12372956 3

Consulta

UNIVERSIDAD DE SEVILLA

Departamento de Matemática Aplicada I

Polinomio de Tutte de Teselaciones Regulares

Tesis 82

068 275

30-07-04

P. D.

Memoria presentada por
Delia Garijo Royo
para optar al grado de
Doctora en Matemáticas
por la Universidad de Sevilla

ESCUELA TECNICA SUPERIOR
INGENIERIA INFORMATICA
- BIBLIOTECA -
N.º ORDEN GENERAL 0 1149745
OBRA N.º ..... TOMO .....
SIGNATURA .....
N.º EN ESPECIALIDAD .....
EJEMPLAR NUMERO R. 14776

Vº Bº de los Directores:

Fdo.: Dr. D. Alberto Márquez Pérez

Fdo.: Dra. Dña. Pastora Revuelta Marchena

Sevilla, Julio del 2004

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA
BIBLIOTECA

UNIVERSIDAD DE SEVILLA

COMISIÓN DE DOCTORADO

UNIVERSIDAD DE SEVILLA
REGISTRO GENERAL
TERCER CICLO
SALIDA
74102 Nº 2004020000651
15-11-2004 13:36:07

Sevilla, 15 de Julio de 2004
N/Ref.: Negociado de Tesis EL/CAR
Asunto: Enviando Tesis Doctoral Leída

ILMO. SR. DIRECTOR DE LA BIBLIOTECA
DE LA E.T.S. DE INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SEVILLA

Adjunto le remito ejemplares de Tesis Doctorales leídas en Departamentos vinculados a esa Facultad a fin de que pasen a formar parte de fondos bibliográficos de consulta de ese Centro.

AUTORES DE LAS TESIS LEÍDAS

- ESCALONA CUARESMA, MARÍA JOSÉ
- GARIJO ROYO, DELIA
- GONZÁLEZ REGAÑA, ALFONSO JOSÉ

LA JEFA DE SECCIÓN DE DOCTORADO

Fdo.: Yolanda Díaz Rolando.

UNIVERSIDAD DE SEVILLA

Sección de Doctorado. Pabellón de México. Paseo de las Delicias. Esq. Eritaña. 41013 Sevilla. Tfno. 95 448 74 18 - 06 - 09. Fax 954 48 74 1

A mi madre,
por su increible capacidad de superación.

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIAL
BIBLIOTECA

Agradecimientos

Tengo que agradecer a muchas personas su colaboración en este proyecto, así que intentaré no olvidarme de ninguna.

En primer lugar quiero dar las gracias a todo el profesorado del Departamento de Matemática Aplicada I de la Escuela de Aparejadores por permitirme y facilitarme en todo momento el uso de sus instalaciones.

Además, quiero agradecer la buena acogida que he tenido por parte del profesorado del Departamento de Matemática Aplicada I en todas las escuelas en las que he impartido clases.

Si tuviera que elegir una persona sin la cual yo considero que este trabajo no hubiera llegado a término es sin duda Pastora Revuelta. A ella tengo que agradecerle su facilidad de trato en todo momento, su disponibilidad y su implicación en este proyecto. Creo que uno no sabe a veces demostrar su agradecimiento por una persona por ello quiero destacar aquí que la cualidad que más me ha ayudado en estos años ha sido su calidad humana.

También quiero agradecer a Alberto Márquez sus consejos determinantes en muchos momentos, a Isidoro Gitler sus revisiones exhaustivas, a Feliú Sagols sus aportaciones informáticas, a Marc Noy y Anna de Mier el haber motivado una parte de esta memoria, a Manuel Ros el apoyo informático que me ha prestado y a Emilio Briales su amabilidad y sus consejos.

A nivel personal me gustaría resaltar el apoyo incondicional que he recibido por parte de toda mi familia, en particular, de mi marido y mi hijo.

Resumen

En esta memoria estudiamos diversos aspectos del polinomio de Tutte de una teselación regular. Comenzamos introduciendo algunas definiciones y resultados significativos de Teoría de Grafos.

En primer lugar nos centramos en el cálculo del polinomio de Tutte de teselaciones o mosaicos del plano mediante cuadrados, triángulos, hexágonos y las posibles combinaciones de estos tres tipos de polígonos regulares, además de estudiar diversos problemas de enumeración en varias áreas de matemáticas relacionados con dicho polinomio. Ofrecemos un sistema que permite codificar estructuras en principio tan dispares como las teselaciones regulares del plano, y un algoritmo efectivo que automatiza la llamada definición recursiva del polinomio de Tutte, y que permite el cálculo de dicho polinomio de fragmentos de teselaciones de grandes dimensiones. Hacemos hincapié en la importancia de este algoritmo ya que permite por primera vez alcanzar dimensiones considerables, no solo en fragmentos de la malla cuadrada, sino en fragmentos de cualquier teselación plana mediante cuadrados, triángulos y hexágonos. Este avance supone el poder mejorar cotas de límites asintóticos y el poder obtener importantes resultados en problemas de enumeración en Teoría de Grafos: el cálculo de orientaciones acíclicas y el cálculo de números de Whitney; y en dos problemas clásicos de Geometría, el cálculo del número de celdas en que arreglos de hiperplanos dividen a espacios euclídeos de altas dimensiones y el cálculo del número de vértices de un zonotopo.

Dejando a un lado los problemas de enumeración, nos preguntamos hasta qué punto el polinomio de Tutte determina al grafo al cual está asociado. Esta cuestión surge debido a la cantidad de invariantes asociados a un grafo contenidos en el polinomio de Tutte. En esta memoria demostramos la existencia de grandes familias de grafos que tienen esta propiedad de unicidad, a la que denominamos Tutte unicidad. Las tres familias estudiadas son los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6. Todas ellas tienen en común la propiedad de ser localmente planas y esto va a permitir probar que todas estas familias de grafos son localmente orientables, herramienta necesaria para

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

demostrar su Tutte unicidad. En primer lugar probamos que los grafos locamente cuadriculados están unívocamente determinados por su polinomio de Tutte. A continuación clasificamos las teselaciones hexagonales y los grafos localmente C_6. Estas clasificaciones adquieren gran importancia, a parte de por su propia complejidad, por ser la rectificación de la clasificación dada por Thomassen [35] en 1991. Estudiamos además distintos invariantes asociados a estas familias de grafos y demostramos la existencia de una relación de menor con los grafos localmente cuadriculados. Por último, establecemos toda la maquinaria necesaria para demostrar la Tutte unicidad de estas familias y debido a la similitud del resto de los casos, que no aportan nada nuevo desde el punto de vista matemático, nos centramos en demostrar la Tutte unicidad de la teselación hexagonal toroidal.

Introducción

La línea de investigación en la que se desarrolla esta memoria es la Teoría de Grafos. Dentro de este ámbito es usual asociar invariantes a grafos a fin de estudiar algunas de sus propiedades. Estos invariantes pueden ser de distinto tipo:

- Estructurales como la conectividad.
- Numéricos como el número de vértices y aristas.
- Algebraicos como el espectro de la matriz de adyacencia.
- Polinómicos como el polinomio cromático.

Esta memoria está destinada al estudio de un invariante de tipo polinómico: el polinomio de Tutte.

El polinomio de Tutte es un polinomio en dos variables originalmente definido para grafos por Tutte y Whitney [36] y posteriormente generalizado para matroides por Crapo [10]. En un principio fue concebido como una extensión del polinomio cromático, pero actualmente se le conoce por tener muchas aplicaciones en combinatoria y otras áreas de matemáticas. A lo largo de esta memoria comentaremos muchas de estas aplicaciones, entre ellas su relación con problemas clásicos de enumeración en Geometría, en concreto, con arreglos de hiperplanos y zonotopos. Comprobaremos como el polinomio de Tutte se convierte en una herramienta muy potente de cálculo tanto del número de celdas en que arreglos de hiperplanos dividen a espacios euclídeos de altas dimensiones, como del número de vértices que poseen determinados zonotopos. Esta estructura geométrica se está utilizando actualmente para modelizar el problema de la rentabilidad máxima en transmisión de datos [25]. Dada una serie de canales la pregunta sería cuantos receptores son necesarios para obtener la mayor rentabilidad en la comunicación. En nuestro caso, los canales son aristas, los receptores son vértices y la estructura resultante es un zonotopo.

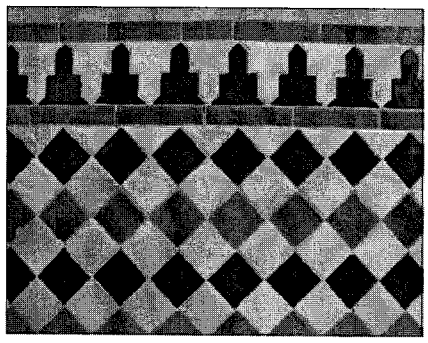
Además de las aplicaciones geométricas del polinomio de Tutte, también nos detendremos en comentar sus aplicaciones en el campo de la Física, en particular en Mecánica Estadística [38] . Mostraremos que dos modelos clásicos para el estudio de la energía de un material, los modelos de Ising (1925) y Potts (1952)

tienen una interpretación como evaluaciones del polinomio de Tutte.

Una de las características más importantes del polinomio de Tutte es que contiene gran información acerca del grafo al cual está asociado. De su estudio podemos obtener resultados sobre coloraciones, orientaciones, flujos y también sobre propiedades estructurales tales como la conectividad.

Esta memoria está estructurada en dos partes, el motivo es que en cada parte utilizamos una definición distinta del polinomio de Tutte. Existen diversas definiciones equivalentes de este polinomio, nosotros nos vamos a centrar en dos. La llamada definición recursiva, que es muy útil para estudiar problemas de enumeración, y está basada en dos de las operaciones más potentes en Teoría de Grafos, la contracción y el borrado de aristas, y una definición combinatoria más clásica que lo relaciona con otro polinomio también asociado a grafos. La primera parte de la memoria está compuesta por tres capítulos y en ella estudiamos el polinomio de Tutte asociado a teselaciones[1] del plano mediante tres tipos de polígonos regulares: cuadrados, triángulos y hexágonos.

El arte de teselar comenzó muy pronto en la historia de la civilización. Tenemos famosos ejemplos en la Alhambra de Granada (Figura 1), en Portugal, en Asia central (anterior a la invasión Mongola en 1259) etc. Estas teselaciones consistían en mosaicos o baldosas hechas de piedra, cerámica o algún material similar que tenían que cubrir el plano o alguna otra superficie sin huecos ni solapamientos apreciables.



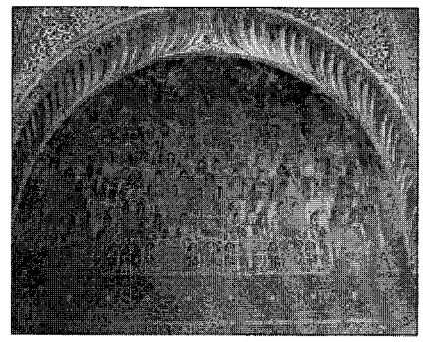


Figura 1: Mosaicos de la Alhambra de Granada

[1]En la literatura matemática también son usados los términos pavimento, mosaico y enlosado con el mismo significado.

Desde muchos puntos de vista, una extensión de estas ideas es natural y útil. Nosotros consideraremos una teselación como cualquier partición del plano en regiones a las que llamaremos baldosas. Las regiones estudiadas en esta memoria son polígonos regulares, concretamente, cuadrados, triángulos y hexágonos. Este tipo de grafos despierta un gran interés, además de por su atractivo como estructura, por el hecho de ser la solución para muchos problemas, fundamentalmente en el campo de la Física. En efecto, los llamados modelos de vértices de Mecánica Estadística [8], que explican entre otros fenómenos el comportamiento del hielo y el comportamiento magnético del grafito, no son más que teselaciones del plano que utilizan fundamentalmente cuadrados y hexágonos.

Aunque el arte de diseñar teselaciones o mosaicos es muy antiguo y está muy desarrollado, el estudio de las propiedades matemáticas de las teselaciones es muy reciente y quedan todavía muchas áreas sin explorar. En el pasado han habido muchos intentos de describir, sistematizar y establecer notaciones para varios tipos de teselaciones. Sin embargo, sin una base matemática estos intentos no han tenido éxito. Esta falta de éxito se puede comprobar en los trabajos de Bourgoin (1873) [6], (1880) [7]; Day (1903) [11] y Wersin (1953) [40] entre otros.

En la segunda parte de la memoria dejamos a un lado los problemas de enumeración y nos planteamos una cuestión bien distinta: hasta qué punto el polinomio de Tutte determina al grafo al cual está asociado. Esta cuestión surge debido a la cantidad de información que el polinomio de Tutte asociado a un grafo posee acerca de dicho grafo. Es natural preguntarse si dos grafos con el mismo polinomio de Tutte son isomorfos. La respuesta a esta pregunta es negativa: el polinomio de Tutte no tiene por qué determinar al grafo al cual esté asociado. Esta parte de la memoria está enfocada a la búsqueda de grandes familias de grafos que tengan esta propiedad de unicidad, a la que se denomina Tutte unicidad.

Hemos trabajado con tres familias de grafos: los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6. Estas familias tienen dos características comunes. La primera de ellas es que son estructuras totalmente regulares y la segunda es que todos los grafos pertenecientes a estas familias son grafos con una estructura local, es decir, en los grafos localmente cuadriculados todo vértice pertenece exactamente a cuatro cuadrados. En las teselaciones hexagonales, todo vértice pertenece exactamente a tres hexágonos y en los grafos localmente C_6 todo vértice pertenece exactamente a seis triángulos; tal y como mostramos en la Figura 2.

Los resultados sobre Tutte unicidad dan idea de la fuerza del polinomio de Tutte como invariante y permiten comparar este polinomio con otros invariantes polinómicos como el polinomio cromático. También es important observar que

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

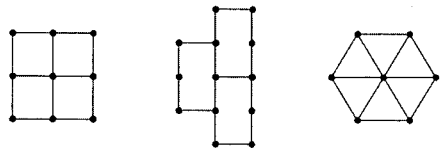


Figura 2: Estructuras Locales

la Tutte unicidad de un grafo proporciona una caracterización de este objeto en términos de algunos invariantes básicos ya que, los conceptos esenciales en la demostración de que un grafo es Tutte único son los invariantes numéricos contenidos en el polinomio de Tutte.

La Tutte unicidad ha sido mayormente estudiada para matroides. Se han encontrado grandes familias de matroides que son Tutte únicas. Respecto a grafos, conocemos diversos resultados sobre Tutte unicidad. Tenemos constancia de que los autores Mier y Noy han demostrado la Tutte unicidad de algunas familias de grafos muy conocidas como los grafos multipartitos completos, las ruedas, escaleras e hipercubos [31]. Además, en [29] se demuestra la Tutte unicidad de la cuadrícula toroidal.

Comenzamos la memoria introduciendo algunas notaciones y terminología de Teoría de Grafos y Algorítmica. En el Capítulo 2 codificamos las teselaciones del plano mediante tres tipos de polígonos regulares: cuadrados, triángulos, hexágonos y sus posibles combinaciones. Conseguimos sistematizar la definición recursiva del polinomio de Tutte, obteniendo sistemas matriciales que permiten calcular el polinomio de Tutte asociado a fragmentos de dichas teselaciones. Debido a las dimensiones de las matrices obtenidas y para un mejor entendimiento del lector realizamos manualmente los cálculos para fragmentos de teselaciones del plano relativamente pequeños y para obtener el polinomio de Tutte asociado a fragmentos mayores ofrecemos un algoritmo que permite sistematizar la definición recursiva de dicho polinomio. Resaltamos el hecho de que este algoritmo permite por primera vez el cálculo del polinomio de Tutte de fragmentos de dimensiones considerables de cualquier teselación del plano mediante cuadrados, triángulos y hexágonos. El Capítulo 3 está dedicado al estudio de dos evaluaciones del polinomio de Tutte: las orientaciones acíclicas y los números de Whitney; y dos aplicaciones de tipo geométrico, los arreglos de hiperplanos y los zonotopos. Mejoramos acotaciones de límites asintóticos y conseguimos métodos de cálculo de orientaciones acíclicas, números de Whitney, del número de celdas en que arreglos de hiperplanos dividen a espacios euclídeos de altas dimensiones y del número de vértices de determinados zonotopos. Finalizamos esta parte de la memoria con una serie de conclusiones y problemas abiertos.

Hemos estructurado la segunda parte de la memoria en cuatro capítulos. En el Capítulo 5, demostramos que los grafos localmente cuadriculados son Tutte únicos. Para ello utilizamos dos invariantes de tipo numérico, la longitud mínima de los ciclos esenciales y el número de ciclos de dicha longitud. Damos una definición de ciclo esencial más general, en el sentido de que no depende de la inmersión del grafo, que la dada por Márquez, Mier, Noy y Revuelta [29]; y demostramos que ambas definiciones son equivalentes. Además, calculamos los valores exactos del número de ciclos esenciales de longitud mínima de los grafos localmente cuadriculados, hecho fundamental para demostrar su Tutte unicidad. En el Capítulo 6, clasificamos las teselaciones hexagonales y los grafos localmente C_6 demostrando además la existencia de una relación de menor muy potente entre estas familias de grafos y los grafos localmente cuadriculados. Estos teoremas de clasificación adquieren gran importancia entre otros motivos por haber sido la rectificación de la clasificación dada por Thomassen [35] en 1991. El Capítulo 7 está destinado a establecer la maquinaria necesaria para demostrar que las teselaciones hexagonales y los grafos localmente C_6 son Tutte únicos. Nos centramos en la Tutte unicidad de las teselaciones hexagonales debido a que reproducir el desarrollo para los grafos localmente C_6 no nos aporta nada nuevo desde el punto de vista matemático. Finalizamos la segunda parte de la memoria con una serie de conclusiones y problemas abiertos.

Por último, queremos resaltar que varias partes de esta memoria han sido presentadas en diversos foros [13, 16, 17]. Además han surgido artículos, algunos de los cuales han sido ya aceptados para publicación y otros se encuentran actualmente en proceso de referee [12, 14, 15].

Índice general

Resumen IX

Introducción XI

Índice general XVII

1. Preliminares 1

1.1. Teoría de Grafos 1

1.2. Algoritmos 7

1.3. Polinomio de Tutte 8

I. Primera parte 13

2. Polinomio de Tutte de Teselaciones Planas 19

2.1. Introducción 19

2.2. Teselaciones Regulares del Plano 22

2.3. Polinomio de Tutte de la Malla Cuadrada 26

2.4. Polinomio de Tutte de la Malla Hexagonal y de la Malla Triangular 35

3. Aplicaciones 41

3.1. Introducción 41

3.2. Orientaciones Acíclicas 42

3.3. Arreglos de Hiperplanos y Zonotopos 53

3.4. Números de Whitney de Primera Clase 58

4. Conclusiones y Problemas Abiertos 65

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

II. Segunda parte **67**

5. Tutte Unicidad de los Grafos Localmente Cuadriculados **71**
5.1. Introducción 71
5.2. Clasificación de los Grafos Localmente Cuadriculados 74
5.3. Tutte Unicidad 77

6. Teselaciones hexagonales y Grafos Localmente C_6 **103**
6.1. Introducción 103
6.2. Clasificación de las Teselaciones Hexagonales 105
6.3. Clasificación de los Grafos Localmente C_6 126
6.4. Relación de Menor con los Grafos Localmente Cuadriculados . . . 128

7. Teselaciones Hexagonales y Grafos Localmente C_6: Tutte Unicidad **137**
7.1. Introducción 137
7.2. Orientación Local en las Teselaciones Hexagonales 139
7.3. Contando Conjuntos de Aristas 145
7.4. Tutte Unicidad 152

8. Conclusiones y Problemas Abiertos **163**

A. Instrucciones para el uso de los programas *poltutte*, *poltutte2* y *poltutte3* **165**

B. Listados **177**

Referencias **237**

CAPÍTULO 1

Preliminares

Este capítulo está dedicado a definir algunos conceptos de Teoría de Grafos y Algorítmica que aparecen con frecuencia en esta memoria. Establecemos además algunas notaciones y recordamos resultados básicos de Teoría de Grafos. Damos dos definiciones equivalentes de polinomio de Tutte, recordamos algunas de sus propiedades y definimos el concepto de Tutte único. Estos conceptos pueden encontrarse en los libros de Harary [23], Bollobas [5] y Welsh [39].

1.1. Teoría de Grafos

Comenzamos esta sección fijando notaciones y terminología referente a Teoría de Grafos que serán utilizadas a lo largo de la memoria.

Un *grafo* G es un par $(V(G), A(G))$ siendo $V(G)$ un conjunto cuyos elementos reciben el nombre de *vértices* y $A(G)$ un conjunto de pares de vértices cuyos elementos se conocen como *aristas*. Siempre que no haya lugar a confusión denotaremos V al conjunto de vértices y A al de aristas. Un grafo $G' = (V', A')$ es un *subgrafo* de $G = (V, A)$ si $V' \subseteq V$ y $A' \subseteq A$.

Una arista de G de extremos u y v se denota $\{u, v\}$. Si $\{u, v\}$ es una arista, decimos que los vértices u y v son *adyacentes* y que la arista $\{u, v\}$ es *incidente* con u y también con v. En todas las figuras a lo largo de la memoria, los vértices están representados por puntos y unimos dos puntos por una línea si el par correspondiente de vértices es una arista.

BIBLIOTECA

Los vértices y aristas de un grafo pueden ser etiquetados con números o letras. Fijada una etiquetación en los vértices de un grafo G, llamamos *matriz de adyacencia del grafo* a una matriz $A = (a_{ij})$ de dimensión $|V| \times |V|$ donde:

$$a_{ij} = \begin{cases} 1 & \text{si} \quad \{u_i, u_j\} \in A \\ 0 & \text{si} \quad \{u_i, u_j\} \notin A \end{cases}$$

La matriz de adyacencia de un grafo nos informa de las adyacencias existentes en dicho grafo. Evidentemente, un grafo tiene distintas matrices de adyacencia asociadas en función de la etiquetación de los vértices del grafo.

Denotamos por $N(v)$ al conjunto de *vecinos* de un vértice v, esto es, el conjunto de vértices adyacentes a v. Llamamos *valencia* de v, $val(v)$, a $|N(v)|$, es decir, al número de aristas incidentes con v. Si todos los vértices de un grafo tienen valencia d se dice que el grafo es *d-regular*. Si $d = 3$, el grafo se denomina *cúbico*.

Para cada n entero positivo, definimos el *grafo completo* K_n como el grafo de n vértices en el que cada par de vértices es adyacente. Por tanto, los vértices de un grafo completo con n vértices tienen valencia $n - 1$ y el número de aristas del grafo es $n(n-1)/2$.

De entre las aristas de un grafo G queremos destacar tres tipos:

1.- Un *lazo* es una arista del tipo $\{u, u\}$.

2.- Una *hoja* es una arista del tipo $\{u, v\}$ con $val(v) = 1$.

3.- G tiene *aristas múltiples* si existe más de una arista uniendo dos de sus vértices. Si existen exactamente dos aristas de extremos u, v se dice que la arista $\{u, v\}$ es *doble*.

Llamamos *grafo simple* al grafo que no contiene lazos ni aristas múltiples. En caso contrario lo llamamos *multigrafo*. A lo largo de esta memoria trabajaremos con ambos tipos de grafos. Al igual que hemos destacado algunos tipos de aristas queremos destacar un tipo de vértices, los *vértices aislados*, definidos como los vértices que tienen valencia 0.

§ 1.1.1. Operaciones de Grafos.— Vamos a definir tres operaciones clásicas en Teoría de Grafos: la contracción y el borrado de aristas y la eliminación de vértices. Consideramos un grafo G con un conjunto de vértices V y un conjunto de aristas A.

• *Borrar una arista* $e \in A$ es considerar el grafo $(V, A - \{e\})$. El borrado de una arista supone perder una única arista del grafo pero nunca un vértice.

• *Contraer una arista* en un grafo consiste en identificar los extremos de dicha arista. El grafo resultante tiene un vértice menos y una arista menos. Si la arista a contraer es doble, la contracción produce un lazo.

• La *eliminación de un vértice* de un grafo G consiste en considerar el grafo que tiene por conjunto de vértices $V - v$ y por conjunto de aristas el formado por todas las aristas de G excepto las incidentes con v.

Mostramos un ejemplo de las tres operaciones definidas en la Figura 1.1.

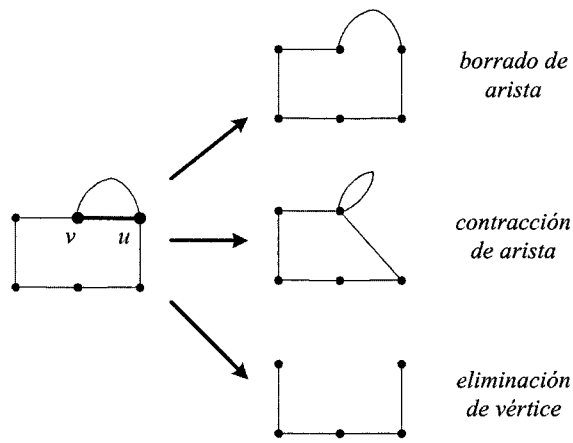


Figura 1.1: Contracción y borrado de una arista y eliminación de un vértice

Un *menor* de un grafo es todo grafo obtenido mediante una sucesión de contracciones y borrados de aristas.

§ 1.1.2. Caminos y Ciclos. Conectividad.— Un *camino* en un grafo G es una sucesión de vértices $v_0, v_1, \ldots, v_k$ tal que $v_i \neq v_j$ para todo i, j y $\{v_i, v_{i+1}\} \in A$ con $0 \leq i \leq k-1$. Si $\{v_0, v_k\} \in A$ decimos que el camino es un *ciclo*. La *longitud* de un camino (análogamente de un ciclo) es el número de aristas pertenecientes al camino (respectivamente al ciclo). Si un camino tiene k vértices entonces tiene $k-1$ aristas pero los ciclos tienen el mismo número de vértices que de aristas. A los ciclos de longitud n los denotamos C_n. Para $n = 3, 4, 5, 6$ también nos referiremos a ellos como triángulos, cuadrados, pentágonos y hexágonos. A los caminos y ciclos

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

de longitud l los denotaremos *$l-$caminos* y *$l-$ciclos* respectivamente. Llamamos *distancia* entre dos vértices u y v, y la denotamos $d(u,v)$, a la longitud mínima de los caminos que unen u con v.

Podemos establecer una relación de equivalencia entre los vértices de un grafo de tal forma que dos vértices están relacionados si existe un camino que los une. Las clases de equivalencia generadas por esta relación se conocen como *componentes conexas* del grafo (Figura 1.2a). Una componente conexa está formada por todos los vértices del grafo que pueden ser unidos entre sí por un camino de aristas. Si el grafo tiene una única componente conexa, decimos que es *conexo.*

Para cada subconjunto X de V denotamos por $G-X$ al grafo que tiene a $V-X$ como conjunto de vértices y cuyas aristas son las aristas de G que tienen ambos extremos en $V-X$. Dado un grafo G y un subconjunto E de A, denotamos por $G|E$ al grafo (V,E).

Se dice que un grafo G es *$k-$conexo* si tiene al menos $k+1$ vértices y para cada conjunto $X \subset V$ con $|X| < k$, $G-X$ es conexo (Figura 1.2b). Al mayor entero k tal que G es $k-$conexo se le denomina *conectividad* de G. Un grafo es *$l-$arista-conexo* si para cada subconjunto $E \subseteq A$ con $|E| < l$ el grafo $(V, A-E)$ es conexo (Figura 1.2c). Al mayor entero l tal que G es $l-$arista-conexo se le llama la *arista-conectividad* de G.

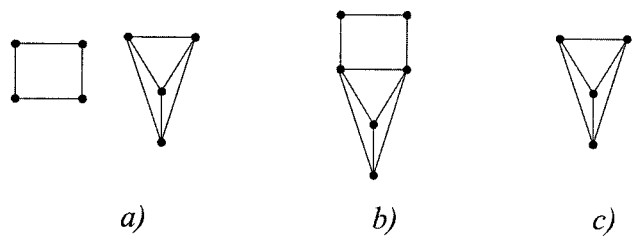


Figura 1.2: a) Grafo con 2 componentes conexas b) Grafo $2-$conexo c) Grafo $3-$arista-conexo

Para cada subconjunto $E \subseteq A$ el *rango* es $r(E) = |V| - k(G|E)$ siendo $k(G|E)$ el número de componentes conexas del grafo $G|E$.

§ 1.1.3. Inmersión de Grafos.— Una *inmersión* de un grafo G en una superficie S es una representación del grafo en la superficie de tal forma que no se

produzcan cortes entre las aristas y que cada componente de $S - G$ sea un disco abierto.

En la primera parte de esta memoria trabajaremos con *grafos planos*. Un grafo se dice plano si admite una inmersión en el plano (Figura 1.3). Los grafos planos están caracterizados mediante uno de los resultados más importantes de Teoría de Grafos, el Teorema de Kuratowsky [27].

En la segunda parte de la memoria trabajaremos con otras dos superficies: el *toro topológico* y *la botella de Klein*. En la Figura 1.4 mostramos dos visiones del toro cuadriculado. Si cortamos y desarrollamos por un paralelo y un meridiano obtenemos la representación mostrada en la Figura 1.4b. Análogamente las etiquetaciones $1, 2, 3$ etc son las identificaciones a realizar para obtener la superficie mostrada en la Figura 1.4a.

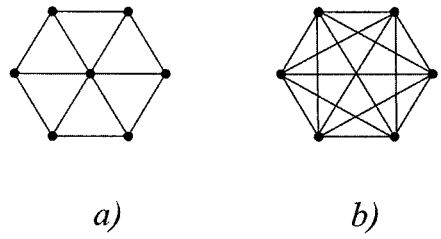


Figura 1.3: a) Grafo plano b) El grafo completo K_6 no es plano

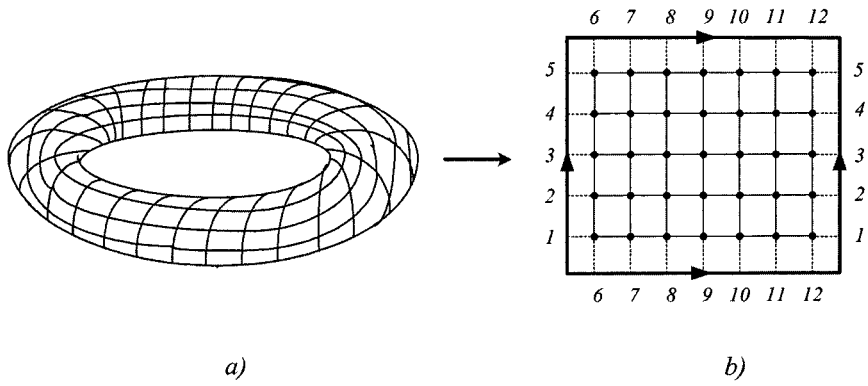


Figura 1.4: Dos visiones del toro cuadriculado

Una inmersión de un grafo en una superficie genera una partición de la superficie en una serie de regiones conexas llamadas *caras*. Con el término conexo nos referimos a que dos puntos cualesquiera de la región pueden ser unidos por una curva que no toca ninguna arista o vértice del grafo.

Sea $G = (V, A)$ un grafo inmerso en una superficie S. Se define la *característica de Euler* de G como $|V| + |A| - |C|$ siendo $|C|$ el número de caras del grafo G.

Lema 1.1. *La característica de Euler de un grafo G es 0 si y sólo si G admite una inmersión en el toro o en la botella de Klein.*

§ 1.1.4. Árboles.— Un *árbol* es un grafo conexo que no contiene ciclos. Un *bosque* es un grafo que no contiene ciclos pero que no tiene que ser necesariamente conexo. Dado un grafo conexo $G = (V, A)$, llamamos *árbol generador de G* a un árbol cuyo conjunto de vértices es V.

En muchas ocasiones se distingue un vértice de un árbol. Este vértice recibe el nombre de *raíz*. Para estudiar los árboles con raíz se disponen los vértices por niveles, estando el vértice raíz en el nivel 0, sus adyacentes en el nivel 1 etc. Un vértice de un árbol con raíz se llama una *hoja* si está en el nivel i y no es adyacente a ningún vértice del nivel $i + 1$. Decimos que un grafo G es de *ancho de árbol no acotado*[1] si a cualquier árbol contenido en G podemos añadirle tantos niveles como queramos.

§ 1.1.5. Isomorfismo de Grafos.— Decimos que dos grafos $G_1 = (V_1, A_1)$ y $G_2 = (V_2, A_2)$ son *isomorfos*, y lo denotamos $G_1 \simeq G_2$, si existe una biyección $\gamma : V_1 \to V_2$ tal que:

$$\{u_1, v_1\} \in V_1 \Leftrightarrow \{\gamma(u_1), \gamma(v_1)\} \in V_2$$

El concepto de grafos isomorfos va a ser muy importante en la segunda parte de esta memoria. Estaremos interesados en demostrar que dos grafos son no isomorfos, para ello utilizaremos los llamados *invariantes de grafos*.

Un *invariante de grafos bajo un isomorfismo* es una función g tal que si $G_1 \simeq G_2$ entonces $g(G_1) = g(G_2)$. Nosotros trabajaremos con diversos invariantes de grafos, entre ellos, el número de vértices, el número de aristas, la conectividad y la vértice-transitividad. Un grafo se dice *vértice-transitivo* si para cualesquiera

[1] unbounded tree width

dos vértices del grafo, u y v existe un automorfismo de grafos, σ sobre el conjunto $V(G)$ tal que $\sigma(u) = v$. Si un grafo es vértice-transitivo es totalmente simétrico, es decir, desde cualquier vértice del grafo se tiene la misma visión de dicho grafo.

§ 1.1.6. Coloraciones. Polinomio Cromático.— Una *coloración* de un grafo G con λ colores es una aplicación $c : V \to \{1, 2, \ldots, \lambda\}$ tal que si $\{u, v\}$ es una arista de G entonces $c(u) \neq c(v)$. El *número cromático* de G, denotado $\chi(G)$, es el mínimo $\lambda \in \mathbb{N}$ tal que existe una coloración de G con λ colores. El *polinomio cromático asociado a* G es un polinomio en una variable $\chi(G, \lambda)$ tal que para todo valor entero positivo de λ, $\chi(G, \lambda)$ es el número de coloraciones propias de G con λ colores.

Un grafo $G = (V, A)$ se dice *bipartito* si existe una partición del conjunto V en dos conjuntos V_1 y V_2 tales que cada arista de A contiene un vértice de V_1 y otro de V_2 y A es no vacío. Son bien conocidas las siguientes caracterizaciones de grafos bipartitos en función del número cromático y de la longitud de sus ciclos.

Lema 1.2. *Un grafo es bipartito si y sólo si tiene número cromático igual a dos.*

Lema 1.3. *Un grafo es bipartito si y sólo si no contiene ciclos de longitud impar.*

1.2. Algoritmos

Un *algoritmo* es una sucesión de instrucciones que tienen como fin resolver un problema. La *complejidad* de un algoritmo depende de dos factores, el tiempo que tarda en obtener la solución del problema y el espacio que ocupa. En los últimos años, el espacio que ocupa un algoritmo ha pasado a ser un factor secundario debido en gran medida al aumento de la capacidad de almacenamiento de los ordenadores y a la disminución del precio de los elementos de memoria. Así, uno de los elementos de estudio más importantes en el diseño de algoritmos es la *eficiencia o complejidad del algoritmo.*

La *eficiencia de un algoritmo* es el término utilizado para establecer la relación entre el esfuerzo necesario para resolver un ejemplo cualquiera de un problema y el tamaño del ejemplo. En nuestro caso, el problema consistirá en estudiar

la eficiencia de un algoritmo para el cálculo del polinomio de Tutte asociado a determinados grafos y diversas evaluaciones de dicho polinomio. Por tanto, una medida del tamaño del ejemplo será el número de vértices del grafo a estudiar. Por otra parte, para medir el esfuerzo contamos el número de operaciones significativas que efectúa el algoritmo.

Para expresar la eficiencia de un algoritmo utilizaremos la *notación asintótica*, debida a Bachmann [2] y popularizada por Knuth [26]. Esta notación se basa en los conjuntos de funciones $O(f(n))$, $\Omega(f(n))$ y $\Theta(f(n))$ pero de todas ellas, en esta memoria, sólo vamos a utilizar la función $O(f(n))$. Para obtener la eficiencia de un algoritmo tenemos que analizar los peores casos que pueden presentarse entre las posibles entradas del problema. Denotaremos $O(f(n))$ al conjunto de funciones $g(n)$ que verifican que para cualquier constante positiva C existe un número natural m tal que $g(n) \leq Cf(n)$ para todo $n \geq m$, es decir:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

Por tanto, si un algoritmo requiere un tiempo que viene dado por un polinomio de segundo grado, por ejemplo $2n^2 + 1 \leq (2+1)n^2 = 3n^2$, diremos que el algoritmo es $O(n^2)$. En este contexto, está claro que una eficiencia $O(log(n))$ será mejor que $O(n)$. Un algoritmo se dice *polinomial* si es $O(n^p)$. Si $p = 0$ el algoritmo es *constante* y si $p = 1$ el algoritmo se denomina *lineal.*

1.3. Polinomio de Tutte

El polinomio de Tutte fue definido inicialmente por Tutte [36] como un invariante de grafos. Actualmente, es uno de los invariantes de grafos más estudiados en combinatoria. En esta memoria nos centraremos en el polinomio de Tutte asociado a grafos, aunque como ya hemos comentado este polinomio también puede estar asociado a matroides [10].

El *polinomio generador de rango-tamaño* asociado a un grafo G se define de la siguiente forma:

$$R(G; x, y) = \sum_{E \subseteq A} x^{r(E)} y^{|E|}$$

Por tanto, el coeficiente de $x^i y^j$ en $R(G; x, y)$ es el número de subgrafos de G de rango i y j aristas.

Definición 1.1. El *polinomio de Tutte* asociado a un grafo $G = (V, A)$ se define como:

$$T(G; x, y) = \sum_{E \subseteq A} (x-1)^{r(A)-r(E)} (y-1)^{|E|-r(E)}$$

El polinomio generador de rango-tamaño y el polinomio de Tutte asociados a un grafo contienen la misma información acerca de dicho grafo, es decir, ambos polinomios están definidos de forma análoga. De hecho podemos obtener uno de los polinomios en función del otro:

$$\begin{aligned} R(G; x, y) &= (xy)^{r(G)} T\left(G; \tfrac{xy+1}{xy}, y+1\right) \\ T(G; x, y) &= (x-1)^{r(G)} R\left(G; \tfrac{1}{(x-1)(y-1)}, y-1\right) \end{aligned}$$

Por tanto, el polinomio de Tutte asociado a un grafo G nos dice para cada i y j el número de conjuntos de aristas en G de rango i y tamaño j; entendiendo por *tamaño de un conjunto de aristas* E a $|E|$. Sin embargo, el polinomio de Tutte tiene propiedades muy interesantes no compartidas por el polinomio generador de rango-tamaño. En particular, el polinomio de Tutte satisface una regla de contracción-borrado que da lugar a la que llamamos *definición recursiva del polinomio de Tutte.*

Definición 1.2. Sea G un grafo y $T(G; x, y)$ su polinomio de Tutte asociado. Si G no tiene aristas entonces $T(G; x, y) = 1$. En caso contrario, dada $e \in A(G)$:

(1) $T(G; x, y) = T(G'_e; x, y) + T(G''_e; x, y)$ donde G'_e es el grafo resultante de borrar la arista e de G y G''_e es el grafo resultante de contraer la arista e de G.

(2) $T(G; x, y) = xT(G'_e; x, y)$ si e es una hoja.

(3) $T(G; x, y) = yT(G'_e; x, y)$ si e es un lazo.

Aplicando de forma sucesiva las condiciones (1), (2) y (3) se obtiene una expresión del polinomio de Tutte. La definición recursiva es consistente, en el sentido de que no importa el orden en que sean elegidas las aristas para borrar y contraer.

A lo largo de esta memoria vamos a utilizar las dos definiciones aquí especificadas de polinomio de Tutte, las Definiciones 1.1 y 1.2. En la primera parte de la memoria, destinada a cálculos de polinomios de Tutte y problemas de enumeración, vamos a utilizar la Definición 1.2. En la segunda parte de la memoria necesitamos tener una visión del polinomio de Tutte como una reformulación del polinomio generador de rango-tamaño, es decir, utilizaremos la Definición 1.1.

Queremos resaltar la importancia del polinomio de Tutte, en el sentido de la gran información que contiene acerca del grafo al cual está asociado. De hecho, el polinomio de Tutte contiene gran cantidad de invariantes numéricos asociados al grafo además de ser una generalización del polinomio cromático y del polinomio de flujo entre otros.

- $T(G;1,1)$ =nº árboles recubridores de un grafo conexo G.
- $T(G;2,1)$ =nº bosques de un grafo.
- $T(G;2,0)$ =nº orientaciones acíclicas de un grafo.
- El polinomio cromático asociado a G es:

$$\chi(G,\lambda) = (-1)^{r(A)}\lambda^{k(G)}T(G;1-\lambda,0)$$

siendo $k(G)$ el número de componentes conexas del grafo $G = (V, A)$.

- El polinomio de flujo asociado a un grafo G es:

$$F(G,\lambda) = (-1)^{|A|-r(A)}T(G;0,1-\lambda)$$

Por último queremos comentar la relación existente entre los polinomios de Tutte asociados a un grafo y su grafo dual. Llamamos *dual geométrico* de un grafo G (Figura 1.5) y lo denotamos G^* al grafo cuyo conjunto de vértices está formado por las caras de G y dos vértices son adyacentes si y sólo si comparten una arista.

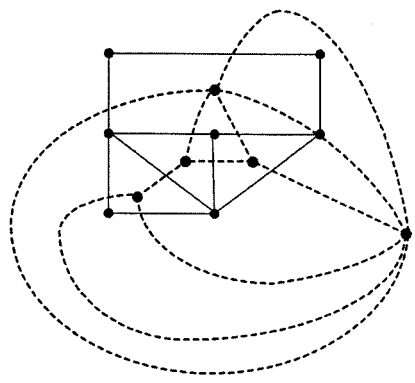


Figura 1.5: Un grafo y su dual (punteado)

Teorema 1.4. *Si G es un grafo plano entonces $T(G;x,y) = T(G^*;y,x)$.*

En el caso de grafos no planos, no tenemos constancia de la existencia de ningún resultado que relacione el polinomio de Tutte de un grafo y su dual.

§ 1.3.1. Tutte Unicidad.— Para definir el concepto de Tutte unicidad suponemos que todos los grafos considerados no contienen vértices aislados.

Dos grafos G y H son *Tutte equivalentes* si $T(G;x,y) = T(H;x,y)$. Una clase de grafos C es $T-$*cerrada* si para cada grafo $G \in C$ todos los grafos Tutte equivalentes a G pertenecen a C. Un grafo G es *Tutte único* si cada grafo Tutte equivalente a G es isomorfo a G.

Los grafos Tutte únicos están unívocamente determinados por su polinomio de Tutte, en otras palabras, basta con estudiar su polinomio de Tutte para distinguirlo de cualquier otro grafo. La segunda parte de esta memoria está destinada a la búsqueda de grandes familias de grafos que tengan esta propiedad de unicidad.

I

Primera parte

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA
BIBLIOTECA

La primera parte de esta memoria está dedicada al cálculo del polinomio de Tutte, mediante su definición recursiva, de un fragmento de dimensión $m \times n$ de una teselación del plano mediante tres tipos de polígonos regulares: cuadrados, hexágonos y triángulos, y sus posibles combinaciones. Así como sus aplicaciones en problemas clásicos de enumeración tanto en Teoría de Grafos: cálculo de orientaciones acíclicas y de números de Whitney; como en Geometría: enumeración de las caras de un arreglo de hiperplanos y de los vértices de un zonotopo.

Queremos hacer hincapié en el hecho de que en 1988 Read y Tutte propusieron como problema abierto y "endiabladamente difícil de resolver"(problema 8.1 en [33]) encontrar el polinomio cromático (que no es más que una evaluación del polinomio de Tutte a lo largo del eje de abcisas) de un fragmento de dimensión $m \times n$ de la malla cuadrada. Resaltamos por tanto el hecho de que en la primera parte de esta memoria calculamos el polinomio de Tutte no solo de un fragmento de la malla cuadrada sino también de un fragmento de cualquier teselación del plano mediante cuadrados, triángulos, hexágonos y sus posibles combinaciones.

A parte de la importancia en sí del polinomio de Tutte como invariante asociado a un grafo que engloba a polinomios ampliamente estudiados como el polinomio cromático y el de flujo, queremos también resaltar su relación con dos modelos clásicos de Mecánica Estadística: los modelos de Ising y Potts [38]. Ambos modelos utilizan una teselación del plano mediante cuadrados conocida por malla cuadrada para situar los spines a fin de maximizar o minimizar la energía de un material. Actualmente se está aplicando el modelo de Ising utilizando la malla hexagonal (comportamiento magnético del grafito).

Un *Tutte-Gorthendieck invariante* ((TG)-invariante) es una función f definida sobre una clase de grafos, cerrada bajos dos operaciones: la suma directa y la toma de menores; verificando dos condiciones:

(1) $f(G) = af(G'_e) + bf(G''_e)$ siendo G'_e el grafo resultante de borrar la arista e de G y G''_e el resultante de contraer dicha arista.

(2) $f(G_1 \oplus G_2) = f(G_1)f(G_2)$ donde $G_1 \oplus G_2$ denota la suma directa de dos grafos.

En 1991, Welsh demuestra que todo (TG)-invariante tiene una interpretación como evaluación del polinomio de Tutte [38], en particular, los modelos de Ising y Potts.

Esta primera parte de la memoria está estructurada en tres capítulos. Comenzamos el primer capítulo definiendo las teselaciones del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos tres tipos de polígonos regulares,

estableciendo un sistema que nos permite codificar fragmentos de estas teselaciones, basado fundamentalmente en tres conceptos: el grafo base, las capas y las bandas. A continuación, y a fin de poder sistematizar la definición recursiva del polinomio de Tutte, definimos los conceptos de vértice mayor y arista mayor de un fragmento de teselación del plano. Para un mejor entendimiento del lector, realizamos manualmente los cálculos de los polinomios de Tutte asociados a las tres únicas teselaciones del plano mediante polígonos regulares cuyas baldosas son todas del mismo tipo y tamaño y tal que toda arista de la teselación pertenece exactamente a dos baldosas [21]: la malla cuadrada, la malla hexagonal y la malla triangular; analizando la eficiencia del algoritmo desarrollado. Por último, ofrecemos una serie de programas informáticos que codifican fragmentos de teselaciones del plano realizadas con los tres tipos de polígonos regulares anteriormente mencionados y que calculan los polinomios de Tutte asociados a dichos fragmentos utilizando la definición recursiva del polinomio.

Basándonos en la potencia del polinomio de Tutte, en el sentido de la cantidad de información que posee acerca del grafo al cual está asociado, en el Capítulo 3 estudiamos dos aplicaciones del algoritmo desarrollado en el Capítulo 2 en problemas de enumeración en Teoría de grafos: el cálculo de orientaciones acíclicas y el cálculo de números de Whitney; y dos aplicaciones del algoritmo en problemas de enumeración en Geometría: el cálculo del número de caras de un arreglo de hiperplanos y el cálculo del número de vértices de un zonotopo.

En primer lugar, tratamos las orientaciones acíclicas como una evaluación del polinomio de Tutte. Esto permite aplicar todos los resultados obtenidos en el Capítulo 2. A continuación estudiamos la mejora que supone tratar el problema de forma independiente, utilizando una definición recursiva propia de las orientaciones acíclicas y no la del polinomio de Tutte. Por último, hacemos una incursión en el tema de las matrices de compatibilidad para abordar el cálculo de las orientaciones acíclicas de la malla cuadrada. Tanto los resultados obtenidos para las orientaciones acíclicas como los obtenidos en el Capítulo 2, son aplicables a números de Whitney, arreglos de hiperplanos y zonotopos. Nuestro interés por los números de Whitney, se basa en que son los coeficientes del polinomio cromático asociado a un grafo. Recordemos en este punto que la conjetura de Read abierta desde 1968 [32] nos dice que estos coeficientes forman una sucesión unimodal. En esta memoria, obtenemos formas de cálculo de números de Whitney de algunas familias de grafos y los programas informáticos que adjuntamos permiten comprobar que la conjetura de Read es cierta para familias de grafos con un número considerable de vértices. Los cálculos realizados también nos permiten ofrecer resultados sobre orientaciones acíclicas con una única fuente, utilizando la relación existente entre este tipo de orientaciones y los números de Whitney.

Por último, nos adentramos en el campo de la Geometría estudiando arreglos de hiperplanos y zonotopos asociados a fragmentos de teselaciones del plano. Los programas informáticos que adjuntamos proporcionan métodos de cálculo del número de celdas en que arreglos de hiperplanos asociados a fragmentos de grandes dimensiones dividen a espacios euclídeos de altas dimensiones. Además también obtenemos métodos de cálculo del número de vértices de zonotopos asociados a dichos fragmentos de teselaciones del plano.

Finalmente, en el Capítulo 4 establecemos algunas conclusiones y problemas abiertos.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

CAPÍTULO 2

Polinomio de Tutte de Teselaciones Planas

Este capítulo está dedicado al cálculo del polinomio de Tutte asociado a un fragmento de dimensión $m \times n$ de una teselación del plano mediante tres tipos de polígonos regulares: cuadrados, triángulos y hexágonos, y sus posibles combinaciones. Necesitamos definir en primer lugar el concepto de fragmento de dimensión $m \times n$ de una teselación regular del plano, no tanto por los casos en los que teselemos el plano únicamente con cuadrados, triángulos o hexágonos, sino por los casos en los que consideremos las combinaciones de estos polígonos regulares. Además, definimos los conceptos de grafo base, banda y capa necesarios para codificar dichas teselaciones. Damos un algoritmo para el cálculo del polinomio de Tutte asociado a estos grafos y analizamos su eficiencia. Para un mejor entendimiento del lector, desarrollamos matemáticamente el método para tres fragmentos de teselaciones distintos: la malla cuadrada, la malla hexagonal y la malla triangular. Por último adjuntamos una serie de programas informáticos que permiten codificar fragmentos de teselaciones planas de dimensiones considerables y calcular su polinomio de Tutte asociado.

2.1. Introducción

Una *teselación plana* Γ es una familia numerable de conjuntos cerrados $\{\Gamma_1, \Gamma_2, \ldots\}$ que recubren el plano sin solapamientos ni huecos. Más explícitamente, la unión de los conjuntos $\Gamma_1, \Gamma_2, \ldots$ es el plano y los interiores de los conjuntos Γ_i tienen que ser dos a dos disjuntos. Por *el plano* nos referimos al plano euclídeo de la geometría elemental y a cada Γ_i se le denomina *baldosa*.

Para nuestro propósito, esta definición de teselación plana es demasiado general, ya que incluye por ejemplo teselaciones cuyas baldosas no sean regiones simplemente conexas (tengan agujeros). Por ello vamos a establecer algunas restricciones en la definición de teselación plana:

a) Cada baldosa tiene que ser un disco topológico cerrado.

b) Cada arista (o lado) de una baldosa Γ_i es también una arista de exactamente una baldosa Γ_j con $i \neq j$.

Esta condición determina el concepto de *baldosas vecinas*. Dos baldosas de una teselación se dicen *vecinas* si comparten una arista.

c) Cualquier baldosa de la teselación tiene que tener un número finito de vecinos.

De entre todas las teselaciones planas que cumplen las condiciones anteriores, en este capítulo nos centramos en las teselaciones del plano mediante tres tipos de polígonos regulares: cuadrados, triángulos y hexágonos y sus posibles combinaciones. Mostramos algunos ejemplos en la Figura 2.1.

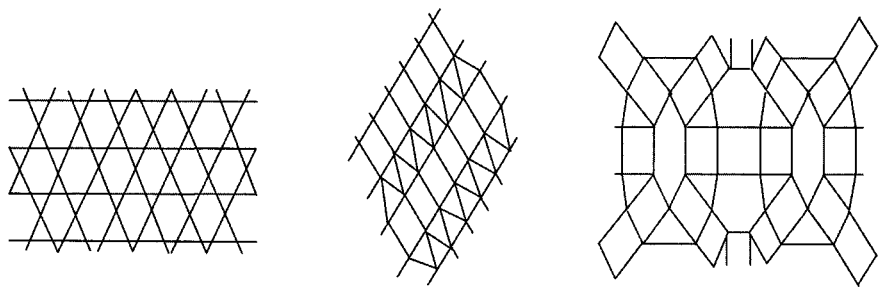


Figura 2.1: Tres fragmentos de teselaciones del plano realizadas con cuadrados, triángulos y hexágonos.

El estudio de las teselaciones del plano por polígonos regulares lo inició Kepler en 1619 [24] y tienen gran aplicación en el campo de la Física. Hemos comentado la relación de este tipo de teselaciones con el comportamiento magnético del grafito, el comportamiento del hielo y dos modelos clásicos de Mecánica Estadística: los modelos de Ising y Potts [8, 38]. Sabemos que ambos modelos tienen una interpretación como una evaluación del polinomio de Tutte. En este capítulo, estudiamos métodos de cálculo del polinomio de Tutte asociado a fragmentos de teselaciones del plano mediante polígonos regulares.

Existen diversas definiciones equivalentes de polinomio de Tutte, entre ellas, la llamada definición recursiva en la cual nos vamos a centrar en esta primera parte de la memoria.

Sea G un grafo y $T(G;x,y)$ su polinomio de Tutte asociado. Si G no tiene aristas entonces $T(G;x,y)=1$. En caso contrario, para toda arista $e \in A(G)$:

(1) $T(G;x,y) = T(G'_e;x,y) + T(G''_e;x,y)$ donde G'_e es el grafo resultante de borrar la arista e de G y G''_e es el grafo resultante de contraer la arista e de G.

(2) $T(G;x,y) = xT(G'_e;x,y)$ si e es una hoja.

(3) $T(G;x,y) = yT(G'_e;x,y)$ si e es un lazo.

Como ya hemos comentado, el polinomio de Tutte asociado a un grafo está bien definido en el sentido de que puede ser calculado recursivamente sin que importe el orden en que elegimos las aristas a borrar y contraer.

A continuación, vamos a poner un ejemplo sencillo de aplicación de la definición recursiva del polinomio de Tutte para que el lector compruebe la complejidad de dicha definición, en el sentido de que con un número pequeño de vértices y aristas ya es necesario aplicar varias veces las operaciones de contracción y borrado para obtener una expresión del polinomio de Tutte. Además, pequeños aumentos en el número de vértices y aristas del grafo provoca un aumento enorme en el número de operaciones a realizar. El polinomio de Tutte asociado a un grafo compuesto por una única arista es x (basta aplicar (2) de la definición recursiva). Por otra parte, la Figura 2.2 muestra el diagrama de contracción y borrado de un cuadrado. Aplicando la definición recursiva tenemos que $T(\square;x,y) = x^3+x^2+x+y$.

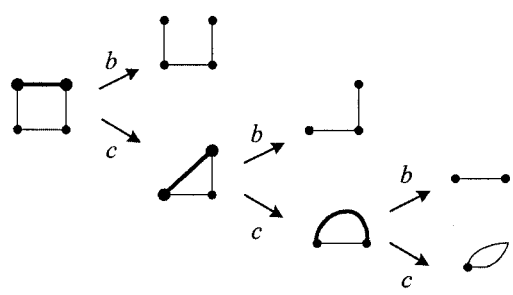


Figura 2.2: Diagrama de contracción y borrado de un cuadrado

Debido al aumento del número de operaciones a realizar en relación al aumento

del número de vértices y aristas del grafo se plantea la necesidad de automatizar la definición recursiva del polinomio de Tutte. Para ello, en primer lugar es necesario codificar los grafos a los cuales vamos a aplicar dicha definición. En la Sección 2, definimos el concepto de fragmento de teselación de dimensión $m \times n$ y damos todas las definiciones necesarias para poder codificarlo. En las Secciones 3 y 4, describimos el algoritmo para sistematizar la definición recursiva del polinomio de Tutte, aplicándolo a tres fragmentos de teselaciones del plano: la malla cuadrada, la malla hexagonal y la malla triangular de dimensión $m \times n$. Estas son las tres únicas teselaciones del plano mediante polígonos regulares cuyas baldosas son todas del mismo tipo y tamaño y tal que toda arista de la teselación pertenece exactamente a dos baldosas [21].

El algoritmo que desarrollamos produce sistemas matriciales que permiten calcular el polinomio de Tutte asociado a fragmentos de teselaciones del plano mediante cuadrados, triángulos, hexágonos y sus posibles combinaciones. Debido a la dimensión de las matrices obtenidas adjuntamos tres programas informáticos, **poltutte**, **poltutte2** y **poltutte3** que sirven para generar fórmulas recursivas para calcular el polinomio de Tutte asociado a fragmentos de dimensiones considerables.

2.2. Teselaciones Regulares del Plano

Esta sección está dedicada a establecer el mecanismo necesario para poder codificar fragmentos de teselaciones del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos tres polígonos regulares. A partir de ahora y por simplicidad, siempre que hablemos de teselaciones del plano nos referiremos a aquellas que se realizan únicamente con cuadrados, triángulos y hexágonos cumpliendo las condiciones (b) y (c) de la sección anterior.

Una propiedad fundamental de las teselaciones planas es que pueden ser cortadas por líneas paralelas, que pueden ser líneas rectas o producirse zigzags de aristas [21] (ver Figura 2.3*a*). Al grafo producido al cortar por dos líneas paralelas consecutivas lo denominamos *capa de la teselación* (Figura 2.3b) y a cada línea la llamamos *frontera de la capa.* Podemos etiquetar los vértices de una capa distinguiendo la *frontera inferior*, cuyos vértices son de la forma $(x, 0)$ y la *frontera superior* cuyos vértices están etiquetados $(x', 1)$ con $x, x' \in \mathbb{N}$.

Definición 2.1. La *unión de una capa* C_1 *con una capa* C_2, denotada $C_1 \cup C_2$, es el grafo resultante de identificar los vértices de la frontera superior de C_1 con los vértices de la frontera inferior de C_2. De forma equivalente podemos definir *la unión de* j *capas*, $C_1 \cup C_2 \cup \ldots \cup C_j$ como $((C_1 \cup C_2) \cup C_3) \cup \ldots$ (Figura 2.3c).

La necesidad de distinguir las fronteras de una capa parte del hecho de que la unión de capas no es en general una propiedad conmutativa. Por otra parte, está claro que toda teselación regular del plano Γ es unión infinita de capas denotadas C_i. Por tanto:

$$\Gamma = \bigcup_{i=1}^{\infty} C_i$$

Definición 2.2. Llamamos *fragmento de tamaño* n de una teselación regular del plano Γ y lo denotamos Γ_n a la unión de n capas consecutivas de Γ, es decir,

$$\Gamma_n = \bigcup_{i=i_0}^{i_0+n-1} C_i$$

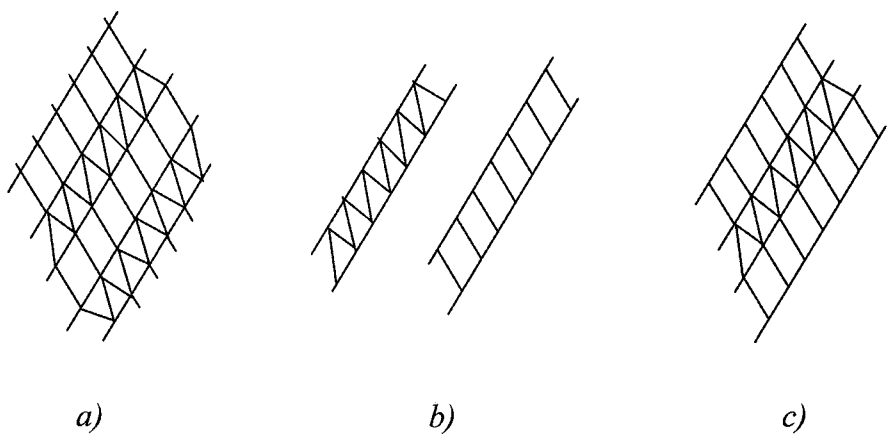


Figura 2.3: *a*) Teselación regular del plano b) Dos capas de la teselación c) Unión de tres capas

Definición 2.3. Dados dos fragmentos de una teselación regular del plano, F_1 y F_2, decimos que F_1 *es menor que* F_2 y lo denotamos $F_1 < F_2$ si el número de capas de F_1 es menor que el número de capas de F_2.

Definición 2.4. El *grafo base* de una teselación regular del plano Γ es el fragmento menor de la teselación tal que la unión infinita de copias de dicho fragmento sea Γ (ver Figuras 2.4 y 2.5).

Un grafo base puede estar formado por una o más capas. En general, cuando en las teselaciones participan los tres tipos de polígonos regulares el grafo base está formado por más de una capa. Además, existen una regiones que se repiten dentro del grafo base, a las que llamamos *bandas*. Las bandas pueden ser estructuras muy simples como un cuadrado o estructuras realmente complejas. Mostramos un ejemplo en las Figuras 2.4 y 2.5. En la primera de ellas damos una teselación regular del plano y en la segunda el grafo base de dicha teselación, el cual, está formado por 4 capas y hemos sombreado las bandas.

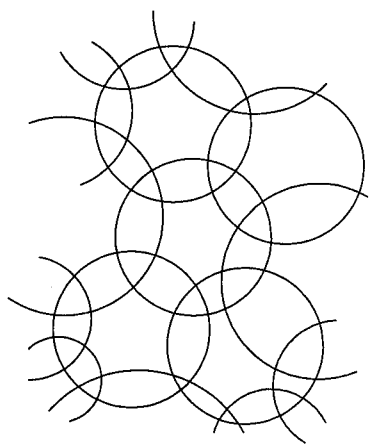


Figura 2.4: Teselación del plano mediante cuadrados, triángulos y hexágonos

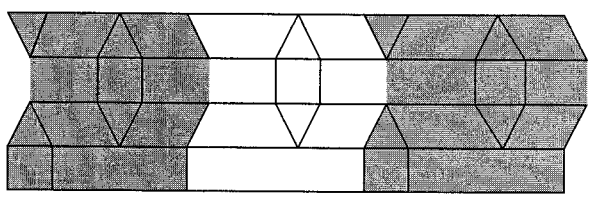


Figura 2.5: Grafo base de la teselación mostrada en la Figura 2.4

Para automatizar la definición recursiva del polinomio de Tutte necesitamos considerar fragmentos finitos de teselaciones, por ello vamos a definir los conceptos de *fragmento de dimensión* $m \times n$ y *grafo base de anchura* m.

Definición 2.5. Llamamos *fragmento de dimensión* $m \times n$ de una teselación regular plana Γ y lo denotamos Γ_{mn} al fragmento de tamaño n resultante de considerar un número máximo de vértices en cada frontera de cada capa igual a m. (Figura 2.6a). Además, definimos el *grafo base de anchura* m como el grafo finito resultante de considerar un número máximo de vértices en cada frontera de cada capa del grafo base de Γ igual a m (Figura 2.6b).

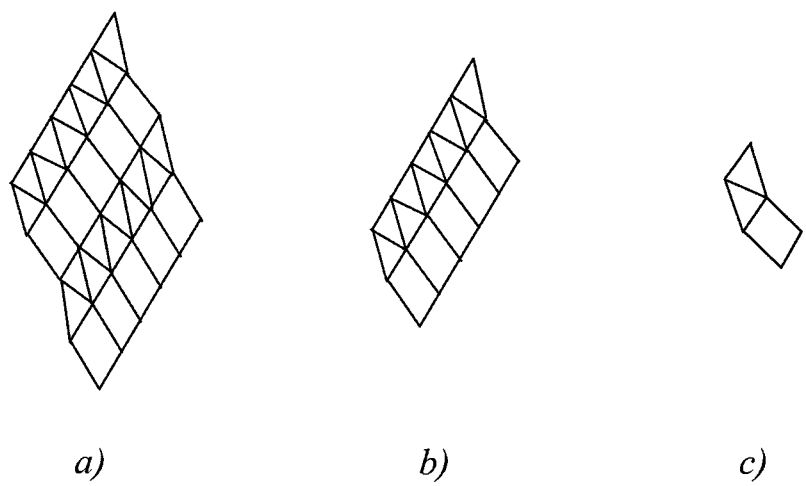


Figura 2.6: a) Fragmento de dimensión 6×4 b) Grafo base de anchura 6 c) Banda del grafo base

Es evidente que los fragmentos de teselaciones son grafos conexos y que la diferencia entre un fragmento de tamaño n y un fragmento de dimensión $m \times n$ es que el primero es un grafo infinito y el segundo no lo es. Fijando m conseguimos que cada capa tenga un número finito de vértices y el resultado es un grafo finito al que podremos aplicar nuestro algoritmo. Los conceptos de *grafo base*, *capa* y *banda* son fundamentales para codificar fragmentos de teselaciones del plano. Además, son los datos necesarios para poder utilizar los programas **poltutte**, **poltutte2** y **poltutte3** para calcular el polinomio de Tutte de fragmentos de teselaciones de gran tamaño.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

2.3. Polinomio de Tutte de la Malla Cuadrada

El polinomio de Tutte asociado a la malla cuadrada está siendo muy estudiado en los últimos años. Parte de su interés se basa en que el polinomio de Tutte de un grafo de ancho de árbol acotado se puede calcular en tiempo polinomial y la malla cuadrada es el menor grafo de ancho de árbol no acotado. Además como ya hemos comentado en 1988 Read y Tutte propusieron como problema abierto (problema 8.1 en [33]) encontrar el polinomio cromático (evaluación del polinomio de Tutte a lo largo del eje de abcisas) de un fragmento de dimensión $m \times n$ de la malla cuadrada.

La *malla cuadrada* $L_{m,n}$ (Figura 2.7) se define como el grafo $P_m \times P_n$ siendo P_l un camino con l vértices. Sus vértices pueden ser identificados con el conjunto de pares ordenados:

$$\{(i,j) \in \mathbb{N}^2 \mid 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$$

y dos vértices, (i,j), (i',j'), son adyacentes si $|i-i'|+|j-j'|=1$.

$L_{m,n}$ tiene nm vértices y $2n(m-1)$ aristas. Además, es un grafo simple, es decir, sin lazos ni aristas múltiples. Su grafo base es $L_{m,2}$, el cual tiene una única capa y $m-1$ bandas (cada una de ellas es un cuadrado). $L_{m,n}$ tiene $n-1$ copias del grafo base y $m-1$ bandas (Figura 2.7). Por otra parte, está claro que:

$$L_{m,n} = \bigcup_{i=1}^{n-1} B_i$$

siendo B_i la copia del grafo base número i.

En esta sección vamos a obtener un algoritmo para el cálculo del polinomio de Tutte asociado a la malla cuadrada $L_{m,n}$ con un valor de m fijado. Para ello vamos a aplicar de forma sucesiva la definición recursiva del polinomio de Tutte, basada en el borrado y la contracción de una arista seleccionada.

En la que llamamos *etapa 0* seleccionamos una arista de $L_{m,n}$ y mediante la operación de contracción y borrado de dicha arista obtenemos dos nuevos grafos (*etapa 1*) en los que a su vez hay que seleccionar una arista y borrar y contraer. Estaremos ante una nueva etapa cada vez que obtengamos una serie de grafos a partir de las operaciones de contracción y borrado de aristas de los grafos de la etapa anterior. No consideramos un cambio de etapa la eliminación de hojas y lazos. A todos estos grafos que surgen a partir de la aplicación de la

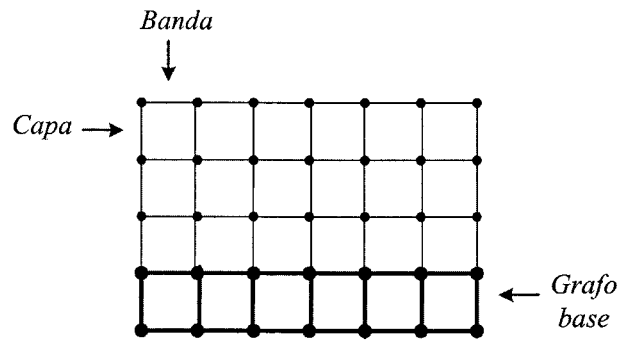


Figura 2.7: Malla cuadrada $L_{7,5}$ con 6 bandas y 4 copias del grafo base. El grafo base es $L_{7,2}$ y tiene una capa y 6 bandas.

regla de contracción y borrado los llamamos *grafos de la descomposición* y los denotamos G_i^j siendo i el índice que distingue dos grafos de la descomposición y j el número de copias del grafo base que posee dicho grafo. Así, $L_{m,n} = G_0^{n-1}$. Además, denotamos por $(G_i^j)_b$ al grafo resultante de borrar una arista en G_i^j y $(G_i^j)_c$ al grafo resultante de contraer una arista en G_i^j.

Es evidente que se plantean dos problemas en la aplicación de la definición recursiva del polinomio de Tutte. El primero de ellos es decidir en cada momento la arista que vamos a borrar y contraer y el segundo es decidir cuando nos tenemos que detener. Para resolver estas cuestiones vamos a utilizar las siguientes definiciones.

Definición 2.6. Sean $u = (i, j)$ y $v = (i', j')$ dos vértices de $L_{m,n}$. Decimos que *u es mayor que v* (Figura 2.8) y lo denotamos $u > v$ si:

$$i^2 + j^2 > (i')^2 + (j')^2$$

ó

$$i^2 + j^2 = (i')^2 + (j')^2 \text{ y } j > j'$$

El *vértice mayor* del grafo es aquél que es mayor que todos los demás vértices (Ver Figura 2.8).

De forma equivalente podemos definir el concepto de *vértice mayor en G_i^j* (Figura 2.8b), ya que los vértices de un grafo cualquiera de la descomposición conservan la etiquetación de los vértices de $L_{m,n}$ y al contraer una arista, el

BIBLIOTECA

vértice resultante de la contracción toma la etiqueta del vértice menor de los dos extremos de la arista contraída.

Definición 2.7. Sea $v \in V(G_i^j)$ y dos aristas, e y e', incidentes con él. Etiquetamos las aristas $e = \{v, (i, j)\}$ y $e' = \{v, (i', j')\}$. Decimos que e *es mayor que* e' y lo denotamos $e > e'$ si $j > j'$ ó $j = j'$ y $i > i'$ (Figura 2.8).

Hay un único caso en el que dos aristas incidentes en un vértice son *iguales*, es decir, ninguna es mayor que la otra. Este es el caso de las aristas dobles.

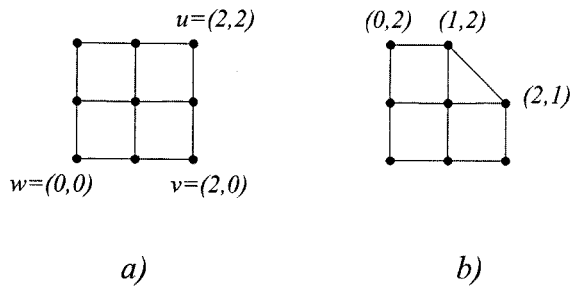


Figura 2.8: *a)* En $L_{3,3}$, $v > w$ y el vértice mayor del grafo es u b) Grafo resultante de la contracción de la arista mayor de $L_{3,3}$. El vértice mayor es $(1, 2)$ y la arista mayor $\{(0, 2), (1, 2)\}$

Con estas definiciones, estamos en condiciones de resolver el primero de los dos problemas planteados en la aplicación de la definición recursiva del polinomio de Tutte: en cada G_i^j seleccionamos la arista mayor del vértice mayor para obtener $(G_i^j)_b$ y $(G_i^j)_c$. Si dicha arista es doble, es decir, entre dos aristas incidentes con el vértice no podemos decidir cuál es la mayor, seleccionamos cualquiera de las dos. Por comodidad en la notación, cuando hablemos de borrar o contraer una arista nos estaremos refiriendo, salvo que se especifique lo contrario, a la arista mayor del vértice mayor.

Definición 2.8. Llamamos *lista asociada a* G_i^j a la 6−upla $[\delta_b, n_b, i_b, \delta_c, n_c, i_c]$ siendo:

- $\delta_b = \begin{cases} 1 & \text{si } (G_i^j)_b \text{ tiene una copia menos del grafo base que } G_i^j \\ 0 & \text{en otro caso} \end{cases}$
- n_b el número de hojas que creamos al borrar la arista en G_i^j.
- i_b el índice i del grafo $(G_i^j)_b$.
- $\delta_c = \begin{cases} 1 & \text{si } (G_i^j)_c \text{ tiene una copia menos del grafo base que } G_i^j \\ 0 & \text{en otro caso} \end{cases}$
- n_c el número de lazos que creamos al contraer la arista seleccionada en G_i^j.
- i_c el índice i del grafo $(G_i^j)_c$.

La lista asociada a un grafo de la descomposición nos ofrece toda la información necesaria para aplicar la fórmula recursiva del polinomio de Tutte a dicho grafo. Por ejemplo, si la lista asociada a un grafo G_i^j de la descomposición es $[0, 0, 10, 1, 1, 8]$ sabemos que el polinomio de Tutte asociado a dicho grafo es $T(G_{10}^j; x, y) + yT(G_8^{j-1}; x, y)$.

Definición 2.9. Llamamos *grafos generadores de* $L_{m,n}$ *en la etapa* k a los grafos G_i^j pertenecientes a dicha etapa tales que $\delta_b(G_i^j) + \delta_c(G_i^j) \geq 1$. Los *grafos generadores de* $L_{m,n}$ son los grafos G_i^1 tales que $\delta_b(G_i^1) + \delta_c(G_i^1) \geq 1$.

Mostramos un ejemplo en la Figura 2.9. Aplicamos las operaciones de contracción y borrado a $L_{2,4}$ hasta la segunda etapa, teniendo en cuenta que no se produce un cambio de etapa con la eliminación de hojas. El grafo base es $L_{2,2}$ (un cuadrado). El grafo etiquetado G_1^2 no es un grafo generador en la etapa 1, ya que al borrar y contraer la arista mayor del vértice mayor (marcada en negro) no perdemos una copia del grafo base (un cuadrado). En cambio, G_2^2 es un grafo generador ya que tiene dos copias del grafo base y al borrar y contraer obtenemos grafos con 1 copia del grafo base, es decir, $\delta_c(G_2^2) + \delta_b(G_2^2) = 2$. Análogamente G_3^1 no es un grafo generador de la etapa 2 porque $\delta_c(G_3^1) + \delta_b(G_3^1) = 0$ y G_4^2 sí lo es ya que $\delta_c(G_4^2) + \delta_b(G_4^2) = 1$. Los grafos generadores de $L_{4,2}$ son G_5^1 y G_6^1.

Los conceptos de lista y grafo generador nos dan la respuesta al segundo problema planteado. El proceso recursivo se detiene cuando hayamos obtenido todos los grafos generadores.

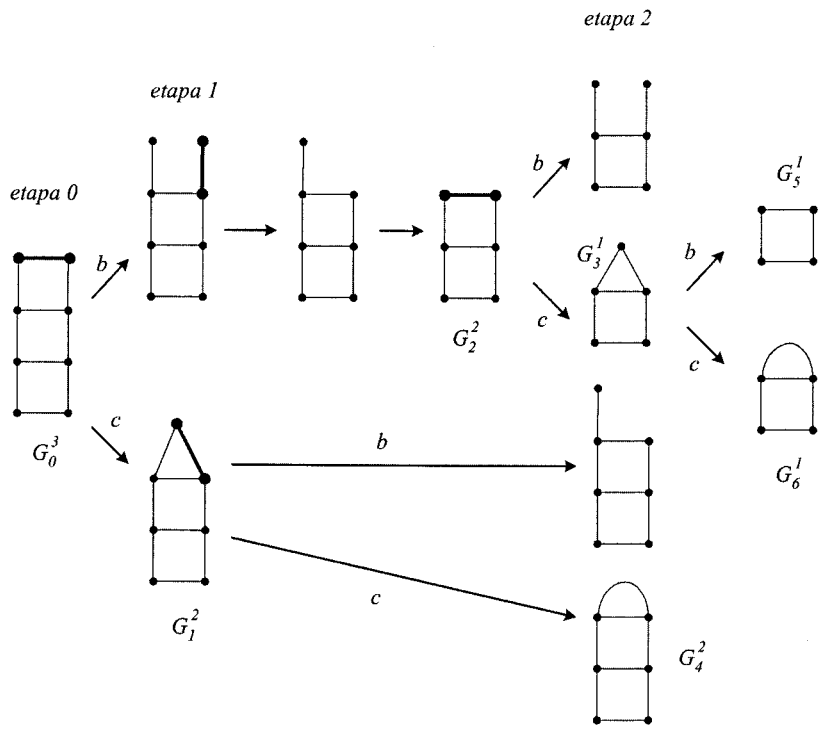


Figura 2.9: Descomposición hasta la etapa 2 de $L_{2,4}$

Para obtener un método de cálculo del polinomio de Tutte asociado a $L_{m,n}$ se describe el polinomio de Tutte de cada uno de los grafos generadores. Por la propia estructura del grafo nos basta saber el efecto que tiene aplicar la regla de contracción y borrado considerando dos copias del grafo base.

En las Figuras 2.10, 2.11 y 2.12 ofrecemos la descomposición de $L_{3,n}$. Estos tres esquemas son necesarios para obtener una ecuación matricial que nos proporcione un método de cálculo del polinomio de Tutte asociado a dicho grafo. A continuación describimos el sistema de notación empleado en los esquemas de descomposición.

Para facilitar su entendimiento, en la primera de estas figuras hemos especificado todos los pasos. Cada vez que aplicamos las operaciones de contracción y borrado escribimos c y b respectivamente. En la Figura 2.10, no hemos considerado como una aplicación de la regla de contracción y borrado la eliminación de hojas y lazos, por ello en todos los pasos consistentes en eliminación de hojas no hemos escrito b. En las Figuras 2.11 y 2.12 hemos suprimido los pasos consistentes

en la eliminación de estos tipos de aristas, pero los hemos incluido en la notación. Si una flecha está etiquetada $b, 3$, significa que para dar el paso correspondiente hay que borrar la arista mayor del vértice mayor y eliminar 3 hojas. Análogamente, $c, 3$ significa que hay que contraer la arista y eliminar 3 lazos. Hay que tener en cuenta que la aparición de lazos está siempre asociada a la operación de contracción y la de hojas a la de borrado. Por último, queremos resaltar que en las tres figuras mencionadas anteriormente, aplicamos de forma sucesiva las operaciones de contracción y borrado hasta obtener un grafo generador o un grafo ya obtenido en el diagrama de la descomposición.

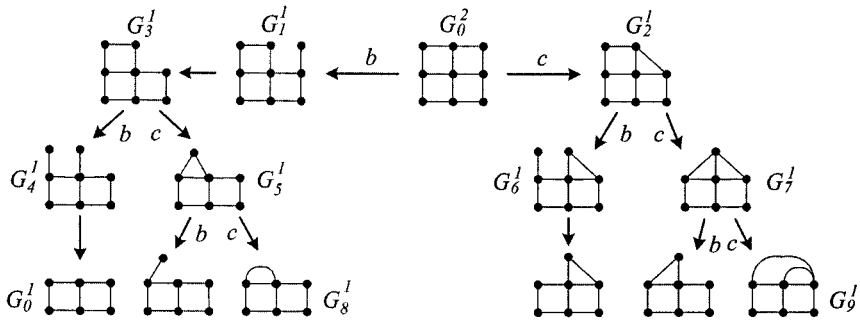


Figura 2.10: Primer esquema de descomposición de $L_{3,n}$

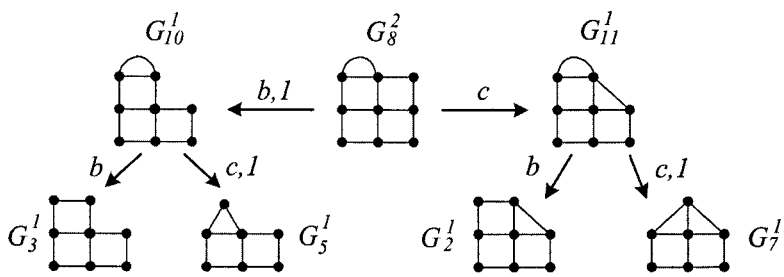


Figura 2.11: Segundo esquema de descomposición de $L_{3,n}$

Los grafos generadores de $L_{3,n}$ son G_0^1, G_8^1, G_9^1 y G_{12}^1. Nuestro siguiente objetivo es obtener una expresión del polinomio de Tutte asociado a G_0^{n-1}, G_8^{n-1}, G_9^{n-1}

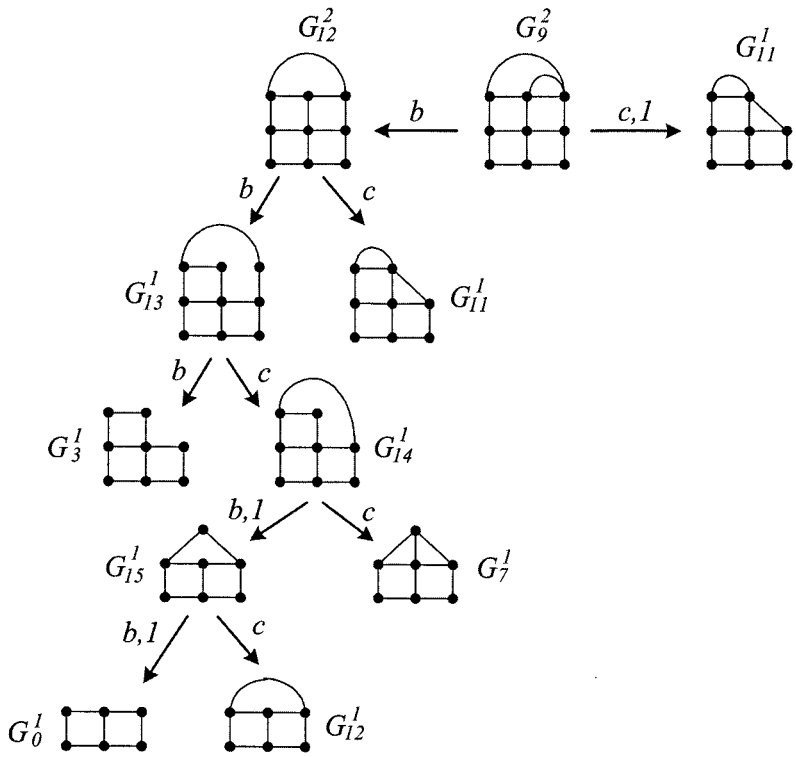


Figura 2.12: Tercer esquema de descomposición de $L_{3,n}$

y G_{12}^{n-1} en función de los grafos generadores. Estas expresiones se obtienen de forma recursiva a partir de los diagramas de descomposición. Por simplicidad, denotamos $T(G)$ al polinomio de Tutte asociado a un grafo G. Veamos como ejemplo el cálculo de $T(G_9^j)$.

La lista de G_9^j es $[0, 0, 12, 1, 1, 11]$, esto implica que $T(G_9^j) = T(G_{12}^j) + yT(G_{11}^j)$. Además, la lista de G_{12}^j es $[1, 0, 13, 1, 0, 11]$ luego

$$T(G_9^j) = T(G_{13}^j) + T(G_{11}^j) + yT(G_{11}^j)$$

Si seguimos aplicando los diagramas de descomposición y la definición recursiva del polinomio de Tutte llegamos a:

$$\begin{aligned} T(G_9^j) &= (x^3 + x^2y + xy^2 + 3x^2 + 2xy + 2x)T(G_0^{j-1}) \\ &+ (xy + y^2 + 2x + 2y + 2)T(G_8^{j-1}) + (y^2 + 2y + 2)T(G_9^{j-1}) + xT(G_{12}^{j-1}) \end{aligned}$$

De forma análoga obtenemos las siguientes expresiones de los polinomios de Tutte asociados a G_0^j, G_8^j y G_{12}^j:

$$
\begin{aligned}
T(G_0^j) &= (x^3+2x^2+x)T(G_0^{j-1})+(2x+1)T(G_8^{j-1})+T(G_9^{j-1}) \\
T(G_8^j) &= (x^3+x^2y+2x^2+xy+x)T(G_0^{j-1})+(xy+2x+y+1)T(G_8^{j-1})+ \\
&\quad (y+1)T(G_9^{j-1}) \\
T(G_{12}^j) &= (x^3+3x^2+xy+2x)T(G_0^{j-1})+(2x+y+2)T(G_8^{j-1})+ \\
&\quad (y+2)T(G_9^{j-1})+xT(G_{12}^{j-1})
\end{aligned}
$$

Matricialmente:

$$
\begin{pmatrix} T(G_0^j) \\ T(G_8^j) \\ T(G_9^j) \\ T(G_{12}^j) \end{pmatrix} = M \begin{pmatrix} T(G_0^{j-1}) \\ T(G_8^{j-1}) \\ T(G_9^{j-1}) \\ T(G_{12}^{j-1}) \end{pmatrix}
$$

siendo M la siguiente matriz:

$$
\begin{pmatrix}
x^3+2x^2+x & 2x+1 & 1 & 0 \\
x^3+x^2y+2x^2+xy+x & xy+2x+y+1 & y+1 & 0 \\
x^3+x^2y+xy^2+3x^2+2xy+2x & xy+y^2+2x+2y+2 & y^2+2y+2 & x \\
x^3+3x^2+xy+2x & 2x+y+2 & y+2 & x
\end{pmatrix}
$$

A partir de la ecuación matricial anterior, es fácil obtener un método de cálculo del polinomio de Tutte asociado a $L_{3,n}$. Recordar que en la notación utilizada $L_{3,n} = G_0^{n-1}$. Para obtener dicho método, en primer lugar hay que calcular el polinomio de Tutte asociado a los grafos generadores. Utilizando la definición recursiva del polinomio de Tutte se comprueba que:

$$
\begin{pmatrix} T(G_0^1) \\ T(G_8^1) \\ T(G_9^1) \\ T(G_{12}^1) \end{pmatrix} = M \begin{pmatrix} x^2 \\ x^2+xy \\ x^2+xy+y^2+x+y \\ x^2+x+y \end{pmatrix}
$$

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

Proposición 2.1. *El polinomio de Tutte asociado a la malla cuadrada $L_{3,n}$ con $n \geq 2$ se obtiene a partir del siguiente sistema matricial:*

$$\begin{pmatrix} T(L_{3,n}) \\ T(G_8^{n-1}) \\ T(G_9^{n-1}) \\ T(G_{12}^{n-1}) \end{pmatrix} = M^{n-2} \begin{pmatrix} T(L_{3,2}) \\ T(G_8^1) \\ T(G_9^1) \\ T(G_{12}^1) \end{pmatrix}$$

El algoritmo desarrollado en esta sección nos proporciona además el siguiente resultado.

Teorema 2.2. *Fijado $m \in \mathbb{N}$, el procedimiento para la evaluación del polinomio de Tutte de la malla cuadrada $L_{m,n}$ en un punto cualquiera del plano (a, b) tiene complejidad computacional $O(log(n))$.*

Demostración. El cálculo del polinomio de Tutte asociado a $L_{m,n}$ en un punto cualquiera del plano (a, b) consta de tres pasos: evaluar los elementos de la matriz M en el punto (a, b), calcular las $n - 2$ potencias de dicha matriz y multiplicar el resultado por el vector formado por los polinomios de Tutte asociados a los grafos generadores evaluados en el punto (a, b).

Todos los pasos mencionados en el párrafo anterior, excepto el cálculo de las $n - 2$ potencias de la matriz se realizan en tiempo constante. Para calcular $n - 2$ potencias de una matriz se necesitan del orden de $log(n)$ operaciones. □

Queremos resaltar que el algoritmo desarrollado nos permite en un tiempo razonable el cálculo del polinomio de Tutte asociado a la malla $12 \times n$, y el poder calcular el polinomio de Tutte de fragmentos mayores depende exclusivamente de la capacidad informática del usuario. Así, este algoritmo adquiere gran importancia como método para el cálculo del polinomio de Tutte pues es la primera vez que se obtienen resultados para fragmentos de dimensiones considerables. Por otra parte, es importante observar que el procedimiento desarrollado en esta sección no nos proporciona el cálculo del polinomio de Tutte asociado a $L_{m,n}$ en tiempo logarítmico sino evaluaciones de dicho polinomio en un punto. La diferencia es que las componentes de la matriz M, si no las evaluamos en un punto, son polinomios y el producto de matrices polinómicas no se realiza en tiempo constante si no limitamos el número de monomios de cada componente de la matriz.

2.4. Polinomio de Tutte de la Malla Hexagonal y de la Malla Triangular

El propósito de esta sección es realizar un desarrollo similar al de la Sección 3 para el cálculo del polinomio de Tutte asociado a la malla hexagonal $H_{m,n}$ y a la malla triangular $T_{m,n}$. Para un mejor entendimiento del lector, en ambas secciones calculamos dicho polinomio para $H_{1,n}$ y $T_{2,n}$ pero los programas **poltutte**, **poltutte2** y **poltutte3** permiten el cálculo para cualquier valor de m. En primer lugar definimos ambas mallas.

Llamamos *malla hexagonal* $m \times n$ (Figura 2.13) y la denotamos $H_{m,n}$ al grafo resultante de eliminar los siguientes vértices y aristas de una malla cuadrada $(m+1) \times (2n+m)$:

$$\{(j,i)|0 \leq j \leq m-2, 0 \leq i \leq m-2-j\}$$
$$\{(j,2k+i)|2 \leq j \leq m+1, m+1-j \leq i \leq m-1\}$$
$$\{\{(i,m-i+2j),(i+1,m-i+2j)\} \mid 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$$

Llamamos *malla triangular* $m \times n$ (Figura 2.13) y la denotamos $T_{m,n}$ al grafo resultante de añadir las siguientes aristas a una malla cuadrada $m \times n$:

$$\{\{(i,j),(i+1,j+1)\} \mid 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$$

Es importante observar que ambas definiciones nos permiten conservar la etiquetación de vértices utilizada en la sección anterior, ya que partimos de una malla cuadrada. Así, todas las definiciones dadas en dicha sección son válidas en los caso de las mallas hexagonal y triangular.

En la Figura 2.13 mostramos los grafos base asociados a estas mallas. Ambos grafos base tienen una única capa y para construir $H_{m,n}$ y $T_{m,n}$ necesitamos respectivamente n y $n-1$ copias del grafo base.

El sistema empleado para obtener un método de cálculo de los polinomios de Tutte asociados a las mallas hexagonal y triangular se basa, al igual que en la sección anterior, en la definición recursiva del polinomio de Tutte. Aplicamos dicha definición sucesivamente, obteniendo los distintos grafos de la descomposición hasta llegar a los grafos generadores. Las Figuras 2.14 y 2.15 muestran los esquemas de descomposición de $H_{1,n}$ y las Figuras 2.16 y 2.17 los correspondientes a $T_{2,n}$.

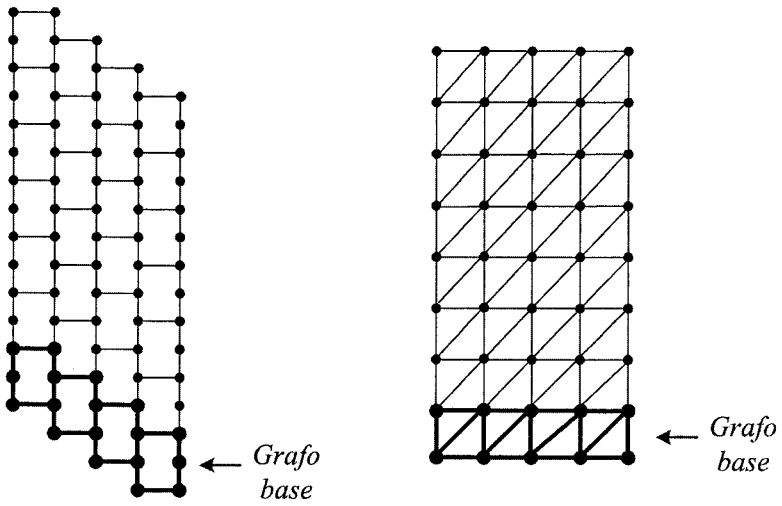


Figura 2.13: Malla hexagonal $H_{4,7}$ y malla triangular $T_{5,9}$ con sus respectivos grafos base

Los grafos generadores de $H_{1,n}$ son G_0^1 y G_4^1. A continuación vamos a obtener una expresión del polinomio de Tutte asociado a G_0^n y G_4^n en función de los grafos generadores. Hay que recordar que $G_0^n = H_{1,n}$.

$$\begin{aligned} T(G_0^j) &= (x^4+x^3+x^2+x)T(G_0^{j-1}) + T(G_4^{j-1}) \\ T(G_4^j) &= (x^4+x^3+x^2+x+y(x^3+x^2+x))T(G_0^{j-1}) + (y+1)T(G_4^{j-1}) \end{aligned}$$

Matricialmente:

$$\begin{pmatrix} T(G_0^j) \\ T(G_4^j) \end{pmatrix} = M \begin{pmatrix} T(G_0^{j-1}) \\ T(G_4^{j-1}) \end{pmatrix}$$

siendo

$$M = \begin{pmatrix} x^4+x^3+x^2+x & 1 \\ x^4+x^3+x^2+x+y(x^3+x^2+x) & y+1 \end{pmatrix}$$

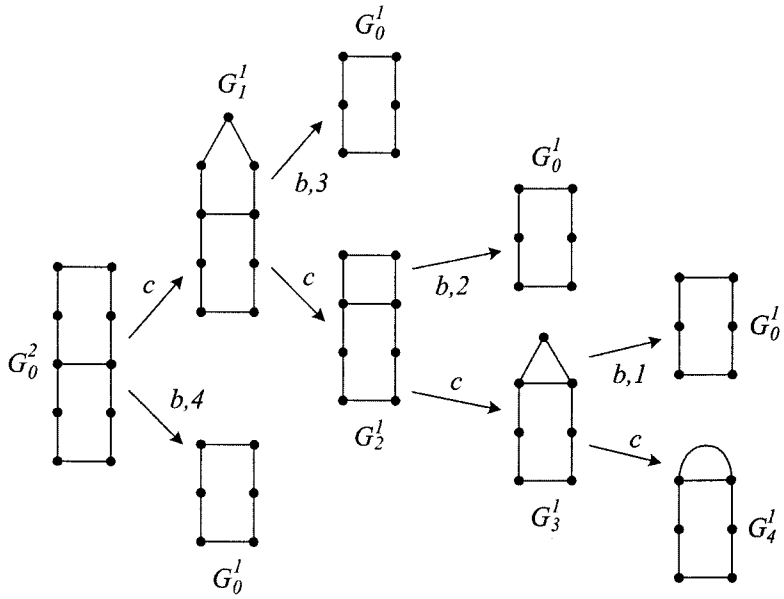


Figura 2.14: Primer esquema de descomposición de $H_{1,n}$

A partir de la ecuación anterior, obtenemos un método de cálculo del polinomio de Tutte asociado a $H_{1,n}$ calculando el polinomio de Tutte asociado a los grafos generadores y $n-1$ potencias de la matriz M.

$$\begin{aligned} T(G_0^1) &= x^5 + x^4 + x^3 + x^2 + x + y \\ T(G_4^1) &= T(G_0^1) + y(x^4 + x^3 + x^2 + x + y) \end{aligned}$$

Proposición 2.3. *El polinomio de Tutte asociado a la malla hexagonal $H_{1,n}$ con $n \geq 1$ se obtiene a partir del siguiente sistema matricial:*

$$\begin{pmatrix} T(H_{1,n}) \\ T(G_4^n) \end{pmatrix} = M^{n-1} \begin{pmatrix} T(H_{1,1}) \\ T(G_4^1) \end{pmatrix}$$

El método desarrollado nos proporciona un resultado equivalente al Teorema 2.2 para la malla hexagonal.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

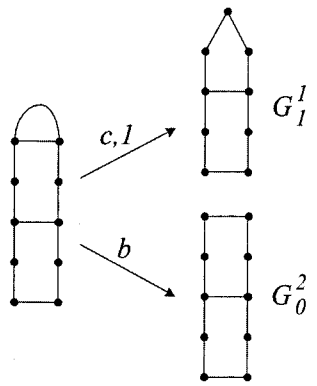


Figura 2.15: Segundo esquema de descomposición de $H_{1,n}$

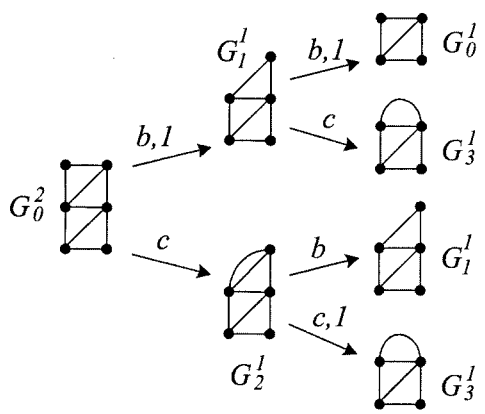


Figura 2.16: Primer esquema de descomposición de $T_{2,n}$

Teorema 2.4. *Fijado $m \in \mathbb{N}$, el procedimiento para la evaluación del polinomio de Tutte de la malla hexagonal $H_{m,n}$ en un punto cualquiera del plano (a, b) tiene complejidad computacional $O(log(n))$.*

Por último, desarrollamos un proceso análogo para obtener un método de cálculo del polinomio de Tutte asociado a la malla triangular $T_{2,n}$. En este caso, los grafos generadores son G_0^1 y G_3^1 (ver Figuras 2.16 y 2.17). Obtenemos una

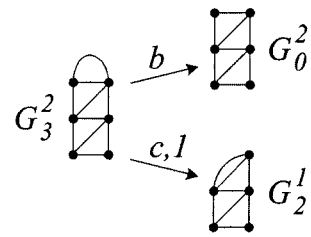


Figura 2.17: Segundo esquema de descomposición de $T_{2,n}$

expresión del polinomio de Tutte asociado a $G_0^{n-1} = T_{2,n}$ y G_3^{n-1} en función de los grafos generadores.

$$\begin{aligned} T(G_0^j) &= (x^2+x)T(G_0^{j-1}) + (x+y+1)T(G_3^{j-1}) \\ T(G_3^j) &= (x^2+x+xy)T(G_0^{j-1}) + (x+y^2+2y+1)T(G_3^{j-1}) \end{aligned}$$

Matricialmente:

$$\begin{pmatrix} T(G_0^j) \\ T(G_3^j) \end{pmatrix} = M \begin{pmatrix} T(G_0^{j-1}) \\ T(G_3^{j-1}) \end{pmatrix}$$

siendo

$$M = \begin{pmatrix} x^2+x & x+y+1 \\ x^2+x+xy & x+(y+1)^2 \end{pmatrix}$$

A partir de la ecuación matricial anterior, obtenemos un método de cálculo del polinomio de Tutte asociado a $T_{2,n}$ calculando el polinomio de Tutte asociado a los grafos generadores y $n-2$ potencias de la matriz M.

$$\begin{aligned} T(G_0^1) &= x^3+2x^2+2xy+y^2+x+y \\ T(G_3^1) &= T(G_0^1) + y(x^2+y^2+xy+x+y) \end{aligned}$$

Proposición 2.5. *El polinomio de Tutte asociado a la malla triangular $T_{2,n}$ con $n \geq 2$ se obtiene a partir del siguiente sistema matricial:*

$$\begin{pmatrix} T(T_{2,n}) \\ T(G_3^{n-1}) \end{pmatrix} = M^{n-2} \begin{pmatrix} T(T_{2,2}) \\ T(G_3^1) \end{pmatrix}$$

El método desarrollado nos proporciona un resultado equivalente a los Teoremas 2.2 y 2.4 para la malla triangular.

Teorema 2.6. *Fijado $m \in \mathbb{N}$, el procedimiento para la evaluación del polinomio de Tutte de la malla triangular $T_{m,n}$ en un punto cualquiera del plano (a, b) tiene complejidad computacional $O(log(n))$.*

Para concluir este capítulo, queremos resaltar que aunque nuestros cálculos se han centrado básicamente en las tres únicas teselaciones del plano mediante polígonos regulares cuyas baldosas son todas del mismo tipo y tamaño y tal que toda arista de la teselación pertenece exactamente a dos baldosas [21], es decir, en las mallas cuadrada, hexagonal y triangular; el algoritmo que hemos desarrollado permite en un tiempo razonable el cálculo del polinomio de Tutte asociado a fragmentos de teselaciones planas realizadas con combinaciones de cuadrados, triángulos y hexágonos. En el apéndice A de esta memoria explicamos el uso tanto de los programas de codificación de teselaciones como de los programas de cálculo de su polinomio de Tutte asociado.

El sistema empleado para la codificación de los fragmentos de teselaciones, realizadas con combinaciones de cuadrados, triángulos y hexágonos, es análogo que el empleado en las mallas cuadrada, hexagonal y triangular. Esta basado en los conceptos de grafo base, banda y capa. Para calcular su polinomio de Tutte asociado analizamos el efecto de la regla de contracción y borrado al considerar dos copias del grafo base, etiquetando los vértices numéricamente de izquierda a derecha y de abajo a arriba y utilizamos de forma análoga los conceptos de vértice mayor del grafo y de arista mayor. Aplicando este algoritmo podemos enunciar el siguiente resultado.

Teorema 2.7. *Fijado $m \in \mathbb{N}$, el procedimiento para la evaluación del polinomio de Tutte de un fragmento de dimensión $m \times n$ de una teselación del plano realizada con combinaciones de cuadrados, triángulos y hexágonos en un punto cualquiera del plano (a, b) tiene complejidad computacional $O(log(n))$.*

Por último, queremos comentar que con nuestra capacidad informática hemos podido calcular en cuestión de horas polinomios de Tutte de fragmentos de teselaciones de dimensión $12 \times n$. El poder obtener el polinomio de Tutte asociado a fragmentos mayores depende únicamente de la capacidad del ordenador y de la posibilidad de trabajar con ordenadores en paralelo.

CAPÍTULO 3

Aplicaciones

El algoritmo desarrollado en el capítulo anterior para la aplicación de la fórmula recursiva del polinomio de Tutte proporciona resultados muy importantes en diversos ámbitos de las matemáticas. En este capítulo estudiamos dos aplicaciones del algoritmo en problemas de enumeración en Teoría de Grafos: el cálculo de orientaciones acíclicas y el cálculo de números de Whitney; y dos aplicaciones del algoritmo en problemas de enumeración en Geometría: el cálculo del número de caras de un arreglo de hiperplanos y el cálculo del número de vértices de un zonotopo. Además desarrollamos un método alternativo para el cálculo de las orientaciones acíclicas de la malla cuadrada y mejoramos la acotación del límite asintótico del número de orientaciones acíclicas de la malla $L_{n,n}$ establecida en 2003 por Calkin, Merino, Noble y Noy [9].

3.1. Introducción

Una característica fundamental del polinomio de Tutte es que contiene gran información acerca del grafo al cual está asociado. Evaluándolo en distintos puntos del plano obtenemos diversos invariantes del grafo, tales como, el número de árboles recubridores, el número de bosques, etc.. Esta propiedad del polinomio de Tutte permite que podamos aplicar el algoritmo desarrollado en el capítulo anterior a diversos problemas clásicos de enumeración de Teoría de Grafos y Geometría. Queremos resaltar que los programas **poltutte**, **poltutte2** y **poltutte3** suponen un avance muy importante en la búsqueda de métodos de cálculo de los invariantes numéricos contenidos en el polinomio de Tutte. En particular, en la

Sección 2, estudiamos la evaluación del polinomio de Tutte asociado a fragmentos de teselaciones del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos tres tipos de polígonos regulares en el punto $(2, 0)$, es decir, las orientaciones acíclicas de dichos grafos. Los desarrollos realizados en la Sección 2 están centrados fundamentalmente en la malla cuadrada pero no hay que olvidar que dichos desarrollos se pueden reproducir para cualquiera de las teselaciones planas estudiadas en el capítulo anterior. Además, en el caso de la malla cuadrada introducimos un método que denominamos método de la matriz de compatibilidad que también proporciona resultados sobre orientaciones acíclicas.

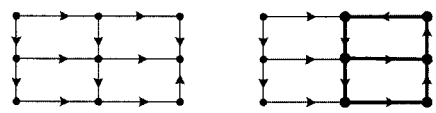
En la Sección 3, relacionamos dos problemas clásicos de enumeración de Geometría con las orientaciones acíclicas de grafos y por tanto con el polinomio de Tutte. El algoritmo desarrollado en el Capítulo 2, no proporciona métodos de cálculo del número de celdas en que arreglos de hiperplanos asociados a fragmentos de teselaciones del plano dividen a espacios euclídeos de altas dimensiones. Además también obtenemos métodos de cálculo del número de vértices de zonotopos asociados a dichos fragmentos de teselaciones. Recordemos que los zonotopos son estructuras que se utilizan para modelizar el problema de la mayor rentabilidad en transmisión de datos en el ámbito de las comunicaciones [25]. Los vértices del zonotopo representan los receptores y los segmentos de línea que unen los vértices, los canales de comunicación.

Finalmente, estudiamos los coeficientes del polinomio cromático asociado a un grafo, también conocidos por números de Whitney de primera clase. Obtenemos métodos de cálculo de dichos números y aplicamos los resultados obtenidos al cálculo de las orientaciones acíclicas con una única fuente.

3.2. Orientaciones Acíclicas

Un *digrafo o grafo orientado*, $D = (V, A)$, es un grafo cuyo conjunto de aristas A es un subconjunto de $V \times V$, es decir, establecemos un sentido de recorrido en las aristas del grafo. Dado un grafo G podemos obtener una orientación del grafo estableciendo un orden entre los extremos de cada una de las aristas, luego el conjunto de aristas de un grafo G admite $2^{|A|}$ orientaciones distintas. Estas orientaciones se pueden clasificar en dos tipos: orientaciones cíclicas y acíclicas. Esta sección está dedicada al estudio de diversos aspectos de las orientaciones acíclicas.

Una orientación de un grafo G es *acíclica* si no contiene ciclos (Figura 3.1a). Más explícitamente, en el digrafo obtenido al fijar una orientación en G nunca podemos partir de un vértice y llegar al mismo vértice siguiendo el sentido de recorrido de las aristas. En caso contrario la orientación se denomina *cíclica* (Figura 3.1b). Al número total de orientaciones acíclicas de un grafo G lo denotamos $\alpha(G)$.



a) *b)*

Figura 3.1: a) Orientación acíclica de la malla cuadrada $L_{3,3}$ b) Orientación cíclica de la malla cuadrada $L_{3,3}$.

Contar el número de orientaciones acíclicas de un grafo es un problema realmente complejo. En particular, el caso de la malla cuadrada $L_{m,n}$ está siendo muy estudiado. En los últimos años, el problema se ha centrado en dar acotaciones del límite asintótico $\lim_{n\to+\infty} \alpha(L_{n,n})^{\frac{1}{n^2}}$. En 1999, Merino y Welsh [30] estudiaron diversos problemas de enumeración en la malla cuadrada $L_{n,n}$. Establecen en 3'7092.. la cota superior del límite asintótico $\lim_{n\to+\infty} \alpha(L_{n,n})^{\frac{1}{n^2}}$. En 2003, esta cota fue mejorada por Calkin, Merino, Noble y Noy [9] estableciéndola en 3'56 . El sistema empleado para la obtención de esta cota se basa fundamentalmente en un algoritmo que permite calcular el polinomio de Tutte asociado a la malla cuadrada. Este algoritmo desarrollado por Calkin, Merino, Noble y Noy permite el cálculo del polinomio de Tutte de un fragmento de malla cuadrada de dimensión máxima 7×7. Queremos, por tanto, resaltar que el algoritmo desarrollado en el Capítulo 2 de esta memoria nos permite, en cuestión de horas, el cálculo del polinomio de Tutte de fragmentos de teselaciones, entre ellos la malla cuadrada, de dimensión $12 \times n$. Con los cálculos realizados con los programas **poltutte**, **poltutte2** y **poltutte3** podemos establecer la cota de las orientaciones acíclicas en 3'49 [18].

En esta memoria presentamos distintos métodos para calcular el número de orientaciones acíclicas de $L_{m,n}$ para determinados valores de m y n arbitrario.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

En 1973, Stanley [34] relaciona las orientaciones acíclicas de un grafo con el polinomio cromático asociado a dicho grafo:

$$\alpha(G) = |\chi(G, -1)|$$

Partiendo de las siguientes fórmulas obtenidas por Marín, Márquez y Revuelta[28] del polinomio cromático asociado a la cuadrícula plana:

$$\begin{aligned}
\chi(L_{2,n}, \lambda) &= \lambda(\lambda - 1)(3 - 3\lambda + \lambda^2)^{n-1} && n \geq 1 \\
\chi(L_{3,n}, \lambda) &= (-10 + 11\lambda - 5\lambda^2 + \lambda^3)\chi(L_{3,n-1}, \lambda) \\
&\quad +(-11 + 24\lambda - 19\lambda^2 + 7\lambda^3 - \lambda^4)\chi(L_{3,n-2}, \lambda) && n \geq 3
\end{aligned}$$

obtenemos que:

$$\begin{aligned}
\alpha(L_{2,n}) &= 2 \cdot 7^{n-1} && n \geq 1 \\
\alpha(L_{3,n}) &= |27\chi(L_{3,n-1}, -1) + 62\chi(L_{3,n-2}, -1)| && n \geq 3
\end{aligned}$$

siendo $\chi(L_{3,1}, -1) = -4$ y $\chi(L_{3,2}, -1) = 98$.

§ 3.2.1. Método recursivo.— Como ya hemos comentado, $\alpha(G)$ es la evaluación en el punto $(2, 0)$ del polinomio de Tutte asociado a G. Por tanto, para calcular el número de orientaciones acíclicas de $L_{m,n}$ podemos aplicar el algoritmo desarrollado en el capítulo anterior. Además, por el Teorema 2.2 sabemos que fijado m, el calculo de $\alpha(L_{m,n})$ puede realizarse en tiempo logarítmico.

Es evidente, que una mejora del algoritmo desarrollado en el Capítulo 2 es reducir la dimensión de las matrices obtenidas, es decir, disminuir el número de grafos generadores de $L_{m,n}$. En esta sección nos planteamos esta reducción como objetivo en el cálculo de $\alpha(L_{m,n})$ para ciertos valores de m y n arbitrario. Para ello no vamos a utilizar la definición recursiva del polinomio de Tutte, sino una definición recursiva para las orientaciones acíclicas.

En 1912, Birkhoff [4] proporciona una definición recursiva del polinomio cromático asociado a un grafo G, sin hojas ni lazos, basada en la contracción y borrado de aristas. Esta definición es la siguiente.

Sea $G = (V, A)$ un grafo y $\chi(G, \lambda)$ su polinomio cromático asociado.

(1) Si G tiene n vértices y no tiene aristas entonces $\chi(G, \lambda) = \lambda^n$.

(2) Si G es la unión de dos grafos, H y K, cuya intersección es un grafo completo entonces:

$$\chi(G, \lambda) = \frac{\chi(H, \lambda)\chi(K, \lambda)}{\chi(H \cap K, \lambda)}$$

(3) $\chi(G,\lambda) = \chi(G'_e,\lambda) - \chi(G''_e,\lambda)$ siendo G'_e el grafo resultante de borrar la arista e de G y G''_e el grafo resultante de contraer la arista e de G.

Mediante el Teorema de Stanley [34] obtenemos una definición similar para calcular el número de orientaciones acíclicas de un grafo sin vértices aislados:

- $\alpha(G) = 2$ si $|A(G)| = 1$.
- $\alpha(G) = \alpha(G'_e) + \alpha(G''_e)$.
- $\alpha(G) = \alpha(G'_e)$ si e es una arista doble.
- $\alpha(G) = 2\alpha(G'_e)$ si e es una hoja.
- $\alpha(G) = 0$ si G tiene algún lazo.

Ambas definiciones no dependen de la arista e elegida.

El poder reducir la dimensión de las matrices obtenidas en el algoritmo desarrollado en el capítulo anterior se debe a la condición sobre las aristas dobles de la definición recursiva. Pensemos, por ejemplo, en el caso $L_{3,n}$ desarrollado en el Capítulo 2. En las Figuras 2.10, 2.11 y 2.12 se muestra que $L_{3,n}$ tiene cuatro grafos generadores pero desde el punto de vista de las orientaciones acíclicas los grafos mostrados en los apartados *a)* y *b)* de la Figura 3.2 son isomorfos. Por tanto, utilizando la definición recursiva de las orientaciones acíclicas obtenemos dos generadores.

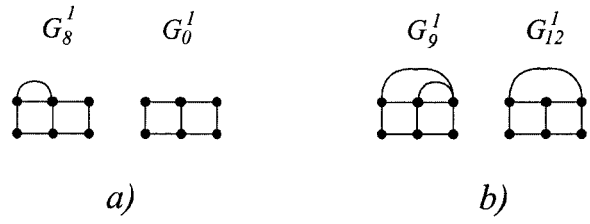


Figura 3.2: Los grafos de *a)* y *b)* son isomorfos

A continuación, aplicamos la definición recursiva de las orientaciones acíclicas a la malla cuadrada. Seguimos el mismo criterio del algoritmo desarrollado en el capítulo anterior para la definición recursiva del polinomio de Tutte: la arista a borrar y contraer es la arista mayor del vértice mayor. En las Figuras 3.3, 3.4 y 3.5 mostramos tres diagramas de descomposición de $L_{4,n}$. Los grafos generadores son $L_{4,2}$, G_1^1, G_2^1 y G_3^1. Para entender los diagramas hay que tener en cuenta que cada

vez que eliminemos una hoja hay que multiplicar por 2 y que las aristas dobles se consideran simples. Hemos seguido la misma notación que en los esquemas del capítulo anterior, así $b, 3$ significa que para realizar el paso correspondiente hay que borrar una arista y eliminar 3 hojas.

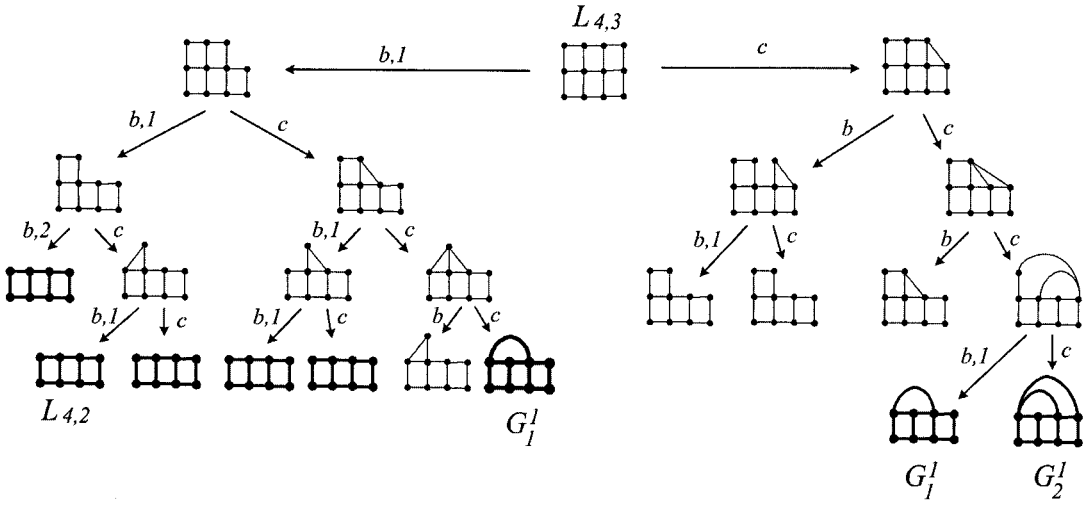


Figura 3.3: Primer diagrama de descomposición de $L_{4,n}$

Aplicando la definición recursiva de las orientaciones acíclicas obtenemos los siguientes resultados (aunque no hay que olvidar que los programas *poltutte* y *poltutte2* permiten obtener $\alpha(L_{m,n})$ para $n > 4$).

Lema 3.1. *El número de orientaciones acíclicas de la malla cuadrada $L_{2,n}$ con $n \geq 2$ es $2 \cdot 7^{n-1}$.*

Lema 3.2. *El número de orientaciones acíclicas de la malla cuadrada $L_{3,n}$ con $n \geq 2$ se obtiene a partir del siguiente sistema matricial:*

$$\begin{pmatrix} \alpha(L_{3,n}) \\ \alpha(G_{12}^n) \end{pmatrix} = \begin{pmatrix} 23 & 1 \\ 30 & 4 \end{pmatrix} \begin{pmatrix} \alpha(L_{3,n-1}) \\ \alpha(G_{12}^{n-1}) \end{pmatrix} = \left[\begin{pmatrix} 23 & 1 \\ 30 & 4 \end{pmatrix} \right]^{n-2} \begin{pmatrix} \alpha(L_{3,2}) \\ \alpha(G_{12}^1) \end{pmatrix}$$

siendo G_{12}^1 el grafo de la Figura 3.2, $\alpha(L_{3,2}) = 98$ y $\alpha(G_{12}^1) = 144$.

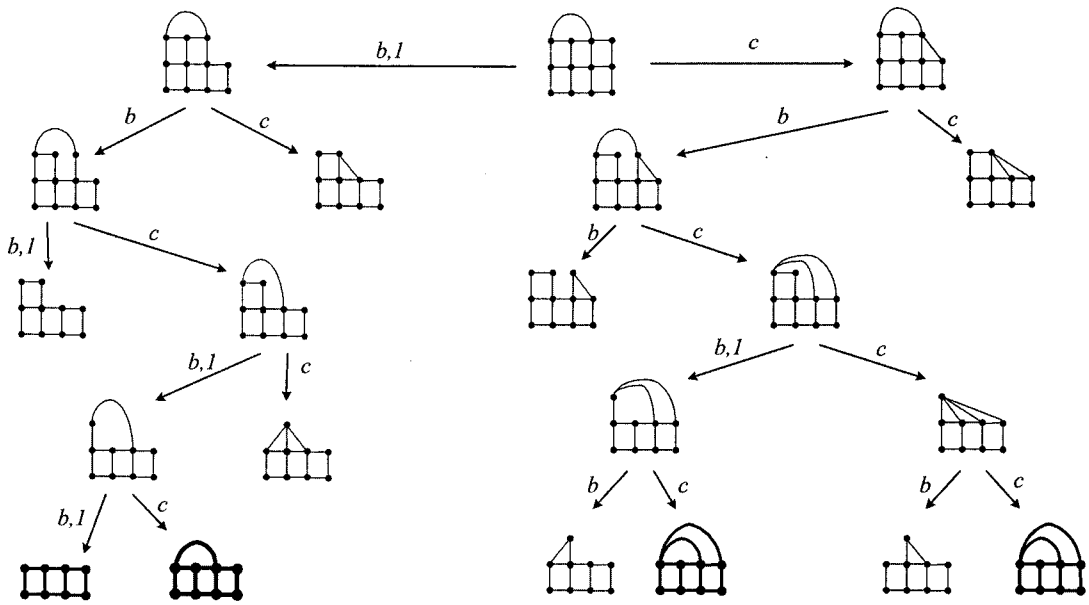


Figura 3.4: Segundo diagrama de descomposición de $L_{4,n}$

Lema 3.3. *El número de orientaciones acíclicas de la malla cuadrada $L_{4,n}$ con $n \geq 2$ se obtiene a partir del siguiente sistema matricial:*

$$\begin{pmatrix} \alpha(L_{4,n}) \\ \alpha(G_1^n) \\ \alpha(G_2^n) \\ \alpha(G_3^n) \end{pmatrix} = \begin{pmatrix} 65 & 4 & 1 & 0 \\ 80 & 10 & 4 & 0 \\ 99 & 10 & 6 & 9 \\ 102 & 6 & 6 & 7 \end{pmatrix} \begin{pmatrix} \alpha(L_{4,n-1}) \\ \alpha(G_1^{n-1}) \\ \alpha(G_2^{n-1}) \\ \alpha(G_3^{n-1}) \end{pmatrix}$$

$$= \left[\begin{pmatrix} 65 & 4 & 1 & 0 \\ 80 & 10 & 4 & 0 \\ 99 & 10 & 6 & 9 \\ 102 & 6 & 6 & 7 \end{pmatrix} \right]^{n-2} \begin{pmatrix} \alpha(L_{4,2}) \\ \alpha(G_1^1) \\ \alpha(G_2^1) \\ \alpha(G_3^1) \end{pmatrix}$$

y las condiciones iniciales son:

$$\begin{pmatrix} \alpha(L_{4,2}) \\ \alpha(G_1^1) \\ \alpha(G_2^1) \\ \alpha(G_3^1) \end{pmatrix} = \begin{pmatrix} 65 & 4 & 1 & 0 \\ 80 & 10 & 4 & 0 \\ 99 & 10 & 6 & 9 \\ 102 & 6 & 6 & 7 \end{pmatrix} \begin{pmatrix} 6 \\ 10 \\ 16 \\ 12 \end{pmatrix}$$

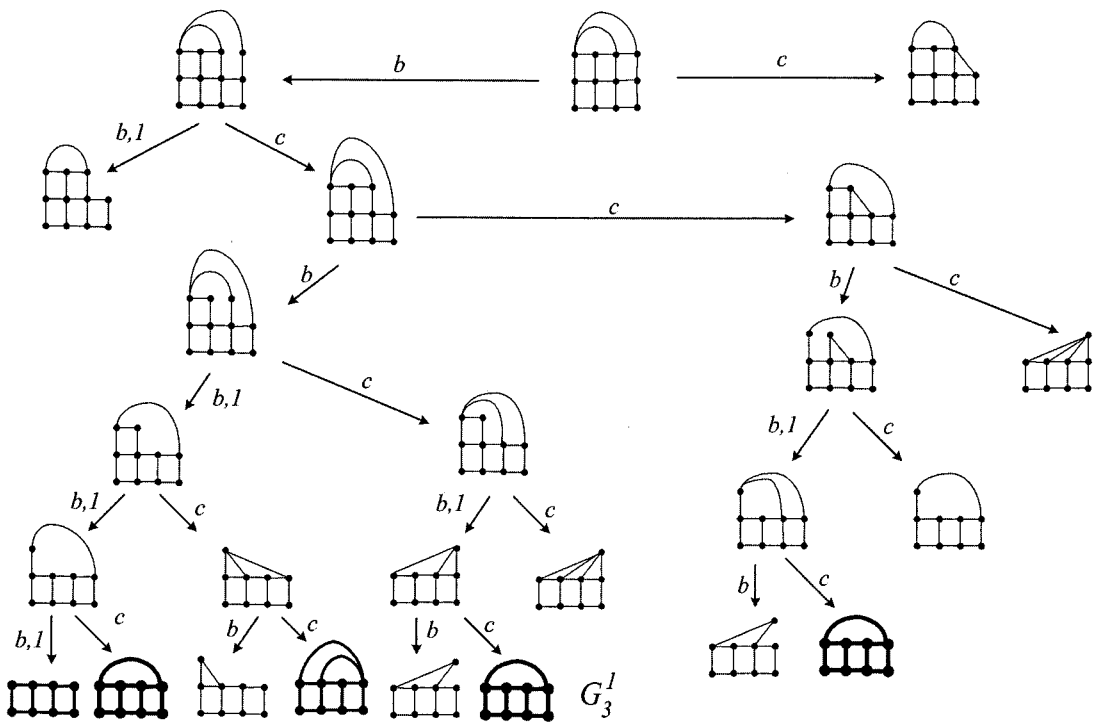


Figura 3.5: Tercer diagrama de descomposición de $L_{4,n}$

Si seguimos aumentando los valores de n obtenemos que el número de grafos generadores de $L_{m,n}$ en cada etapa es $m-1+\binom{m-2}{2}$ si $m \neq 3$ y $m-1$ si $m=3$, ya que para cualquier valor fijado de n los grafos generadores se obtienen considerando todas los posibles formas de añadir una arista $\{(0,1),(x,1)\}$ con $x \neq 1$ al grafo base $L_{2,n}$ y todas las combinaciones dos a dos de dichas aristas. Mostramos un ejemplo en la Figura 3.6.

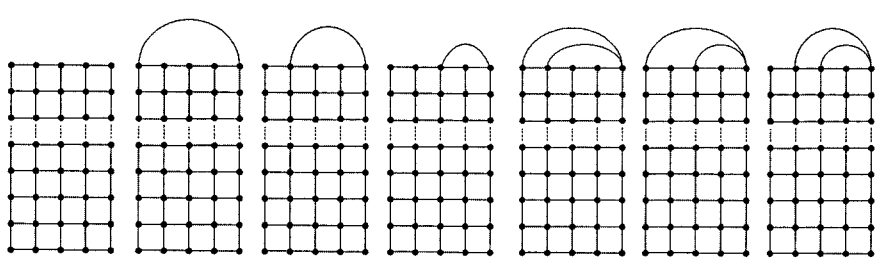


Figura 3.6: Grafos generadores de $L_{5,n}$ en una etapa j de la descomposición.

Por último, nos parece importante resaltar que con la eliminación de aristas dobles en la definición recursiva de las orientaciones acíclicas disminuimos de forma considerable el número de grafos generadores de $L_{m,n}$. De hecho, el número de grafos generadores a obtener si utilizamos la definición recursiva del polinomio de Tutte y evaluamos posteriormente en el punto $(2,0)$ es $m+\binom{m-1}{2}$ luego la aplicación de la definición recursiva de las orientaciones acíclicas supone obtener $m-1$ grafos generadores menos. Por ejemplo, para $L_{3,n}$ disminuimos la dimensión de la matriz a la mitad.

Por otra parte, queremos hacer constar que el desarrollo efectuado para el cálculo de las orientaciones acíclicas de la malla cuadrada mediante el método recursivo puede realizarse de igual forma para cualquiera de las teselaciones planas estudiadas en el capítulo anterior.

§ 3.2.2. Matriz de Compatibilidad.— Otra forma de afrontar el problema de las orientaciones acíclicas en la malla cuadrada es tratarlo como un problema matricial, dividiendo el grafo en subgrafos y estudiando lo que ocurre al unirlos de cierta forma. Esto es lo que llamamos el método de la matriz de compatibilidad y que desarrollamos a continuación.

Un *peine de longitud m* se define como el grafo $\varphi = (V, A)$ (Figura 3.7) siendo:

$$\begin{aligned} V &= \{ v_1, \ldots, v_m, v'_1, \ldots, v'_m \} \\ A &= \{ \{v_j, v_{j+1}\}, \{v_i, v'_i\} \mid 1 \leq j \leq m-1 \,,\ 1 \leq i \leq m \} \end{aligned}$$

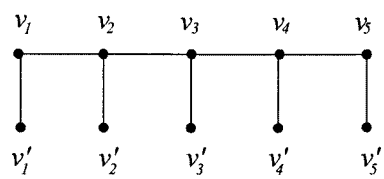


Figura 3.7: Peine de longitud 5

Es fácil comprobar que un peine tiene 2^{2m-1} orientaciones acíclicas distintas.

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

Definición 3.1. Llamamos *peines orientados de longitud* m a los grafos orientados que resultan de considerar cada una de las orientaciones acíclicas de φ.

Fijado $m \geq 2$ denotamos Ω_m al conjunto de peines orientados de longitud m. En este conjunto podemos definir la siguiente operación entre dos peines de longitud m, $\varphi_1 = (V_1, A_1)$ y $\varphi_2 = (V_2, A_2)$.

Definición 3.2. Llamamos *pegada de* φ_1 *con* φ_2 y lo denotamos $\varphi_1\varphi_2$ al grafo que se obtiene al identificar los vértices v_i de φ_1 con los vértices u'_i de φ_2 siendo $V_1 = \{ v_1, \ldots, v_m, v'_1, \ldots, v'_m \}$ y $V_2 = \{ u_1, \ldots, u_m, u'_1, \ldots, u'_m \}$ los conjuntos de vértices de φ_1 y φ_2 respectivamente.

Análogamente, podemos definir la pegada de k peines φ_1, φ_2, $\ldots$,$\varphi_k \in \Omega_m$ con $k \geq 3$ como el grafo resultante de pegar φ_1 con φ_2, a este pegarle φ_3 y así sucesivamente.

Sean $\varphi, \psi \in \Omega_m$. La operación de pegar cumple las siguientes propiedades:

- No es una operación conmutativa, es decir, $\varphi\psi \neq \psi\varphi$.
- $\varphi\varphi$ no contiene ciclos.
- Es una operación asociativa.

Con las definiciones dadas es fácil comprobar que para cada orientación de $L_{m,n}$ existen n peines orientados de longitud m cuya pegada da lugar a la orientación dada en la malla cuadrada. Recíprocamente, dados $\psi_1, \ldots, \psi_n$ peines orientados de longitud m, $\psi_1\psi_2 \ldots \psi_n$ nos proporciona una única orientación de $L_{m,n}$.

Definición 3.3. Sean φ_1, φ_2, $\ldots$, φ_k peines de la misma longitud. La $k-$upla $(\varphi_1, \ldots, \varphi_k)$ se dice *compatible* si el grafo orientado $\varphi_1\varphi_2 \ldots \varphi_k$ no contiene ciclos.

Definición 3.4. Fijados $m, n \geq 2$ se define una *matriz de compatibilidad* $n-$*ésima asociada a* $L_{m,n}$, $A = (a_{i_1 \ldots i_n})$, de la siguiente forma:

$$a_{i_1 \ldots i_n} = \begin{cases} 1 & \text{si } (\varphi_{i_1}, \ldots, \varphi_{i_n}) \text{ es compatible.} \\ 0 & \text{si } (\varphi_{i_1}, \ldots, \varphi_{i_n}) \text{ no es compatible.} \end{cases} \quad \forall \varphi_j \in \Omega_m$$

$L_{m,n}$ tiene muchas matrices de compatibilidad asociadas distintas ya que, si tomamos un peine de longitud m y consideramos todas sus orientaciones tenemos que $|\Omega_m| = 2^{2m-1}$ y en función del orden en que coloquemos los elementos de este conjunto a la hora de pegar obtendremos distintas matrices. En cualquier caso, desde el punto de vista de las orientaciones, todas estas matrices nos ofrecen la misma información. Como ejemplo mostramos una de las matrices de compatibilidad 2−ésima asociada a $L_{2,2}$:

$$A_2 = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{18} \\ a_{21} & a_{22} & \cdots & a_{28} \\ a_{31} & a_{32} & \cdots & a_{38} \\ \vdots & \vdots & \vdots & \vdots \\ a_{81} & a_{82} & \cdots & a_{88} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

donde a_{ij} con $1 \leq i, j \leq 8$ es 1 si $\varphi_i\varphi_j$ no contiene ciclos y 0 en caso contrario.

Lema 3.4. *El número de orientaciones acíclicas de la malla cuadrada $L_{m,n}$ se obtiene a partir del siguiente sistema matricial:*

$$\alpha(L_{m,n}) = \frac{1}{2^m}\left(A\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right)\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}^t \qquad m, n \in \mathbb{N}$$

siendo A una matriz de compatibilidad $n-$ésima cualquiera asociada a $L_{m,n}$.

Demostración. Sea A una matriz de compatibilidad $n-$ésima asociada a $L_{m,n}$. Cada elemento de A nos proporciona, o bien, una orientación acíclica (representada por un 1), o una orientación cíclica (representada por un 0) ; por tanto $(A(1,1,\ldots,1)^t)(1,\ldots,1)$ es el número de orientaciones acíclicas del grafo de la Figura 3.8 y dividiendo por 2^m obtenemos el resultado deseado. □

Lema 3.5. *Una orientación de $L_{2,n}$ es acíclica si y sólo si los n peines orientados asociados a dicha orientación, $\varphi_1, \ldots, \varphi_n$, verifican que $(\varphi_i, \varphi_{i+1})$ son pares compatibles para todo $i \in [1, n-1]$.*

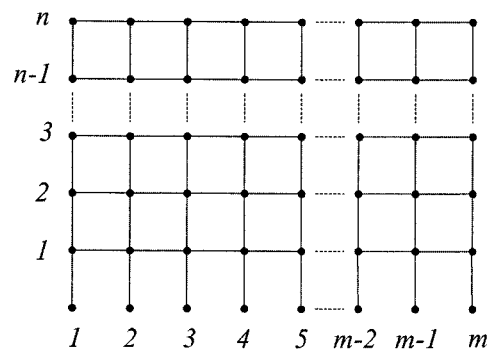


Figura 3.8: Pegada de n peines de longitud m

Demostración. Por definición, es evidente que si una orientación es acíclica, los pares $(\varphi_i, \varphi_{i+1})$ con $1 \leq i \leq n-1$ son compatibles.

Supongamos que dichos pares son compatibles y que la orientación asociada no es acíclica. Si $(\varphi_i, \varphi_{i+1})$ es compatible para todo i, entonces $\varphi_i\varphi_{i+1}$ no contiene ningún ciclo. Por tanto, la única posibilidad de que en la orientación dada en $L_{2,n}$ haya un ciclo es que existan $k, l \in \mathbb{N}$ tales que $\varphi_k\varphi_{k+1}\ldots\varphi_l$ con $1 \leq k \leq n-2$ y $k+2 \leq l \leq n$ contenga algún ciclo; pero es fácil comprobar que al estar trabajando en $L_{2,n}$ esta última afirmación implica que existe $t \in \mathbb{N}$, $k \leq t \leq l-1$ tal que $\varphi_t\varphi_{t+1}$ contiene algún ciclo, lo cual no es posible por hipótesis. Se produce una contradicción del hecho de suponer que la orientación no es acíclica. □

Como consecuencia del lema anterior obtenemos el siguiente resultado.

Lema 3.6. *El número de orientaciones acíclicas de la malla cuadrada $L_{2,n}$ se obtiene a partir del siguiente sistema matricial:*

$$\alpha(L_{2,n}) = \frac{1}{4}\left(A^{n-1}\begin{pmatrix}1\\ \vdots\\ 1\end{pmatrix}\right)\begin{pmatrix}1\\ \vdots\\ 1\end{pmatrix}^t \qquad n \geq 2$$

siendo A cualquier matriz de compatibilidad $2-$ésima asociada a $L_{2,2}$.

3.3. Arreglos de Hiperplanos y Zonotopos

El algoritmo desarrollado en el capítulo anterior para la aplicación de la fórmula recursiva del polinomio de Tutte y el estudio realizado sobre orientaciones acíclicas en la Sección 2, tienen aplicación en otras áreas de las matemáticas. En esta sección vamos a estudiar dos de estas aplicaciones: la enumeración de las caras de un arreglo de hiperplanos y la enumeración de los vértices de un zonotopo. Además, vamos a comprobar el avance que suponen los programas **poltutte**, **poltutte2** y **poltutte3** en estos problemas.

§ 3.3.1. Arreglos de Hiperplanos.— Un *arreglo de hiperplanos* es una colección de n hiperplanos en el espacio euclídeo $d-$dimensional $\mathbb{R}^d$, junto con la descomposición celular de $\mathbb{R}^d$ asociada (Figura 3.9).

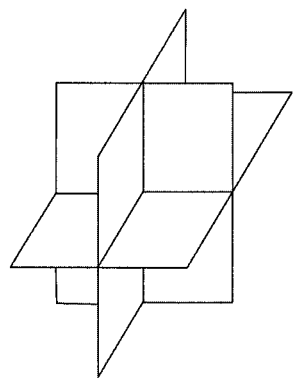


Figura 3.9: Arreglo central formado por 3 hiperplanos en $\mathbb{R}^3$

A una celda $k-$dimensional de un arreglo de hiperplanos se la denomina $k-$*cara*. Como el conjunto de todas las caras forman una partición del espacio, un arreglo es también llamado una *partición del espacio por hiperplanos*. Llamamos *arreglo central de hiperplanos* a un arreglo en el cual todos los hiperplanos se cortan en un punto (Figura 3.9).

El centro de interés en la teoría de arreglos es el estudio de las propiedades combinatorias de las caras, en particular, cuestiones de enumeración como contar el número de $d-$caras de un determinado arreglo de hiperplanos. El más conocido de estos problemas es contar el número de regiones de un arreglo de líneas. La

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

solución se basa en contar las regiones en términos del número de vértices de grado k o del número de líneas conteniendo k vértices [41]. Para dimensión mayor que dos no hay una solución similar a esta. En altas dimensiones, hay únicamente dos soluciones generales para el problema de la enumeración de las caras de un arreglo de hiperplanos: las fórmulas de Roberts para dimensión tres y la enumeración de Winders para arreglos centrales de hiperplanos en $\mathbb{R}^d$ (basada en los llamados conjuntos de corte) [41]. No queremos profundizar en estos problemas ya que no son el objetivo de esta memoria; pero nos parece interesante resaltar que muchas demostraciones en la teoría de arreglos de hiperplanos, entre ellas la demostración del teorema de Winders, se basan en el concepto de $TG-$invariante, explicado en la introducción de esta primera parte de la memoria y que tiene una interpretación como una evaluación del polinomio de Tutte [38].

A todo grafo $G = (V, A)$ le podemos asociar el siguiente arreglo central de hiperplanos:

$$H(G) = \{h_{ij} \ ; \ e_{ij} \in A\} \quad \text{con} \quad h_{ij} = \{x \in \mathbb{R}^{|V|} \ ; \ x_i = x_j\}$$

Debido al siguiente resultado demostrado por Greene y Zaslavsky en 1983 [20] que relaciona el número de orientaciones acíclicas de un grafo con el número de celdas del arreglo de hiperplanos asociado a dicho grafo, el algoritmo desarrollado en el Capítulo 2 se convierte en una herramienta muy potente para el cálculo del número de caras de un arreglo de hiperplanos asociado a toda teselación del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos polígonos regulares.

Teorema 3.7. *[20] El número de orientaciones acíclicas de G es igual al número de celdas $|V|-$dimensionales en que $H(G)$ divide al espacio $\mathbb{R}^{|V|}$.*

Así, teniendo en cuenta el algoritmo desarrollado en el capítulo anterior, el estudio realizado sobre orientaciones acíclicas y los Teoremas 2.7 y 3.7, obtenemos diversos resultados sobre enumeración de caras de arreglos de hiperplanos asociados a teselaciones del plano.

Teorema 3.8. *Sea Γ_{mn} un fragmento de dimensión $m \times n$ de una teselación del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos tres tipos de polígonos regulares. Fijado $m \in \mathbb{N}$, la enumeración de las caras $|V(\Gamma_{mn})|$ dimensionales de un arreglo de hiperplanos en $\mathbb{R}^{|V(\Gamma_{mn})|}$ asociado a Γ_{mn} tiene complejidad computacional $O(log(n))$.*

Lema 3.9.

- *El número de celdas $2n-$dimensionales en que $H(L_{2,n})$ divide al espacio $\mathbb{R}^{2n}$ es $2 \cdot 7^{n-1}$ $n \geq 1$.*
- *Denotando $f(G)$ al número de celdas $|V|-$dimensionales en que $H(G)$ divide al espacio $\mathbb{R}^{|V|}$ obtenemos los siguientes sistemas matriciales:*

$$\begin{pmatrix} f(L_{3,n}) \\ f(G_{12}^n) \end{pmatrix} = \left[\begin{pmatrix} 23 & 1 \\ 30 & 4 \end{pmatrix}\right]^{n-2} \begin{pmatrix} f(L_{3,2}) \\ f(G_{12}^1) \end{pmatrix}$$

$$\begin{pmatrix} f(L_{4,n}) \\ f(G_1^n) \\ f(G_2^n) \\ f(G_3^n) \end{pmatrix} = \left[\begin{pmatrix} 65 & 4 & 1 & 0 \\ 80 & 10 & 4 & 0 \\ 99 & 10 & 6 & 9 \\ 102 & 6 & 6 & 7 \end{pmatrix}\right]^{n-2} \begin{pmatrix} f(L_{4,2}) \\ f(G_1^1) \\ f(G_2^1) \\ f(G_3^1) \end{pmatrix}$$

Queremos resaltar que los programas **poltutte**, **poltutte2** y **poltutte3** representan un método de cálculo muy efectivo del número de caras de un arreglo de hiperplanos asociado a fragmentos de teselaciones del plano de dimensiones considerables.

§ 3.3.2. Zonotopos.— Un *zonotopo* es el vector suma de un número finito de segmentos de línea cerrados en el espacio real $\mathbb{R}^d$. Más explícitamente, llamamos zonotopo generado por un conjunto de vectores $E = \{e^1, \ldots, e^k\} \subseteq \mathbb{Q}^d$ a la suma de Minkowsky:

$$Z = \left\{ \sum_{i=1}^{k} \lambda_i e^i \mid 0 \leq \lambda_i \leq 1 \right\}$$

Mostramos un ejemplo en la Figura 3.10.

Cada cara de un zonotopo es también un zonotopo. El conjunto de todas las caras que tengan un segmento de línea como un sumando común se llama *banda* del zonotopo. Si el zonotopo tiene menos de cuatro generadores, entonces es un segmento de línea (1 generador), un paralelogramo (2 generadores) o un paralelepípedo (3 generadores). Para más de tres generadores la geometría es mucho más compleja.

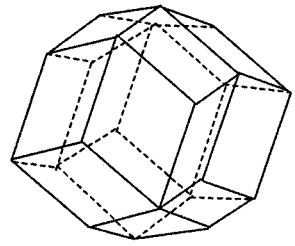


Figura 3.10: Zonotopo en dimensión 3 con 5 generadores

En la Sección 1, hemos comentado la importancia que están adquiriendo estructuras geométricas como los zonotopos, para modelizar problemas en el ámbito de las comunicaciones. El estudio de los zonotopos se centra fundamentalmente en calcular o acotar el número de vértices que posee. Usando un teorema clásico de geometría discreta se demuestra que los vértices de un zonotopo pueden ser calculados en tiempo polinomial [25]. Además, existen diversas acotaciones de este número pero la cota más utilizada a lo largo del tiempo, para un zonotopo $d-$dimensional con k generadores, ha sido $2\sum_{i=0}^{d-1}\binom{k-1}{i}$. En esta sección demostramos que los vértices de zonotopos asociados a fragmentos de teselaciones pueden ser calculados en tiempo logarítmico. Además comprobamos que los programas **poltutte**, **poltutte2** y **poltutte3** representan un método efectivo de cálculo del número de vértices de determinados zonotopos.

En general, a una matriz $A = (a_{ij})$ de dimensión $r \times l$ le podemos asociar el siguiente zonotopo:

$$Z(A) = \left\{ p \in \mathbb{R}^l \quad | \quad p = \sum_{k=1}^{l} \lambda_k a_k, \quad 0 \leq \lambda_k \leq 1 \right\}$$

siendo a_k los vectores columna de la matriz A.

Se define un *zonotopo asociado a un grafo* G como $Z(A)$ siendo A una matriz de adyacencia del grafo. Dependiendo de la etiquetación dada a los vértices de G obtendremos distintas matrices de adyacencia del grafo, y por tanto distintos zonotopos asociados a G; pero todos ellos tienen el mismo número de vértices. En particular, si Γ_{mn} es un fragmento de teselación del plano de dimensión $m \times n$, le podemos asociar el siguiente zonotopo:

$$Z(\Gamma_{mn}) = \left\{ p \in \mathbb{R}^{|V(\Gamma_{mn})|} \quad | \quad p = \sum_{k=1}^{|V(\Gamma_{mn})|} \lambda_k a_k, \quad 0 \leq \lambda_k \leq 1 \right\}$$

En 1997, Bartels, Mount y Welsh establecen la siguiente relación entre el número de orientaciones acíclicas de un grafo y el número de vértices de un zonotopo asociado a dicho grafo.

Teorema 3.10. *[1] El número de orientaciones acíclicas de G coincide con el número de vértices de cualquier zonotopo asociado a G.*

Basándonos en este Teorema y en los Teoremas 2.2, 2.6 y 2.4 podemos enunciar el siguiente resultado.

Teorema 3.11. *Fijado $m \in \mathbb{N}$, el cálculo del número de vértices de cualquier zonotopo asociado a un fragmento de dimensión $m \times n$ de una teselación del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos tres tipos de polígonos regulares tiene complejidad computacional $O(log(n))$.*

El algoritmo desarrollado en el Capítulo 2 y los programas **poltutte**, **poltutte2** y **poltutte3** nos ofrecen la posibilidad de calcular el número de vértices de cualquier zonotopo asociado a fragmentos de teselaciones de grandes dimensiones. Debido a las dimensiones de las matrices obtenidas ponemos como ejemplo el caso de la malla cuadrada.

La matriz de adyacencia de $L_{m,n}$ es $A = (a_{(i,j)(i',j')})$ con $1 \leq i, i' \leq m$ y $1 \leq j, j' \leq n$ siendo:

$$a_{(i,j)(i',j')} = \begin{cases} 1 & \text{si} \quad |i-i'| + |j-j'| = 1 \\ 0 & \text{si} \quad |i-i'| + |j-j'| \neq 1 \end{cases}$$

y su zonotopo asociado es:

$$Z(L_{m,n}) = \left\{ p \in \mathbb{R}^{nm} \quad | \quad p = \sum_{k=1}^{nm} \lambda_k a_k \quad 0 \leq \lambda_k \leq 1 \right\}$$

donde $a_k = (\mu_k^1, \ldots, \mu_k^{nm})$ con $\mu_k^j \in \{0,1\}$, $1 \leq j \leq nm$ son los vectores columna de la matriz A.

Utilizando el estudio realizado sobre orientaciones acíclicas en la Sección 2 obtenemos el siguiente resultado.

Lema 3.12.

- *El número de vértices de cualquier zonotopo asociado a* $L_{2,n}$ *es* $2 \cdot 7^{n-1}$.
- *Si* $\Delta(G)$ *es el número de vértices de cualquier zonotopo asociado a* G *entonces:*

$$\begin{pmatrix} \Delta(L_{3,n}) \\ \Delta(G_{12}^n) \end{pmatrix} = \left[\begin{pmatrix} 23 & 1 \\ 30 & 4 \end{pmatrix} \right]^{n-2} \begin{pmatrix} \Delta(L_{3,2}) \\ \Delta(G_{12}^1) \end{pmatrix}$$

$$\begin{pmatrix} \Delta(L_{4,n}) \\ \Delta(G_1^n) \\ \Delta(G_2^n) \\ \Delta(G_3^n) \end{pmatrix} = \left[\begin{pmatrix} 65 & 4 & 1 & 0 \\ 80 & 10 & 4 & 0 \\ 99 & 10 & 6 & 9 \\ 102 & 6 & 6 & 7 \end{pmatrix} \right]^{n-2} \begin{pmatrix} \Delta(L_{4,2}) \\ \Delta(G_1^1) \\ \Delta(G_2^1) \\ \Delta(G_3^1) \end{pmatrix}$$

3.4. Números de Whitney de Primera Clase

En esta sección estudiamos dos aspectos de los números de Whitney de primera clase: su relación con el polinomio cromático asociado a un grafo y su relación con las orientaciones acíclicas con una única fuente.

Existen diversas definiciones de número de Whitney de primera clase. La que utilizamos en esta memoria es la que relaciona los números de Whitney de primera clase con la Teoría de Grafos. *El número de Whitney de grado* k *de primera clase asociado a un grafo* G, denotado $w_k(G)$, es el coeficiente de grado k del polinomio cromático asociado a G. Por tanto:

$$\chi(G, \lambda) = \sum_{k=1}^{d} w_k(G)\lambda^k$$

Además, Stanley y Zaslavsky [34, 41] demuestran de forma independiente que:

$$\alpha(G) = \sum_{k=1}^{n} w_k^+(G)$$

siendo $w_k^+(G)$ el módulo del número de Whitney de grado k de primera clase asociado a G.

El algoritmo desarrollado en el Capítulo 2 y el estudio realizado sobre orientaciones acíclicas nos ofrece gran información sobre números de Whitney pues no hay que olvidar que el polinomio cromático asociado a un grafo G es $T(G;x,0)$ con $x<0$. Además, los programas **poltutte**, **poltutte2** y **poltutte3** nos permiten calcular los números de Whitney asociados a diversos fragmentos de teselaciones del plano mediante cuadrados, triángulos, hexágonos y combinaciones de ellos. En el caso de la malla cuadrada, $L_{m,n}$, también hemos conseguido formas de cálculo de los números de Whitney de primera clase para ciertos valores fijos de m y n arbitrario, partiendo de las fórmulas del polinomio cromático de $L_{2,n}$ y $L_{3,n}$ obtenidas en [28] por Marín, Márquez y Revuelta.

Lema 3.13. *Los números de Whitney de la malla $L_{2,n}$ se obtienen a partir de la siguiente ecuación en recurrencia:*

$$w_k(L_{2,n}) = \begin{cases} 3w_k(L_{2,n-1}) - 3w_{k-1}(L_{2,n-1}) + w_{k-2}(L_{2,n-1}) & si \quad 0 \le k \le 2n,\ n \ge 2 \\ 0 \quad si \quad k > 2n \ \ ó \ \ k < 0 \end{cases}$$

$$\begin{aligned} w_0(L_{2,1}) &= 0 \\ w_1(L_{2,1}) &= -1 \\ w_2(L_{2,1}) &= 1 \end{aligned}$$

Lema 3.14. *Los números de Whitney de la malla $L_{3,n}$ se obtienen a partir de la siguiente ecuación en recurrencia:*

$$w_k(L_{3,n}) = \begin{cases} -10w_k(L_{3,n-1}) + 11w_{k-1}(L_{3,n-1}) - 5w_{k-2}(L_{3,n-1}) + w_{k-3}(L_{3,n-1}) \\ -11w_k(L_{3,n-2}) + 24w_{k-1}(L_{3,n-2}) - 19w_{k-2}(L_{3,n-2}) + 7w_{k-3}(L_{3,n-2}) \\ -w_{k-4}(L_{3,n-2}) \quad si \quad 0 \le k \le 3n, \quad n \ge 3 \\ \\ 0 \quad si \quad k > 3n \ \ ó \ \ k < 0 \end{cases}$$

$$\begin{aligned} w_0(L_{3,1}) &= 0 & w_2(L_{3,2}) &= 27 \\ w_1(L_{3,1}) &= 1 & w_3(L_{3,2}) &= -33 \\ w_2(L_{3,1}) &= -2 & w_4(L_{3,2}) &= 21 \\ w_3(L_{3,1}) &= 1 & w_5(L_{3,2}) &= -7 \\ w_0(L_{3,2}) &= 0 & w_6(L_{3,2}) &= 1 \\ w_1(L_{3,2}) &= -9 \end{aligned}$$

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

Lema 3.15. *Un ciclo de longitud n, C_n, tiene asociados los siguientes números de Whitney de primera clase:*

$$\begin{aligned} w_1(C_n) &= (-1)^{n-1}(n-1) \\ w_j(C_n) &= (-1)^{n-j}\binom{n}{j} \qquad 2 \leq j \leq n \end{aligned}$$

Demostración. En 1977, Biggs [3] prueba que $\chi(C_n, \lambda) = (\lambda-1)^n + (-1)^n(\lambda-1)$

$$\begin{aligned} \chi(C_n, \lambda) &= \sum_{k=0}^{n} \binom{n}{k} \lambda^k (-1)^{n-k} + (-1)^n(\lambda-1) \\ &= \sum_{k=2}^{n} \binom{n}{k} \lambda^k (-1)^{n-k} + (-1)^{n-1}(n-1)\lambda \end{aligned}$$

□

§ 3.4.1. Propiedades de los números de Whitney.— Una sucesión de números reales, $v_0, v_1, \ldots, v_k$, se dice *unimodal* si existe $j \in \{0, \ldots, k\}$ tal que:

$$v_0 \leq v_1 \leq \ldots \leq v_j \geq v_{j+1} \geq v_{j+2} \geq \ldots \geq v_k$$

Además, dicha sucesión se dice que es *logaritmo cóncava* si $v_i^2 \geq v_{i-1}v_{i+1}$ con $1 \leq i \leq k-1$. Además, toda sucesión logaritmo cóncava de términos positivos es también unimodal.

En 1968, Read [32] conjeturó que el módulo de los coeficientes del polinomio cromático asociado a un grafo forman una sucesión logaritmo cóncava y por tanto unimodal. En esta memoria, obtenemos formas de cálculo de números de Whitney de algunas familias de grafos y los programas informáticos que adjuntamos permiten comprobar que la conjetura de Read es cierta para familias de grafos con un número considerable de vértices. Veamos como ejemplo las tablas obtenidas para las mallas $L_{2,n}$ y $L_{3,n}$.

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	$\dots$
$L_{2,1}$	0	-1	1	0	0	0	0	0	0	0	0	$\dots$
$L_{2,2}$	0	-3	6	-4	1	0	0	0	0	0	$\dots$	
$L_{2,3}$	0	-9	27	-33	21	-7	1	0	0	0	$\dots$	
$L_{2,4}$	0	-27	108	-189	189	-117	43	10	1	0	$\dots$	
$\vdots$												

Tabla 3.1: Números de Whitney de $L_{2,n}$

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	$\dots$
$L_{3,1}$	0	1	-2	1	0	0	0	0	0	0	0	$\dots$
$L_{3,2}$	0	-9	27	-33	21	-7	1	0	0	0	0	$\dots$
$L_{3,3}$	0	79	-323	594	-648	459	-216	66	-12	1	0	$\dots$
$\vdots$												

Tabla 3.2: Números de Whitney de $L_{3,n}$

Además obtenemos valores exactos para algunos números de Whitney:

- $w_{2n}(L_{m,n}) = 1$.
- $w_1(L_{2,n}) = -3^{n-1}$.
- $w_2(L_{2,n}) = n3^{n-1}$.
- $w_{2n-1}(L_{2,n}) = -3n + 2$.
- $w_{3n-1}(L_{3,n}) = -5n + 3$.

Por último, debido a las alternancias de signos de los coeficientes del polinomio cromático se tiene que:

$$w_k^+(L_{2,n}) = \begin{cases} 3w_k^+(L_{2,n-1}) + 3w_{k-1}^+(L_{2,n-1}) + w_{k-2}^+(L_{2,n-1}) & \text{si} \quad 0 \leq k \leq 2n,\ n \geq 2 \\ 0 \quad \text{si} \quad k > 2n \ \text{ó} \ k < 0 \end{cases}$$

ESCUELA TÉCNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

$$w_k^+(L_{3,n}) = \begin{cases} 10w_k^+(L_{3,n-1}) + 11w_{k-1}^+(L_{3,n-1}) + 5w_{k-2}^+(L_{3,n-1}) + w_{k-3}^+(L_{3,n-1}) \\ +11w_k^+(L_{3,n-2}) + 24w_{k-1}^+(L_{3,n-2}) + 19w_{k-2}^+(L_{3,n-2}) \\ +7w_{k-3}^+(L_{3,n-2}) + w_{k-4}^+(L_{3,n-2}) \quad \text{si} \quad 0 \leq k \leq 3n, \quad n \geq 3 \\ \\ 0 \quad \text{si} \quad k > 3n \text{ ó } k < 0 \end{cases}$$

§ 3.4.2. Orientaciones acíclicas de grafos con una única fuente.— Para cualquier vértice $v \in V$ de un digrafo podemos definir dos tipos de valencia, la de entrada $\delta^+(v)$ y la salida $\delta^-(v)$, de la siguiente forma:

$$\begin{aligned} \delta^+(v) &= |\{u \in V : (v,u) \in A\}| \\ \delta^-(v) &= |\{w \in V : (w,v) \in A\}| \end{aligned}$$

Un vértice s de un grafo orientado se dice que es una *fuente* si todos los arcos que contienen a s están dirigidos desde s, es decir, $\delta(s) = \delta^+(s)$.

En 1977, Green relaciona el número de orientaciones acíclicas con una única fuente de un grafo con la derivada del polinomio cromático asociado a dicho grafo.

Teorema 3.16. *[19] El número de orientaciones acíclicas de un grafo G con una única fuente, denotado $\alpha_s(G)$, es $|\chi'(G,0)|$.*

Como $\chi(G,\lambda) = \sum_{k=1}^{d} w_k(G)\lambda^k$ entonces $\chi'(G,0) = w_1(G)$ y por tanto obtenemos un método de cálculo muy potente del número de orientaciones acíclicas con una única fuente de fragmentos de teselaciones regulares del plano. En el caso particular de la malla cuadrada:

Lema 3.17. *El número de orientaciones acíclicas con una única fuente de $L_{2,n}$ es 3^{n-1}.*

Lema 3.18. *El número de orientaciones acíclicas con una única fuente de $L_{3,n}$ viene dado por:*

$$\alpha_s(L_{3,n}) = \begin{cases} 10\alpha_s(L_{3,n-1}) - 11\alpha_s(L_{3,n-2}) & n \text{ impar} \\ 10\alpha_s(L_{3,n-1}) + 11\alpha_s(L_{3,n-2}) & n \text{ par} \end{cases} \qquad n \geq 3$$

$$\begin{aligned} \alpha_s(L_{3,1}) &= 1 \\ \alpha_s(L_{3,2}) &= 9 \end{aligned}$$

Demostración. A partir de la expresión del polinómio cromático de $L_{3,n}$ [28] obtenemos que:

$$\alpha_s(L_{3,1}) = w_1^+(L_{3,n}) = |-10w_1(L_{3,n-1}) - 11w_1(L_{3,n-2})| \quad n \geq 3$$

- Sea $n \geq 3$ un número par. Observando la tabla 3.2 es fácil comprobar que $w_1(L_{3,n}) < 0$ y debido a la alternancia en signo de los números de Whitney:

$$\begin{array}{rcl} w_1(L_{3,n-1}) > 0 & \Rightarrow & \alpha_s(L_{3,n-1}) = w_1(L_{3,n-1}) \\ w_1(L_{3,n-2}) < 0 & \Rightarrow & \alpha_s(L_{3,n-2}) = -w_1(L_{3,n-2}) \end{array}$$

- Razonando de forma análoga, si n es impar, $w_1(L_{3,n}) > 0$; por tanto:

$$\begin{array}{rcl} w_1(L_{3,n-1}) < 0 & \Rightarrow & \alpha_s(L_{3,n-1}) = -w_1(L_{3,n-1}) \\ w_1(L_{3,n-2}) > 0 & \Rightarrow & \alpha_s(L_{3,n-2}) = w_1(L_{3,n-2}) \end{array}$$

□

Para concluir este capítulo queremos resaltar la importancia del algoritmo desarrollado en el Capítulo 2 y de los programas **poltutte**, **poltutte2** y **poltutte3** para resolver problemas de enumeración, tanto en Teoría de Grafos como en otros ámbitos de las matemáticas. Además, queremos hacer notar que aunque los desarrollos de este capítulo están centrados en la malla cuadrada, todos ellos excepto el método de la matriz de compatibilidad, se pueden reproducir para cualquier fragmento de teselaciones del plano, en particular para la malla triangular y la malla hexagonal.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

CAPÍTULO 4

Conclusiones y Problemas Abiertos

En esta primera parte de la memoria hemos calculado el polinomio de Tutte de fragmentos de grandes dimensiones de teselaciones del plano realizadas con cuadrados, triángulos, hexágonos y sus posibles combinaciones; además de estudiar diversos problemas de enumeración en Teoría de Grafos y Geometría. Hemos conseguido codificar, a través de los conceptos de grafo base, capa y banda, estructuras en principio tan dispares como las teselaciones del plano mediante cuadrados, triángulos, hexágonos y cualquier combinación de estos polígonos regulares. Para ello hemos tenido que definir el concepto de fragmento de dimensión $m \times n$ de una teselación del plano mediante combinaciones de cuadrados, triángulos y hexágonos. Además hemos desarrollado un algoritmo que permite el cálculo del polinomio de Tutte asociado a fragmentos de dichas teselaciones del plano ampliando y mejorando los algoritmos existentes.

Hacemos hincapié una vez más en la importancia de este algoritmo, ya que si en 1988 Read y Tutte consideraron un problema abierto y "endiabladamente difícil de resolver"(problema 8.1 en [33]) encontrar el polinomio cromático de un fragmento de dimensión $m \times n$ de la malla cuadrada, nos podemos hacer una idea de lo que supone encontrar un método de cálculo del polinomio de Tutte (generalización del cromático), no solo de un fragmento de la malla cuadrada sino de un fragmento de cualquier teselación del plano realizada con cuadrados, triángulos y hexágonos.

Los programas **poltutte**, **poltutte2** y **poltutte3** son herramientas muy potentes de cálculo del polinomio de Tutte asociado a fragmentos de teselaciones del plano de grandes dimensiones. Su efectividad depende exclusivamente de la capacidad informática del usuario. Queremos resaltar que utilizando estos programas

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

hemos podido mejorar la cota superior del límite asintótico $\lim_{n\to+\infty} \alpha(L_{n,n})^{\frac{1}{n^2}}$ establecida en primer lugar por Merino y Welsh en 1999 [30] y mejorada en 2003 por Calkin, Merino, Noble y Noy [9].

Nos parece que podría ser interesante la programación de la definición recursiva de las orientaciones acíclicas, pues como ya hemos comentado debido a la eliminación de aristas dobles en dicha definición, las dimensiones de las matrices se reducen de forma considerable y podemos obtener con la misma capacidad informática resultados para fragmentos de teselaciones del plano de dimensión mayor que $12 \times n$. Esto supondría el poder mejorar aun más la cota superior del límite asintótico anterior.

Por último, la comunicación que se establece entre la Teoría de Grafos y la Geometría es una vía interesante para obtener métodos de cálculo de algunos problemas clásicos. En esta memoria hemos tratado los arreglos de hiperplanos y los zonotopos asociados a fragmentos de teselaciones del plano. Tenemos constancia de la existencia de diversos algoritmos que calculan el número de caras de determinados arreglos de hiperplanos y el número de vértices de zonotopos en tiempo lineal [25]. El sistema que nosotros planteamos en esta memoria permite realizar este cálculo en tiempo logarítmico y además, a diferencia de los demás algoritmos, utilizamos la Teoría de Grafos como puente para obtener resultados sobre estos problemas geométricos.

II

Segunda parte

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA
BIBLIOTECA

En esta segunda parte de la memoria damos un nuevo enfoque a nuestro estudio del polinomio de Tutte. Debido a la gran información que este polinomio posee acerca del grafo al cual está asociado, parece lógico plantearse si el polinomio de Tutte asociado a un grafo determina unívocamente a dicho grafo, es decir, si dos grafos con el mismo polinomio de Tutte son isomorfos. Sabemos que la respuesta a esta cuestión es negativa: dos grafos con el mismo polinomio de Tutte no tienen por qué ser isomorfos. Esta parte de la memoria está destinada a la búsqueda de grandes familias de grafos que tengan esta propiedad de unicidad, denominada Tutte unicidad. En esta línea damos por primera vez tres grandes familias de grafos que son Tutte únicas: los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6. Además, clasificamos las teselaciones hexagonales y los grafos localmente C_6. Queremos resaltar la importancia de estos resultados, en el sentido de que rectificamos las clasificaciones realizadas en 1991 por Thomassen [35]. Definimos dos familias de teselaciones no descritas hasta el momento y demostramos que incluyendo dichas familias hemos considerado todos los casos posibles. También establecemos una relación de menor muy potente entre los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6; relacionando así las tres familias de grafos estudiadas en esta segunda parte de la memoria.

La demostración de la Tutte unicidad de estas familias de grafos supone un conocimiento en profundidad de todos los grafos pertenecientes a dichas familias, ya que hay que realizar el siguiente estudio:

1.- Clasificación: caracterizar todos los grafos pertenecientes a la familia.

2.- Estudio de invariantes numéricos contenidos en el polinomio de Tutte: números cromáticos, vértice-transitividad, ciclos esenciales y cualquier otro invariante de grafos que permita distinguir los polinomios de Tutte asociados a dos grafos no isomorfos de la familia. Veremos que en todas las familias no es necesario estudiar todos los invariantes mencionados.

3.- Tutte unicidad: comparativa dos a dos entre todos los grafos clasificados en el paso 1, utilizando los invariantes estudiados en el paso 2.

Hemos estructurado esta parte de la memoria en cuatro capítulos. En el Capítulo 5, demostramos la Tutte unicidad de los grafos localmente cuadriculados. Para ello hacemos uso del teorema de clasificación de dichos grafos demostrado de forma independiente por Thomassen [35] y por Márquez, Mier, Noy y Revuelta [29]. En este resultado se prueba que los grafos localmente cuadriculados se clasifican en cinco familias y que todos ellos admiten una inmersión en el toro topológico o en la botella de Klein.

Utilizamos dos invariantes para distinguir los polinomios de Tutte asociados

a dos grafos localmente cuadriculados no isomorfos: la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud. Damos una nueva definición de ciclo esencial, más general que la dada por Márquez, Mier, Noy y Revuelta en [29], ya que la definición dada en esta memoria no depende de la inmersión del grafo; y demostramos que ambas definiciones son equivalentes. Calculamos valores exactos del número de ciclos esenciales de longitud mínima en cada grafo localmente cuadriculados, hecho fundamental para poder realizar una comparativa dos a dos entre todos los grafos dados en el teorema de clasificación. Tendremos que manejar expresiones de gran complejidad con números combinatorios y calcular el número de conjuntos de aristas de determinados rangos y tamaños, lo cual implica el estudio en profundidad de los grafos localmente cuadriculados.

En el Capítulo 6, estudiamos los números cromáticos, la vértice-transitividad y los ciclos esenciales de las teselaciones hexagonales a fin de poder clasificarlas. Damos una clasificación completa de esta familia de grafos, rectificando el teorema enunciado por Thomassen en [35]. Además, mostraremos, que al igual que los grafos localmente cuadriculados, las teselaciones hexagonales admiten una inmersión en el toro topológico o en la botella de Klein. El teorema de clasificación de las teselaciones hexagonales y la relación de dualidad existente entre dicha familia de grafos y los grafos localmente C_6 demostrada por Thomassen [35], nos permiten clasificar los grafos localmente C_6.

Finalmente, demostramos que existe una relación de menor entre los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6. Esta relación está basada en dos operaciones muy potentes en Teoría de Grafos: la contracción y el borrado de aristas; y es fundamental en el Capítulo 7, el cual está destinado a establecer la maquinaria necesaria para demostrar la Tutte unicidad de las teselaciones hexagonales y de los grafos localmente C_6. Debido a la similitud del desarrollo nos centramos en la Tutte unicidad de las teselaciones hexagonales ya que no nos aporta nada nuevo desde el punto de vista matemático repetir el proceso para los grafos localmente C_6. Probamos toda una serie de resultados necesarios para poder establecer una comparativa dos a dos entre todos los grafos dados en el teorema de clasificación. Entre estos resultados destacan dos. El primero de ellos es que el polinomio de Tutte preserva la estructura local de las teselaciones hexagonales y el segundo, la existencia de un orientación local en esta familia de grafos. Estos resultados también son ciertos para los grafos localmente C_6. Por último, realizamos las comparaciones correspondientes a la teselación hexagonal toroidal a fin de demostrar su Tutte unicidad. Realizando las demás comparaciones se demuestra la Tutte unicidad del resto de las teselaciones hexagonales y como ya hemos comentado esta técnica se reproduce para los grafos localmente C_6.

CAPÍTULO 5

Tutte Unicidad de los Grafos Localmente Cuadriculados

El objetivo de este capítulo es demostrar, que además de la cuadrícula toroidal [29], los demás grafos localmente cuadriculados también están unívocamente determinados por su polinomio de Tutte. Para ello damos una definición más general de ciclo esencial a la dada por Márquez, Mier, Noy y Revuelta en [29] y calculamos valores exactos del número de ciclos esenciales de longitud mínima en todos los grafos localmente cuadriculados, hecho fundamental para poder demostrar su Tutte unicidad. A través del estudio de propiedades combinatorias nos enfrentaremos al difícil problema de establecer en qué grafos localmente cuadriculados, pueden coincidir la longitud mínima de los ciclos esenciales, el número de ciclos esenciales de dicha longitud y el número de vértices. Además, mediante el cálculo de conjuntos de aristas de determinado rango y tamaño probaremos que aunque las cantidades anteriores coincidan, en dos grafos localmente cuadriculados no isomorfos con pq vértices hay al menos un coeficiente del polinomio generador de rango-tamaño en el que ambos grafos difieren.

5.1. Introducción

Thomassen en 1991 [35] y Márquez, Mier, Noy y Revuelta en 2003 [29] demuestran un teorema de clasificación de los grafos localmente cuadriculados, mostrando que todos ellos admiten una inmersión en el toro o en la botella de Klein. Se obtienen cinco familias no isomorfas de grafos, en función de las posibles formas de añadir aristas entre los vértices de valencia dos y tres de una cuadrícula

$p \times q$. Además, Márquez, Mier, Noy y Revuelta demuestran la Tutte unicidad de uno de estos grafos: la cuadrícula toroidal $C_p \times C_q$ [29].

Nuestro objetivo en este capítulo es demostrar que los demás grafos localmente cuadriculados también son Tutte únicos. Para ello en primer lugar damos una definición de ciclo esencial que no depende de la inmersión del grafo y por tanto más general a la dada por Márquez, Mier, Noy y Revuelta en [29]. A continuación demostramos que ambas definiciones son equivalentes y calculamos los valores exactos del número de ciclos esenciales de longitud mínima en todos los grafos localmente cuadriculados, hecho fundamental para poder establecer comparaciones dos a dos entre todos estos grafos, a fin de demostrar su Tutte unicidad.

Con estas herramientas podemos abordar el arduo problema de establecer todas las posibles comparaciones dos a dos entre todos los grafos localmente cuadriculados. Esta comparativa está basada en los ciclos esenciales de dichos grafos. Si dos de estos grafos tienen el mismo polinomio de Tutte entonces tienen que coincidir el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud [29]. El poder comparar estas cantidades supone el manejo de expresiones con números combinatorios de gran complejidad, por ejemplo, demostraremos que no existen $p, q, \delta, p', q' \in \mathbb{N}$ con $q' \geq 2p'$ tales que:

$$pq = p'q'$$
$$2p' = p = q + \delta$$
$$q + p\binom{q+\delta-1}{\delta} = q'(q'-p')\binom{2p'-1}{p'}$$

Además, en aquellos casos en los que sí puedan coincidir las cantidades anteriores nos veremos obligados a establecer biyecciones entre conjuntos de aristas de grafos muy complejos inmersos en el toro topológico o en la botella de Klein. En estos grafos también tendremos que contar conjuntos de aristas de determinados rangos y tamaños, lo que supone realizar un estudio en profundidad de cada grafo localmente cuadriculado.

Debido a la gran cantidad de cálculos que nos vemos obligados a realizar ofrecemos varias tablas en las que el lector podrá ver fácilmente el motivo por el cual dos grafos localmente cuadriculados no isomorfos no tienen el mismo polinomio de Tutte.

A continuación, definimos los grafos localmente cuadriculados y comentamos algunas de sus propiedades.

Sea $N(x)$ el conjunto de vecinos de un vértice x. Un grafo G conexo y 4−regular es *localmente cuadriculado* si para cada vértice $x \in V(G)$ existe una ordenación x_1, x_2, x_3, x_4 de $N(x)$ y cuatro vértices diferentes y_1, y_2, y_3, y_4, tales que, tomando índices módulo 4,

$$\begin{aligned} N(x_i) \cap N(x_{i+1}) &= \{x, y_i\} \\ N(x_i) \cap N(x_{i+2}) &= \{x\} \end{aligned}$$

y las únicas adyacencias entre los vértices $\{x, x_1, \ldots, x_4, y_1, \ldots, y_4\}$ son las requeridas por esta condición (Figura 5.1). Esta restricción es necesaria para demostrar que el polinomio de Tutte preserva la estructura local de estos grafos [29].

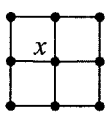


Figura 5.1: Estructura alrededor de un vértice x

Una definición informal de grafo localmente cuadriculado, es que son aquellos grafos en los cuales la estructura alrededor de cada vértice es una cuadrícula 3×3.

Los grafos localmente cuadriculados son simples, es decir, no tienen lazos ni aristas múltiples; además estos grafos no contienen triángulos y cada vértice pertenece exactamente a cuatro cuadrados. Es fácil demostrar que son 2−conexos y que su arista-conectividad es igual a cuatro.

Dado un grafo H, un grafo conexo G se dice que es *localmente* H si para cada vértice x el subgrafo inducido por el conjunto de vecinos de x es isomorfo a H. Por ejemplo, si P es el grafo de Petersen, hay tres grafos localmente P [22]. La condición de ser localmente cuadriculado es diferente ya que no solo afecta a un vértice y sus vecinos, sino también a cuatro vértices a distancia dos.

5.2. Clasificación de los Grafos Localmente Cuadriculados

Sea $H = P_p \times P_q$ la cuadrícula $p \times q$ siendo P_p y P_q dos caminos con p y q vértices respectivamente. Etiquetamos los vértices de H con los elementos del grupo abeliano $\mathbb{Z}_p \times \mathbb{Z}_q$. Es importante observar que los vértices de grado cuatro de H ya verifican la propiedad de ser localmente cuadriculados, luego para obtener grafos localmente cuadriculados se añaden aristas entre los vértices de grado dos y tres de H. En función de como se añaden estas aristas, Márquez, Mier, Noy y Revuelta [29] construyen los siguientes grafos localmente cuadriculados (ver también la definiciones dadas por Thomassen [35]).

A) Toro $T^{\delta}_{p,q}$ con $p \geq 5$, $0 \leq \delta \leq p/2$, $\delta + q \geq 5$ si $q \geq 4$, $\delta + q \geq 6$ si $q = 2, 3$ ó $4 \leq \delta < p/2$ con $\delta \neq p/3, p/4$ si $q = 1$.

$$\begin{aligned} A(T^{\delta}_{p,q}) &= A(H) \cup \{\{(i,0),(i+\delta,q-1)\},\ 0 \leq i \leq p-1\} \\ &\quad \cup \{\{(0,j),(p-1,j)\},\ 0 \leq j \leq q-1\} \end{aligned}$$

Para $\delta = 0$ obtenemos la cuadrícula toroidal, a la que denotaremos $T_{p,q}$. Todos los grafos de esta familia están inmersos en el toro topológico (Figura 5.2a). Queremos hacer notar que en todas las figuras presentadas en esta segunda parte de la memoria, los vértices del grafo están representados por puntos y dos vértices con la misma etiqueta corresponden a puntos que están identificados en la superficie.

B) Botella de Klein $K^1_{p,q}$ con $p \geq 5$, p impar y $q \geq 4 + \lceil i/2 \rceil$. (Figura 5.2b).

$$\begin{aligned} A(K^1_{p,q}) &= A(H) \cup \{\{(j,0),(p-j-1,q-1)\},\ 0 \leq j \leq p-1\} \\ &\quad \cup \{\{(0,j),(p-1,j)\},\ 0 \leq j \leq q-1\} \end{aligned}$$

C) Botella de Klein $K^0_{p,q}$ con $p \geq 5$, p par y $q \geq 4 + \lceil i/2 \rceil$. (Figura 5.3$a$).

$$\begin{aligned} A(K^0_{p,q}) &= A(H) \cup \{\{(j,0),(p-j-1,q-1)\},\ 0 \leq j \leq p-1\} \\ &\quad \cup \{\{(0,j),(p-1,j)\},\ 0 \leq j \leq q-1\} \end{aligned}$$

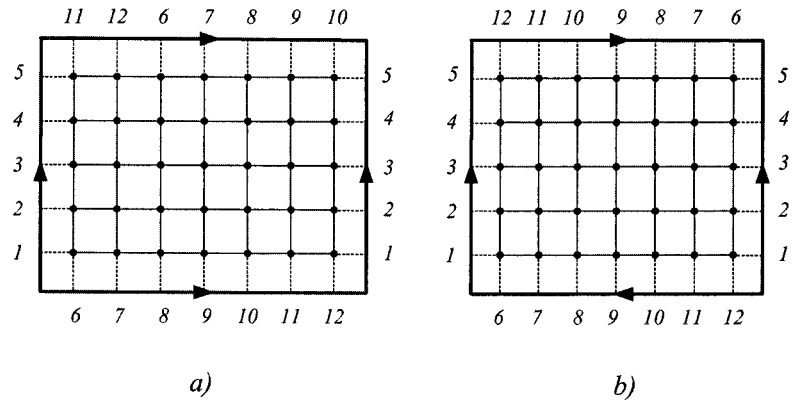


Figura 5.2: *a)* $T^2_{7,5}$ b) $K^1_{7,5}$

D) Botella de Klein $K^2_{p,q}$ con $p \geq 5$, p par y $q \geq 4 + \lceil i/2 \rceil$. (Figura 5.3b).

$$\begin{aligned} A(K^2_{p,q}) &= A(H) \cup \{\{(j,0),(p-j,q-1)\},\ 0 \leq j \leq p-1\} \\ &\quad \cup \{\{(0,j),(p-1,j)\},\ 0 \leq j \leq q-1\} \end{aligned}$$

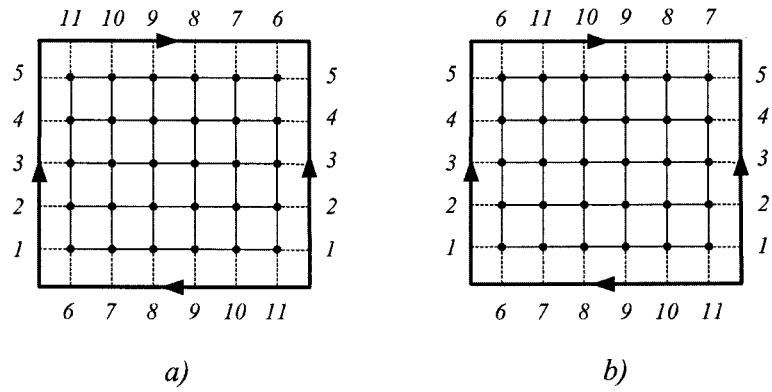


Figura 5.3: *a)* $K^0_{6,5}$ b) $K^2_{6,5}$

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

En las botellas de Klein, $K^i_{p,q}$, el índice i indica el número de identificaciones del tipo $\{(c,0),(c,q-1)\}$ que tiene el grafo. A estas identificaciones las llamamos *identificaciones fijas*. Así, en $K^0_{p,q}$ no hay ninguna identificación fija. En $K^1_{p,q}$ hay una, $\{((p-1)/2,0),((p-1)/2,q-1)\}$ y en $K^2_{p,q}$ hay dos.

A continuación definimos otra familia de grafos cuyos miembros admiten una inmersión en la botella de Klein. Debido a su extraña pegada los conocemos como los *grafos raros*.

E) Los grafos $S_{p,q}$ con $p \geq 3$ y $q \geq 6$. (Figura 5.4).

$$\begin{aligned}
\text{Si } p \leq q \quad A(S_{p,q}) &= A(H) \cup \{(j,0),(p-j,q-p+j)\},\ 0 \leq j \leq p-1\} \\
&\cup \{\{(0,i),(i,q-1)\},\ 0 \leq i \leq p-1\} \\
&\cup \{\{(0,i),(p-1,i-p)\},\ p \leq i \leq q-1\}
\end{aligned}$$

$$\begin{aligned}
\text{Si } q \leq p \quad A(S_{p,q}) &= A(H) \cup \{\{(j,0),(0,q-1-j)\},\ 0 \leq j \leq q-1\} \\
&\cup \{\{(p-1-i,q-1),(p-1,i)\},\ 0 \leq i \leq q-1\} \\
&\cup \{\{(i,q-1),(i+q,0)\},\ 0 \leq i \leq p-q-1\}
\end{aligned}$$

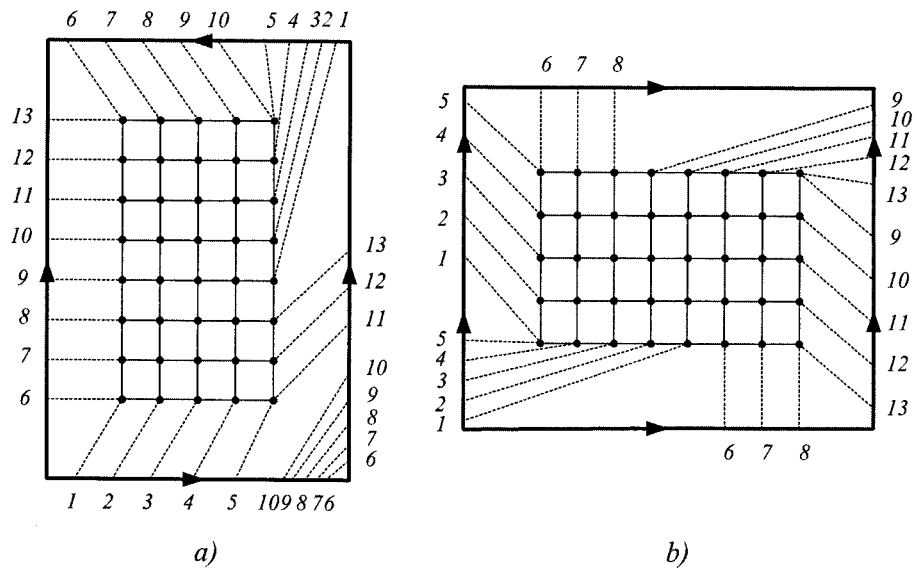


Figura 5.4: *a)* $S_{5,8}$ b) $S_{8,5}$

Una vez definidas las familias de grafos localmente cuadriculados, enunciamos el teorema de clasificación de dichos grafos demostrado por Márquez, Mier, Noy y Revuelta [29] (ver también la demostración realizada por Thomassen [35]). Con este teorema se demuestra que todo grafo localmente cuadriculado es isomorfo a un único grafo perteneciente a una de las familias de grafos definidas en los apartados anteriores.

Teorema 5.1. *[29] Si G es un grafo localmente cuadriculado con N vértices, entonces G es isomorfo a uno y sólo uno de los siguientes grafos:*

A) $G \cong T^{\delta}_{p,q}$ *con* $pq = N$, $p \geq 5$, $\delta \leq p/2$ *y* $\delta + q \geq 5$ *si* $q \geq 4$ *ó* $\delta + q \geq 6$ *si* $q = 2, 3$ *ó* $4 \leq \delta \leq p/2$ *si* $q = 1$.

B) $G \cong K^{i}_{p,q}$ *con* $pq = N$, $p \geq 5$, $i \equiv p \ (mod\ 2)$ *para* $i \in \{0, 1, 2\}$ *y* $q \geq 4 + \lceil i/2 \rceil$.

C) $G \cong S_{p,q}$ *con* $pq = N$, $p \geq 3$ *y* $q \geq 6$.

5.3. Tutte Unicidad

El objetivo de esta sección es demostrar la Tutte unicidad de todos los grafos localmente cuadriculados. El concepto clave que vamos a utilizar a lo largo de toda la sección es el de ciclo esencial, que definimos a continuación.

Definición 5.1. Dado un grafo localmente cuadriculado G y dos ciclos de G, C y C', decimos que C *es localmente homotópico a* C' si existe un ciclo de longitud 4, H, tal que $C \cap H$ es conexo y C' se obtiene de C sustituyendo $C \cap H$ por $H - (C \cap H)$. Una *homotopía* es una sucesión de homotopías locales.

Definición 5.2. Un ciclo C de un grafo localmente cuadriculado es un *ciclo esencial* si no es homotópico a un ciclo de longitud 4. En caso contrario decimos que el ciclo es *contractible*.

Algunos ejemplos de ciclo esencial y contractible se muestran en la Figura 5.5.

En 2003, Márquez, Mier, Noy y Revuelta [29] definen los ciclos contractibles y esenciales de la siguiente forma.

Definición 5.3. [29] Sea M la superficie en la cual un grafo localmente cuadriculado G está inmerso de forma natural, es decir, M es un toro o una botella de Klein. Se dice que un ciclo C en G es contractible si $M - C$ tiene dos componentes conexas, una de ellas contraible a un punto. En caso contrario C es un ciclo esencial. En otras palabras, los ciclos contractibles determinan una región simplemente conexa y los esenciales no la determinan.

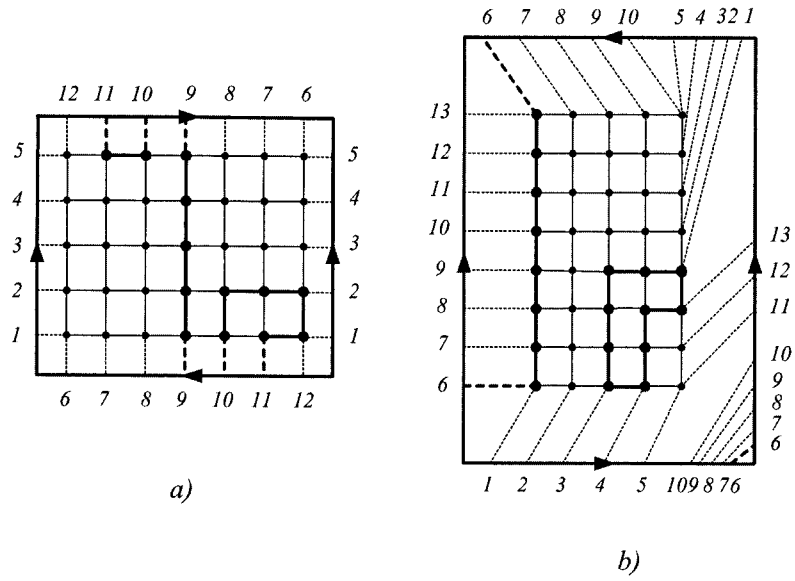


Figura 5.5: *a*) Ciclo esencial (determinado por la arista etiquetada 9) y ciclo contractible en $K^1_{7,5}$ b) Ciclo esencial (determinado por la arista etiquetada 6) y ciclo contractible en $S_{5,8}$.

A continuación vamos a demostrar que ambas definiciones determinan el mismo tipo de ciclo. Aunque hay que tener en cuenta que la Definición 5.2 no depende de la inmersión del grafo y por tanto es más general que la dada por Márquez, Mier, Noy y Revuelta en 2003.

Lema 5.2. *Las definiciones 5.2 y 5.3 son equivalentes.*

Demostración. Vamos a demostrar que las dos definiciones de ciclo contractible son equivalentes y por tanto también lo serán las de ciclo esencial. Sea C un ciclo homotópico a un ciclo de longitud 4, C', en un grafo localmente cuadriculado G. Sabemos que C' se obtiene a partir de C realizando una serie de homotopías locales. Cada vez que aplicamos una de estas homotopías locales aparece un ciclo de longitud 4, H tal que $C \cap H$ es conexo. Supongamos que C' se obtiene a partir de C realizando r homotopías locales. Realizamos ahora la operación inversa, añadiéndole a C', $C \cap H_r$ y eliminando $H_r - (C \cap H_r)$. Al resultado le añadimos $C \cap H_{r-1}$ y le eliminamos $(H_{r-1}) - (C \cap H_{r-1})$. Si repetimos el proceso r veces, llegamos a C y por tanto este ciclo determina una región simplemente conexa ya que H la determina y en cada paso lo único que hacemos es ampliar la región determinada por H. Recíprocamente, si C es un ciclo que determina una región simplemente conexa, seleccionamos un cuadrado H del interior de la región que tenga al menos una arista en C y tal que $C \cap H$ sea conexo. Reemplazamos $C \cap H$ por $H - (C \cap H)$. El resultado es un ciclo que determina una región simplemente conexa que contiene un cuadrado menos en su interior. Aplicando este proceso un número finito de veces (pues el grafo tiene un número finito de aristas) obtenemos que C es homotópico a un ciclo de longitud 4. □

Un conjunto de aristas $A \subseteq A(G)$ se dice que es un *conjunto normal de aristas* si no contiene ningún ciclo esencial. Denotamos por l_G a la longitud mínima de los ciclos esenciales de G. El número de ciclos esenciales de longitud l_G contribuye al coeficiente a_{l_G-1,l_G} del polinomio de rango-tamaño de G, $R(G;x,y)$, el cual cuenta el número de conjuntos de aristas de rango $l_G - 1$ y tamaño l_G. A fin de demostrar la Tutte unicidad de los grafos localmente cuadriculados vamos a utilizar dos resultados probados por Márquez, Mier, Noy y Revuelta en [29]. El primero de ellos, nos dice que el polinomio de Tutte preserva la estructura de grafo localmente cuadriculado, y el segundo establece que si dos grafos localmente cuadriculados tienen el mismo polinomio de Tutte, entonces tienen que coincidir la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud.

Lema 5.3. *[29] Dados dos grafos G y G', si G es un grafo localmente cuadriculado y $T(G;x,y) = T(G';x,y)$ entonces G' es localmente cuadriculado.*

Lema 5.4. *[29] Sean G, G' un par de grafos localmente cuadriculados con pq vértices. Entonces:*

- *$l_G \neq l_{G'}$ implica $T(G;x,y) \neq T(G';x,y)$.*
- *Si $l_G = l_{G'}$ pero G y G' no tienen el mismo número de ciclos esenciales de longitud mínima, entonces $T(G;x,y) \neq T(G';x,y)$.*

El proceso a seguir para demostrar la Tutte unicidad es comparar dos a dos todos los grafos dados en el Teorema 5.1. Esta comparativa está basada en los ciclos esenciales. Por tanto, el primer paso es calcular valores exactos de la longitud mínima de estos ciclos y del número de ciclos esenciales de dicha longitud en cada uno de los grafos localmente cuadriculados.

Lema 5.5. *Si G es un grafo localmente cuadriculado con pq vértices, entonces la longitud l_G y el número de ciclos esenciales de dicha longitud vienen dados en la siguiente tabla.*

G	l_G	nº ciclos esenciales	
$T_{p,q}$	$\min\{p,q\}$	q	*si* $p<q$
		$2p$	*si* $p=q$
		p	*si* $p>q$
$T^{\delta}_{p,q}$	$\min\{p,q+\delta\}$	q	*si* $p<q+\delta$
		$q+p\binom{q+\delta-1}{\delta}$	*si* $p=q+\delta$
		$p\binom{q+\delta-1}{\delta}$	*si* $p>q+\delta$
$K^0_{p,q}$	$\min\{p,q+1\}$	q	*si* $p<q+1$
		$5q$	*si* $p=q+1$
		$4q$	*si* $p>q+1$
$K^1_{p,q}$	$\min\{p,q\}$	q	*si* $p<q$
		$q+1$	*si* $p=q$
		1	*si* $p>q$
$K^2_{p,q}$	$\min\{p,q\}$	q	*si* $p<q$
		$q+2$	*si* $p=q$
		2	*si* $p>q$
$S_{p,q}$	$\min\{2p,q\}$	$2p\sum_{j=0}^{p-1}\binom{q-1}{j}$	*si* $p\leq q\leq 2p$
		$q(q-p)\binom{2p-1}{p}$	*si* $2p\leq q$
		2^q	*si* $q\leq p$

Demostración. Los casos $T_{p,q}$, $T^{\delta}_{p,q}$ y $K^{i}_{p,q}$ se prueban en [29], donde también se muestra que $l_{S_{p,q}} = \min(2p, q)$ y que si $q \leq p$ el número de ciclos esenciales de longitud mínima es 2^q. Por tanto, quedan por demostrar dos casos en los cuales se dieron cotas inferiores del número de ciclos esenciales de longitud $l_{S_{p,q}}$. Nosotros estamos interesados en calcular el número exacto.

Los grafos localmente cuadriculados con pq vértices se construyen añadiendo aristas a la cuadrícula $p \times q$. A estas aristas se les llama *aristas exteriores.* Los ciclos esenciales de longitud mínima se obtienen uniendo los extremos de una arista exterior por un camino de longitud mínima contenido en la cuadrícula $p \times q$. En $S_{p,q}$ distinguimos dos casos:

Caso 1: Si $2p \leq q$, cada arista exterior de la forma $\{(0,i),(p-1,i-p)\}$ determina $\binom{2p-1}{p}$ ciclos esenciales de longitud $2p$. Tenemos $q-p$ aristas exteriores de este tipo y cada una de ellas puede ocupar q vértices distintos. Por tanto, el número de ciclos esenciales de longitud $2p$ es $q(q-p)\binom{2p-1}{p}$.

Caso 2: Si $p \leq q \leq 2p$, cada arista exterior de la forma $\{(0,i),(i,q-1)\}$ ó $\{(i,0),(p-1,q-p+i)\}$ da lugar a $\binom{q-1}{i}$ ciclos esenciales de longitud q. Como $0 \leq i \leq (p-1)$, el número de ciclos esenciales de longitud q por el número de aristas que generan estos ciclos es $2\sum_{i=0}^{p-1}\binom{q-1}{i}$. Además, estas aristas pueden ocupar p vértices distintos, luego el número de ciclos esenciales de longitud mínima q es $2p\sum_{i=0}^{p-1}\binom{q-1}{i}$. □

A continuación, demostramos que el toro con desplazamiento $T^{\delta}_{p,q}$ está unívocamente determinado por su polinomio de Tutte.

Teorema 5.6. *Sean $p, q \geq 6$ verificando las siguientes condiciones:*

1.- $p\binom{q+\delta-1}{\delta}$ *no es una potencia de 2.*

2.- $pq \neq p'q'$ *si* $p = q+\delta = q'+\delta' < p'$ *y* $q + p\binom{p-1}{\delta} = p'\binom{p-1}{\delta'}$.

Entonces el grafo $T^{\delta}_{p,q}$ es Tutte único.

Demostración. Sean $p, q \geq 6$ y G un grafo tal que $T(G; x, y) = T(T^{\delta}_{p,q}; x, y)$. Por el Teorema 5.1 y el Lema 5.3, G tiene que ser isomorfo a uno y sólo uno de los siguientes grafos:

$$T^{\delta'}_{p',q'},\ K^{i}_{p',q'},\ S_{p',q'}$$

Vamos a demostrar que G es isomorfo a $T^{\delta'}_{p',q'}$ con $p = p'$, $q = q'$ y $\delta = \delta'$; para ello suponemos que G es isomorfo a cada uno de los grafos anteriores y obtendremos una contradicción en todos los casos, excepto en el anteriormente mencionado.

Por el Lema 5.4, sabemos que $l_{T^{\delta}_{p,q}} = l_G$ y que el número de vértices y el número de ciclos esenciales de longitud mínima tienen que coincidir en ambos grafos. El procedimiento que vamos a seguir es comparar el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud en $T^{\delta}_{p,q}$ y en cada uno de los grafos dados en el listado anterior. En los casos en los que estas cantidades no coincidan, tendremos demostrado que los grafos considerados no son Tutte equivalentes. Si ambas cantidades coinciden utilizaremos otros métodos para demostrar el resultado deseado. Podemos considerar $\delta > 0$ y G no isomorfo a $T_{p',q'}$ ya que la cuadrícula toroidal está unívocamente determinada por su polinomio de Tutte [29].

CASO 1: Supongamos $G \simeq K^{0}_{p',q'}$.

Por el Lema 5.4, $l_{T^{\delta}_{p,q}} = l_{K^{0}_{p',q'}}$ y el número de ciclos esenciales de longitud mínima es el mismo en ambos grafos.

Caso 1.1. $l_{T^{\delta}_{p,q}} = p$, $l_{K^{0}_{p',q'}} = p'$ con $p < q + \delta$ y $p' < q' + 1$.

Por ser ambos grafos Tutte equivalentes tenemos que $p = p'$ y $q = q'$ con $p < q + 1$. Vamos a demostrar que con estas igualdades, el número de conjuntos de aristas de rango q y tamaño $q + 1$ es distinto en $T^{\delta}_{p,q}$ y $K^{0}_{p,q}$. Esto supone una contradicción, ya que este número es el coeficiente de $x^q y^{q+1}$ del polinomio generador de rango-tamaño; y por tanto $T(T^{\delta'}_{p,q}; x, y) \neq T(K^{0}_{p,q}; x, y)$. Definimos dos tipos de conjuntos de aristas de rango q y tamaño $q + 1$:

(1) Ciclos esenciales de longitud q.

(2) Conjuntos de aristas de rango q y tamaño $q + 1$ no ciclos esenciales.

Haciendo un recuento del número de conjuntos de aristas de cada tipo que hay en cada grafo, obtenemos la siguiente tabla:

Tipo	$T^{\delta}_{p,q}$	$K^{0}_{p,q}$
1	n si $\delta > 1$ $n + pq$ si $\delta = 1$	$n + 4q$
2	?	?

donde n es el número de ciclos esenciales de longitud q generados por las aristas del tipo $\{(0,j),(p-1,j)\}$ con $0 \leq j \leq q-1$. Si demostramos que existe una biyección entre los conjuntos del tipo 2 de ambos grafos, tendremos probado que $T^{\delta}_{p,q}$ y $K^{0}_{p,q}$ tienen distinto número de conjuntos de aristas de rango q y tamaño $q+1$; y obtendremos una contradicción.

Definimos los siguientes conjuntos de aristas en la cuadrícula plana $p \times q$:

$$E_r = \{((i,r),(i,r+1))\,;\, 0 \leq i \leq p-1\},\ \ 0 \leq r \leq q-2$$

Si A es un conjunto de aristas, no ciclo esencial, contenido en $A(T^{\delta}_{p,q})$ ó en $A(K^{0}_{p,q})$, entonces es fácil comprobar que $s(A)$ =min$\{r \in [0,q-2]\ ;\ A \cap E_r = \emptyset\}$ siempre existe. Para cada $r \in [0,q-2]$ definimos una biyección φ_r entre los siguientes conjuntos de aristas:

$$\{A \subseteq A(T^{\delta}_{p,q}) | r(A) = q, |A| = q+1, s(A) = r\}$$
$$\{A \subseteq A(K^{0}_{p,q}) | r(A) = q, |A| = q+1, s(A) = r\}$$

Si $A \subset A(T^{\delta}_{p,q})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(p-1-i'+\delta,j')\} & j' = q-1, j = 0 \\ \{(i,j),(i',j')\} & j,j' \in [0,r] \\ \{(p-1-i+\delta,j),(p-1-i'+\delta,j')\} & j,j' \in [r+1,q-1] \end{cases}$$

Caso 1.2. $l_{T^{\delta}_{p,q}} = p$, $l_{K^{0}_{p',q'}} = p' = q'+1$ con $p < q+\delta$ ó $l_{T^{\delta}_{p,q}} = q+\delta < p$, $l_{K^{0}_{p',q'}} = p'$ con $p' < q'+1$.

La contradicción en estos dos casos se produce de forma inmediata, debido a que no pueden coincidir el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud en ambos grafos.

Caso 1.3. $l_{T^{\delta}_{p,q}} = p$, $l_{K^{0}_{p',q'}} = q'+1$ con $p < q+\delta$ y $q'+1 < p'$.

Para obtener una contradicción, vamos a probar que el número de conjuntos de aristas de rango $q'+2$ y tamaño $q'+3$ es mayor en $T^{\delta}_{p,q}$ que en $K^{0}_{p',q'}$, y por tanto los polinomios de Tutte asociados a ambos grafos no pueden ser iguales. Estos conjuntos se pueden clasificar en tres grupos:

1.- Conjuntos normales de aristas (no contienen ciclos esenciales).

2.- Conjuntos conteniendo un ciclo esencial de longitud $q'+1$ y dos aristas más. (Figura 5.6a)

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

3.- Ciclos esenciales de longitud $q' + 3$. (Figuras 5.6b y 5.7)

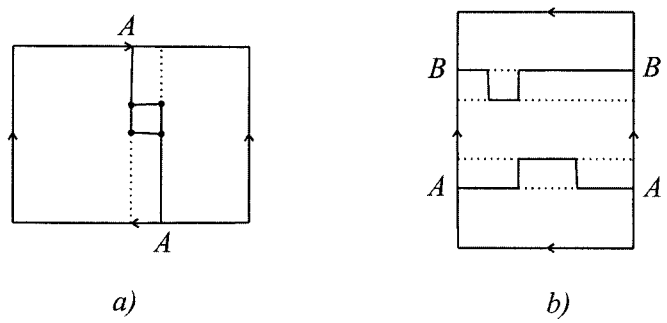


Figura 5.6: *a*) Conjunto de aristas en $K^0_{p',q'}$ conteniendo un ciclo esencial de longitud $q' + 1$ y dos aristas más. b) Ciclos esenciales de longitud $q' + 3$ en $T^\delta_{p,q}$

(1) Conjuntos Normales.

En el Corolario 16 de [29] Márquez, Mier, Noy y Revuelta demuestran que el número de conjuntos normales de rango r y tamaño s es el mismo en todo grafo localmente cuadriculado G con pq vértices, siempre y cuando se verifiquen dos condiciones: $s \leq l_G + 2$ y si algún conjunto normal de los que estamos evaluando contiene un ciclo contractible, G no puede contener conjuntos prohibidos de aristas de tamaño s cuyo ciclo contractible sea de igual longitud.

No queremos entrar en definir formalmente los conjuntos prohibidos de aristas, pues ello requeriría explicar muchos conceptos que no nos interesan en nuestro estudio. Nosotros entenderemos como *conjunto prohibido de aristas de un grafo localmente cuadriculado*, aquellos conjuntos que contienen un ciclo esencial y un ciclo contractible. Con esta definición, es fácil comprobar que $T^\delta_{p,q}$ no contiene conjuntos prohibidos de aristas de tamaño $p+2 = q'+3$ pero $K^0_{p',q'}$ sí los contiene. En la Figura 5.6*a* se muestra que en $K^0_{p',q'}$ hay conjuntos de aristas que contienen un ciclo esencial y un ciclo de longitud 4. De hecho, esta es la única posibilidad de formar un conjunto prohibido de aristas de tamaño $q' + 3$.

Aplicando el Corolario 16 de [29], sabemos que $T^\delta_{p,q}$ y $K^0_{p',q'}$ tienen el mismo número de conjuntos normales de rango $q' + 2$ y tamaño $q' + 3$ que no contienen un ciclo de longitud 4. Vamos a probar que $T^\delta_{p,q}$ tiene más conjuntos normales de rango $q' + 2$ y tamaño $q' + 3$, que contienen un ciclo de longitud 4, que $K^0_{p',q'}$. Otra vez por el Corolario 16 de [29], el número de conjuntos de aristas de rango $q' + 1$ y tamaño $q' + 2$ conteniendo un cuadrado, es el mismo en ambos grafos,

llamémosle $s_{q'+1}$. Al reducir el tamaño del conjunto hemos evitado los conjuntos prohibidos de $K^0_{p',q'}$. Añadimos una arista para obtener un conjunto de tamaño $q'+3$. Hay tres posibilidades en función de como estemos añadiendo la arista:

A) Que sea un conjunto normal de rango $q'+2$.

B) Que sea un conjunto normal conteniendo dos ciclos contractibles y por tanto con rango $q'+1$.

C) Que sea un conjunto de aristas conteniendo un ciclo esencial de longitud $q'+1$ y un ciclo contractible de longitud 4.

Llamamos $A(G)$, $B(G)$, $C(G)$ al número de conjuntos del tipo A, B, C respectivamente de G, siendo G el grafo $T^\delta_{p,q}$ ó $K^0_{p',q'}$. Tenemos que probar que $A(T^\delta_{p,q}) > A(K^0_{p',q'})$. $A(G)+B(G)+C(G)$ es el número de posibilidades de añadir una arista a los conjuntos de aristas de tamaño $q'+2$ conteniendo un ciclo de longitud 4 (cuadrado), y tal que la arista añadida no pertenezca a un cuadrado fijado.

$$A(G)+B(G)+C(G) = \frac{s_{q'+1}(2pq-q'-2)}{q'+3-4}$$

por tanto,

$$s_{q'+1}(2pq-q'-2) = A(G)(q'-1)+B(G)(q'-1)+C(G)(q'-1)$$

Además, $(q'-1)B(G) = \sum_{B\in B(B)}(q'+3-\delta(B))$, donde $\delta(B)$ es el número de aristas de B que no pertenecen a todos los ciclos de longitud 4 de B. Esta última igualdad se deduce del hecho de que para obtener un conjunto de aristas $B \in B(G)$ añadimos una arista a un conjunto de aristas de tamaño $q'+2$ que contiene un cuadrado, formando un nuevo ciclo contractible. Como el ciclo es contractible, podemos considerar que la arista añadida tiene que pertenecer a un nuevo cuadrado. Así,

$$B(G) = \frac{\sum_{B\in B(G)}(q'+3-\delta(B))}{q'+3-4}$$

Si aplicamos el mismo corolario de [29] para cada $B \in B(G)$, el número de conjuntos normales de aristas de tamaño $q'+3-\delta(B)$ es igual en $T^\delta_{p,q}$ y $K^0_{p',q'}$ entonces:

$$\sum_{B\in B(T^\delta_{p,q})}(q'+3-\delta(B)) = \sum_{B\in B(K^0_{p',q'})}(q'+3-\delta(B))$$

Como $C(T^\delta_{p,q}) = 0$ y $C(K^0_{p',q'}) \neq 0$ tenemos que:

$$A(T^\delta_{p,q})(q'-1) = A(K^0_{p',q'})(q'-1)+C(K^0_{p',q'})(q'-1)$$

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

(2) Conjuntos de aristas conteniendo un ciclo esencial de longitud $q'+1$ y dos aristas más.

En $T^{\delta}_{p,q}$, cada ciclo esencial de longitud $q'+1$ al añadirle dos aristas, se convierte en un conjunto de aristas de rango $q'+2$. Tal y como se muestra en la Figura 5.6*a* esto no es cierto para todo ciclo esencial de $K^0_{p',q'}$; en este grafo hay ciclos esenciales que al añadirle dos aristas dan lugar a conjuntos de rango $q'+1$. Como por hipótesis, el número de ciclos esenciales de longitud $q'+1$ es el mismo en ambos grafos tenemos que hay más conjuntos de aristas conteniendo un ciclo esencial de longitud $q'+1$ y dos aristas más en $T^{\delta}_{p,q}$ que en $K^0_{p',q'}$.

(3) Ciclos esenciales de longitud $q'+3$.

Por cada ciclo esencial de longitud $p=q'+1$ en $T^{\delta}_{p,q}$ tenemos $2\binom{p}{2}$ formas de añadir 2 aristas de modo que el conjunto de aristas que obtengamos siga siendo un ciclo esencial (Ver Figura 5.6b). Por tanto, hay $2q\binom{p}{2}$ ciclos esenciales de longitud $q'+3$.

Figura 5.7: Ciclos esenciales de longitud $q'+3$ en $K^0_{p',q'}$

En 2003, Márquez, Mier, Noy y Revuelta [29] demuestran que en $K^0_{p',q'}$ hay $4q'\binom{q'}{2}+4\binom{q'+2}{3}$ ciclos esenciales de longitud $q'+3$. (Ver Figura 5.7). Como $p=q'+1$ y $q=4q'$, tenemos que el número de ciclos esenciales de longitud $q'+3$ es mayor en $T^{\delta}_{p,q}$ que en $K^0_{p',q'}$.

Caso 1.4. $l_{T^{\delta}_{p,q}}=p=q+\delta$, $l_{K^0_{p',q'}}=p'$ con $q'+1>p'$.

$p=p'$ y $pq=p'q'$, entonces $q=q'$. Además, $q'+1>p'=p=q+\delta=q'+\delta$ luego $\delta<1$. Obtenemos una contradicción porque $\delta\geq 1$.

Caso 1.5. $l_{T^{\delta}_{p,q}} = p = q + \delta$, $l_{K^0_{p',q'}} = p' = q' + 1$.

Si la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud coincidiera en ambos grafos, tendríamos $p = p'$, $q = q'$, $\delta = 1$ y $q + pq = 5q'$ por tanto $p = 4$.

Caso 1.6. $l_{T^{\delta}_{p,q}} = p = q + \delta$, $l_{K^0_{p',q'}} = q' + 1$ con $p' > q' + 1$.

$$q' + 1 = p = q + \delta \quad \Rightarrow \quad 4q' = q + p\binom{q'}{\delta}$$

$$q' = \binom{q'}{q'-1} < \binom{q'}{\delta} \quad \Rightarrow \quad 4q' < p\binom{q'}{\delta}$$

Caso 1.7. $l_{T^{\delta}_{p,q}} = q + \delta < p$, $l_{K^0_{p',q'}} = p' = q' + 1$.

$$q' + 1 = p' = q + \delta \quad \text{y} \quad 5q' = p\binom{q+\delta-1}{\delta}$$

Si $\delta = 1$ entonces $q = q'$, $p = p' = q + \delta < p$ luego $\delta > 1$.

$$p > 5 \quad \text{y} \quad \binom{q+\delta-1}{\delta} = \binom{q'}{\delta} > q' \quad \Rightarrow \quad 5q' < p\binom{q+\delta-1}{\delta}$$

Caso 1.8. $l_{T^{\delta}_{p,q}} = q + \delta < p$, $l_{K^0_{p',q'}} = q' + 1 < p'$.

$q' + 1 = q + \delta$. Podemos suponer $\delta > 1$ ya que si $\delta = 1$ entonces $q = q'$, $p = p'$ y no se produciría la igualdad del número de ciclos esenciales de longitud mínima. La contradicción en este caso se obtiene de forma similar a la del caso anterior pues tenemos que $4q' = p\binom{q'}{\delta}$.

Con estos casos podemos concluir que $T^{\delta}_{p,q}$ no es Tutte equivalente a $K^0_{p,q}$.

CASO 2: Supongamos $G \simeq T^{\delta'}_{p',q'}$.

$T(T^{\delta}_{p,q}; x, y) = T(G; x, y) = T(T^{\delta'}_{p',q'}; x, y)$ luego $l(T^{\delta}_{p,q}) = l(T^{\delta'}_{p',q'})$ y el número de ciclos esenciales de longitud mínima es el mismo en ambos grafos.

Caso 2.1. Supongamos que $l_{T^{\delta}_{p,q}} = p$, $l_{T^{\delta'}_{p',q'}} = p'$ con $p < q + \delta$ y $p' < q' + \delta'$.

Por el Lema 5.5, sabemos que el número de ciclos esenciales de longitud p de $T^{\delta}_{p,q}$ es q y el número de ciclos esenciales de longitud p' de $T^{\delta'}_{p',q'}$ es q', entonces $p = p'$ y $q = q'$. Si $\delta = \delta'$ entonces $G \simeq T^{\delta}_{p,q}$. Veamos que no puede ser $\delta \neq \delta'$, ya que en tal caso el número de conjuntos de aristas de rango $q + \delta' - 1$ y tamaño $q + \delta'$ es distinto en $T^{\delta}_{p,q}$ y $T^{\delta'}_{p,q}$. Esto supondría una contradicción, ya que este número es el coeficiente de $x^{q+\delta'-1}y^{q+\delta'}$ del polinomio generador de rango-tamaño; y por tanto $T(T^{\delta'}_{p,q}; x, y) \neq T(T^{\delta}_{p,q}; x, y)$. Supongamos $\delta' < \delta$. El procedimiento que vamos a seguir es análogo al del caso 1.1.

Definimos dos tipos de conjuntos de aristas de rango $q+\delta'-1$ y tamaño $q+\delta'$:

(1) Ciclos esenciales de longitud $q + \delta'$.

(2) Conjuntos de rango $q+\delta'-1$ y tamaño $q+\delta'$ que no sean ciclos esenciales.

Haciendo el recuento del número de conjuntos de aristas de cada tipo que hay en cada grafo, obtenemos la siguiente tabla:

Tipo	$T^{\delta}_{p,q}$	$T^{\delta'}_{p,q}$
1	n	$n + p\binom{q+\delta'-1}{\delta'}$
2	?	?

Demostrando que existe una biyección entre los conjuntos del tipo 2 de ambos grafos, tendremos probado que $T^{\delta}_{p,q}$ tiene menos conjuntos de aristas de rango $q + \delta' - 1$ y tamaño $q + \delta'$ que $T^{\delta'}_{p,q}$.

Sea A un subconjunto de aristas del tipo 2. Para cada $r \in [0, q-2]$ definimos la biyección φ_r entre los siguientes conjuntos:

$$\{A \subseteq A(T^{\delta'}_{p,q}) | r(A) = q + \delta' - 1, |A| = q + \delta', s(A) = r\}$$
$$\{A \subseteq A(T^{\delta}_{p,q}) | r(A) = q + \delta' - 1, |A| = q + \delta', s(A) = r\}$$

Si $A \subset A(T^{\delta'}_{p,q})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(i'+\delta-\delta',j')\} & \text{si} \quad j' = q-1, j = 0 \\ \{(i,j),(i',j')\} & \text{si} \quad j, j' \in [0,r] \\ \{(i+\delta-\delta',j),(i'+\delta-\delta',j')\} & \text{si} \quad r+1 \leq j, j' \leq q-1 \end{cases}$$

Caso 2.2. Supongamos que $l_{T^{\delta}_{p,q}} = p$, $l_{T^{\delta'}_{p',q'}} = p' = q' + \delta'$ con $p < q + \delta$.

Como $T(T^{\delta'}_{p',q'}; x, y) = T(T^{\delta}_{p,q}; x, y)$, por el Lema 5.4 se tiene que $p = p'$ y $q = q' + p' \binom{q' + \delta' - 1}{\delta'}$. Además, $pq = p'q'$ luego $q = q'$, y obtenemos una contradicción.

Caso 2.3. $l_{T^{\delta}_{p,q}} = p$, $l_{T^{\delta'}_{p',q'}} = q' + \delta'$ con $p < q + \delta$ y $q' + \delta' < p'$.

$$q' + \delta' = p \quad \Rightarrow \quad q = p' \binom{p-1}{\delta'}$$

$$pq = p'q' \quad \Rightarrow \quad p \binom{q' + \delta' - 1}{\delta'} = q' = p - \delta'$$

$\delta' < p - 1$ luego $p \binom{q' + \delta' - 1}{\delta'} > p$. La contradicción se produce por haber asumido que ambos grafos tienen el mismo polinomio de Tutte.

Por la hipótesis 2, tenemos que el caso $l_{T^{\delta}_{p,q}} = p = q + \delta$, $l_{T^{\delta'}_{p',q'}} = q' + \delta'$ con $q' + \delta' < p'$ no puede producirse. Con un proceso análogo al seguido en el caso 1.3 demostramos que el número de conjuntos de aristas de rango $q' + \delta' + 1$ y tamaño $q' + \delta' + 2$ es diferente en $T^{\delta}_{p,q}$ y $T^{\delta'}_{p',q'}$. Por tanto, el caso en que $l_{T^{\delta}_{p,q}} = q + \delta$ y $l_{T^{\delta'}_{p',q'}} = q' + \delta' < p'$ no es posible. En las demás comparaciones se llega a una contradicción de forma análoga a los casos ya especificados, por tanto hay un único caso que puede producirse: $l_{T^{\delta}_{p,q}} = p = q + \delta$, $l_{T^{\delta'}_{p',q'}} = p' = q' + \delta'$, lo cual implica $p = p'$, $q = q'$ y $\delta = \delta'$.

CASO 3. Si $G \simeq K^1_{p',q'}$, entonces $T(T^{\delta}_{p,q}; x, y) = T(K^1_{p',q'}; x, y)$. Como consecuencia del Lema 5.5 no puede ser $p' > q'$ ya que si $p, q \geq 6$ no podrían coincidir el número de ciclos esenciales de longitud mínima. Así, podemos considerar $p' \leq q'$.

Caso 3.1. $l_{T^{\delta}_{p,q}} = p < q + \delta$ y $l_{K^1_{p',q'}} = p' < q'$.

Como en el caso 1.1 definimos una biyección para demostrar que el número de conjuntos de aristas de rango $q - 1$ y tamaño q es diferente en cada grafo.

Si $A \subset A(T^{\delta}_{p,q})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(p-1-i'+\delta,j')\} & j'=q-1, j=0 \\ \{(i,j),(i',j')\} & j,j' \in [0,r] \\ \{(p-1-i+\delta,j),(p-1-i'+\delta,j')\} & j,j' \in [r+1,q-1] \end{cases}$$

Omitimos el resto de los casos pues es inmediato demostrar que no pueden coincidir la longitud mínima de los ciclos esenciales, el número de ciclos esenciales de dicha longitud y el número de vértices.

CASO 4. Supongamos $G \simeq K^2_{p',q'}$.

Las comparaciones con $K^2_{p',q'}$ son análogas a las del caso anterior. Especificamos la biyección en el caso $p = p' < q'$ y $q = q'$:

Si $A \subset A(T^\delta_{p,q})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(p-i'+\delta,j')\} & j'=q-1, j=0 \\ \{(i,j),(i',j')\} & j,j' \in [0,r] \\ \{(p-i+\delta,j),(p-i'+\delta,j')\} & j,j' \in [r+1,q-1] \end{cases}$$

CASO 5. Por último, suponemos que $G \simeq S_{p',q'}$. En los casos en los cuales $l_{T^\delta_{p,q}} = p < q+\delta$ y $l_{S_{p',q'}} = 2p' \leq q'$ ó $l_{T^\delta_{p,q}} = p = q+\delta$ y $l_{S_{p',q'}} = q'$ con $q' \leq p'$ tenemos que la longitud mínima de los ciclos esenciales, el número de ciclos esenciales de dicha longitud y el número de vértices no coinciden en ambos grafos. La contradicción es debida a que G y $S_{p',q'}$ no pueden tener el mismo polinomio de Tutte. Por la hipótesis 1, el caso $l_{T^\delta_{p,q}} = q+\delta < p$, $l_{S_{p',q'}} = q'$ con $q' \leq p'$ no se puede producir.

Caso 5.1. $l_{T^\delta_{p,q}} = p < q+\delta$, $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$.

Dado que $p = q'$, $q = p'$ y a la igualdad del número de ciclos esenciales de longitud mínima, obtenemos $q \geq 2^{q'-1}$. Esto nos conduce a una contradicción ya que:

$$p' \geq 2^{q'-1} \geq 2^{p'-1} \Rightarrow 2p' \geq 2^{p'}$$

Caso 5.2. $l_{T^{\delta}_{p,q}} = p < q + \delta$, $l_{S_{p',q'}} = q'$ con $q' \leq p'$

Utilizando las mismas ideas que en el caso 1.3 demostramos que el número de conjuntos de aristas de rango $q' + 1$ y tamaño $q' + 2$ es mayor en $T^{\delta}_{p,q}$ que en $S_{p',q'}$. Por motivos de brevedad damos un esquema de la demostración. Estos conjuntos se pueden clasificar en tres grupos: conjuntos normales de aristas, conjuntos de aristas conteniendo un ciclo esencial de longitud q' y dos aristas más, y ciclos esenciales de longitud $q' + 2$ (ver Figura 5.8). Probamos que $T^{\delta}_{p,q}$ tiene más conjuntos de cada tipo que $S_{p',q'}$. Como las ideas son análogas a las del caso 1.3, comentamos únicamente el último tipo.

Por cada ciclo esencial de longitud $p + 2 = q' + 2$ en $T^{\delta}_{p,q}$ tenemos $2\binom{p}{2}$ formas de añadir dos aristas de modo que el conjunto de aristas que obtengamos siga siendo un ciclo esencial. Por tanto, en $T^{\delta}_{p,q}$ hay $2q\binom{p}{2}$ ciclos esenciales de longitud $q' + 2$. En $S_{p',q'}$ hay aristas exteriores que generan ciclos esenciales en los cuales podemos añadir dos aristas de $\binom{q'}{2} + \binom{q'-1}{2}$ formas distintas, de tal modo que el resultado siga siendo un ciclo esencial (Figura 5.8b). Como $p = q'$ tenemos que el número de ciclos esenciales de longitud $q' + 2$ es mayor en $T^{\delta}_{p,q}$ que en $S_{p',q'}$.

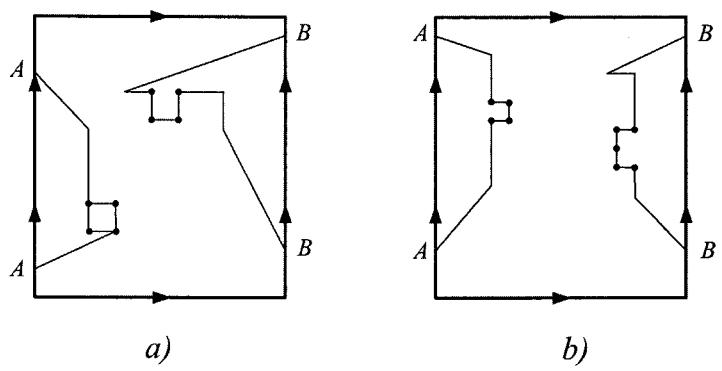


Figura 5.8: *a*) Conjunto de aristas en $S_{p',q'}$ con $p' \geq q'$ conteniendo un ciclo esencial de longitud q' y dos aristas más. b) Ciclos esenciales de longitud $q' + 2$ en $S_{p',q'}$.

Caso 5.3. $l_{T^{\delta}_{p,q}} = p = q + \delta$, $l_{S_{p',q'}} = 2p'$ con $q' \geq 2p'$.

Dado que $2p' = p = q + \delta$, tenemos que $q' = 2q$. Obtendremos una contradicción, suponiendo que se produce la igualdad del número de ciclos esenciales

de longitud mínima en ambos grafos. En este caso y en los siguientes vamos a utilizar la siguiente propiedad, si $n < m \leq \lfloor t/2 \rfloor \leq s < u$ entonces:

$$\binom{t}{n} < \binom{t}{m} \leq \binom{t}{\lfloor t/2 \rfloor} \geq \binom{t}{s} > \binom{t}{u}$$

Si $q + p\binom{q+\delta-1}{\delta} = q'(q'-p')\binom{2p'-1}{p'}$ entonces

$$q'(q'-p')\binom{2p'-1}{p'} < q + p\binom{2p'-1}{p'-1} = q + p\binom{2p'-1}{p'} <$$

$$< q + q'\binom{2p'-1}{p'} < q'\left(1 + \binom{2p'-1}{p'}\right) < q'(q'-p')\binom{2p'-1}{p'}$$

Caso 5.4. $l_{T^{\delta}_{p,q}} = p = q + \delta$, $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$.
$q' = p$ entonces $p' = q$. Supongamos $q + p\binom{q+\delta-1}{\delta} = 2p'\sum_{j=0}^{p'-1}\binom{q'-1}{j}$.
$\delta \leq p/2 = q'/2 \leq p' = q$. Si $\delta < q \leq p' - 1$ tenemos que:

$$q + p\binom{q+\delta-1}{\delta} = q + p\binom{q'-1}{\delta} = p' + q'\binom{q'-1}{\delta} <$$

$$< q'\left(1 + \binom{q'-1}{\delta}\right) \leq 2p'\left(1 + \binom{q'-1}{\delta}\right) < 2p'\sum_{j=0}^{p'-1}\binom{q'-1}{j}$$

Si $\delta = q$, $\binom{q'-1}{\delta} \leq \binom{q'-1}{q-1}$ porque $\lfloor (q'-1)/2 \rfloor = q - 1$. La diferencia entre este caso y el anterior es que la última acotación se obtiene de la siguiente forma:

$$2p'\left(1 + \binom{q'-1}{\delta}\right) \leq 2p'\left(1 + \binom{q'-1}{q-1}\right) < 2p'\sum_{j=0}^{p'-1}\binom{q'-1}{j}$$

Caso 5.5. $l_{T^{\delta}_{p,q}} = q + \delta < p$, $l_{S_{p',q'}} = 2p'$ con $2p' \leq q'$.

$$2p' = q + \delta \quad \text{y} \quad q'(q'-p')\binom{2p'-1}{p'} = p\binom{2p'-1}{\delta}$$

Como $\lfloor (2p'-1)/2 \rfloor = p'-1$ entonces $\binom{2p'-1}{\delta} \leq \binom{2p'-1}{p'}$. Por tanto:

$$q'(q'-p')\binom{2p'-1}{p'} \geq p'q'\binom{2p'-1}{p'} \geq pq\binom{2p'-1}{\delta} > p\binom{2p'-1}{\delta}$$

Caso 5.6. $l_{T^{\delta}_{p,q}} = q+\delta < p$, $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$.

$$q' = q+\delta \quad \text{y} \quad p\binom{q+\delta-1}{\delta} = 2p'\sum_{j=0}^{p'-1}\binom{q'-1}{j}$$

Si demostramos que $\binom{q'-1}{\delta} < 2\sum_{j=0}^{p'-1}\binom{q'-1}{j}$ obtendremos una contradicción.

$$q' \leq 2p' \Rightarrow \lfloor (q'-1)/2 \rfloor \leq p'-1 \Rightarrow \exists j_0 \in [0, p'-1], \binom{q'-1}{\lfloor (q'-1)/2 \rfloor} = \binom{q'-1}{j_0}$$

$$\binom{q'-1}{\delta} \leq \binom{q'-1}{\lfloor (q'-1)/2 \rfloor} < 2\sum_{j=0}^{p'-1}\binom{q'-1}{j}$$

□

Con los mismos argumentos que en el teorema anterior vamos a demostrar que la botella de Klein $K^0_{p,q}$ está unívocamente determinada por su polinomio de Tutte.

Teorema 5.7. *El grafo $K^0_{p,q}$ es Tutte único para $p, q \geq 6$.*

Demostración. Sean $p, q \geq 6$ y G un grafo tal que $T(G; x, y) = T(K^0_{p,q}; x, y)$. Por el Lema 5.3 y los Teoremas 5.1 y 5.6, sabemos que G tiene que ser isomorfo a uno y sólo uno de los siguientes grafos: $K^i_{p',q'}$ y $S_{p',q'}$ con $p'q' = pq$. Vamos a demostrar que G es isomorfo a $K^0_{p,q}$.

Supongamos que G es isomorfo a $K^0_{p',q'}$, entonces $l_{T^{\delta}_{p,q}} = l_{K^0_{p',q'}}$ y el número de ciclos esenciales de longitud mínima tiene que coincidir en ambos grafos. Basta estudiar el caso en que $l_{K^0_{p,q}} = p < q+1$, $l_{K^0_{p',q'}} = q'+1$ con $p' > q'+1$. Esto es debido a que, si $l_{K^0_{p,q}} = q+1 < p$ y $l_{K^0_{p',q'}} = p'$ con $p' < q'+1$ el razonamiento sería análogo y en el resto de los casos es fácil comprobar que el número de vértices y la longitud mínima de los ciclos esenciales no coincide en ambos grafos.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

Si $l_{K^0_{p,q}} = p < q+1$, $l_{K^0_{p',q'}} = q'+1$ con $p' > q'+1$ podemos probar que el número de conjuntos de aristas de rango $q'+2$ y tamaño $q'+3$ es distinto en $K^0_{p,q}$ y $K^0_{p',q'}$. Omitimos la demostración porque es análoga a la del caso 1.3 del Teorema 5.6.

Supongamos $G \cong K^1_{p',q'}$. Como p es par y p' impar, todos los casos en los cuales la longitud mínima de los ciclos esenciales en $K^0_{p,q}$ sea p y en $K^1_{p',q'}$ sea p' están probados. Por el Lema 5.5, sabemos que el número de ciclos esenciales de longitud mínima en $K^0_{p,q}$ es siempre mayor que uno, por tanto obtenemos una contradicción en todos aquellos casos en los cuales este número en $K^1_{p',q'}$ sea uno. Así, basta estudiar dos casos: $l_{K^0_{p,q}} = q+1 < p$, $l_{K^1_{p',q'}} = p' < q'$ y $l_{K^0_{p,q}} = q+1 < p$, $l_{K^1_{p',q'}} = p' = q'$. En el primero de ellos obtenemos una contradicción probando que el número de conjuntos de aristas de rango $p'+1$ y tamaño $p'+2$ es diferente en cada grafo (siguiendo el mismo razonamiento que en el caso 1.3 del Teorema 5.6). En el segundo caso mostramos que si $p' = q' = q+1$, $pq = p'q'$, p es par y p' es impar entonces q' tiene que ser par.

Por los Lemas 5.4 y 5.5, $T(K^0_{p,q}; x, y) \neq T(K^2_{p',q'}; x, y)$ si $p' > q'$. De forma análoga al caso 1.1 del Teorema 5.6 probamos que no puede ser $l_{K^0_{p,q}} = p < q+1$ y $l_{K^2_{p',q'}} = p' < q'$. Especificamos únicamente la biyección entre los conjuntos:

$$\{A \subseteq A(K^0_{p,q}) | r(A) = q, |A| = q+1, s(A) = r\}$$
$$\{A \subseteq A(K^2_{p',q'}) | r(A) = q, |A| = q+1, s(A) = r\}$$

Si $A \subset A(K^0_{p,q})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(i'+1,j')\} & \text{si} \quad j' = q-1, j = 0 \\ \{(i,j),(i',j')\} & \text{si} \quad j, j' \in [0,r] \\ \{(i+1,j),(i'+1,j')\} & \text{si} \quad r+1 \leq j, j' \leq q-1 \end{cases}$$

Por otra parte, si $l_{K^0_{p,q}} = q+1 < p$ y $l_{K^2_{p',q'}} = p' < q'$ demostramos (como en el caso 1.3 del Teorema 5.6) que el número de conjuntos de aristas de rango $p'+1$ y tamaño $p'+2$ es diferente en cada grafo. El resto de los casos, que se obtienen considerando todas las posibles combinaciones de las longitudes mínimas de los ciclos esenciales de $K^0_{p,q}$ y $K^2_{p',q'}$, no se pueden producir ya que la longitud mínima de los ciclos esenciales, el número de estos ciclos y el número de vértices no coinciden en ambos grafos.

Finalmente, supongamos $G \simeq S_{p',q'}$.

Caso 1. $l_{K^0_{p,q}} = p < q+1$ y $l_{S_{p',q'}} = q' \leq p'$.
Como hicimos en el caso 1.3 del Teorema 5.6 demostramos que hay un número distinto de conjuntos de aristas de rango $q'+1$ y tamaño $q'+2$ en $K^0_{p,q}$ y $S_{p',q'}$, por tanto, estos grafos no tienen el mismo polinomio de Tutte.

Caso 2. Si $l_{K^0_{p,q}} = p < q+1$ y $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$ obtenemos una contradicción de la siguiente forma:

$$p < q+1 \Rightarrow p' \leq q' < p'+1 \Rightarrow p' = q' \Rightarrow p = q = p' = q'$$

$$q \geq 2^{q'-1} = 2^{q-1}$$

Caso 3. $l_{K^0_{p,q}} = p = q+1$ y $l_{S_{p',q'}} = 2p' \leq q'$.
$2p' = p = q+1$ entonces $q' = 2q$. Como $5q = q'(q'-p')\begin{pmatrix} 2p'-1 \\ p' \end{pmatrix}$ obtendremos una contradicción si demostramos que $5q < q'(q'-p')$.

$$q'(q'-p') = 2q(2q-(p/2)) = 4q^2 - pq = 4q^2 - q(q+1) = q(3q-1) > q5$$

Caso 4. $l_{K^0_{p,q}} = q+1 < p$ y $l_{S_{p',q'}} = 2p' \leq q'$.
$2p' = q+1$.
$4q = q'(p'-q')\begin{pmatrix} 2p'-1 \\ p' \end{pmatrix} \geq 2p'^2\begin{pmatrix} 2p'-1 \\ p' \end{pmatrix} > 8p' > 4(2p'-1) = 4q.$

Al igual que en las comparaciones entre $K^0_{p,q}$ y $K^2_{p',q'}$, el resto de los casos no se pueden producir porque como $q \geq 6$ la longitud mínima de los ciclos esenciales, el número de estos ciclos y el número de vértices no puede coincidir en ambos grafos. □

Con el siguiente resultado demostramos que, $K^1_{p,q}$ también está unívocamente determinado por su polinomio de Tutte.

Teorema 5.8. *El grafo $K^1_{p,q}$ es Tutte único para $p, q \geq 6$.*

Demostración. Los argumentos utilizados en esta demostración son básicamente los mismos que los de los Teoremas 5.6 y 5.7. Debido a la Tutte unicidad de

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

$T^{\delta}_{p,q}$ y $K^0_{p,q}$ únicamente tenemos que probar que $T(G;x,y) \neq T(K^1_{p,q};x,y)$ con $G \in \{K^1_{p',q'}$ (excepto si $p = p'$ y $q = q'$), $K^2_{p',q'}, S_{p',q'}\}$. En cada caso vamos a suponer que G y $K^1_{p,q}$ son Tutte equivalentes y obtendremos una contradicción.

Caso 1. Si $G \simeq K^1_{p',q'}$ es fácil demostrar que el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud solo coinciden si $p = p'$ y $q = q'$.

Caso 2. Si $G \simeq K^2_{p',q'}$, por los Lemas 5.4 y 5.5 obtenemos una contradicción en todos aquellos casos en los cuales el número de ciclos esenciales de longitud mínima de $K^2_{p',q'}$ sea 2 o el número de ciclos esenciales de longitud mínima de $K^1_{p,q}$ sea 1. En los demás casos la contradicción se alcanza por el hecho de ser p impar y p' par.

Caso 3. Si $G \simeq S_{p',q'}$ podemos considerar $p \leq q$ porque si $p > q$ el número de ciclos esenciales de longitud mínima de $K^1_{p,q}$ es uno y en ningún caso puede coincidir con el número de ciclos esenciales de longitud mínima de $S_{p',q'}$.

Si $l_{K^1_{p,q}} = p < q$ y $l_{S_{p',q'}} = 2p' \leq q'$ ó $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$, por los Lemas 5.4 y 5.5 es fácil obtener una contradicción. Análogamente si $l_{K^1_{p,q}} = p = q$ y $l_{S_{p',q'}} = q'$ con $q' \leq p'$ ó $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$.

Si $l_{K^1_{p,q}} = p < q$ y $l_{S_{p',q'}} = q' \leq p'$ se demuestra (como en casos anteriores) que hay un número distinto de conjuntos de aristas de rango $q' + 1$ y tamaño $q' + 2$ en ambos grafos, por tanto no pueden tener el mismo polinomio de Tutte.

Si $l_{K^1_{p,q}} = p = q$ y $l_{S_{p',q'}} = 2p' \leq q'$, entonces $2p' = p = q$ y $q' = 2q$ luego

$$q + 1 = q'(q' - p')\binom{2p' - 1}{p'} = 2q(q' - p')\binom{2p' - 1}{p'} > q + 1$$

□

Siguiendo el mismo proceso que en los teoremas anteriores probamos que todo grafo Tutte equivalente al grafo $K^2_{p,q}$ tiene que ser isomorfo a él.

Teorema 5.9. *El grafo $K^2_{p,q}$ es Tutte único para $p, q \geq 6$.*

Demostración. Debido a los Teoremas 5.6, 5.7 y 5.8 basta demostrar que $K^2_{p,q}$ no es Tutte equivalente a un grafo G donde $G \in \{K^2_{p',q'}$ (excepto si $p = p'$ y $q = q'$), $S_{p',q'}\}$ y $pq = p'q'$.

Por los Lemas 5.4 y 5.5, $T(K^2_{p',q'}; x, y) \neq T(K^2_{p,q}; x, y)$ si $p \neq p'$ y $q \neq q'$ porque el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud solo coinciden si $p = p'$ y $q = q'$.

Si $G \simeq S_{p',q'}$ podemos suponer $p \leq q$ porque si $q < p$, $K^2_{p,q}$ tiene dos ciclos esenciales de longitud mínima y por el Lema 5.5, siendo $p, q \geq 6$, el número de ciclos esenciales de longitud mínima de $S_{p',q'}$ es mayor que dos.

Caso 1. Si $l_{K^2_{p,q}} = p < q$ y $l_{S_{p',q'}} = q' \leq p'$ se prueba como en casos anteriores que hay un número distinto de conjuntos de aristas de rango $q' + 1$ y tamaño $q' + 2$ en ambos grafos. Por tanto, estos dos grafos no pueden tener el mismo polinomio de Tutte en consecuencia, obtenemos una contradicción y G no puede ser isomorfo a $S_{p',q'}$.

Caso 2. Si $l_{K^2_{p,q}} = p = q$ y $l_{S_{p',q'}} = 2p' \leq q'$ entonces $2p' = p = q$ y $q' = 2q$ por tanto

$$q + 2 = q'(q' - p') \binom{2p' - 1}{p'} = 2q(q' - p') \binom{2p' - 1}{p'} > q + 2$$

Los demás casos no se pueden producir porque no coinciden la longitud mínima de los ciclos esenciales, el número de estos ciclos y el número de vértices en ambos grafos. □

Por último, demostramos un resultado equivalente a los Teoremas 5.6, 5.7, 5.8 y 5.9 para la familia de grafos $S_{p,q}$.

Teorema 5.10. *El grafo $S_{p,q}$ es Tutte único para $p, q \geq 6$ y $2^q \neq p' \binom{q' + \delta - 1}{\delta}$ para todo p', q' con $p'q' = pq$ y $\delta > 0$.*

Demostración. Supongamos que $S_{p,q}$ no es Tutte único. Entonces, por los Teoremas 5.6, 5.7, 5.8, 5.9 y el Lema 5.3, $S_{p,q}$ es Tutte equivalente a $S_{p',q'}$ con $p' \neq p$, $q' \neq q$ y $pq = p'q'$.

Caso 1. Si $l_{S_{p,q}} = 2p \leq q$ y $l_{S_{p',q'}} = q'$ con $p' \leq q' \leq 2p'$, por el Lema 5.4 tenemos que $q' = 2p$ y $q = 2p'$, entonces:

$$q(q-p)\binom{2p-1}{p} = 2p'(2p'-(q'/2))\binom{q'-1}{q'/2} > (2p'-(q'/2))\binom{q'-1}{q'/2} \geq$$

$$\geq (2p'-p')\binom{q'-1}{q'/2} = p'\binom{q'-1}{q'/2} > \sum_{j=0}^{p'-1}\binom{q'-1}{j}$$

Por tanto, el número de ciclos esenciales de longitud mínima es distinto en cada grafo y por el Lema 5.4, deducimos que $S_{p,q}$ no es isomorfo a $S_{p',q'}$.

Caso 2. Si $l_{S_{p,q}} = 2p \leq q$ y $l_{S_{p',q'}} = q' \leq p'$ entonces $2p = q'$ y $q = 2p'$.

$$2^{q'} = 2\sum_{j=0}^{2p-1}\binom{2p-1}{j} < 2{\cdot}2p\binom{2p-1}{p} \leq q\binom{2p-1}{p} < q(q-p)\binom{2p-1}{p}$$

Obtenemos una contradicción del hecho de suponer que el número de ciclos esenciales de longitud mínima es igual en ambos grafos.

El resto de los casos son análogos a los anteriores. □

A continuación ofrecemos varias tablas en las que el lector podrá ver fácilmente el motivo por el cuál dos grafos localmente cuadriculados no isomorfos no tienen el mismo polinomio de Tutte. En cada una de las tablas mostramos la comparación de un grafo localmente cuadriculado con pq vértices con un grafo localmente cuadriculado con $p'q'$ vértices. En dichas tablas hemos indicado lo siguiente:

I significa que en los grafos correspondientes no pueden coincidir el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud.

II significa que aunque en los grafos correspondientes puedan coincidir las cantidades anteriores, dichos grafos no tienen el mismo número de conjuntos de aristas de rango r y tamaño $r+1$. En cada casilla especificaremos el valor de r.

Además hay que tener en cuenta que en el caso de $T^{\delta}_{p,q}$ existen dos condiciones en el enunciado de teorema a las que llamamos hipótesis 1 y 2.

Como ejemplo, si el lector quiere saber el motivo por el cuál $T^{\delta}_{p,q}$ con $p < q+\delta$ y $K^{0}_{p',q'}$ con $p' < q'+1$ no tienen el mismo polinomio de Tutte, basta con mirar el elemento de la fila 4, columna 2 de la primera tabla y encontrará $r = q$, lo cuál quiere decir que aunque pueden coincidir el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud (según los valores de p, q, δ, p' y q'), ambos grafos no tienen en ningún caso el mismo número de conjuntos de aristas de rango q y tamaño $q+1$.

	$T^{\delta}_{p,q}$	$p < q+\delta$	$p = q+\delta$	$p > q+\delta$
$T^{\delta'}_{p',q'}$	$p' < q'+\delta'$	$r = q+\delta'-1$	I	I
	$p' = q'+\delta'$	I	I	Hipótesis 2
	$p' > q'+\delta'$	I	Hipótesis 2	$r = q'+\delta'+1$
$K^{0}_{p',q'}$	$p' < q'+1$	$r = q$	I	I
	$p' = q'+1$	I	I	I
	$p' > q'+1$	$r = q'+2$	I	I
$K^{1}_{p',q'}$	$p' < q'$	$r = q-1$	I	I
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$K^{2}_{p',q'}$	$p' < q'$	$r = q-1$	I	I
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$S_{p',q'}$	$p' \leq q' \leq 2p'$	I	I	I
	$2p' \leq q'$	I	I	I
	$q' \leq p'$	$r = q'+1$	I	Hipótesis 1

	$K^0_{p,q}$	$p < q+1$	$p = q+1$	$p > q+1$
$K^0_{p',q'}$	$p' < q'+1$	I	I	$r = q'+2$
	$p' = q'+1$	I	I	I
	$p' > q'+1$	$r = q'+2$	I	I
$K^1_{p',q'}$	$p' < q'$	I	I	$r = p'+1$
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$K^2_{p',q'}$	$p' < q'$	$r = q$	I	$r = p'+1$
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$S_{p',q'}$	$p' \leq q' \leq 2p'$	I	I	I
	$2p' \leq q'$	I	I	I
	$q' \leq p'$	$r = q'+1$	I	I

	$K^1_{p,q}$	$p < q$	$p = q$	$p > q$
$K^1_{p',q'}$	$p' < q'$	I	I	I
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$K^2_{p',q'}$	$p' < q'$	I	I	I
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$S_{p',q'}$	$p' \leq q' \leq 2p'$	I	I	I
	$2p' \leq q'$	I	I	I
	$q' \leq p'$	$r = q'+1$	I	I

	$K^2_{p,q}$	$p < q$	$p = q$	$p > q$
$K^2_{p',q'}$	$p' < q'$	I	I	I
	$p' = q'$	I	I	I
	$p' > q'$	I	I	I
$S_{p',q'}$	$p' \leq q' \leq 2p'$	I	I	I
	$2p' \leq q'$	I	I	I
	$q' \leq p'$	$r = q' + 1$	I	I

	$S_{p,q}$	$p \leq q \leq 2p$	$2p \leq q$	$q \leq p$
$S_{p',q'}$	$p' \leq q' \leq 2p'$	I	I	I
	$2p' \leq q'$	I	I	I
	$q' \leq p'$	I	I	I

Para finalizar, queremos resaltar la dificultad que supone establecer todas las comparaciones dos a dos entre los grafos localmente cuadriculados y el hecho de que con todas estas comparaciones podemos por primera vez dar una gran familia de grafos unívocamente determinada por su polinomio de Tutte.

CAPÍTULO 6

Teselaciones hexagonales y Grafos Localmente C_6

Los tres resultados principales que demostramos en este capítulo son: el teorema de clasificación de las teselaciones hexagonales, el teorema de clasificación de los grafos localmente C_6 y la relación de menor existente entre estas dos familias de grafos y los grafos localmente cuadriculados, cuya Tutte unicidad hemos estudiado en el capítulo anterior. Para llegar a estos resultados estudiamos distintos invariantes de grafos, tales como el número cromático, los ciclos esenciales de longitud mínima y la vértice transitividad.

6.1. Introducción

Un grafo conexo y cúbico, G, es una *teselación hexagonal* si la longitud mínima de cualquier ciclo de G es 6 y dada C una colección de 6−ciclos, todo 2−camino de G está contenido exactamente en un ciclo de C. A esta última condición la llamaremos *condición de* 2−*camino*. Es importante observar que las teselaciones hexagonales son grafos simples, es decir, no contienen lazos ni aristas múltiples y que cada vértice pertenece exactamente a tres ciclos de longitud 6 (hexágonos). A estos tres hexágonos nos referiremos cuando hablemos de la estructura local alrededor de un vértice dado (Figura 6.1). A cada hexágono de la teselación le llamaremos una *célula*.

En el estudio de las teselaciones hexagonales nos planteamos fundamentalmente dos objetivos: su clasificación y demostrar su Tutte unicidad. El primero de estos objetivos lo estudiamos en este capítulo y el segundo en el capítulo siguiente.

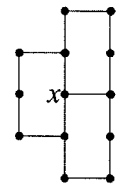


Figura 6.1: Estructura hexagonal alrededor de x

Para dar una clasificación de las teselaciones hexagonales, en primer lugar describimos todas las estructuras necesarias para construir dichos grafos. Algunas de estas estructuras, tales como, el cilindro hexagonal y el doble circuito hexagonal de Möbius fueron definidas por Thomassen en 1991 [35]. En este trabajo realizado por Thomassen hay un primer intento de clasificación de las teselaciones hexagonales. Nosotros damos una clasificación completa de estos grafos, añadiendo dos familias al teorema de clasificación dado por Thomassen en [35] y demostrando que incluyendo estas dos familias hemos considerado todos los casos posibles.

Por otra parte, introducimos diferentes variantes respecto al trabajo realizado por Thomassen. En primer lugar, damos definiciones diferentes de las estructuras básicas a partir de las cuales se construyen las teselaciones hexagonales. Obtenemos estas estructuras a partir de la cuadrícula plana borrando aristas. Este sistema nos va a reportar un gran beneficio en el siguiente capítulo cuando hablemos de Tutte unicidad, pues lo que hacemos es adaptar la notación de los grafos localmente cuadriculados a las teselaciones hexagonales. En segundo lugar, definimos los cilindros hexagonales torcidos, que son las estructuras necesarias para llegar a las dos familias de teselaciones hexagonales no incluidas por Thomassen.

Para clasificar las teselaciones hexagonales hacemos un estudio de distintos invariantes de grafos que nos permitan probar que las familias de teselaciones que definimos son no isomorfas entre sí. Estos invariantes son: el número cromático, los ciclos esenciales de longitud mínima y la vértice- transitividad. Demostramos que toda teselación hexagonal tiene número cromático dos o tres. Definimos el concepto de ciclo esencial en una teselación hexagonal y calculamos la longitud mínima y el número de ciclos esenciales de dicha longitud en las siete familias de teselaciones hexagonales que definimos. Este resultado será fundamental en el estudio de la Tutte unicidad. Por último, estudiamos cuales de las teselaciones definidas son vértice-transitivas.

Dos consecuencias interesantes del teorema de clasificación son, la arista-conectividad de estos grafos y el hecho de que todos ellos admitan una inmersión en el toro o en la botella de Klein.

Un grafo conexo G *se dice que es localmente* C_n si el conjunto de vecinos de cada vértice de G induce un subgrafo isomorfo a C_n. Los únicos grafos localmente C_3, C_4 y C_5 son, respectivamente el 1−esqueleto del tetraedro, octaedro e icosaedro [29]. En este capítulo clasificamos los grafos localmente C_6 y comprobamos que a diferencia de los grafos localmente C_3, C_4 y C_5 existen infinitos grafos no isomorfos entre sí que son localmente C_6 y están clasificados en siete familias. Para ello, hacemos uso del teorema demostrado por Thomassen [35] que establece una relación de dualidad entre las teselaciones hexagonales y los grafos localmente C_6.

Finalmente, queremos resaltar el último de los resultados que demostramos en este capítulo: la relación de menor existente entre los grafos localmente C_6, las teselaciones hexagonales y los grafos localmente cuadriculados. Hacemos uso del concepto de *emparejamiento perfecto* para demostrar que los grafos localmente cuadriculados son menores por contracción y eliminación de aristas dobles de las teselaciones hexagonales, y por la operación dual de los grafos localmente C_6. A parte de la importancia en sí de este resultado, ya que las operaciones de contracción y borrado de aristas son muy potentes en Teoría de Grafos, este resultado nos va a permitir en el próximo capítulo demostrar que, al igual que los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6 son localmente orientables.

6.2. Clasificación de las Teselaciones Hexagonales

Para obtener la clasificación de las teselaciones hexagonales, definimos en primer lugar todas las estructuras necesarias para dicha clasificación. Estas estructuras las obtenemos, dependiendo del caso, borrando ó añadiendo aristas y eliminando vértices en una cuadrícula plana. Esto nos va a permitir mantener la etiquetación de los vértices seguida en el capítulo anterior, relativo a los grafos localmente cuadriculados, lo cual nos será muy útil en la última sección de este capítulo para establecer la relación entre las teselaciones hexagonales, los grafos localmente cuadriculados y los grafos localmente C_6.

Sea $H = P_p \times P_q$ la cuadrícula $p \times q$, donde P_l es un camino con l vértices. Etiquetamos los elementos del grupo abeliano $\mathbb{Z}_p \times \mathbb{Z}_q$ de la forma habitual. Si a esta cuadrícula le añadimos las aristas $\{(j,0),(j,q-1) \mid 0 \leq j \leq p-1\}$ obtenemos

un *cilindro cuadriculado* $p \times q$.

Llamamos *muro hexagonal de longitud k y anchura m* al grafo obtenido borrando las aristas $\{(2i, 2j), (2i+1, 2j)\}$ y $\{(2i+1, 2j+1), (2i+2, 2j+1)\}$ con $0 \leq i \leq \lfloor (m-1)/2 \rfloor$, $0 \leq j \leq k-1$ en una cuadrícula $(m+1) \times 2k$. De forma equivalente, un *cilindro hexagonal de longitud k y anchura m* es el grafo resultante de borrar las mismas aristas en un cilindro cuadriculado $(m+1) \times 2k$ (Figura 6.2*a*). En esta estructura hay únicamente dos ciclos, llamados *ciclos periféricos*, cuyos vértices tienen de forma alternada valencia dos y tres. Cada uno de estos ciclos tienen $2k$ vértices, de los cuales k tienen valencia dos. A estos k vértices, fundamentales para clasificar las teselaciones hexagonales, los etiquetamos de la siguiente forma:

$$\begin{array}{llll} z_j = (0, 2j) & \text{y} \quad x_j = (m, 2j) & \text{con } 0 \leq j \leq k-1 & \text{si } m \text{ impar} \\ z_j = (0, 2j) & \text{y} \quad x_j = (m, 2j+1) & \text{con } 0 \leq j \leq k-1 & \text{si } m \text{ par.} \end{array}$$

Un *circuito cilíndrico hexagonal de longitud k* es un cilindro hexagonal de longitud k y anchura 1. Un *circuito hexagonal de Möbius de longitud k* es el grafo obtenido añadiendo las aristas $\{(0,0), (1, 2k-1)\}$ y $\{(1,0), (0, 2k-1)\}$ a un muro hexagonal de longitud k y anchura 1.

Al grafo resultante de borrar, en una cuadrícula $3 \times (2k+1)$, las aristas $\{(0, 2j+1), (1, 2j+1) | 0 \leq j \leq k-1\}$ y $\{(1, 2j), (2, 2j) | 0 \leq j \leq k\}$, y añadir posteriormente las aristas $\{(0,0), (2, 2k)\}$, $\{(1,0), (1, 2k)\}$ y $\{(2,0), (0, 2k)\}$ se le denomina *doble circuito hexagonal de Möbius* (Figura 6.2b)

Además del cilindro hexagonal hay otra estructura básica necesaria para definir las teselaciones hexagonales, *el cilindro torcido de longitud k y anchura m*. Vamos a definir dos cilindros torcidos: $CT_{k,m,1}$ si $k \leq m-2$ (Figura 6.3b) y $CT_{k,m,2}$ si $k \geq m+1$ (Figura 6.3*a*). Ambos grafos se obtienen a partir de una estructura (denotada malla hexagonal $m \times n$ en el Capítulo 2) que en este capítulo y para adaptarnos a la notación, vamos a llamar *escalera hexagonal de longitud k y anchura m* (Figura 6.2c) y es el grafo resultante de eliminar los siguientes vértices y aristas de una cuadrícula $(m+1) \times (2k+m)$:

$$\{(j, i) | 0 \leq j \leq m-2, 0 \leq i \leq m-2-j\}$$
$$\{(j, 2k+i) | 2 \leq j \leq m+1, m+1-j \leq i \leq m-1\}$$
$$\{\{(i, m-i+2j), (i+1, m-i+2j)\} | 0 \leq i \leq m-1, 0 \leq j \leq k-1\}$$

Definición 6.1. Sea $k \leq m-2$. El *cilindro hexagonal torcido* $CT_{k,m,1}$ (Figura 6.3b) es el grafo que se obtiene añadiendo dos vértices, etiquetados $(0, 2k+m)$ y

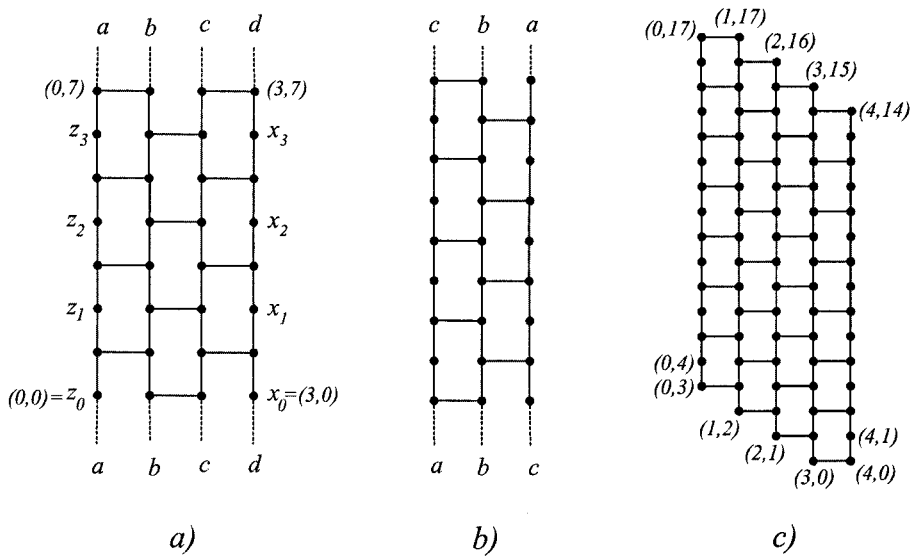


Figura 6.2: *a*) Cilindro hexagonal de longitud 4 y anchura 3 b) Doble circuito hexagonal de Möbius de longitud 9 c) Escalera hexagonal de longitud 7 y anchura 4

$(m-k-1, 3k+2)$, y las siguientes aristas a una escalera hexagonal de longitud k y anchura m:

$$\{(0, 2k+m), (0, 2k+m-1)\},\ \{(0, 2k+m), (k+1, m-k-2)\}$$
$$\{(m-k-1, 3k+2), (m-k-1, 3k+1)\},\ \{(m-k-1, 3k+2), (m, 0)\}$$
$$\{(j, 2k+m-j), (k+j+1, m-k-j-2)\} \text{ con } 1 \leq j \leq m-k-2$$

Definición 6.2. Sea $k \geq m+1$. Si $k \neq m+1$, llamamos *cilindro hexagonal torcido* $CT_{k,m,2}$ (Figura 6.3*a*) al grafo resultante de eliminar los vértices $(m+1, i)$ con $0 \leq i \leq 2(k-m)-3$ en una escalera hexagonal de longitud k y anchura $m+1$, y añadir las aristas:

$$\{(0, 2k+m), (m+1, 2(k-m-1)\} \text{ y } \{(0, 2k+m-(2j+1)), (m, 2(k-m-1)-(2j+2))\}$$

con $0 \leq j \leq k-m-2$. Si $k = m+1$, no eliminamos ningún vértice y añadimos una única arista, $\{(0, 2k+m), (m+1, 0)\}$.

Al igual que hablamos de ciclos periféricos en los cilindros hexagonales, podemos hablar de *ciclos periféricos en los cilindros hexagonales torcidos.*

Definición 6.3. Llamamos *ciclos periféricos del cilindro hexagonal torcido* $CT_{k,m,1}$ *con* $k \leq m-2$ y los denotamos C_1 y C_2 a los dos ciclos de longitud mínima que contienen a los siguientes vértices de valencia 2 (Figura 6.3b):

C_1: $x_j = (j, m-(j+1))$ y $z_j = (0, 2k+m-2j)$ con $0 \leq j \leq k$

C_2: $v_0 = (m-k-1, 3k+2)$, $v_{i+1} = (m-k+1, 3k-i)$, $w_0 = (m, 2k)$ y $w_{i+1} = (m, 2k-(2i+1))$ con $0 \leq i \leq k-1$

Definición 6.4. Llamamos *ciclos periféricos del cilindro hexagonal torcido* $CT_{k,m,2}$ *con* $k \geq m+1$ y los denotamos C_1 y C_2 a los dos ciclos de longitud mínima que contienen a los siguientes vértices de valencia 2 (Figura 6.3*a*):

C_1: $x_i = (m-i, i)$ y $z_i = (0, m+2i+1)$ con $0 \leq i \leq m$.

C_2: $w_i = (i+1, 2k+m-i)$ y $v_i = (m+1, 2k-(2i+1))$ con $0 \leq i \leq m$.

La diferencia fundamental con los ciclos periféricos de un cilindro hexagonal es que no todos los vértices de valencia 2 de los ciclos periféricos de los cilindros torcidos están a distancia dos. En $CT_{k,m,1}$, v_0 y v_1 están a distancia 4 y las parejas x_0, z_k y w_0, w_1 están a distancia 1. En $CT_{k,m,2}$, las parejas z_0, x_m y v_0, w_m están a distancia 1.

A fin de obtener teselaciones hexagonales, debemos añadir adecuadamente aristas entre los vértices de valencia dos de dos de las estructuras definidas: el cilindro hexagonal y el cilindro hexagonal torcido. En primer lugar, añadimos aristas entre los vertices de valencia dos de los ciclos periféricos de un cilindro hexagonal, H, de longitud k y anchura m, obteniendo las siguientes familias de grafos.

A) $H_{k,m,r}$ con $r, k, m \in \mathbb{N}$, $0 \leq r \leq \lfloor k/2 \rfloor$, $m \geq 2$, $k \geq 3$. Si $m = 1$ entonces $k > 3$ y $\lfloor k/2 \rfloor \geq r \geq 2$ (Figura 6.5*a*).

$$E(H_{k,m,r}) = E(H) \cup \{\{z_j, x_{j+r}\} | 0 \leq j \leq k-1\}$$

Hay un caso degenerado, llamado $H_{k',m',e}$ por Thomassen [35] que queremos comentar. Se obtiene a partir de un ciclo de longitud par k' añadiendo las adyacencias $\{z_i, z_{i+m'}\}$ tomando índices módulo m' y teniendo en cuenta la condición

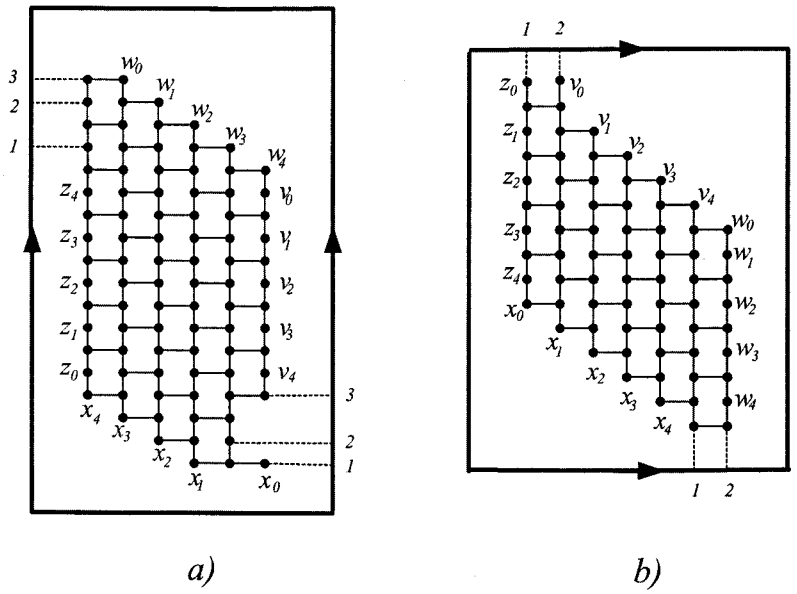


Figura 6.3: $a)$ $CT_{7,4,2}$ b) $CT_{4,6,1}$

de 2−camino, esto es, si z_i es adyacente a $z_{i+m'}$ entonces z_{i+1} es adyacente a z_l con $l \equiv (i+1)(\mathrm{mod}\, m')$ y $l \neq i+m'+1$. Este grafo es un tipo de espiral hexagonal y está tratado por Thomassen como un caso independiente [35], pero es el grafo degenerado $H_{(k'/2),0,m'}$ (Figura 6.4).

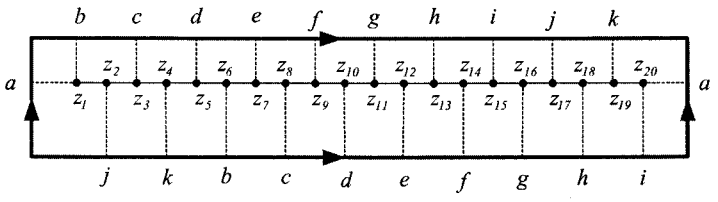


Figura 6.4: $H_{10,0,5}$

B) $H_{k,m,a}$ con $m \geq 2, \quad k \geq 3$ (Figura 6.5b).

$$E(H_{k,m,a}) = E(H) \cup \{\{z_0, x_1\}, \{z_1, x_0\}, \{z_i, x_{k+1-i}\} | 2 \leq 1 \leq k-1\}$$

BIBLIOTECA

C) $H_{k,m,b}$ con k par, m impar, $m \geq 3$, $k \geq 4$ (Figura 6.5c).

$$E(H_{k,m,b}) = E(H) \cup \{\{z_0, x_0\}, \{z_i, x_{k-1}\} | 1 \leq i \leq k-1\}$$

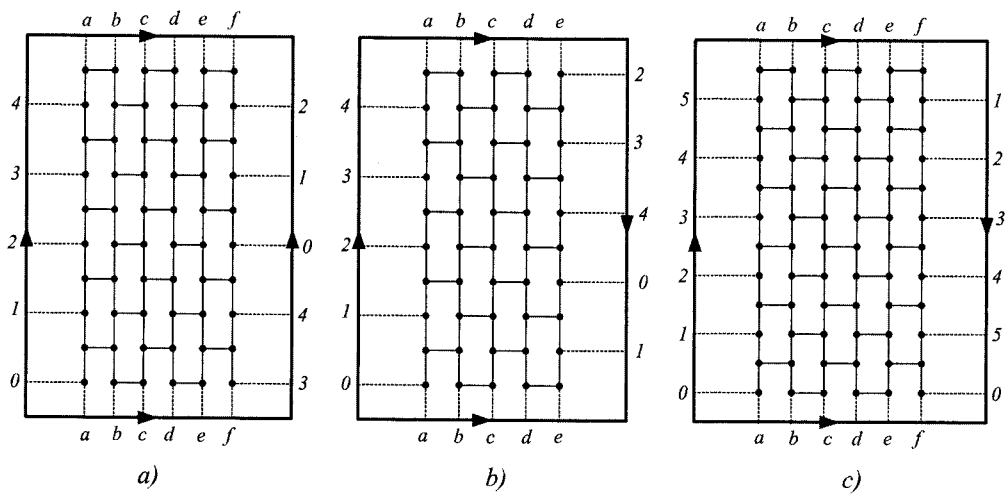


Figura 6.5: $a)$ $H_{5,5,r}$ b) $H_{5,4,a}$ c) $H_{6,5,b}$

D) $H_{k,m,c}$ con k par, $k \geq 6$, $m \geq 1$. (Figura 6.6a).

$$E(H_{k,m,c}) = E(H) \cup \{\{z_i, z_{i+k/2}\}, \{z_i, x_{i+k/2}\};\ 0 \leq i \leq (k/2) - 1\}$$

Tal y como mostramos en la Figura 6.6b los grafos pertenecientes a esta familia admiten una inmersión en la botella de Klein. Esta inmersión se obtiene borrando las aristas $\{(2i, 2j+1), (2i+1, 2j+1)\}$ y $\{(2i+1, 2j), (2i+2, 2j)\}$ con $0 \leq j \leq (k/2) - 1$ y $0 \leq i \leq m$ de una cuadrícula $(2m+2) \times k$. Posteriormente, añadimos las aristas $\{(0, 2i+1), (2m+1, 2i+1) | 0 \leq i \leq (k/2) - 1\}$ obteniendo dos ciclos periféricos, cuyos vértices de grado dos los etiquetamos $z_i = (i, 0)$ y $x_i = (i, k-1)$ con $0 \leq i \leq 2m+1$. Finalmente, añadimos el siguiente conjunto de aristas: $\{\{z_i, x_{2m+1-i}\} | 0 \leq i \leq 2m+1\}$.

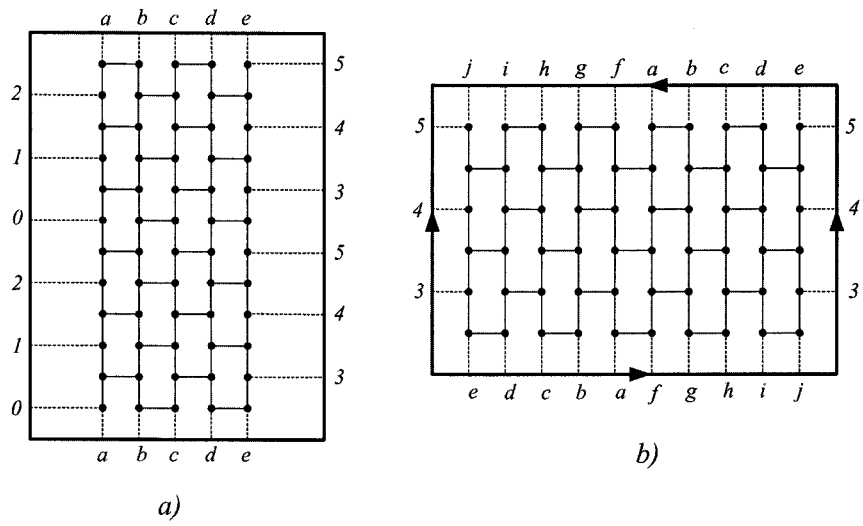


Figura 6.6: *a)* $H_{6,4,c}$ b) Inmersión de $H_{6,4,c}$ en la botella de Klein

E) $H_{k,m,f}$ con k impar, $m \geq 0$, $k \geq 7$ (Figura 6.7*a*).

Añadimos dos ciclos $w_0w_1 \ldots w_{k-1}w_0$ y $v_0v_1 \ldots v_{k-1}v_1$, a un cilindro hexagonal de longitud k y anchura m: $\{\{z_i, w_{2i}\}, \{x_i, v_{2i}\} | 0 \leq i \leq (k-1)/2\}$. Una teselación hexagonal se obtiene añadiendo aristas entre los vértices de grado dos de esta nueva estructura. Estas aristas son:

$$\{\{z_{(k+2i+1)/2}, w_{2i+1}\}, \{x_{(k+2i+1)/2}, v_{2i+1}\} | 0 \leq i \leq (k-1)/2\}$$

Todos los grafos de esta familia también admiten una inmersión en la botella de Klein (Figura 6.7b), obtenida borrando las mismas aristas que en el caso anterior a una cuadrícula $(2m+4) \times k$. Además se añaden las aristas:

$$\{(0, 2i+1), (2m+3, 2i+1) | 0 \leq i \leq (k-1/2) - 1\}$$

dando lugar a los dos ciclos periféricos, cuyos vértices de grado dos se etiquetan: $z_i = (i, 0)$ y $x_i = (i, k-1)$ con $0 \leq i \leq 2m+3$. Por último, añadimos las siguientes aristas: $\{\{z_0, x_0\}, \{z_i, x_{2m+4-i}\} | 0 \leq i \leq 2m+3\}$.

Las cinco familias de teselaciones hexagonales definidas son las obtenidas al añadir aristas entre los vértices de valencia dos de los ciclos periféricos de un cilindro hexagonal de longitud k y anchura m. Si añadimos aristas entre los vértices de valencia dos de los ciclos periféricos de los cilindros hexagonales torcidos, obtenemos dos nuevas familias de teselaciones hexagonales.

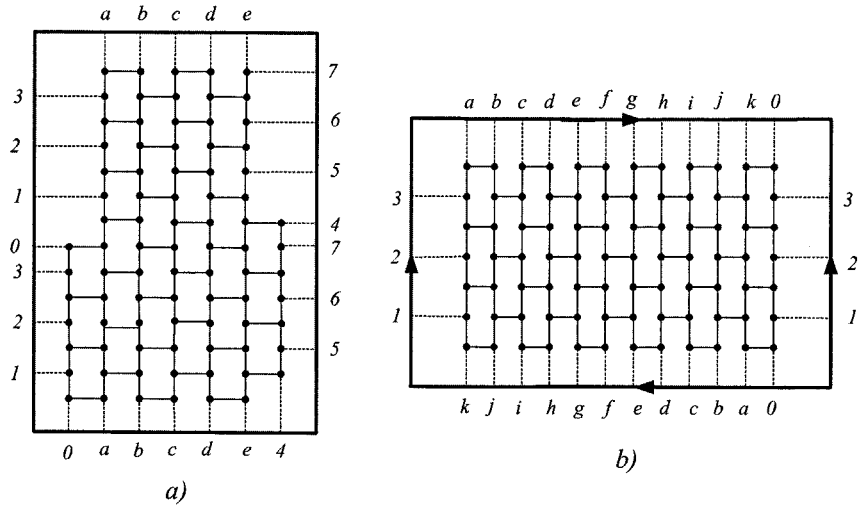


Figura 6.7: $a)$ $H_{7,4,f}$ b) Inmersión de $H_{7,4,f}$ en la botella de Klein

F) $H_{k,m,g}$ con $k \geq m+1$ y $m \geq 3$ (Figura 6.8a).

$$E(H_{k,m,g}) = E(CT_{k,m,2}) \cup \{\{z_i, w_i\}, \{x_i, v_i\};\ 0 \leq i \leq m\}$$

G) $H_{k,m,h}$ con $k \leq m-2$ y $k \geq 2$ (Figura 6.8b).

$$E(H_{k,m,h}) = E(CT_{k,m,1}) \cup \{\{z_i, x_i\} | 0 \leq i \leq k\} \cup \{v_i, w_i\};\ 0 \leq i \leq k-1\}$$

Es inmediato comprobar que todos los grafos que hemos definido son teselaciones hexagonales. El siguiente paso es demostrar que estas familias engloban todos los casos posibles. Para ello, vamos a demostrar, en primer lugar, que todas estas familias de grafos son no isomorfas entre sí, estudiando algunos invariantes de grafos: número cromático, vértice transitividad y ciclos esenciales de longitud mínima. En segundo lugar, demostraremos que cualquier teselación hexagonal tiene que ser isomorfa a una teselación perteneciente a alguna de las familias definidas en los apartados anteriores.

Todas las teselaciones hexagonales que hemos definido admiten una inmersión en el Toro o en la botella de Klein. De hecho, esto es un resultado general ya que si G es una teselación hexagonal con v vértices, e aristas y h hexágonos entonces

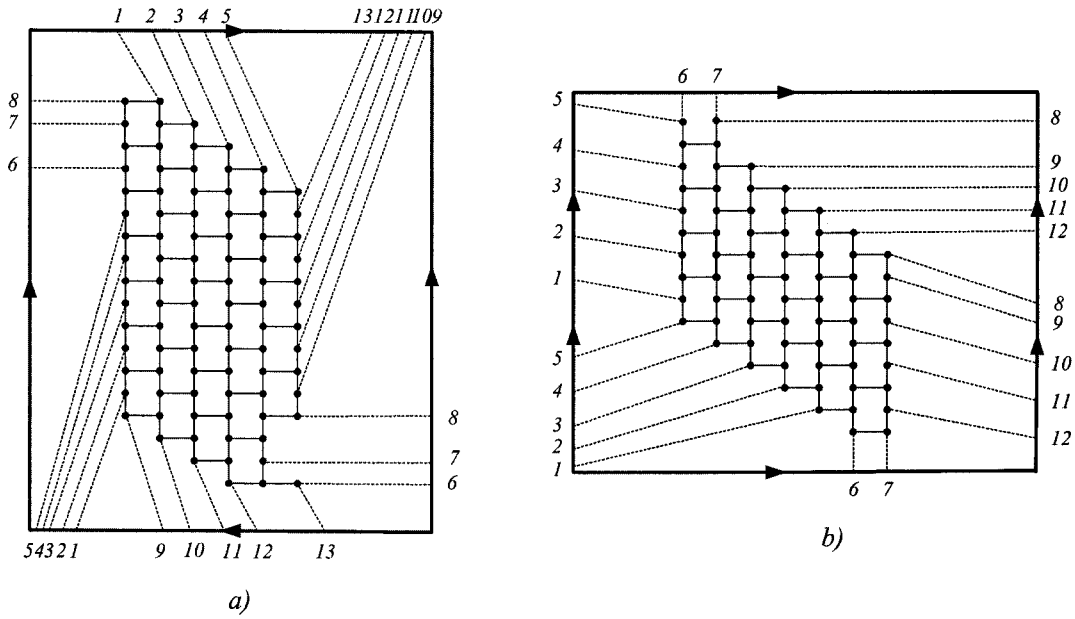


Figura 6.8: *a)* $H_{7,4,g}$ b) $H_{6,4,h}$

$v = 2h$ y $2a = 3v$, por tanto la característica de Euler de G es $v + h - a = 0$. Por el Lema 1.1, G admite una inmersión en el toro o en la botella de Klein.

Lema 6.1. *Si G es una teselación hexagonal de las definidas en los apartados A), B), C), D), E), F) y G), entonces el número cromático de G viene dado en la siguiente tabla:*

G	$H_{k,m,r}$	$H_{k,m,a}$	$H_{k,m,b}$	$H_{k,m,c}$	$H_{k,m,f}$	$H_{k,m,g}$	$H_{k,m,h}$
$\chi(G)$	2	2	2	3	3	3	2

Demostración. Demostramos en primer lugar que el número cromático de un cilindro hexagonal H de longitud k y anchura m es 2 para todo k y m. En efecto, H tiene $m + 1$ ciclos, C_0, C_1, ..., C_m, de longitud $2k$ (donde C_0 y C_m son los ciclos periféricos). Como la longitud de estos ciclos es par, cada uno de ellos admite una 2−coloración. Para obtener una 2−coloración de H basta considerar la 2−coloración $\{a, b, a, \ldots, b\}$ en C_i y $\{b, a, b, \ldots, a\}$ en C_{i+1}. Debido a la condición de 2−camino, cualquier 2−coloración de un cilindro hexagonal de longitud k y anchura m se caracteriza porque todos los vértices de valencia 2 de un mismo ciclo periférico tienen asociado el mismo color y el color contrario a los vértices de valencia 2 del otro ciclo periférico.

$H_{k,m,r}$, $H_{k,m,a}$ y $H_{k,m,b}$ se construyen añadiendo aristas entre los vértices de valencia 2 de los ciclos periféricos de un cilindro hexagonal de longitud k y anchura m. Como estos vértices tienen asociados colores distintos y el número cromático del cilindro hexagonal es 2 se tiene que el número cromático de $H_{k,m,r}$, $H_{k,m,a}$ y $H_{k,m,b}$ es 2.

Como $H_{k,m,c}$ se obtiene añadiendo aristas entre los vértices de valencia 2 de un mismo ciclo periférico de H, su número cromático es mayor que 2. Para obtener una 3−coloración de $H_{k,m,c}$ basta cambiar el color de los vértices $x_{k/2+1}, \ldots, x_{k-1}$ de C_m y $z_{k/2+1}, \ldots, z_{k-1}$ de C_0.

Para construir $H_{k,m,f}$ añadimos dos ciclos, $w_0w_1 \ldots w_{k-1}w_0$ y $v_0v_1 \ldots v_{k-1}v_0$ a un cilindro hexagonal de longitud k y anchura m. Estos ciclos tienen un número impar de vértices siendo w_0, w_{k-1} y v_0, v_{k-1} adyacentes, por tanto el número cromático de $H_{k,m,f}$ es mayor que 2. Obtenemos que el número cromático de este grafo es 3 cambiando el color de los vértices w_{k-1} y $z_{(k-1)/2+l}$ con $1 \leq l \leq (k-1)/2$ en C_0 y análogamente en C_m.

Para estudiar el número cromático de $H_{k,m,g}$ y $H_{k,m,h}$ tenemos que calcular el número cromático de los cilindros hexagonales torcidos $CT_{k,m,1}$ y $CT_{k,m,2}$. El cilindro $CT_{k,m,2}$ se construye a partir de una escalera hexagonal de longitud k y anchura $m+1$. Como cada escalera hexagonal admite una 2−coloración, podemos suponer que el vértice $(0, m)$ tiene el color a y por tanto $(0, 2k+m)$ y $(m, 0)$ tienen también el color a.

Debido a la condición de 2−camino, los vértices $(m, 2(k-m-1)-(2j+2))$ con $0 \leq j \leq k-m-2$ también tienen asignado el color a. Si $(0, 2k+m)$ tiene el color a entonces $(0, 2k+m-1)$ tiene el color b y por la condición de 2−camino, $(0, 2k+m-(2j+1))$ con $0 \leq j \leq k-m-2$ tiene el color b. Así, $(m, 2(k-m-1)$ tiene el color b y $(m+1, 2(k-m-1)$ el color a. Esto prueba que $CT_{k,m,2}$ admite una 2−coloración. Usando argumentos similares obtenemos que el número cromático de $CT_{k,m,1}$ es también dos.

$H_{k,m,h}$ se obtiene a partir de $CT_{k,m,1}$, añadiendo aristas entre los vértices de valencia 2 de un mismo ciclo periférico. Cada ciclo periférico de $CT_{k,m,1}$ tiene $2k+2$ vértices de valencia 2, de los cuales $k+1$ tienen asignados el mismo color y son adyacentes a los otros $k+1$ vértices que tienen asignados el color contrario; en consecuencia $H_{k,m,h}$ tiene número cromático 2.

Por otra parte, los dos ciclos periféricos de $CT_{k,m,2}$ tienen $2m+2$ vértices de

valencia 2 coloreados exactamente con los mismos colores. Como $H_{k,m,g}$ se construye añadiendo aristas entre estos dos ciclos periféricos, tiene número cromático mayor que 2. Supongamos que tenemos una 2−coloración del cilindro $CT_{k,m,2}$ tal que los vértices de valencia 2 de los ciclos periféricos, C_1 y C_2 tienen asociados los colores que se muestran en la tabla:

C_1:	x_0	x_1	...	x_m	z_0	...	z_m
	0	0	...	0	1	...	1
C_2:	v_0	v_1	...	v_m	w_0	...	w_m
	0	0	...	0	1	...	1

Introducimos un nuevo color, cambiando la coloración de C_2:

C_2:	v_0	v_1	...	v_m	w_0	...	w_m
	3	3	...	3	0	...	0

Además, para obtener una 3−coloración de $H_{k,m,g}$, tenemos que cambiar el color 0 de $m+1$ vértices de valencia de 3 de C_2 por el color 3. □

Una vez estudiados los números cromáticos de las teselaciones hexagonales definidas, pasamos a estudiar otro de los invariantes que nos van a permitir distinguir unas familias de otras: los ciclos esenciales de longitud mínima.

Definición 6.5. Dados dos ciclos C y C' en una teselación hexagonal G, decimos que C *es localmente homotópico a* C' si existe una célula H, con $C \cap H$ conexo y C' se obtiene a partir de C reemplazando $C \cap H$ por $H - (C \cap H)$. Una *homotopía* es una secuencia de homotopías locales. Un ciclo en G se dice *esencial* si no es homotópico a una célula. En caso contrario se dice que el ciclo es *contractible.*

Con los mismos argumentos que en el Lema 5.2 del Capítulo 5 se demuestra que la definición anterior es equivalente a la de ciclo esencial de un grafo inmerso en una superficie. Denotamos l_G a la longitud mínima de los ciclos esenciales de G. l_G es invariante bajo isomorfismo.

Lema 6.2. *Sea G una teselación hexagonal de los tipos definidos en A), B), C), D), E), F), G) entonces la longitud l_G y el número de ciclos esenciales de dicha longitud viene dado en las siguientes tablas:*

G	*número de ciclos esenciales de longitud mínima*	
$H_{k,m,r}$	$m+1$	*si* $k < m+1$
	$k\binom{m+1}{\lfloor (m+1)/2 \rfloor - r}$	*si* $r < \lfloor (m+1)/2 \rfloor < \lfloor k/2 \rfloor$
	$k\binom{r+\lfloor (m+1)/2 \rfloor}{m}$	*si* $\lfloor (m+1)/2 \rfloor \leq r \leq \lfloor k/2 \rfloor$
	$m+1+k\binom{m+1}{\lfloor (m+1)/2 \rfloor - r}$	*si* $k = m+1$
$H_{k,m,a}$	$m+1$	*si* $k < m+1$
	2^{m+1}	*si* $k > m+1$
	$2^{m+1}+m+1$	*si* $k = m+1$
$H_{k,m,b}$	$m+1$	*si* $k < m+1$
	$2\binom{m+1}{(m+1)/2} + 4\sum_{j=1}^{\lfloor (m+1)/4 \rfloor}\binom{m+1}{(m+1)/2-2j}$	*si* $k > m+1$
	$k+2\binom{k}{k/2} + 4\sum_{j=1}^{\lfloor k/4 \rfloor}\binom{k}{k/2-2j}$	*si* $k = m+1$ *y* $k > 4$
	18	*si* $k = 4$ *y* $m = 3$
$H_{k,m,c}$	$(k/2)\binom{2m+2}{m+1}$	*si* $4m+4 < k+1$
	$2k$	*si* $4m+4 > k+1$
$H_{k,m,f}$	$(k-1)/2\binom{2m+4}{m+2}$	*si* $4m+8 < k$
	2	*si* $4m+8 > k$
$H_{k,m,h}$	2^{k+1}	
$H_{k,m,g}$	*2*	*si* $k \leq 2m+1$ *y* k *impar*
	$2(k+2)$	*si* $k < 2m+1$ *y* k *par*

G	l_G	
$H_{k,m,r}$	$2k$	$si\ k \leq m+1$
	$2(m+1)$	$si\ r < \lfloor (m+1)/2 \rfloor < \lfloor k/2 \rfloor$
	$2(m+1+r-\lfloor (m+1)/2 \rfloor)$	$si\ \lfloor (m+1)/2 \rfloor \leq r \leq \lfloor k/2 \rfloor$
$H_{k,m,a}$	$min(2k, 2m+2)$	
$H_{k,m,b}$	$min(2k, 2m+2)$	
$H_{k,m,c}$	$min(k+1, 4m+4)$	
$H_{k,m,f}$	$min(k, 4m+8)$	
$H_{k,m,h}$	$2k+2$	
$H_{k,m,g}$	$2(k-m)-2\lfloor (m+1)/2 \rfloor+3$	$si\ k > 2m+1$
	$k+2$	$si\ k \leq 2m+1\ y\ k\ impar$
	$k+3$	$si\ k < 2m+1\ y\ k\ par$

Demostración. Tenemos dos formas distintas de unir j escaleras conteniendo i hexágonos cada una obteniendo dos estructuras que llamamos *escalera* $i \times j$ y *escalera desplazada* $i \times j$, mostradas en la Figura 6.9.

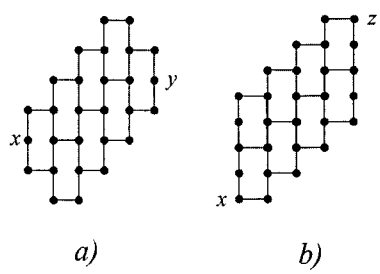


Figura 6.9: $a)$ Escalera desplazada 4×2 b) Escalera 4×2

A lo largo de toda la demostración usaremos que el número de caminos de longitud mínima entre x e y ó entre x y z en las escaleras $i \times j$ o en las escaleras desplazadas $i \times j$ es $\binom{i+j}{j}$ y la longitud de dichos caminos es $2(i+j)-1$.

BIBLIOTECA

Recordemos que todas las teselaciones hexagonales que hemos definido se obtienen añadiendo aristas entre los vértices de valencia 2 de un muro hexagonal o de una escalera hexagonal (excepto en el caso de $H_{k,m,h}$ en el cuál también añadimos dos vértices). Estas aristas son llamadas *aristas exteriores*. Está claro que cualquier ciclo esencial debe contener al menos una de estas aristas (en $H_{k,m,h}$ las aristas $\{(0,2k+m),(0,2k+m-1)\}$ y $\{(m-k-1,3k+2),(m-k-1,3k+1)\}$ no las consideramos aristas exteriores).

(1) $H_{k,m,r}$

Si $k < m+1$, cada una de las $m+1$ aristas exteriores de la forma $\{(i,0),(i,2k-1)\}$ determina un único camino de longitud mínima $2k-1$ uniendo sus extremos. Por tanto, el número de ciclos esenciales de longitud mínima es $m+1$ y la longitud es $2k$.

Si $r < \lfloor (m+1)/2 \rfloor < \lfloor k/2 \rfloor$, las k aristas de la forma $\{z_i, x_{i+r}\}$ dan lugar a los ciclos esenciales de longitud mínima. Los caminos de longitud mínima que unen los dos extremos de estas aristas tienen longitud $2m+1$ y cada una de ellas determina una escalera desplazada $r+(m+1)/2 \times (m-1)/2+1-r$ si m impar y $(m+2)/2+r\times(m-2)/2+1-r$ si m par; por tanto tenemos $k\begin{pmatrix} m+1 \\ \lfloor (m+1)/2 \rfloor - r \end{pmatrix}$ ciclos esenciales de longitud mínima $2m+2$.

Si $\lfloor (m+1)/2 \rfloor \leq r \leq \lfloor k/2 \rfloor$, los caminos de longitud mínima en el muro hexagonal que unen los dos extremos de una arista exterior de la forma $\{z_i, x_{i+r}\}$ constan de dos partes. La primera parte es un camino de longitud $2m+1$ atravesando el muro hexagonal y la segunda es un camino de longitud $2r-2\lfloor (m+1)/2 \rfloor$ a lo largo de un ciclo periférico del cilindro hexagonal. Cada una de estas k aristas exteriores genera una escalera desplazada $r-(m+1)/2 \times m$ si m impar ó $r-(m+2)/2 \times m$ si m par. Por tanto, tenemos $k\begin{pmatrix} r+\lfloor (m+1)/2 \rfloor \\ m \end{pmatrix}$ ciclos esenciales de longitud mínima $2(m+1+r-\lfloor (m+1)/2 \rfloor)$.

Si $k = m+1$ la longitud es $2k$ y el número de ciclos esenciales de dicha longitud es la suma de los obtenidos en el primer y segundo caso. En la Figura 6.10a mostramos un ejemplo. Los ciclos esenciales de $H_{5,4,2}$ los determinan las aristas exteriores, que son a,b,c,d,e,f y $0,1,2,3,4$. Los de longitud mínima son los ciclos de longitud 10 determinados por a,b,c,d,e y f. Las aristas $0,1,2,3$ y 4 dan lugar a ciclos esenciales de longitud mínima 12 y los caminos de dicha longitud que unen los extremos de cada una de estas aristas generan escaleras desplazadas 1×5 (marcada la correspondiente a la arista (1)).

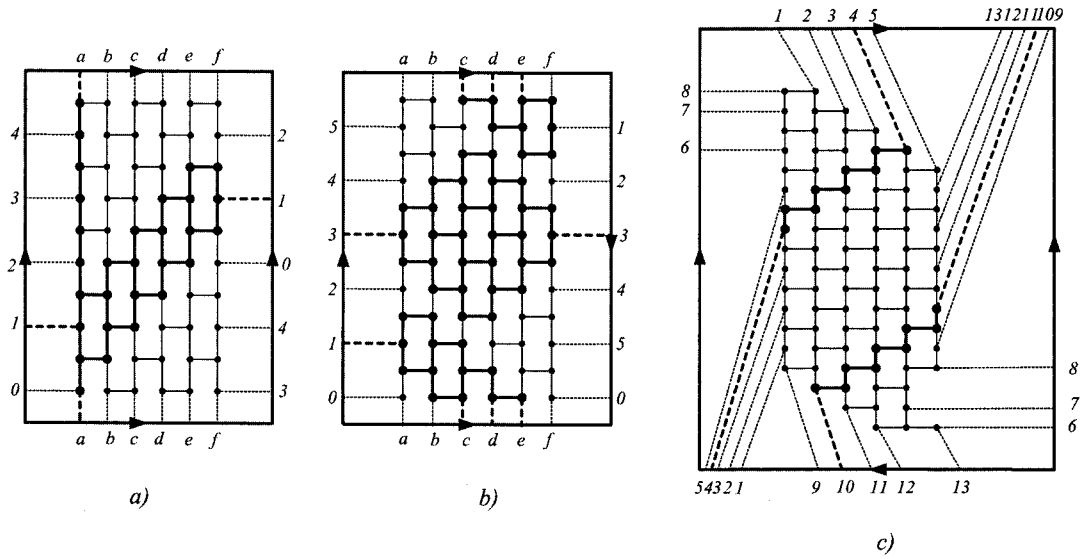


Figura 6.10: *a*) Ciclos esenciales en $H_{5,4,2}$ b) Ciclos esenciales en $H_{6,5,b}$ c) Ciclos esenciales en $H_{7,4,g}$

(2) $H_{k,m,a}$

Si $k < m+1$, las $m+1$ aristas de la forma $\{(i,0),(i,2k-1)\}$ dan lugar al mismo número de ciclos esenciales de longitud mínima que en el caso anterior.

Si $k > m+1$, los ciclos esenciales de longitud mínima los generan aristas exteriores del tipo $\{z_0,x_1\},\{z_1,x_0\},\{z_i,x_{k+1-i}\}$ con $2 \leq i \leq k-1$. Los camino que atraviesan el muro hexagonal y que unen los extremos de estas aristas tienen longitud mínima $2m+1$, por tanto la longitud $l_{H_{k,m,a}}$ es $2m+2$. Estos caminos generan escaleras desplazadas $i \times j$ con $i = m+1-j$ y $0 \leq j \leq m+1$ entendiéndose la escalera $m+1 \times 0$ ó $0 \times m+1$ como una escalera habitual de longitud $m+1$. Así, el número de ciclos esenciales de longitud mínima es $\sum_{j=0}^{m+1} \binom{m+1}{j} = 2^{m+1}$.

Si $k = m+1$ sumamos los ciclos esenciales obtenidos en los dos casos anteriores.

(3) $H_{k,m,b}$

Si $k < m+1$, tenemos la misma situación que en el caso anterior. Si $k > m+1$, las aristas exteriores del tipo $\{z_0,x_0\},\{z_i,x_{k-i}\}$ con $1 \leq i \leq k-1$ son las que determinan los ciclos esenciales de longitud mínima $2m+2$. Dos de estas aristas dan lugar a escaleras desplazadas $(m+1)/2 \times (m+1)/2$ y el resto agrupadas

BIBLIOTECA

de 4 en 4 dan lugar a escaleras desplazadas $(m+1)/2+2j \times (m+1)/2-2j$, $1 \leq j \leq \lfloor (m+1)/4 \rfloor$; por tanto el número de ciclos esenciales de longitud mínima es:

$$2\binom{m+1}{(m+1)/2}+4\sum_{j=1}^{\lfloor (m+1)/4 \rfloor}\binom{m+1}{(m+1)/2-2j}$$

Si $k=m+1$ y $k>4$ la longitud es $2k$ y el número de ciclos esenciales de longitud mínima se obtiene sumando las cantidades anteriores.

Si $k=4$ y $m=3$ hay $4+2+2\binom{4}{2}=18$ ciclos esenciales de longitud mínima.

En la Figura 6.10b se muestra un ejemplo. Todas las aristas exteriores de $H_{6,5,b}$ generan ciclos esenciales de longitud mínima 12, ya que $k=m+1$. En la figura hemos marcado las escaleras desplazadas 5×1 y 3×3 generadas por los caminos de longitud 12 que unen los extremos de las aristas 1 y 3 respectivamente.

(4) $H_{k,m,c}$

Para contar los ciclos esenciales de longitud mínima utilizamos la inmersión de este grafo en la botella de Klein.

Si $4m+4<k+1$, cada una de las $k/2$ aristas exteriores de la forma $\{(0,2i+1),(2m+1,2i+1)\}$ con $0 \leq i \leq (k/2)-1$ determina una escalera desplazada $(m+1)\times(m+1)$, en la cual la longitud mínima de los caminos de que unen los extremos de dicha arista es $4m+3$. Por tanto, el número de ciclos esenciales de longitud mínima es $(k/2)\binom{2m+2}{m+1}$ y la longitud de estos ciclos $4m+4$.

Si $4m+4>k+1$, hay exactamente 4 aristas exteriores que generan ciclos esenciales de longitud mínima $k+1$. Los caminos de longitud mínima que unen los dos extremos de cada una de estas aristas generan una escalera $(k/2)-1\times 1$; luego el número de ciclos esenciales de longitud mínima es $4\binom{(k/2)}{1}=2k$.

(5) $H_{k,m,f}$

En este caso también utilizamos la inmersión de este grafo en la botella de Klein. Si $4m+8<k$, razonando como en el caso anterior obtenemos $(k-1)/2$ aristas exteriores dando lugar a escaleras desplazadas $(m+2)\times(m+2)$. Si $4m+8>k$, solo tenemos dos ciclos esenciales de longitud k generados por las

aristas:

$$\{(2m+3,0),(2m+3,k-1)\} \text{ y } \{(m-1)/2,0),((m-1)/2,k-1)\}$$

(6) $H_{k,m,h}$

La arista $\{x_0, z_0\}$ determina un ciclo esencial de longitud mínima $2m+2$. Las aristas $\{(m-k-1,3k+2),(m,0)\}$, $\{(0,2k+m),(k+1,m-k-2)\}$ y $\{(j,2k+m-j),(k+j+1,m-k-j-2)\}$ con $1 \leq j \leq m-k-2$ no dan lugar a ningún ciclo esencial de longitud mínima. De las aristas restantes, hay $2k+1$ que generan ciclos esenciales de longitud mínima distintos y 2 que generan el mismo ciclo esencial de longitud mínima. Cada $k+1$ de estas aristas dan lugar a escaleras $i \times j$ con $i = k-j$ y $0 \leq j \leq k$ luego el número de ciclos esenciales de longitud mínima es $2\sum_{j=0}^{k} \binom{k}{j} = 2^{k+1}$.

(7) $H_{k,m,g}$

Si $k = 2m+1$ las aristas exteriores que dan lugar a los ciclos esenciales de longitud mínima son $\{z_m, w_m\}$ y $\{x_m, v_m\}$ porque son las únicas aristas cuyos extremos están unidos por un camino que atraviesa el cilindro torcido usando $m+1$ hexágonos y dos aristas por cada hexágono, luego $l_{H_{k,m,g}} = 2m+3$. Cada una de estas aristas exteriores genera una escalera desplazada $0 \times m+1$ por tanto hay dos ciclos esenciales de longitud $2m+3$.

Si $k < 2m+1$ y k par, hay 4 aristas exteriores que producen ciclos de longitud mínima, y son aquellas cuyos extremos están unidos por un camino que atraviesa el cilindro torcido hexagonal utilizando $k/2$ hexágonos, luego la longitud mínima es $2(k/2)+3$. Cada una de estas aristas exteriores genera una escalera desplazada $1 \times k/2$ por tanto hay $4(k/2+1)$ ciclos esenciales de longitud $k+3$.

Si $k < 2m+1$ y k impar, con razonamientos análogos al caso anterior se obtiene que hay 2 aristas exteriores que producen ciclos de longitud mínima y son las que permiten atravesar el cilindro torcido hexagonal utilizando $(k+1)/2$ hexágonos, luego la longitud mínima es $2((k+1)/2)+1 = k+2$. Cada una de estas aristas exteriores genera una escalera desplazada $0 \times (k+1)/2$ por tanto hay 2 ciclos esenciales de longitud $k+2$ (Figura 6.10c).

Si $k > 2m+1$ tenemos que la longitud mínima de los ciclos esenciales es la suma de las longitudes mínimas de dos caminos. El primero de ellos es un camino de longitud mínima que atraviesa la escalera hexagonal y el segundo es un camino en un ciclo periférico de $CT_{k,m,2}$. Calculando las longitudes de estos caminos tenemos que $l_{H_{k,m,g}} = 2(k-m) - 2\lfloor (m+1)/2 \rfloor + 3$. □

El último invariante de grafos que vamos a estudiar en este capítulo es la vértice-transitividad.

Lema 6.3. *Si G es una teselación hexagonal de las definidas en los apartados A), B), C), D), E), F) y G), entonces G es vértice-transitivo si y sólo si G es isomorfo a $H_{k,m,r}$ ó $H_{4,m,a}$ con m impar.*

Demostración. Un automorfismo de grafos conserva las adyacencias, es decir, si u, v son adyacentes entonces $\sigma(u), \sigma(v)$ también lo son. Por tanto, es evidente que si C es un ciclo, $\sigma(C)$ es un ciclo de la misma longitud. Además, los automorfismos de grafos llevan ciclos esenciales en ciclos esenciales. En efecto, si C no es esencial entonces es homotópico a una célula, luego C determina una región simplemente conexa y como los automorfismos de grafos llevan regiones simplemente conexas en regiones simplemente conexas, la imagen por σ de C tiene que ser un ciclo no esencial.

$H_{k,m,r}$ es vértice-transitivo, ya que dados dos vértices del grafo u y v existe una traslación que me lleva u en v. Análogamente con $H_{4,m,a}$. Los grafos pertenecientes a las demás familias no pueden ser vértice-transitivos porque por lo comentado anteriormente, si un grafo es vértice-transitivo entonces todos los vértices del grafo tienen que pertenecer al mismo número de ciclos esenciales de longitud mínima. Por el Lema 6.2 es fácil comprobar que todos los grafos pertenecientes a las familias $H_{k,m,a}$ con $k \neq 4$ ó m par, $H_{k,m,b}$, $H_{k,m,c}$, $H_{k,m,f}$, $H_{k,m,g}$ y $H_{k,m,h}$ no cumplen esta última condición. □

Lema 6.4. *Las siguientes familias de teselaciones hexagonales son no isomorfas:*

A) $H_{k,m,r}$ *con* $0 \leq r \leq \lfloor k/2 \rfloor$, $m \geq 2$ *y* $k \geq 3$. *Si* $m = 1$ *entonces* $k > 3$ *y* $\lfloor k/2 \rfloor \geq r \geq 2$. *Si* $m = 0$ *entonces* $k > 4$, $5 \leq r \leq k$ *y* r *impar.*

B) $H_{k,m,a}$ *con* $m \geq 2$, $k \geq 3$.

C) $H_{k,m,b}$ *con* k *par,* m *impar,* $m \geq 3$, $k \geq 4$.

D) $H_{k,m,c}$ *con* $m \geq 1$, k *par,* $k \geq 6$.

E) $H_{k,m,f}$ *con* k *impar,* $m \geq 0$, $k \geq 7$.

F) $H_{k,m,g}$ *con* $k \geq m + 1$, $m \geq 3$.

G) $H_{k,m,h}$ *con* $k < m - 1$, $k \geq 2$.

Demostración. Por los Lemas 6.1 y 6.3, basta probar que los grafos dados en los siguientes casos son no isomorfos:

(1) $H_{k,m,a}$ y $H_{k,m,b}$ no pueden ser isomorfos ya que todos los grafos de la primera familia contienen a lo sumo un doble circuito hexagonal de Möbius y los de la segunda contienen exactamente dos.

(2) Para probar que $H_{k,m,a}$ y $H_{k,m,h}$ son familias no isomorfas, vamos a suponer que para cada $k \geq 3$ y $m \geq 2$ existen k_1 y m_1 tales que $H_{k,m,a}$ y $H_{k_1,m_1,h}$ son isomorfos y así obtener una contradicción. Si ambos grafos son isomorfos, tienen igual número de vértices, de ciclos esenciales de longitud mínima y $l_{H_{k,m,a}} = l_{H_{k,m,h}}$, es decir:

$$2k(m+1) = 2(k_1+1)(m_1+1), \quad k_1 = m, \quad k = m_1 + 1$$

Por tanto, las longitudes mínimas de los ciclos no desorientadores de $H_{k,m,a}$ y $H_{k,m,h}$ son $2k$ y $4(m+1)$ respectivamente, y así llegamos a una contradicción. Con un razonamiento análogo se obtiene que $H_{k,m,b}$ y $H_{k,m,h}$ son familias no isomorfas.

(3) Por construcción las familias $H_{k,m,f}$ y $H_{k,m,c}$ no pueden ser isomorfas, ya que en general, no coincide el número de vértices de los grafos pertenecientes a cada familia. Dado $H_{k_1,m_1,f}$ con k_1 y m_1 impares no existen k, m con k par tales que $H_{k_1,m_1,f}$ y $H_{k,m,c}$ tengan el mismo número de vértices:

$$\left.\begin{array}{rcl} |V(H_{k_1,m_1,f})| & = & 2k_1(m_1+2) \\ |V(H_{k,m,c})| & = & 2k(m+1) \end{array}\right\} \Rightarrow \underbrace{k_1(m_1+2)}_{\text{impar}} = \underbrace{k(m+1)}_{\text{par}}$$

Además, la familia $H_{k,m,c}$ no está contenida en la familia $H_{k_1,m_1,f}$ ya que, por el Lema 6.2 es evidente que para cualquier valor de k y m no existen k_1 y m_1 tales que $H_{k,m,c}$ y $H_{k_1,m_1,f}$ tengan la misma longitud mínima de los ciclos esenciales y el mismo número de ciclos de dicha longitud.

(4) Por el Lemma 6.2, $H_{k,m,g}$ no es isomorfo ni a $H_{k,m,c}$ ni a $H_{k,m,f}$ porque la longitud l_G y el número de ciclos esenciales de longitud mínima no coincide en estos grafos. □

Teorema 6.5. *Si G es una teselación hexagonal con N vértices, entonces se verifica uno y sólo uno de los siguientes apartados:*

A) $G \simeq H_{k,m,r}$ *con* $N = 2k(m+1)$, $0 \leq r \leq \lfloor k/2 \rfloor$, $m \geq 2$, $k \geq 3$. *Si* $m = 1$ *entonces* $k > 4$ *y* $\lfloor k/2 \rfloor \geq r \geq 2$. *Si* $m = 0$ *entonces* $k > 4$, $5 \leq r \leq k$ *y* r *impar.*

B) $G \simeq H_{k,m,a}$ *con* $N = 2k(m+1)$, $m \geq 2$, $k \geq 3$.

C) $G \simeq H_{k,m,b}$ *con* $N = 2k(m+1)$, k *par*, m *impar*, $m \geq 3$, $k \geq 4$.

D) $G \simeq H_{k,m,c}$ *con* $N = 2k(m+1)$, $m \geq 1$, k *par*, $k \geq 6$.

E) $G \simeq H_{k,m,f}$ *con* $N = 2k(m+2)$, k *impar*, $m \geq 0$, $k \geq 7$.

F) $G \simeq H_{k,m,g}$ *con* $N = 2(m+1)(k+2)$, $k \geq m+1$, $m \geq 3$.

G) $G \simeq H_{k,m,h}$ *con* $N = 2(m+1)(k+1)$, $k < m-1$, $k \geq 2$.

Demostración. Los argumentos utilizados en esta demostración son básicamente los mismos que los utilizados por Thomassen en [35]. La diferencia fundamental entre ambas pruebas es que nosotros incluimos dos nuevas familias a la lista dada en el Teorema 3.1 de [35], $H_{k,m,g}$ y $H_{k,m,h}$. Además, consideramos las familias $H_{k,d}$ y $H_{k,m,e}$ de [35] como casos degenerados de las familias $H_{k,m,f}$ y $H_{k,m,r}$ respectivamente.

Sea G una teselación hexagonal con N vértices. Consideramos dos casos.

Caso 1: G contiene un circuito cilíndrico hexagonal de longitud k.

Tenemos dos opciones, extender dicho circuito a un cilindro hexagonal máximo, H, o a un cilindro hexagonal torcido, H'. Supongamos en primer lugar que extendemos el circuito de longitud k a cilindro hexagonal máximo de anchura m, H. Sean C_0 y C_m los ciclos periféricos de H, cuyos vértices de valencia 2 están etiquetados de la forma habitual: $C_0 : z_0, z_1, \ldots, z_{k-1}$ y $C_m : x_0, x_1, \ldots, x_{k-1}$. Si existen dos vértices $x \in V(C_0)$, $y \in V(C_m)$ tales que $\{x, y\} \in A(G)$ entonces por ser G una teselación hexagonal, tiene que cumplir la condición de 2−camino, es decir, cada vértice de C_0 que esté a distancia 2 de x es adyacente a un vértice de C_m que esté a distancia 2 de y. Repitiendo este proceso con todos los vértices de valencia 2 de C_0 y C_m obtenemos tres posibilidades distintas de añadir aristas que dan lugar a las familias de grafos $H_{k,m,r}$, $H_{k,m,a}$ y $H_{k,m,b}$. Es fácil comprobar que no hay más posibilidades, ya que el grafo que obtendríamos no sería una teselación hexagonal.

$$\text{Si } \{z_0, x_0\} \in A(G) \Rightarrow \begin{cases} \{z_1, x_1\} \in A(G) \Rightarrow \{z_i, x_i\} \in A(G) \Rightarrow G \simeq H_{k,m,r},\ r = 0. \\ \text{ó} \\ \{z_2, x_{k-1}\} \in A(G) \Rightarrow \{z_i, x_{k-i+1}\} \in A(G) \Rightarrow G \simeq H_{k,m,b}. \end{cases}$$

$$\text{Si } \{z_0, x_i\} \in A(G), i \neq 1 \Rightarrow \begin{cases} \{z_1, x_{i+1}\} \in A(G) \Rightarrow G \simeq H_{k,m,r},\ r > 0. \\ \text{ó} \\ \{z_1, x_{i-1}\} \in A(G) \Rightarrow G \simeq H_{k,m,a}. \end{cases}$$

Si ningún vértice de C_0 es adyacente a un vértice de C_m, entonces Thomassen demuestra que G se obtiene o bien, a partir de H uniendo vértices diametralmente opuestos de valencia dos de C_0 y análogamente de C_m, o añadiendo dos ciclos a los vértices de valencia 2 de cada ciclo periférico [35]. Así, G es de la forma $H_{k,m,c}$ ó $H_{k,m,f}$.

Supongamos ahora que extendemos el circuito cilíndrico hexagonal de longitud k a un cilindro hexagonal torcido, $CT_{(k/2)-1,m,1}$ ó $CT_{l,(k/2)-1,2}$, cuyos ciclos periféricos C_1 y C_2 están etiquetados como se muestra en la Figura 6.3. Si algún vértice de C_1 se une a algún vértice de C_2, entonces por la condición de 2−camino cada vértice de valencia 2 de C_1 tiene que ser adyacente a un vértice de valencia 2 de C_2. En un cilindro hexagonal torcido tenemos que dos vértices consecutivos de valencia 2 de un ciclo periférico están a distancia 2, excepto las parejas x_0z_k, w_0w_1 y v_0v_1 en $CT_{k,m,1}$ y z_0x_k, w_kv_0 en $CT_{k,m,2}$, que están a distancia 1 ó 4 dependiendo del caso (Figura 6.3). Estas parejas determinan la forma de unir vértices en un cilindro hexagonal torcido a fin de obtener una teselación hexagonal. Hay dos posibilidades, si z_i es adyacente a v_i y x_i a w_i estaríamos en el caso anterior. Si z_i es adyacente a w_i y x_i a v_i, G es isomorfo a $H_{l,(k/2)-1,g}$.

Si ningún vértice de C_1 es adyacente a un vértice de C_2, cada vértice de valencia 2 de un ciclo periférico tiene que ser adyacente a un vértice de valencia 2 del mismo ciclo periférico. Solo hay una posibilidad, z_i adyacente a x_i y v_i a w_i, por tanto G es isomorfo a $H_{k/2-1,m,h}$.

Caso 2: G no contiene un circuito cilíndrico hexagonal de longitud k.

En 1991, Thomassen [35] demuestra que una teselación hexagonal G que no contenga un circuito cilíndrico hexagonal verifica lo siguiente:

• Si G contiene un doble circuito hexagonal de Möbius entonces G es isomorfo al grafo degenerado $H_{k,d} \simeq H_{k,0,f}$.

• Si G no contiene un doble circuito hexagonal de Möbius entonces G es isomorfo a la teselación hexagonal degenerada $H_{k,m,e} \simeq H_{k/2,0,m}$. □

Corolario 6.6. *La arista conectividad de una teselación hexagonal es igual a 3.*

La demostración de este resultado es inmediata a partir de la demostración del teorema de clasificación de las teselaciones hexagonales.

6.3. Clasificación de los Grafos Localmente C_6

Un grafo conexo G *se dice que es localmente* C_6 si el conjunto de vecinos de cada vértice de G induce un subgrafo isomorfo a C_6. Como C_6 es un ciclo de longitud 6, la estructura alrededor de un vértice cualquiera x de G es la mostrada en la Figura 6.11. Los grafos localmente C_6 son simples y cada vértice pertenece exactamente a 6 triángulos.

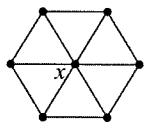


Figura 6.11: Estructura local alrededor de x

El objetivo de esta sección es clasificar los grafos localmente C_6, mostrando que existen infinitos grafos no isomorfos que son localmente C_6 y que todos ellos admiten una inmersión en el toro o en la botella de Klein.

El Teorema de clasificación de estos grafos se obtiene directamente a partir del Teorema de clasificación de las teselaciones hexagonales (Teorema 6.5) haciendo uso del siguiente resultado demostrado por Thomassen [35], en el cual se prueba que los grafos localmente C_6 son los duales geométricos de las teselaciones hexagonales (Figura 6.12).

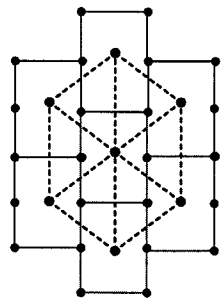


Figura 6.12: Estructura hexagonal y su dual

Teorema 6.7. [35] *Sea G un grafo $6-$regular conexo y C una colección de $3-$ciclos en G tal que, para cada vértice v de G, hay exactamente seis ciclos en C que contienen a v y su unión es una $6-$rueda cuyo centro es v. Supongamos además que G no tiene un subgrafo no plano de radio 1. Entonces G es el grafo dual de una teselación hexagonal.*

Teorema 6.8. *Sea G un grafo localmente C_6 con N vértices. Entonces se verifica uno y sólo uno de los siguientes apartados:*

A) $G \simeq (H_{k,m,r})^*$ *con* $N = k(m+1)$, $0 \leq r \leq \lfloor k/2 \rfloor$, $m \geq 2$, $k \geq 3$. *Si* $m = 1$ *entonces* $k > 4$ *y* $\lfloor k/2 \rfloor \geq r \geq 2$. *Si* $m = 0$ *entonces* $k > 4$, $5 \leq r \leq k$ *y* r *impar.*

B) $G \simeq (H_{k,m,a})^*$ *con* $N = k(m+1)$, $m \geq 2$, $k \geq 3$.

C) $G \simeq (H_{k,m,b})^*$ *con* $N = k(m+1)$, k *par*, m *impar*, $m \geq 3$, $k \geq 4$.

D) $G \simeq (H_{k,m,c})^*$ *con* $N = k(m+1)$, $m \geq 1$, k *par*, $k \geq 6$.

E) $G \simeq (H_{k,m,f})^*$ *con* $N = k(m+2)$, k *impar*, $m \geq 0$, $k \geq 7$.

F) $G \simeq (H_{k,m,g})^*$ *con* $N = (m+1)(k+2)$, $k \geq m+1$, $m \geq 3$.

G) $G \simeq (H_{k,m,h})^*$ *con* $N = (m+1)(k+1)$, $k < m-1$, $k \geq 2$.

Corolario 6.9. *La arista conectividad de un grafo localmente C_6 es igual a 6.*

Mostramos dos ejemplos de grafos localmente C_6 en la Figura 6.13.

6.4. Relación de Menor con los Grafos Localmente Cuadriculados

En esta sección establecemos una relación de menor entre los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6. Esta relación va a ser fundamental en el próximo capítulo a la hora de demostrar la orientabilidad local de las teselaciones hexagonales y de los grafos localmente C_6.

Lema 6.10. *Si G es un grafo localmente cuadriculado entonces $G^* = G$ si $G \in \{T^r_{p,q}, K^1_{p,q}, S_{p,q}\}$ y $(K^0_{p,q})^* = K^2_{p,q}$, siendo G^* el dual geométrico de G.*

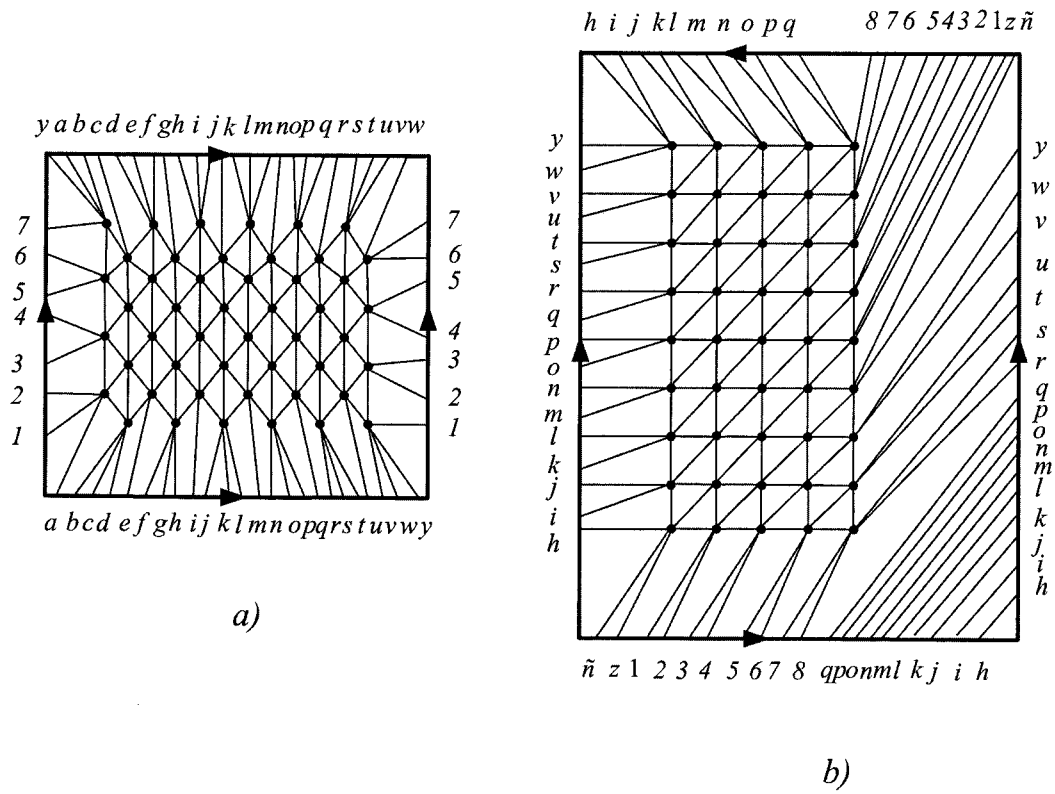


Figura 6.13: *a)* $(H_{4,11,0})^*$ b) $(H_{7,4,g})^*$

Demostración. Debido al Teorema de clasificación de los grafos localmente cuadriculados enunciado en el capítulo anterior, sabemos que si G es un grafo localmente cuadriculado con N vértices, entonces G es isomorfo a un único grafo de una de las siguientes familias:

$$T^r_{p,q}, \quad K^0_{p,q}, \quad K^1_{p,q}, \quad K^2_{p,q}, \quad S_{p,q}$$

Sea H la cuadrícula $p \times q$. H tiene $(p-1)(q-1)$ cuadrados (ciclos de longitud cuatro). Sustituimos cada cuadrado, por un vértice y dos vértices son adyacentes si sus correspondientes cuadrados comparten una arista. El grafo resultante es una cuadrícula $(p-1) \times (q-1)$. Para construir grafos localmente cuadriculados, añadimos aristas entre los vértices de valencia 2 y 3 de H, es decir, añadimos $p+q-1$ cuadrados. Por tanto, si G es un grafo localmente cuadriculado con pq vértices, entonces G^* tiene pq vértices y es obtenido añadiendo aristas entre los vértices de valencia 2 y 3 de una cuadrícula $p \times q$, que denotamos H'. Etiquetamos los vértices de H y H', (i,j) y $(i,j)^*$ respectivamente, con $0 \leq i \leq p-1$ y $0 \leq j \leq q-1$.

(1) Si $G \simeq T^\delta_{p,q}$, cada vértice $(0,j)^*$ con $0 \leq j \leq q-1$ está asociado al cuadrado de vértices $(0,j-1)$, $(1,j-1)$, $(0,j+1)$ y $(1,j+1)$ (tomando coordenadas módulo q para $j = q-1$), entonces tiene que ser adyacente al vértice del cuadrado $(0,j-1)$, $(p-1,j-1)$, $(0,j+1)$, $(p-1,j+1)$, que es $(p-1,j)^*$. Además, los vértices $(i,0)^*$ y $(i+\delta,q-1)^*$ con $0 \leq i \leq p-1$ tienen que ser adyacentes porque los cuadrados $(i,0)$, $(i+1,0)$, $(i+\delta,q-1)$, $(i+\delta+1,q-1)$ y $(i+\delta,q-1)$, $(i+\delta+1,q-1)$, $(i+\delta,q-2)$, $(i+\delta+1,q-2)$ (tomando coordenadas módulo p para $i = p-1$) comparten una arista. Por tanto, $V((T^\delta_{p,q})^*) \simeq V(T^\delta_{p,q})$ y $A((T^\delta_{p,q})^*) \simeq A(T^\delta_{p,q})$. (Figura 6.14*a*)

(2) Si $G \simeq K^1_{p,q}$, al igual que ocurría en el caso anterior cada vértice $(0,j)^*$ está asociado al cuadrado de vértices $(0,j-1)$, $(1,j-1)$, $(0,j+1)$ y $(1,j+1)$, entonces tiene que ser adyacente al vértice del cuadrado $(0,j-1)$, $(p-1,j-1)$, $(0,j+1)$, $(p-1,j+1)$, que es $(p-1,j)^*$.

El vértice $(i,0)^*$ con $0 \leq i \leq p-1$ está asociado al cuadrado $(i,0)$, $(i+1,0)$, $(p-i-1,q-1)$, $(p-i-2,q-1)$, luego tiene que ser adyacente al vértice del cuadrado $(p-i-1,q-1)$, $(p-i-2,q-1)$, $(p-i-1,q-2)$, $(p-i-2,q-2)$, es decir, adyacente a $(p-i-1,q-1)^*$. Por tanto, $V((K^1_{p,q})^*) \simeq V(K^1_{p,q})$ y $A((K^1_{p,q})^*) \simeq A(K^1_{p,q})$.

(3) Si $G \simeq S_{p,q}$ con $p \leq q$, cada vértice $(0,j)^*$ con $0 \leq j \leq p-1$ está asociado al cuadrado de vértices $(0,j)$, $(1,j)$, $(0,j+1)$ y $(1,j+1)$ (tomar coordenadas módulo p para $j = p-1$), luego tiene que ser adyacente al vértice del cuadrado $(0,j)$, $(j,q-1)$, $(0,j+1)$, $(j+1,q-1)$, que es $(j,q-1)^*$.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

Por otra parte, los vértices $(i,0)^*$ y $(p-1,q-p+i)^*$ con $0 \leq i \leq p-1$ tienen que ser adyacentes porque los cuadrados $(i,0)$, $(i+1,0)$, $(i,1)$, $(i+1,1)$ y $(i,0)$, $(i+1,0)$, $(p-1,q-p+i)$, $(p-1,q-p+i+1)$ comparten una arista.

Por último, los vértices $(0,i)^*$ y $(p-1,i-p)^*$ con $p \leq i \leq q-1$ son adyacentes porque sus cuadrados respectivos, $(0,i)$, $(0,i+1)$, $(1,i)$, $(1,1+i)$ y $(0,i)$, $(0,i+1)$, $(p-1,i-p)$, $(p-1,i+1-p)$ (tomando coordenadas módulo q para $i=q-1$) comparten una arista. Por tanto, $V((S_{p,q})^*) \simeq V(S_{p,q})$ y $A((S_{p,q})^*) \simeq A(S^{\delta}_{p,q})$. Si $G \simeq S_{p,q}$ con $p > q$ se demuestra de forma análoga.

(4) Si $G \simeq K^0_{p,q}$, razonando como en (1), cada vértice $(0,j)^*$ es adyacente a $(p-1,j)^*$. Los vértices $(p-1,0)^*$ y $(p-1,q-1)^*$ tienen que ser adyacentes porque los cuadrados $(p-1,0)$, $(0,0)$, $(p-1,q-1)$, $(0,q-1)$ y $(p-1,q-1)$, $(p-1,q-2)$, $(0,q-2)$, $(0,q-1)$ comparten una arista. Como por definición p es par, G^* es un grafo localmente cuadriculado y tiene dos adyacencias fijas entonces G^* es isomorfo a $K^2_{p,q}$. (Figura 6.14b) □

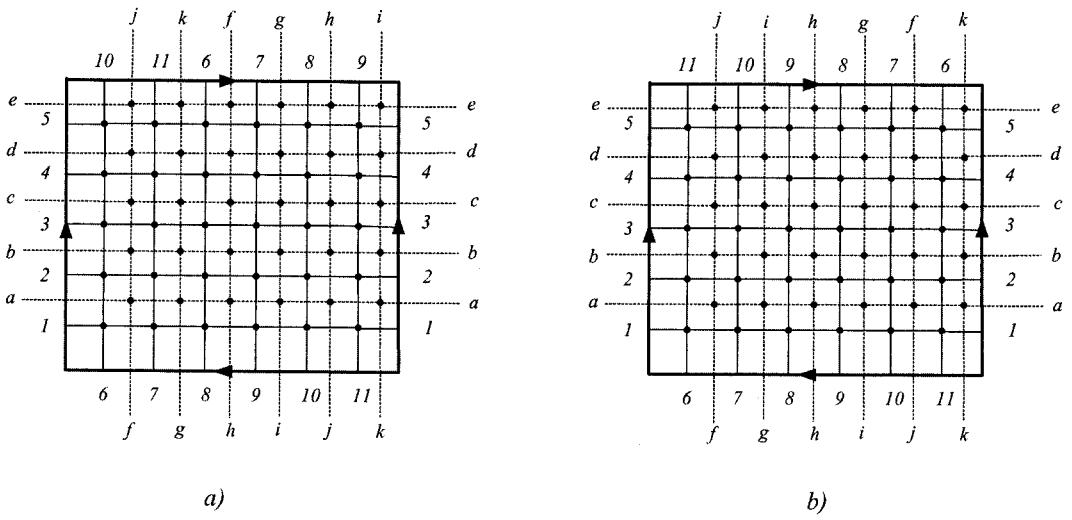


Figura 6.14: *a)* $T^2_{6,5}$ y su dual, $T^2_{6,5}$ (punteado) b) $K^0_{6,5}$ y su dual, $K^2_{6,5}$ (punteado)

Teorema 6.11. *Los grafos localmente cuadriculados son menores de las teselaciones hexagonales y por dualidad de los grafos localmente C_6.*

Demostración. Sea H una teselación hexagonal. Por el Teorema 6.5, H pertenece a una de las familias:

$$H_{k,m,r},\quad H_{k,m,a},\quad H_{k,m,b},\quad H_{k,m,c},\quad H_{k,m,f},\quad H_{k,m,g},\quad H_{k,m,h}$$

Para probar que los grafos localmente cuadriculados son menores de las teselaciones hexagonales, vamos a seleccionar un *emparejamiento perfecto* en cada una de las familias, esto es, una selección de aristas de $A(H)$ que contienen a todos los vértices del grafo. Entonces obtenemos el grafo localmente cuadriculado contrayendo las aristas de este emparejamiento y eliminando aristas dobles si fuera necesario. Hay exactamente dos casos en los cuales tenemos que eliminar aristas dobles, $H_{k,m,f}$ y $H_{k,m,h}$. El emparejamiento perfecto debe verificar algunas condiciones. Debe contener a lo sumo una arista exterior de H y dos aristas no incidentes por cada hexágono de H. Hay solo un caso en el cual el emparejamiento contiene cuatro aristas de $m+1$ hexágonos, este es $H_{k,m,f}$.

Sea C un cilindro hexagonal de longitud k y anchura m. Tomamos el siguiente emparejamiento perfecto, P, en C:

$$\{\{(i,2j),(i,2j+1)\} \mid 0 \leq i \leq m, 0 \leq j \leq k-1\}$$

Contrayendo las aristas pertenecientes a P obtenemos un cilindro cuadriculado $k \times (m+1)$. Este emparejamiento es un emparejamiento perfecto de $H_{k,m,r}$, $H_{k,m,a}$ y $H_{k,m,b}$ y ninguna arista exterior de estos grafos está contenida en P. Así, el resultado de contraer las aristas de P en $H_{k,m,r}$ es el grafo $T^r_{k,m+1}$, en $H_{k,m,a}$ es $K^0_{k,m+1}$ si k par y $K^1_{k,m+1}$ si k impar, y en $H_{k,m,b}$ es el grafo $K^2_{k,m+1}$.

Para seleccionar un emparejamiento en $H_{k,m,c}$ utilizamos la inmersión de este grafo en la botella de Klein, y tomamos el mismo emparejamiento perfecto que especificamos en el caso anterior. Contrayendo las aristas de P obtenemos el grafo $K^0_{2m+2,k/2}$.

El caso de $H_{k,m,f}$ es diferente. Tomamos la inmersión de este grafo en la botella de Klein, y los hexágonos de vértices $(0,l)(0,l+1)(0,l+2)(1,l)(1,l+1)(1,l+2)$ con $2 \leq l \leq 2k-5$. Entonces, un emparejamiento perfecto viene dado por: $P = P_1 \cup P_2$ con:

$$P_1 = \{\{(i,2j),(i,2j+1)\} \mid 0 \leq i \leq m, 0 \leq j \leq l/2\}$$
$$P_2 = \{\{(i,l+1),(i,l+2)\},\ldots\{(i,2k-2),(i,2k-1)\} \mid 0 \leq i \leq m\}$$

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

Si las aristas de P son contraídas eliminando las aristas dobles resultantes, obtenemos $K^2_{2m+4,(k-1)/2}$. Como ilustración de estas operaciones ver el ejemplo desarrollado en la Figura 6.16. En este ejemplo, empezamos con el grafo $H_{7,4,f}$ y seleccionamos un emparejamiento P, formado por las aristas punteadas. Contrayendo estas aristas y eliminando las aristas dobles obtenemos el grafo $K^2_{12,3}$.

En una escalera hexagonal de longitud k y anchura m, consideramos las siguientes aristas para el emparejamiento:
$\{(0,m-1),(0,m)\}$, $\{(0,m+1),(0,m+2)\},\ldots,\{(0,2k+m-3),(0,2k+m-2)\}$, $\{(i,m-(i+1)),(i,m-i)\}$, $\{(i,m-(i+1)+2),(i,m-i+2)\},\ldots,\{(i,2k+m-(i+1)),(i,2k+m-i)\}$ con $1 \leq i \leq m$.

Para obtener un emparejamiento perfecto en $H_{k,m,h}$ añadimos las aristas $\{(0,2k+m-1),(0,2k+m)\}$ y $\{(m-k-1,3k+2),(m,0)\}$. Contrayendo las aristas de P y eliminando las aristas dobles resultantes de la contracción, obtenemos el grafo localmente cuadriculado $S_{m+1,k+1}$ con $k < m$.

Si consideramos una selección similar de aristas en una escalera hexagonal de longitud k y anchura $m+1$, y añadimos $\{(0,2k+m),(m+1,2(k-m-1))\}$ conseguimos un emparejamiento perfecto de $H_{k,m,g}$ cuya contracción da lugar al grafo $S_{m+1,k+2}$ con $k \geq m+1$. En la Figura 6.17 se muestran estas operaciones.

En los grafos localmente C_6, seguimos el mismo procedimiento que para las teselaciones hexagonales. En cada caso tomamos el conjunto de aristas duales de las aristas pertenecientes al emparejamiento P. (Figura 6.15).

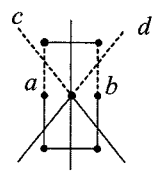


Figura 6.15: c y d son las aristas duales de a y b respectivamente

Si G es un grafo localmente cuadriculado obtenido a partir de la contracción de las aristas de P y eliminación de las aristas dobles resultantes en una teselación hexagonal H, entonces G^* se obtiene por el borrado en un grafo localmente C_6, H^*, del conjunto P^* de aristas duales asociadas al emparejamiento perfecto P y la contracción del conjunto de aristas duales asociadas a las aristas dobles resultantes por la contracción de P. Por los Teoremas 6.5 y 6.8 y el Lema 6.10,

todos los casos quedan perfectamente determinados. En las Figuras 6.16 y 6.17 mostramos dos ejemplos. En la Figura 6.16, empezamos por $H^*_{7,4,f}$ seleccionando las aristas duales de aquellas pertenecientes al emparejamiento perfecto de $H_{7,4,f}$. Aplicando las operaciones de menor obtenemos $K^0_{12,3}$, que es el grafo dual de $K^2_{12,3}$. En la Figura 6.17, eliminamos las aristas duales de aquellas pertenecientes al emparejamiento de $H_{7,4,g}$ obteniendo $S_{4,9}$.

Para concluir:

Teselación Hexagonal	Menor por contracción y eliminación de arstas dobles
$H_{k,m,r}$	$T^r_{k,m+1}$
$H_{k,m,a}$	$K^0_{k,m+1}$ si k par $K^1_{k,m+1}$ si k impar
$H_{k,m,b}$	$K^2_{k,m+1}$
$H_{k,m,c}$	$K^0_{2m+2,k/2}$
$H_{k,m,f}$	$K^2_{2m+4,(k-1)/2}$
$H_{k,m,g}$	$S_{m+1,k+2}$
$H_{k,m,h}$	$S_{m+1,k+1}$

$\updownarrow$ dual

Grafo Localmente C_6	Menor por borrado y contracción de aristas duales de aristas dobles
$H^*_{k,m,r}$	$T^r_{k,m+1}$
$H^*_{k,m,a}$	$K^2_{k,m+1}$ si k par $K^1_{k,m+1}$ si k impar
$H^*_{k,m,b}$	$K^0_{k,m+1}$
$H^*_{k,m,c}$	$K^2_{2m+2,k/2}$
$H^*_{k,m,f}$	$K^0_{2m+4,(k-1)/2}$
$H^*_{k,m,g}$	$S_{m+1,k+2}$
$H^*_{k,m,h}$	$S_{m+1,k+1}$

$\square$

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES BIBLIOTECA

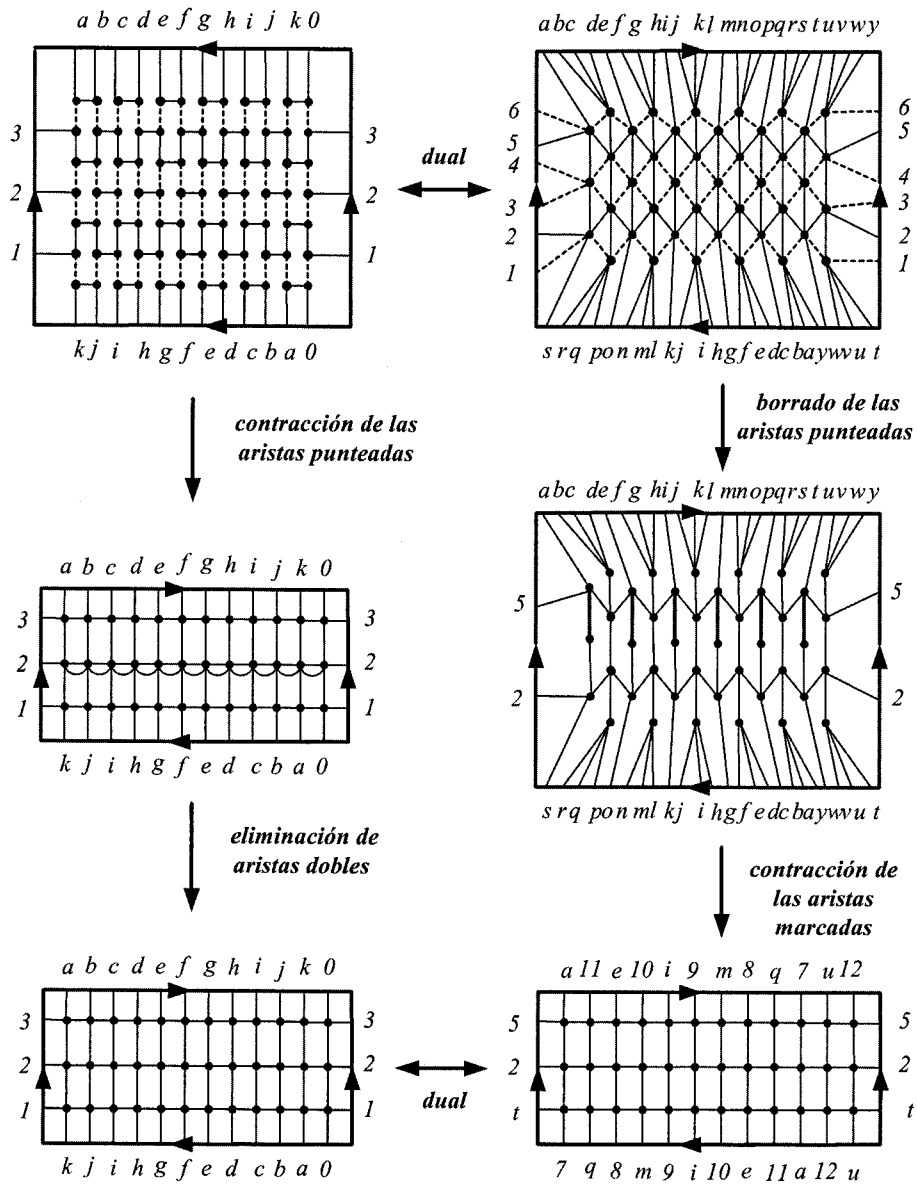


Figura 6.16: Borrado y contracción de las aristas de un emparejamiento perfecto en $H_{7,4,f}$ y $H^*_{7,4,f}$

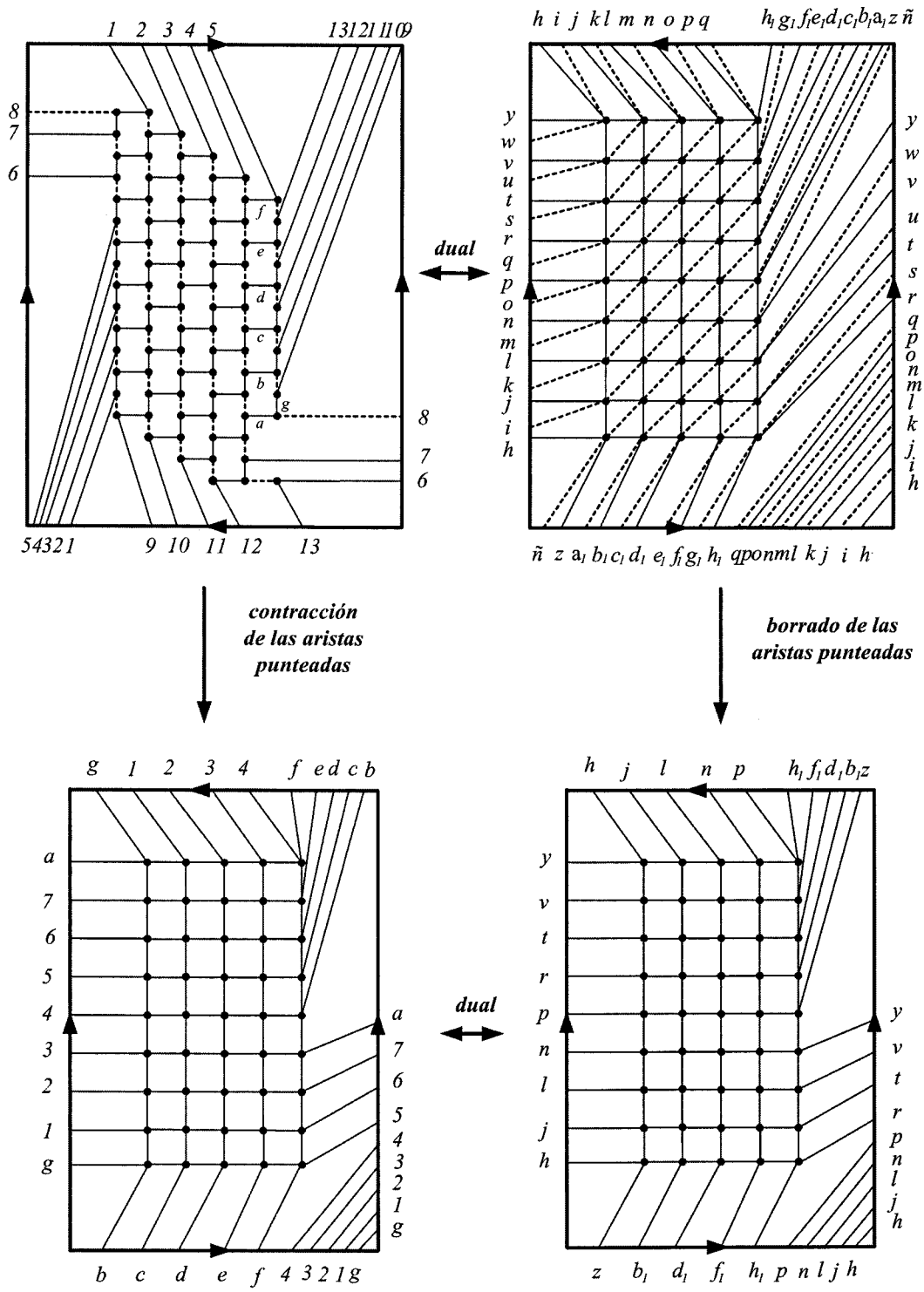


Figura 6.17: Borrado y contracción de las aristas de un emparejamiento perfecto en $H_{7,4,g}$ y $H^*_{7,4,g}$

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

CAPÍTULO 7

Teselaciones Hexagonales y Grafos Localmente C_6: Tutte Unicidad

Una vez demostrados los teoremas de clasificación de las teselaciones hexagonales y de los grafos localmente C_6 y siguiendo con la línea de investigación desarrollada para los grafos localmente cuadriculados, en este capítulo establecemos la maquinaria necesaria para demostrar que estas familias de grafos son Tutte únicas. Debido a la similitud en los cálculos nos centramos en el caso de las teselaciones hexagonales pues realizar el mismo desarrollo para los grafos localmente C_6 no aporta nada nuevo desde el punto de vista matemático.

7.1. Introducción

Para demostrar la Tutte unicidad de los grafos localmente cuadriculados se desarrolló (ver [29]) un sistema que permitió codificar y contar conjuntos de aristas en cada uno de los grafos localmente cuadriculados dados en el teorema de clasificación. El objetivo era poder determinar que dados dos grafos localmente cuadriculados no isomorfos existía un coeficiente distinto del polinomio de Tutte asociado a cada uno de estos grafos, basándose en que el polinomio de Tutte asociado a un grafo cualquiera H nos dice, para cada i y j, el número de conjuntos de aristas que hay en H de rango i y tamaño j.

El sistema que se desarrolla para codificar conjuntos de aristas está basado fundamentalmente en la orientación local definida en estos grafos. Dos aristas

incidentes con un vértice v de un grafo localmente cuadriculado G se dicen *adyacentes* si existe un cuadrado (ciclo de longitud 4) conteniéndolas a ambas; en caso contrario se dice que las aristas son *opuestas*. Una *orientación en un vértice* v consiste en etiquetar las cuatro aristas incidentes con él biyectivamente con las etiquetas N, S, E, W de tal forma que las aristas etiquetadas N y S sean opuestas, y análogamente las etiquetadas E y W. La orientación en un vértice v se puede extender a todos los vértices de un camino que comience en v. Así, partiendo de este vértice y siguiendo las instrucciones N, S, E y W podemos recorrer conjuntos de aristas contenidos en el grafo.

En este capítulo desarrollamos un sistema similar para codificar y contar conjuntos de aristas en las teselaciones hexagonales. En primer lugar, demostramos que estos grafos son localmente orientables, a partir de la relación de menor existente entre las teselaciones hexagonales y los grafos localmente cuadriculados (análogamente se demuestra que los grafos localmente C_6 también son localmente orientables a partir de su respectiva relación de menor con los grafos localmente cuadriculados). Esta propiedad, nos va a permitir codificar los conjuntos de aristas, utilizando las etiquetas N, S, E y W.

Hay varias diferencias fundamentales entre el alfabeto que definimos en este capítulo y el que se definió en los grafos localmente cuadriculados. La más importante es que este alfabeto tiene que contemplar el hecho de que en un vértice de una teselaciones hexagonal solamente podemos tomar tres direcciones. Esto implica dos hechos muy importantes y que no ocurrían en los grafos localmente cuadriculados. El primero de ellos es que fijado un vértice origen (a partir del cual comenzamos a recorrer las aristas) dos vértices cualesquiera no son intercambiables en el sentido de la orientación, es decir, hay vértices en los que podremos tomar las direcciones N, S y E y vértices en los que podremos tomar las direcciones N, S y W. El segundo factor a tener en cuenta es que todas las secuencias en el alfabeto $\{N, S, E, W\}$ no representan conjuntos de aristas en una teselación hexagonal. Nos veremos obligados a definir un alfabeto con algunas restricciones.

Una vez establecido el sistema de codificación, demostramos que cada conjunto normal de aristas es el resultado de una única palabra sobre el alfabeto definido, y esto es lo que nos permitirá contar estos conjuntos de aristas.

En la última sección, demostramos que el polinomio de Tutte preserva la estructura local de las teselaciones hexagonales (con argumentos similares también se demuestra que el polinomio de Tutte preserva la estructura local de los grafos localmente C_6) y comparamos $H_{k,m,0}$ con cada una de las teselaciones hexagonales especificadas en el teorema de clasificación con número cromático 2. Estudiando los ciclos esenciales de longitud mínima y los conjuntos normales, probamos que

o bien $H_{k,m,0}$ no tiene la misma longitud mínima de los ciclos esenciales que el grafo con el que estemos comparando, o no coincide el número de dichos ciclos, o el número de conjuntos de aristas de cierto rango y tamaño difiere en ambos grafos. Por último, queremos hacer notar que con los mismos argumentos utilizados para $H_{k,m,0}$ podemos demostrar que todas las teselaciones hexagonales y los grafos localmente C_6 están unívocamente determinados por su polinomio de Tutte.

7.2. Orientación Local en las Teselaciones Hexagonales

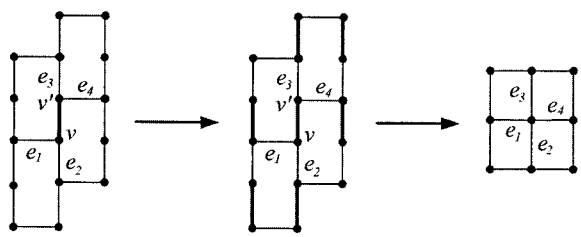
En esta sección vamos a demostrar que toda teselación hexagonal es localmente orientable. Para ello, vamos a hacer uso de la relación de menor existente entre los grafos localmente cuadriculados y las teselaciones hexagonales, demostrada en la Sección 4 del capítulo anterior, y de la orientación local definida en los grafos localmente cuadriculados (ver [29]). Haciendo uso de su respectiva relación de menor con los grafos localmente cuadriculados, también se demuestra que los grafos localmente C_6 son localmente orientables.

En primer lugar, vamos a recordar como se define una orientación en un vértice de un grafo localmente cuadriculado, G. Por definición, todo vértice $v \in V(G)$ pertenece exactamente a 4 cuadrados. Dos aristas incidentes con v se dicen *adyacentes* si existe un cuadrado (ciclo de longitud 4) conteniéndolas a ambas; en caso contrario se dice que las aristas son *opuestas*. Una orientación en un vértice v consiste en etiquetar las cuatro aristas incidentes con él biyectivamente con las etiquetas N, S, E, W de tal forma que las aristas etiquetadas N y S sean opuestas, y análogamente las etiquetadas E y W.

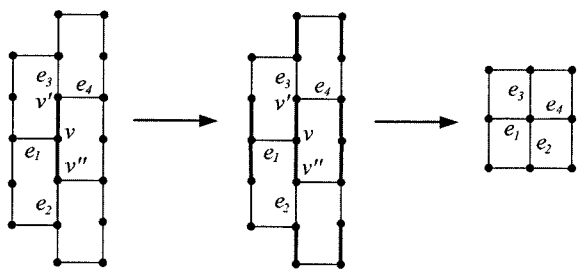
Lema 7.1. *Las teselaciones hexagonales son localmente orientables.*

Demostración. Sea H una teselación hexagonal. Por el Teorema 6.5 del Capítulo 6, H es isomorfo a uno de los siguientes grafos: $H_{k,m,r}$, $H_{k,m,a}$, $H_{k,m,b}$, $H_{k,m,c}$, $H_{k,m,h}$, $H_{k,m,g}$, $H_{k,m,f}$. Consideramos el emparejamiento perfecto correspondiente a cada una de estas familias de grafos (definidos en la Sección 4 del capítulo anterior) y lo denotamos P. Sabemos que contrayendo las aristas de este emparejamiento y eliminando las aristas dobles resultantes de la contracción, si las hubiera, obtenemos un grafo localmente cuadriculado.

Supongamos primero que H no es isomorfo a $H_{k,m,f}$. Para cada vértice $v \in V(H)$ hay exactamente una arista del emparejamiento P incidente con v, llamémosla $\{v, v'\}$. Consideramos la unión de las estructuras locales de v y v', etiquetando las otras dos aristas incidentes con v como e_1 y e_2 y las otras dos aristas incidentes con v' como e_3 y e_4. Contraemos todas las aristas de P pertenecientes a esta unión, eliminando aristas dobles si fuera necesario. Tal y como se muestra en la Figura 7.1a obtenemos una cuadrícula 3×3, por tanto podemos etiquetar las aristas e_1, e_2, e_3 y e_4 biyectivamente con las etiquetas N, S, E, W siguiendo el criterio establecido en los grafos localmente cuadriculados. $\{v, v'\}$ la consideramos opuesta a la única arista incidente con v y perteneciente al emparejamiento perfecto, P', formado por las aristas $\{(i,l)(i,l+1) \mid 0 \leq i \leq m\}$ no pertenecientes a P.



a)



b)

Figura 7.1: a) Operaciones a realizar para obtener una orientación local en un vértice v de H no isomorfo a $H_{k,m,f}$ b) Operaciones para obtener una orientación local en $v \in V(H_{k,m,f})$

Si H es isomorfo a $H_{k,m,f}$, para cada vértice del grafo hay una o dos aristas

incidentes con él pertenecientes al emparejamiento perfecto P. Si v no es un vértice de un hexágono que tenga más de dos aristas pertenecientes a P, podemos seguir el mismo procedimiento que hemos explicado en el párrafo anterior.

Supongamos que v es un vértice perteneciente a uno de los $m+1$ hexágonos que contienen (cada uno) 4 aristas pertenecientes a P. Consideramos dos casos:

(1) Hay dos aristas del emparejamiento P incidentes con v, denotémoslas $\{v, v'\}$ y $\{v, v''\}$. La unión de las estructuras hexagonales de v, v' y v'' está compuesta por cinco hexágonos. La única arista incidente con v y no perteneciente a P es etiquetada e_1; e_2 es la única arista incidente con v'' y perteneciente a P'. Por último, las dos aristas incidentes con v' y no pertenecientes a P son etiquetadas e_3 y e_4 (ver Figura 7.1b). Contrayendo todas las aristas de P pertenecientes a dicha unión y eliminando aristas dobles, también obtenemos una cuadrícula 3×3, y podemos etiquetar e_1, e_2, e_3 y e_4 biyectivamente con las etiquetas N, S, E, W (Figura 7.1b). $\{v, v'\}$ y $\{v, v''\}$ las consideramos opuestas a las aristas incidentes con v' y v'' respectivamente, perteneciente al emparejamiento perfecto, P' (e_2 y e_3 en la Figura 7.1b).

(2) Hay exactamente una arista del emparejamiento P incidente con v, denotémosla $\{v, v'\}$. En v' tiene que haber una arista incidente con él que pertenezca a P y sea distinta de $\{v, v'\}$, llamémosla $\{v', v''\}$. Consideramos la unión de las estructuras locales de v y v'', compuesta por cinco hexágonos, etiquetando las otras dos aristas incidentes con v como e_1 y e_2, la única arista no perteneciente a P e incidente con v' como e_3 y la arista incidente con v'' no perteneciente a P y no perteneciente al hexágono que contiene 4 aristas de P, la etiquetamos e_4. Contrayendo todas las aristas de P pertenecientes a dicha unión y eliminando aristas dobles, también obtenemos una cuadrícula 3×3. Por tanto, también podemos etiquetar las aristas e_1, e_2 e_3 y e_4 biyectivamente como N, S, E, W. Las aristas $\{v, v'\}$ y $\{v', v''\}$ las consideramos opuestas a las dos aristas no pertenecientes al hexágono que contiene 4 aristas de P, e incidentes con v y v'' respectivamente. □

Queremos resaltar que utilizando la relación de menor existente entre los grafos localmente cuadriculados y los grafos localmente C_6 podemos reproducir el resultado anterior para esta última familia de grafos. Basta considerar el emparejamiento perfecto correspondiente en cada grafo localmente C_6. Borramos las aristas de dicho emparejamiento y etiquetamos las restantes siguiendo el criterio de los grafos localmente cuadriculados. Cada arista del emparejamiento es la diagonal de un cuadrado al que pertenecen dos de las aristas ya etiquetadas. Así, por ejemplo si estas aristas están etiquetadas N y E, la del emparejamiento la etiquetamos NE.

La demostración anterior nos proporciona una definición de *aristas adyacentes* y *opuestas* en una teselación hexagonal heredada, en cierta forma, de los grafos localmente cuadriculados. Dos aristas incidentes con v y pertenecientes a P son siempre opuestas y dos aristas incidentes con v y no pertenecientes a P son siempre adyacentes. De las tres aristas adyacentes con v hay una única arista adyacente a las otras dos.

Por la orientación (v, v', e, f) entenderemos que v es el vértice origen, $\{v, v'\} \in P$, e es la única arista incidente con v y adyacente a las otras dos aristas incidentes con v. Esta arista nunca pertenece a P y la etiquetamos E. Por último, f es una arista incidente con v y etiquetada N. Análogamente, por la orientación (v, v', v'', e, f) entendemos que v es el vértice origen, $\{v, v'\}, \{v, v''\} \in P$ (ó $\{v, v'\}, \{v', v''\} \in P$), e es la única arista incidente con v y opuesta a las otras dos aristas incidentes con v y la etiquetamos E, y f es una arista incidente con v etiquetada N. Siempre que no genere confusión, denotaremos ambos casos por (v, v', e, f).

Si $\alpha \in \{N, S, E, W\}$, denotaremos α^{-1} la etiqueta opuesta a α. Si fijamos una orientación en $v \in V(H)$ (v es el vértice origen) entonces cada vértice u adyacente a v esta orientado sin ambigüedad, porque la orientación en v induce una orientación en u: si $\{u, v\}$ está etiquetada α desde v, entonces $\{u, v\}$ está etiquetada α^{-1} desde u.

Si $tzyxvw$ es un hexágono y $\{x, v\}$, $\{w, v\}$ están etiquetadas β y α respectivamente desde v, entonces $\{x, y\}$ es β desde x, $\{z, y\}$ es α desde y, $\{z, t\}$ es β^{-1} desde z y $\{t, w\}$ es β^{-1} desde t.

También, podemos trasladar la orientación en v a todos los vértices de un camino que comience en v, pero dos caminos diferentes uniendo v y u pueden inducir una orientación diferente en u. Esto no ocurre si la unión de estos dos caminos es un ciclo contractible (no esencial). De hecho, podemos transformar un camino en el otro usando tres transiciones elementales y sus inversas (ver Figura 7.2): si e, f, g, h, i, j son las aristas de un hexágono ordenadas cíclicamente, podemos cambiar e, f, g, h, i por j; e, f, g, h por i, j y e, f, g por h, i, j y estas operaciones no cambian la orientación en el vértice final.

Sea A un conjunto de aristas normal y conexo en una teselación hexagonal H. Al igual que en los grafos localmente cuadriculados, los *conjuntos normales* son aquellos que no contienen ciclos esenciales. Fijamos una orientación en un vértice $v \in V(A)$, entonces todos los vértices en $V(A)$ están orientados sin ambigüedad.

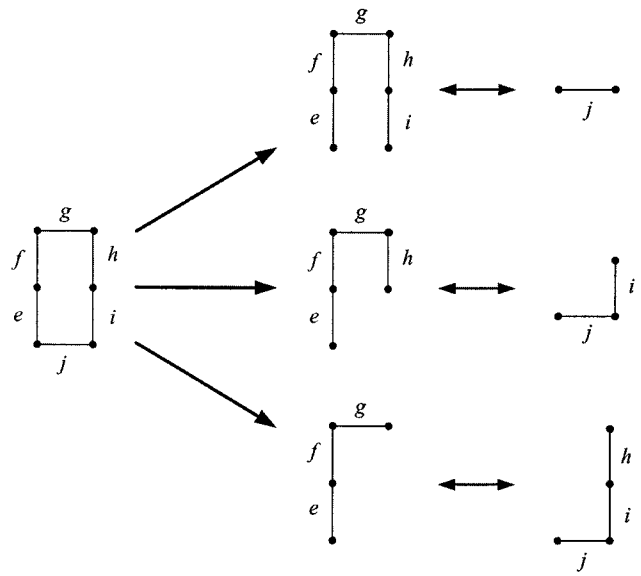


Figura 7.2: Transiciones elementales de un hexágono.

Cada camino en A puede ser descrito como una secuencia (no única) de etiquetas $\{N, S, E, W\}$ (Figura 7.3a). Esto nos permite asignar coordenadas a cada vértice de $V(A)$:

(1) El vértice origen v tiene coordenadas $(0, 0)$.

(2) Un vértice cualquiera $u \in V(A)$ tiene coordenadas (i, j) si en un camino en A uniendo v con u (teniendo en cuenta que la orientación en el vértice final inducida por diferentes caminos no cambia), i es el número de etiquetas E menos el número de etiquetas W y j es el número de etiquetas N menos el número de etiquetas S.

Como ejemplo, el vértice etiquetado u en la Figura 7.3a tiene coordenadas $(3, 0)$ y obtenemos las mismas coordenadas aunque tomemos distintos caminos uniendo v con u, tales como $ESENE$ y $ENENESS$.

Lema 7.2. *Sea A un conjunto de aristas normal y conexo en una teselación hexagonal H con $|A| \leq l_H + 3$, entonces dos vértices distintos de $V(A)$ tienen diferentes coordenadas.*

Demostración. En primer lugar, vamos a demostrar que con una orientación fijada (v, v', e, f) ningún vértice excepto el vértice origen puede tener coordenadas $(0, 0)$. Supongamos que existe $x \in V(A)$ tal que x tiene coordenadas $(0, 0)$, entonces existe un camino R uniendo v con x con tantas etiquetas N como S y tantas

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

etiquetas E como W. Vamos a probar que $x = v$ por inducción en la longitud de este camino. Si $|R| = 6$ y v es el vértice origen, las primeras etiquetas de R pueden ser: NN, NW, E, SS, ó SW y utilizando las tres transiciones elementales descritas anteriormente, podemos cambiar estas etiquetas por $ENNW$, $SWNN$, $NNESS$, $ESSW$ y $NWSS$ respectivamente. Para obtener tantas etiquetas N como S y tantas etiquetas E como W tiene que ser $v = x$. Supongamos ahora que todo camino R (que no sea un ciclo) con $|R| < n$ no tiene tantas etiquetas N como S y tantas etiquetas E como W y probamos el resultado para $|R| = n$. Supongamos que existe un camino R (no ciclo) uniendo v con x, con $|R| = n$ y con tantas etiquetas N como S y E como W, entonces existe una subsecuencia etiquetada $\beta(\alpha^{-1})^l\beta\alpha^s\beta^{-1}(\alpha^{-1})^m\beta^{-1}$ ó $\beta\alpha^l\beta\alpha^s\beta^{-1}\alpha^m\beta^{-1}$ (u otras subsecuencias análogas). En el primer caso, podemos cambiar la subsecuencia por α^{s-l-m} manteniendo la orientación en el vértice final. Análogamente, en el segundo caso la subsecuencia puede ser cambiada por α^{s+l+m} (ver Figura 7.3b).

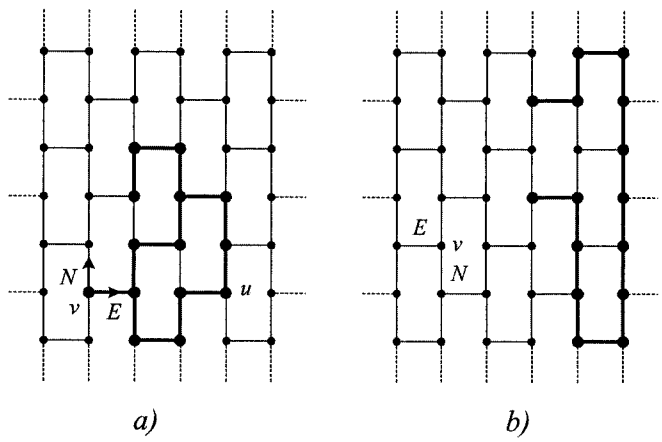


Figura 7.3: *a)* Un conjunto de aristas descrito usando dos secuencias distintas, la primera de ellas es $ENENNWSNESESSWSWN$ y la segunda es $ESENENNWNWSNESSWS$ b) El camino marcado $W(N^3)WS^6ENE$ se puede cambiar por S^{6-3-1}

En ambos casos, el cambio da lugar a un nuevo camino R' uniendo v y x con $|R'| < n$, en un nuevo conjunto de aristas conexo A'. Este conjunto es también un conjunto normal de aristas porque $|A'| \leq |A| - 4 \leq l_H - 1$. Por hipótesis de inducción, R' no tiene tantas etiquetas N como S y E como W. Así, obtenemos una contradicción ya que R tiene las mismas etiquetas de R' y además tantas etiquetas α como α^{-1} y tantas etiquetas β como β^{-1}. Por tanto, si R' no tiene

tantas etiquetas N como S y E como W, tampoco las tiene R. Como ejemplo, en la Figura 7.3b, R tiene las etiquetas de R' y 12 etiquetas más, de las cuales, 4 son N, 4 son S, 2 son E y 2 son W.

Dados dos vértices diferentes, u y w en $V(A)$, consideramos los caminos R_u y R_w uniendo u con v y w con v respectivamente. Como A es conexo, existe un camino $R_w \cup R$ uniendo v con u y conservando la orientación en u. Si u y v tuvieran las mismas coordenadas, el camino R tendría que unir dos vértices distintos y tener tantas etiquetas N como S y E como W, lo cual no es posible. □

A partir de ahora, y a menos que especifiquemos lo contrario, consideramos que todos los conjuntos normales y conexos de aristas tienen a lo sumo $l_H + 3$ elementos.

7.3. Contando Conjuntos de Aristas

Para probar la Tutte unicidad de las teselaciones hexagonales, hay que demostrar que dadas dos teselaciones hexagonales no isomorfas hay al menos un coeficiente del polinomio generador de rango-tamaño en el cual ambos grafos difieren, es decir, que ambos grafos tienen diferente número de conjuntos de aristas de rango i y tamaño j, para ciertos i y j (análogamente, para los grafos localmente C_6). El propósito de esta sección es encontrar un sistema que nos permita contar conjuntos de aristas. Nuestro primer paso es codificar los conjuntos de aristas de una teselación hexagonal.

Sea L^∞ la cuadrícula plana infinita, esto es, el grafo infinito que tiene por vértices $\mathbb{Z}\times\mathbb{Z}$ y en el cual (i,j) es adyacente a $(i-1,j),(i+1,j),(i,j-1),(i,j+1)$. Borramos las aristas $\{(i,2j),(i-1,2j)\}$ si i par y $\{(i,2j+1),(i+1,2j+1)\}$ si i impar, obteniendo una *teselación hexagonal plana infinita* H^∞.

Sea Ω el grupo de los automorfismos de grafos de H^∞. Este grupo está generado por las traslaciones, simetrías y giros del plano que manda vértices en vértices. Denotamos por $\Sigma(H^\infty)$ el conjunto de los conjuntos de aristas conexos y finitos de H^∞. Definimos una relación de equivalencia en $\Sigma(H^\infty)$ de la siguiente forma:

$$B_1 \sim B_2 \Leftrightarrow \exists\sigma \in \Omega,\ \sigma(B_1) = B_2$$

Si Λ es un conjunto de representantes de $\Sigma(H^\infty)/\sim$ tal que cada $B \in \Lambda$ contiene al vértice $(0,0)$, entonces, intuitivamente, Λ cubre todas las posibles formas que un conjunto normal puede tener.

A continuación, definimos un conjunto Γ de palabras que representen a todos los conjuntos de aristas de $\Sigma(H^\infty)/\sim$. Para ello utilizamos el alfabeto $\{N, S, E, W\}$.

Etiquetamos la arista $\{(0,0),(0,1)\}$ como N, la arista $\{(0,0),(1,0)\}$ como E, $\{(0,1),(-1,1)\}$ desde $(0,1)$ como W y $\{(0,0),(-1,0)\}$ como S. Para cada $B \in \Lambda$ existe una secuencia γ_B sobre el alfabeto $\{N, S, E, W\}$ tal que comenzando en $(0,0)$ y siguiendo las instrucciones dadas por γ_B las únicas aristas cubiertas sean las pertenecientes a B. Observar que una arista puede ser cubierta más de una vez y que la secuencia γ_B no es única. Debido a que toda secuencia sobre el alfabeto $\{N, S, E, W\}$ no representa a un conjunto de Λ (por ejemplo, una secuencia con dos etiquetas E consecutivas) vamos a establecer algunas restricciones sobre el alfabeto:

(1) Ninguna secuencia puede comenzar con la etiqueta W.

(2) No pueden haber dos etiquetas E (análogamente W) consecutivas.

(3) Si una secuencia comienza con la etiqueta N (análogamente S), tiene que haber un número impar de etiquetas N y S antes de tener una etiqueta W; y un número par antes de tener una etiqueta E. Por tanto, tiene que haber un número impar de etiquetas N y S entre dos etiquetas W y un número par entre dos etiquetas E.

(4) Si una secuencia comienza con la etiqueta E, tiene que haber un número impar de etiquetas N y S entre dos etiquetas E y un número par entre dos etiquetas W.

Denotamos por Γ el conjunto de palabras sobre el alfabeto $\{N, S, E, W\}$ verificando las cuatro condiciones anteriores. Estas restricciones no modifican el hecho de que cada $B \in \Lambda$ tiene una palabra γ_B asociada, pero nos proporcionan el recíproco, es decir, dada $\gamma \in \Gamma$ existe $B \in \Lambda$, tal que $\gamma = \gamma_B$. A este conjunto $B \in \Lambda$ lo denotaremos $B(\gamma)$.

El siguiente paso es asignar una palabra de Γ a cada conjunto de aristas normal y conexo de una teselación hexagonal, H. Fijamos una orientación (v, v', e, f) en H, siendo $v \in V(H)$ el vértice origen. Siguiendo el código dado por γ desde v con orientación (v, v', e, f), se obtiene un conjunto de aristas $A^H(\gamma, v, v', e, f)$, que llamamos *instancia de γ en H*. Decimos que (v, v', e, f) es una *orientación de la instancia*. Una instancia de γ puede tener diferentes orientaciones asociadas, el número de ellas depende solamente de las simetrías de $B(\gamma)$ y lo denotamos

$sym(\gamma)$. Llamamos *longitud de* γ al número de etiquetas N, S, E, W que componen la secuencia γ.

Lema 7.3. *Sea H una teselación hexagonal y $\gamma \in \Gamma$. Si $r(B(\gamma)) = l_H - 1$ y $|B(\gamma)| \leq l_H$, entonces $A^H(\gamma, v, v', e, f)$ es un conjunto normal de aristas y tiene el mismo rango y tamaño que $B(\gamma)$.*

Demostración. Vamos a probar este resultado por inducción en la longitud de γ. Si γ tiene longitud igual a 1, el resultado es trivial. Sea γ' la palabra resultante de eliminar la última etiqueta de γ. Por hipótesis de inducción $A^H(\gamma', v, v', e, f)$ es normal y tiene el mismo rango y tamaño que $B(\gamma')$. Cuando añadimos la última etiqueta de γ a γ', hay tres posibilidades en $B(\gamma')$:

(1) Que estemos añadiendo una arista entre dos vértices pertenecientes a $B(\gamma')$.

(2) Que estemos añadiendo una arista cuyos extremos sean dos vértices, uno perteneciente a $B(\gamma')$ y el otro no.

(3) Que no estemos añadiendo ninguna arista.

(1) Denotamos por x' al último vértice de $A^H(\gamma', v, v', e, f)$ cubierto por γ'. Teniendo en cuenta que $A^H(\gamma', v, v', e, f)$ es normal, todos sus vértices están orientados sin ambigüedad. Sea c la arista añadida. Esta arista está etiquetada α desde x' siguiendo la orientación (v, v', e, f). Supongamos que $A^H(\gamma, v, v', e, f)$ no es un conjunto normal de aristas, entonces al añadir la arista c hemos creado un ciclo esencial. Como $|B(\gamma)| \leq l_H$, $A^H(\gamma, v, v', e, f)$ tiene que ser un ciclo esencial. $B(\gamma)$ contiene al menos un ciclo porque ha sido obtenido uniendo dos vértices de $B(\gamma')$. Por tanto, γ contiene una subpalabra del tipo:

$$\beta(\alpha^{-1})^s\beta^{-1}\alpha^s$$
$$\beta(\alpha^{-1})^s\beta(\alpha^{-1})^t\beta^{-1}\alpha^r\beta^{-1}$$
$$\beta(\alpha^{-1})^s\beta(\alpha^{-1})^t\beta^{-1}(\alpha^{-1})^r\beta^{-1}$$

(u otros casos análogos a los dos últimos) (Figura 7.4). Cambiamos estas subpalabras por $\beta(\alpha^{-1})^{s-2}\beta^{-1}\alpha^{s-2}$, $(\alpha^{-1})^{t+s-r}$, $(\alpha^{-1})^{t+s+r}$ respectivamente.

En cualquier caso, se obtiene una palabra γ^* de longitud menor que la longitud de γ, por tanto $A^H(\gamma^*, v, v', e, f)$ es un conjunto normal de aristas y contiene un ciclo. Si consideramos la región simplemente conexa determinada por $A^H(\gamma^*, v, v', e, f)$ y añadimos, dependiendo del caso, un hexágono o dos columnas de hexágonos, se obtiene una región simplemente conexa determinada por γ. Como $A^H(\gamma, v, v', e, f)$ es un ciclo esencial, al menos un vértice de los hexágonos añadidos pertenece a $A^H(\gamma^*, v, v', e, f)$. Así, $A^H(\gamma, v, v', e, f)$ contiene al menos dos ciclos y obtenemos una contradicción.

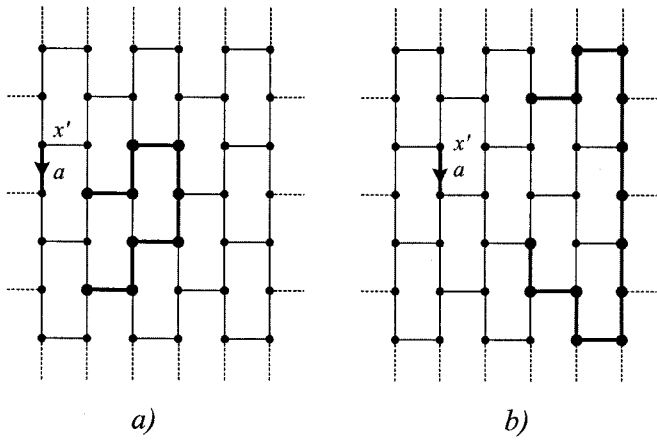


Figura 7.4: *a)* El camino etiquetado $b(a^{-1})b(a^{-1})^2b^{-1}ab^{-1}$ se puede cambiar por $(a^{-1})^{2+1-1}$ b) El camino etiquetado $abab(a^{-1})^6b^{-1}ab^{-1}$ se puede cambiar por $(a^{-1})^3$

Queda probar la igualdad de rangos y tamaños. Por hipótesis de inducción:

$$r(A^H(\gamma', v, v', e, f)) = r(B(\gamma')) \text{ y } |A^H(\gamma', v, v', e, f)| = |B(\gamma')|$$

Además $|B(\gamma)| = |B(\gamma')| + 1$ y $r(B(\gamma)) = r(B(\gamma'))$. Como $A^H(\gamma, v, v', e, f)$ es normal entonces:

$$r(A^H(\gamma, v, v', e, f)) = r(A^H(\gamma', v, v', e, f))$$
$$|A^H(\gamma, v, v', e, f)| = |A^H(\gamma', v, v', e, f)| + 1$$

(2) y (3) Utilizando un razonamiento similar obtenemos que si $A^H(\gamma, v, v', e, f)$ no es un conjunto normal de aristas, entonces es un ciclo esencial. Pero un ciclo esencial se crea uniendo dos vértices pertenecientes a $B(\gamma')$. El tercer caso es trivial pues si no añadimos ninguna arista, $A^H(\gamma, v, v', e, f) = A^H(\gamma', v, v', e, f)$. □

Lema 7.4. *Sea $A \subseteq A(H)$ un conjunto de aristas normal y conexo. Fijada una orientación (v, v', e, f) con $v \in V(A)$, existe un único conjunto $B \subseteq A(H^\infty)$ y un único isomorfismo de grafos $\varphi : V(A) \to V(B)$ tal que:*

(1) $\varphi(v) = (0, 0)$.

(2) Si la arista $\{x, y\} \in A$ está etiquetada α desde x, entonces $\{\varphi(x), \varphi(y)\}$ está etiquetada α desde $\varphi(x)$ de acuerdo con la orientación:

$$((0,0), \{(0,0), (1,0)\}, \{(0,0), (0,1)\})$$

Demostración. Fijada una orientación (v, v', e, f) con $v \in V(A)$, podemos asignar coordenadas a los vértices de A. Como A es normal, esta asignación no es ambigua. Definimos $\varphi : V(A) \to V(H^\infty)$ tal que $\varphi(x) = (i, j)$ si en cada camino en A uniendo v con x, i es el número de etiquetas E menos el número de etiquetas W y j es el número de etiquetas N menos el número de etiquetas S. Además, definimos $\bar{\varphi} : A \to A(H^\infty)$ como $\bar{\varphi}(\{x, y\}) = \{\varphi(x), \varphi(y)\}$. Tomando $B = \bar{\varphi}(A)$, es fácil probar que φ es un isomorfismo de grafos y que $\bar{\varphi}$ es el isomorfismo inducido sobre las aristas por φ.

(1) y (2) son consecuencia de la asignación de coordenadas siguiendo la orientación (v, v', e, f) en una teselación hexagonal H, y de la asignación de etiquetas en las aristas incidentes con el vértice origen v.

Demostramos la unicidad de B por inducción en $m = \max\{d(v, x), x \in V(A)\}$. Supongamos que existe un par (φ', B') verificando el enunciado del lema. Si $m = 1$, $d(v, x) = 1$ $\forall x \in V(A)$ con $x \neq v$; por tanto $|A| \leq 3$. Por (1) y (2), es inmediato que $B = B'$. Supongamos cierto el resultado para $m \leq l$ y lo probamos para $m = l + 1$.

Sean $x_1, \ldots, x_n \in V(A)$ tales que $d(v, x_i) = l + 1$ y $d(v, x) < l + 1$ para todo $x \neq x_i$. Por inducción en n demostramos que B es único para $m = l + 1$. Si $n = 1$, sea x_1 el único vértice tal que $d(v, x_1) = l + 1$ y $A(x_1)$ el conjunto de aristas incidentes con x_1 y pertenecientes a A. Llamamos A' al conjunto de aristas normal conexo que resulta de eliminar el vértice x_1. Por hipótesis de inducción, existe un único par $(\psi, \tilde{B})$ asociado a A', entonces basta tomar $B = \tilde{B} \cup \bar{\varphi}(A(x_1))$. Supongamos que B es único para $m = l + 1$ y teniendo $n - 1$ vértices tales que $d(v, x_i) = l + 1$. Utilizando el mismo razonamiento que en el caso anterior es fácil probar el resultado para n vértices. □

Lema 7.5. *Dado un conjunto de aristas normal y conexo $A \subseteq A(H)$, existe una única palabra $\gamma \in \Gamma$ y una orientación (v, v', e, f), no necesariamente única, tal que $A = A^H(\gamma, v, v', e, f)$.*

Demostración. Fijamos $u \in V(A)$ y dos aristas adyacentes en u, denotémoslas e' y f'. Aplicando el Lema 7.4 a A con orientación (u, u', e', f') obtenemos un par (φ, B). Sea $B' \in \Lambda$ el representante de la clase de equivalencia de B y $\sigma \in \Omega$ el automorfismo de H^∞ que manda B' en B. Definimos v como el vértice $\varphi^{-1}\sigma((0,0))$. Supongamos que $\sigma(\{(0,0),(1,0)\})$ está etiquetado α desde $\sigma((0,0))$, entonces tomamos e como la arista etiquetada α desde v de acuerdo con la orientación (u, u', e', f'). Análogamente, si $\sigma(\{(0,0),(0,1)\})$ está etiquetada β desde $\sigma((0,0))$, entonces f es la arista etiquetada β desde v.

Finalmente, tomamos $\gamma = \gamma_B$, entonces $A = A^H(\gamma, v, v', e, f)$. Para probar la unicidad de γ, supongamos que existe $B'' \in \Lambda$ tal que $A = A^H(\gamma_{B''}, w, w', e'', f'')$. Por definición de instancia, cambiar la orientación de la instancia solamente produce simetrías de $B(\gamma)$, es decir, existe $\sigma' \in \Omega$ tal que $\sigma'(B'')$ satisface el Lema 7.4 cuando se aplica a A con orientación (v, v', e, f). Por tanto, $\sigma'(B'') = B'$, es decir, B,B' y B'' pertenecen a la misma clase de equivalencia. □

Lema 7.6. *Para todo $n > 0$, el número de conjuntos de aristas normales y conexos de rango $n-1$ y tamaño n es el mismo para toda teselación hexagonal H con $2pq$ vértices y tal que $n \leq l_H$.*

Demostración. Por el lema anterior, cada conjunto de aristas normal y conexo A es la instancia de una única palabra $\gamma(A)$. Por el Lema 7.4, A es isomorfo a $B(\gamma(A))$ entonces tienen el mismo rango y tamaño. Por el Lema 7.3, la instancia de una palabra $\gamma(A)$ es un conjunto de aristas normal y conexo. Por tanto, el número de conjuntos de aristas normales y conexos de rango $n-1$ y tamaño n es igual al número de distintas instancias de palabras correspondientes a los conjuntos de aristas de H^∞ de rango $n-1$ y tamaño n.

Para cada γ fijada, podemos elegir $4pq$ orientaciones distintas. Sea $\Gamma^{r,s}$ el conjunto de palabras de Γ tales que $r(B(\gamma)) = r$ y $|B(\gamma)| = s$. Entonces el número de conjuntos de aristas normales y conexos de rango $n-1$ y tamaño n es:

$$\sum_{\gamma \in \Gamma^{n-1,n}} \frac{4pq}{sym(\gamma)}$$

Como $sym(\gamma)$ depende solamente de la palabra, esta cantidad no depende del grafo. □

Una versión no conexa de este lema se demuestra utilizando los mismos argumentos que los usados por Márquez, Mier, Noy y Revuelta en [29] para los grafos localmente cuadriculados.

Lema 7.7. *Para todo $n > 0$ y $t > 1$, el número de conjuntos normales de aristas de rango $n-1$, tamaño n y t componentes conexas es el mismo para toda teselación hexagonal H con $2pq$ vértices y tal que $n \leq l_H$.*

Como una consecuencia de los Lemas 7.6 y 7.7, tenemos el siguiente resultado.

Lema 7.8. *Para todo $n > 0$. El número de conjuntos de aristas de rango $n-1$ y tamaño n que no contienen ciclos esenciales es el mismo para toda teselación hexagonal H con $2pq$ vértices y tal que $n \leq l_H$.*

Corolario 7.9. *Sean H y H' un par de teselaciones hexagonales con $2pq$ vértices. Entonces:*

- *$l_H \neq l_{H'}$ implica $T(H;x,y) \neq T(H';x,y)$.*
- *Si $l_H = l_{H'}$ pero H y H' no tienen el mismo número de ciclos esenciales de longitud mínima entonces $T(H;x,y) \neq T(H';x,y)$.*

Demostración. Si $l_H < l_{H'}$, hay ciclos esenciales de longitud l_H en H pero no en H'. Como por el lema anterior, el número de conjuntos normales de aristas de rango $l_H - 1$ y tamaño l_H es el mismo en H y H', el coeficiente de $x^{l_H-1}y^{l_H}$ en el polinomio generador de rango-tamaño es mayor en H que en H'. De forma análoga, se demuestra el segundo punto. □

El procedimiento que hemos seguido para contar conjuntos normales de aristas de tamaño menor que $l_H + 1$, es mostrar que podemos pasar de conjuntos normales de aristas en las teselaciones hexagonales a conjuntos de aristas contenidos en $A(H^\infty)$. El fijar un tamaño menor o igual que l_H es para asegurar que en esta correspondencia siempre manejamos conjuntos normales. Nuestro próximo objetivo es demostrar que en ciertos casos podemos contar conjuntos normales de aristas de tamaño mayor que l_H. Este hecho va a ser fundamental en la próxima sección.

Denotamos por $N(\gamma, H, n, r)$ al número de conjuntos normales de aristas contenidos en $A(H)$ de rango r, tamaño n y teniendo a $B(\gamma)$ como subgrafo. Un conjunto de aristas $A \subseteq A(H)$ es un *conjunto prohibido de aristas para* γ si contiene un ciclo esencial y un subconjunto $B \subseteq A$ isomorfo a $B(\gamma)$.

Corolario 7.10. *Sea $\gamma \in \Gamma$ tal que $B(\gamma)$ contiene al menos un ciclo. Entonces $N(\gamma, H, n, r)$ es el mismo para toda teselación hexagonal H con $2pq$ vértices, que no contenga conjuntos prohibidos de aristas para γ de tamaño n y tal que $n \leq l_H + 3$.*

Demostración. Con una revisión detallada de las demostraciones de todos los lemas anteriores es fácil comprobar que, para demostrar este resultado, basta probar que fijada $\gamma \in \Gamma$ tal que $B(\gamma)$ contiene al menos un ciclo y dado $\gamma' \in \Gamma$ tal que $B(\gamma) \subseteq B(\gamma')$, $r(B(\gamma')) = l_H+2$ y $|B(\gamma')| \leq l_H+3$, entonces $A^H(\gamma', v, v', e, f)$ es un conjunto normal de aristas con igual rango y tamaño que $B(\gamma')$. Este resultado es análogo al Lema 7.3 pero tomando tamaño menor que $l_H + 4$.

Si $B(\gamma) \subseteq B(\gamma')$ entonces $B(\gamma')$ contiene al menos un ciclo. Si $n \leq l_H$, la demostración es igual que la del Lema 7.3. Si $n = l_H + 1$, probamos que $A^H(\gamma', v, v', e, f)$ tiene que ser un ciclo esencial (entonces el resto de la prueba sería análoga a la del Lema 7.3). Supongamos que $A^H(\gamma', v, v', e, f)$ no es un ciclo esencial, entonces $A^H(\gamma', v, v', e, f)$ contiene un ciclo esencial y una arista más. Como:

$$A^H(\gamma, v, v', e, f) \simeq B(\gamma) \subseteq B(\gamma') \simeq A^H(\gamma', v, v', e, f)$$

$A^H(\gamma', v, v', e, f)$ es un conjunto prohibido de aristas para γ de tamaño n y obtenemos una contradicción. Los casos en los cuales $n = l_H + 2$ y $n = l_H + 3$ se demuestran de forma análoga. □

7.4. Tutte Unicidad

En esta sección demostramos la Tutte unicidad de la teselación hexagonal toroidal $H_{k,m,0}$. La técnica utilizada se puede aplicar al resto de las teselaciones hexagonales y a los grafos localmente C_6. En primer lugar probamos que el polinomio de Tutte preserva la estructura local de las teselaciones hexagonales, utilizando el siguiente lema.

Lema 7.11. *[31] Si G es un grafo $2-$conexo y simple, y H es Tutte equivalente a G, entonces H es también simple y $2-$conexo. Además, si G es un grafo $2-$conexo y simple, los siguientes parámetros están determinados por su polinomio de Tutte:*

(1) El número de vértices y aristas.

(2) La arista-conectividad.

(3) El número de ciclos de longitud tres, cuatro y cinco.

A continuación, demostraremos que hay al menos un coeficiente, en el cual, el polinomio de Tutte asociado a $H_{k,m,0}$ y el polinomio de Tutte asociado a cualquier

otra teselación hexagonal no isomorfa a $H_{k,m,0}$ difieren. Sabemos que el polinomio de Tutte asociado a un grafo cualquiera H nos dice para cada i y j el número de conjuntos de aristas que hay en H de rango i y tamaño j. Estos conjuntos de aristas se pueden clasificar en dos grupos: conjuntos normales de aristas (no contienen ciclos esenciales) y conjuntos de aristas conteniendo al menos un ciclo esencial.

Recordemos que en el Lema 6.2 del capítulo anterior, calculamos para cada teselación hexagonal la longitud mínima de los ciclos esenciales y el número de ciclos de dicha longitud; y en la sección anterior hemos establecido un sistema para poder contar, en determinados casos, conjuntos normales. Por tanto, estamos en condiciones de evaluar el número de conjuntos de aristas de rango i y tamaño j para determinados i y j.

Teorema 7.12. *Sea G un grafo Tutte equivalente a una teselación hexagonal H, entonces G es una teselación hexagonal.*

Demostración. Por el Teorema 6.5 del capítulo anterior y el Lema 6.6, sabemos que H tiene $2uv$ vértices para ciertos u, v; $3uv$ aristas y arista-conectividad igual a tres. Por el Lema 7.11, G tiene los mismos parámetros entonces, G es cúbico, tiene $2uv$ vértices y $3uv$ aristas. Como H es una teselación hexagonal, la longitud mínima de un ciclo cualquiera de H es 6; por tanto, por el Lema 7.11 la longitud mínima de un ciclo cualquiera de G es al menos 6. Para probar que G es una teselación hexagonal, nos queda demostrar que ambos grafos tienen el mismo número de ciclos de longitud 6 (hexágonos). Como G y H son Tutte equivalentes, tienen el mismo número de conjuntos de aristas de rango 5 y tamaño 6. Veamos que estos conjuntos son los ciclos de longitud 6.

Sea $C \subseteq A(G)$ un conjunto de aristas de rango 5 y tamaño 6. Si y es el número de componentes conexas de C, entonces $5 = rg(C) = |V(C)| - y$. Supongamos que C no es un ciclo, como la longitud mínima de un ciclo cualquiera en G y H es al menos 6, tenemos que $|V(C_i)| = |A(C_i)| + 1$ donde C_i es una componente conexa de C. Así, $|V(C)| = y + 6$ y obtenemos una contradicción. □

Teorema 7.13. *El grafo $H_{k,m,0}$ es Tutte único para $m \geq 2$ y $k \geq 3$.*

Demostración. Sea G un grafo Tutte equivalente a $H_{k,m,0}$, es decir,

$$T(G; x, y) = T(H_{k,m,0}; x, y)$$

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

Por el Teorema 6.5 y el Lema 6.1 del capítulo anterior, G tiene que ser isomorfo a uno y sólo uno de los siguientes grafos:

$$H_{k',m',r}, \quad H_{k',m',a}, \quad H_{k',m',b}, \quad H_{k',m',h}$$

Veamos que G es isomorfo a $H_{k',m',0}$ con $k = k'$ y $m = m'$. Para ello vamos a suponer que G es isomorfo a cada uno de los grafos anteriores y obtendremos una contradicción en todos los casos, excepto en el anteriormente mencionado. Por el Lema 7.9, sabemos que $l_{H_{k,m,0}} = l_G$ y que el número de vértices y el número de ciclos esenciales de longitud mínima tiene que coincidir en ambos grafos. El procedimiento que vamos a seguir es análogo al de los grafos localmente cudriculados (Capítulo 5). Comparamos $H_{k,m,0}$ con cada uno de los grafos dados en el listado anterior. En los casos en los cuales la longitud mínima de los ciclos esenciales o el número de ciclos esenciales de dicha longitud no coincida en ambos grafos , tendremos demostrado que los grafos considerados no son Tutte equivalentes. Si ambas cantidades coinciden utilizaremos otros métodos para demostrar el resultado deseado.

Caso 1. $G \cong H_{k',m',0}$ con $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',0}} = 2k'$, $k < m+1$ y $k' < m'+1$.

Por el Lema 7.9 es fácil comprobar que $k = k'$ y $m = m'$. Hay dos casos más, análogos a este. El primero de ellos es $G \cong H_{k',m',0}$ con $k > m+1$ y $k' > m'+1$; y el segundo es $G \cong H_{k',m',0}$ con $k = m+1$ y $k' = m'+1$.

Caso 2. $G \cong H_{k',m',0}$ con $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',0}} = 2(m'+1)$, $k < m+1$ y $k' > m'+1$.

Sabemos que tienen que coincidir el número de vértices de ambos grafos, la longitud mínima de los ciclos esenciales y el número de ciclos de dicha longitud. Por tanto:

$$\begin{aligned} k(m+1) &= k'(m'+1) \\ 2k &= 2(m'+1) \\ m+1 &= k'\begin{pmatrix} m'+1 \\ \lfloor (m'+1)/2 \rfloor \end{pmatrix} \end{aligned}$$

A partir de estas expresiones se deduce que $k' = k'\begin{pmatrix} m'+1 \\ \lfloor (m'+1)/2 \rfloor \end{pmatrix}$ y obtenemos la contradicción deseada.

Análogo a este caso es $G \cong H_{k',m',0}$ con $k > m+1$ y $k' < m'+1$.

Caso 3. Supongamos $G \cong H_{k',m',0}$ con $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',0}} = 2k' = 2(m'+1)$, $k < m+1$ y $k' = m'+1$ (los casos $k = m+1$ y $k' < m'+1$; $k > m+1$ y $k' = m'+1$; $k = m+1$ y $k' > m'+1$ se prueban de forma análoga).

Por la igualdad del número de vértices y de la longitud mínima de los ciclos esenciales, obtenemos $k = k' = m' + 1 = m + 1$ pero $k < m + 1$.

Con estos casos hemos demostrado que $T(H_{k,m,0}; x, y) \neq T(H_{k',m',0}; x, y)$ si $k \neq k'$ ó $m \neq m'$. A continuación realizamos las comparaciones correspondientes a $H_{k',m',r}$ con $r > 0$.

Caso 4. $G \cong H_{k',m',r}$ con $r > 0$, $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',r}} = 2k'$, $k < m + 1$ y $k' < m' + 1$.

Como resultado del Lema 7.9, $k = k'$ y $m = m'$. Vamos a demostrar que el número de conjuntos de aristas de rango $2(m + 1) - 1$ y tamaño $2(m + 1)$ es diferente en cada grafo. Esto nos llevaría a una contradicción ya que este número es el coeficiente de $x^{2(m+1)-1}y^{2(m+1)}$ del polinomio generador de rango-tamaño.

Si $H_{k,m,0}$ tiene $k\begin{pmatrix} m+1 \\ \lfloor (m+1)/2 \rfloor \end{pmatrix} + x$ ciclos esenciales de longitud $2(m + 1)$, entonces $H_{k,m,r}$ tiene $x + \begin{pmatrix} m+1 \\ \lfloor (m+1)/2 - r \rfloor \end{pmatrix}$ de estos ciclos, siendo x el número de ciclos esenciales de longitud $2(m+1)$ que generan las $m+1$ aristas exteriores que dan lugar a los ciclos esenciales de longitud mínima $2k$. Por tanto, si demostramos la existencia de una biyección entre los conjuntos de aristas de rango $2(m+1)-1$ y tamaño $2(m + 1)$ que no sean ciclos esenciales, tendremos probado el resultado deseado.

Para cada α con $0 \leq \alpha \leq m - 1$, denotamos por A_α al conjunto de aristas que une un vértice del tipo (α, x) con un vértice del tipo $(\alpha + 1, x)$. Estos conjuntos están formados por k aristas.

Sea A un conjunto de aristas, no ciclo esencial, de rango $2(m + 1) - 1$ y tamaño $2(m + 1)$ en $H_{k,m,0}$. Definimos $s(A) = \min\{\alpha \in [0, m - 1]\ ;\ A \cap A_\alpha = \emptyset\}$. Es fácil comprobar que este mínimo siempre existe para todo conjunto de aristas contenido en $A(H_{k,m,0})$ ó $A(H_{k,m,r})$ que no sea un ciclo esencial.

Para cada α con $0 \leq \alpha \leq m-1$ definimos una biyección, φ_r entre los siguientes conjuntos de aristas:

$$\{A \subseteq A(H_{k,m,0}) | r(A) = 2(m + 1) - 1, |A| = 2(m + 1), s(A) = \alpha\}$$
$$\{A \subseteq A(H_{k,m,r}) | r(A) = 2(m + 1) - 1, |A| = 2(m + 1), s(A) = \alpha\}$$

Si $A \subset A(H_{k,m,0})$, $\varphi_r(A) = \cup\{\psi(\{(i, j), (i', j')\}); \{(i, j), (i', j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(i',j')\} & \text{si} \quad r+1 \leq i,i' \leq m \\ \{(i,j-2r),(i',j')\} & \text{si} \quad i=0 \text{ y } i'=m \\ \{(i,j-2r),(i',j'-2r)\} & \text{si} \quad i,i' \in [0,r] \end{cases}$$

Caso 5. $G \cong H_{k',m',r}$ con $r > 0$, $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',r}} = 2(m'+1)$, $k < m+1$ y $r < \lfloor (m'+1)/2 \rfloor < \lfloor k'/2 \rfloor$.

La demostración de este caso es análoga a la del caso 2.

Caso 6. Si $G \cong H_{k',m',r}$ con $r > 0$, $l_{H_{k',m',r}} = 2(m'+1+r-\lfloor (m'+1)/2 \rfloor)$, $k < m+1$ y $\lfloor (m'+1)/2 \rfloor \leq r \leq \lfloor k'/2 \rfloor$.

Como resultado del Lema 7.9 y de la igualdad del número de vértices tenemos:

$$k'(m'+1) = (m'+1+r-\lfloor (m'+1)/2 \rfloor)k' \binom{r+\lfloor (m'+1)/2 \rfloor}{m'} > k'(m'+1)$$

Caso 7. $G \cong H_{k',m',r}$ con $r > 0$, $l_{H_{k,m,0}} = 2(m+1)$, $l_{H_{k',m',r}} = 2(m'+1)$, $k > m+1$ y $r < \lfloor (m'+1)/2 \rfloor < \lfloor k'/2 \rfloor$.

Por la igualdad en el número de vértices y en las longitudes mínimas de ciclos esenciales, obtenemos $k = k'$ y $m = m'$. Pero entonces no se puede producir la igualdad en el número de ciclos esenciales de longitud mínima ya que:

$$k \binom{m+1}{\lfloor (m+1)/2 \rfloor} > k \binom{m+1}{\lfloor (m+1)/2 - r \rfloor}$$

Caso 8. Si $G \cong H_{k',m',r}$ con $r > 0$, $l_{H_{k',m',r}} = 2(m'+1+r-\lfloor (m'+1)/2 \rfloor)$, $k > m+1$ y $\lfloor (m'+1)/2 \rfloor \leq r \leq \lfloor k'/2 \rfloor$.

Si $m'+1$ es par (el caso impar se prueba de la misma forma), a partir de las siguientes expresiones:

$$\begin{aligned} k(m+1) &= k'(m'+1) \\ m+1 &= m'+1+r-(m'+1)/2 \\ k\binom{m+1}{\lfloor (m+1)/2 \rfloor} &= k'\binom{r+(m'+1)/2}{m'} \end{aligned}$$

deducimos que $2m+1-r = m' < r+(m'+1)/2 = m+1$ entonces $r > m$ pero como $m+1 = m'+1+r-(m'+1)/2$, se tiene que $m+1 > r+1$ luego $m > r$ y obtenemos una contradicción. El caso $G \cong H_{k',m',r}$ con $r > 0$, $k = m+1$ y $\lfloor (m'+1)/2 \rfloor \leq r \leq \lfloor k'/2 \rfloor$ se demuestra de forma análoga.

En el resto de los casos es trivial probar que no pueden coincidir el número de vértices, las longitudes mínimas de los ciclos esenciales y el número de ciclos esenciales de dicha longitud.

Con estas comparaciones, hemos demostrado que $H_{k,m,0}$ y $H_{k',m',r}$ con $r > 0$ no pueden tener el mismo polinomio de Tutte. Realizamos ahora las comparaciones más significativas correspondientes a $H_{k',m',a}$. Las comparaciones que no especificamos es debido a que, al igual que en los casos anteriores, su demostración es trivial.

Caso 9. Supongamos $G \cong H_{k',m',a}$, $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',a}} = 2k'$ con $k < m+1$ y $k' < m'+1$.

Los argumentos para probar que ambos grafos no tienen el mismo polinomio de Tutte son análogos a los del caso 1. Queremos especificar dos cosas: el número de ciclos esenciales de longitud $2(m+1)$ en cada grafo, y la biyección entre los conjuntos de aristas, no ciclos esenciales, de rango $2(m+1)-1$ y tamaño $2(m+1)$ contenidos en cada grafo.

En $H_{k,m,a}$ tenemos k aristas exteriores dando lugar a escaleras desplazadas $i \times j$ con $i = m+1-j$ y $0 \leq j \leq k < m+1$ (ver la Sección 2, Capítulo 6). Por otra parte, en $H_{k,m,0}$ cada una de las k aristas exteriores determina una escalera desplazada $(m+1)/2 \times (m+1)/2$ si m impar ó $(m+2)/2 \times m/2$ si m par. Por tanto, haciendo uso de la siguiente propiedad:

$$\binom{p}{n} < \binom{p}{\lfloor p/2 \rfloor} > \binom{p}{l} \quad \text{si} \quad n < \lfloor p/2 \rfloor < l$$

el número de ciclos esenciales de longitud $2(m+1)$ en $H_{k,m,a}$ es menor que en $H_{k,m,0}$. Definimos la biyección φ_r entre los siguientes conjuntos de aristas:

$$\{A \subseteq A(H_{k,m,0}) | r(A) = 2m+1, |A| = 2m+2, s(A) = \alpha\}$$
$$\{A \subseteq A(H_{k,m,a}) | r(A) = 2m+1, |A| = 2m+2, s(A) = \alpha\}$$

Si $A \subset A(H_{k,m,0})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(i',j')\} & i,i' \in [r+1,m] \\ \{(i,2k-1-j+3),(i',j')\} & i=0 \text{ y } i'=m \\ \{(i,2k-1-j+3),(i',2k-1-j'+3)\} & i,i' \in [0,r] \end{cases}$$

Caso 10. Supongamos $G \cong H_{k',m',a}$, $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',a}} = 2(m'+1)$ con $k < m+1$ y $k' > m'+1$.

Por el Lema 7.9, $k = m'+1$ y $k' = m+1$. Vamos a demostrar que hay más conjuntos de aristas de rango $2k+2$ y tamaño $2k+3$ en $H_{k,m,0}$ que en $H_{k',m',a}$. Estos conjuntos se pueden clasificar en tres grupos:

1.- Ciclos esenciales de longitud $2k+3$.

2.- Conjuntos de aristas normales.

3.- Conjuntos conteniendo un ciclo esencial de longitud $2k$ y tres aristas más (Figura 7.5).

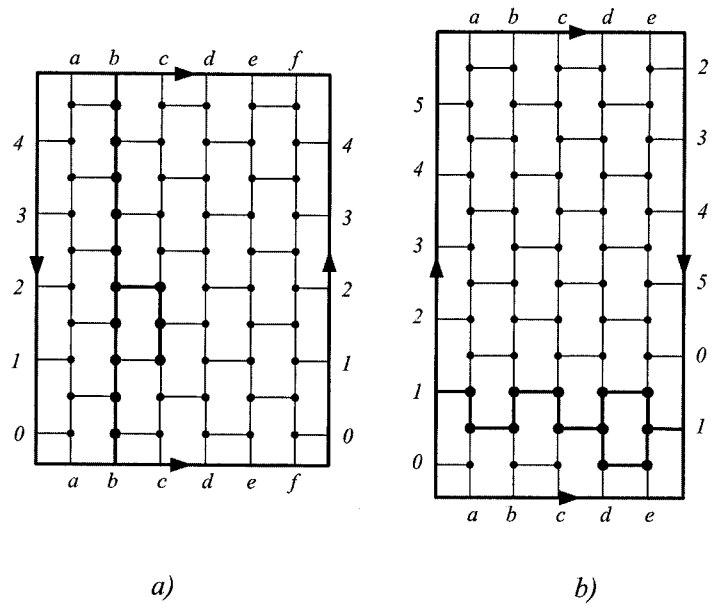


Figura 7.5: *a)* Ciclo esencial de longitud $2k$ más tres aristas en $H_{k,m,0}$ b) Ciclo esencial de longitud $2k = 2(m'+1)$ más tres aristas en $H_{k',m',a}$

(1) Por el Lema 6.1, $H_{k,m,0}$ y $H_{k',m',a}$ son bipartitos (tienen número cromático igual a 2), entonces no pueden tener ciclos esenciales de longitud impar.

(2) Por el Corolario 7.10 sabemos que $H_{k,m,0}$ y $H_{k',m',a}$ tienen el mismo número de conjuntos de aristas normales de rango $2k+2$ y tamaño $2k+3$ que no contienen ciclos de longitud 6. Vamos a demostrar que el número de conjuntos normales de aristas de rango $2k+2$ y tamaño $2k+3$ conteniendo un ciclo de longitud 6 es mayor en $H_{k,m,0}$ que en $H_{k',m',a}$. No podemos demostrar directamente con el

Corolario 7.10 que este número es igual en ambos grafos, ya que $H_{k',m',a}$ contiene conjuntos prohibidos de tamaño $2k+3$, que contienen un ciclo esencial de longitud $2m'+2$ y un ciclo de longitud 6 (ver Figura 7.5b).

Otra vez por el Corolario 7.10, el número de conjuntos normales de aristas de rango $2k+1$ y tamaño $2k+2$ conteniendo un hexágono es el mismo en ambos grafos, llamémosle s_{2k+1}. Añadimos una arista a cada uno de estos conjuntos a fin de obtener un conjunto de tamaño $2k+3$. Este conjunto es de uno de los siguientes tipos, dependiendo de como estemos añadiendo la arista:

(A) Un conjunto normal de aristas de rango $2k+2$.

(B) Un conjunto normal de aristas conteniendo dos ciclos contractibles (no esenciales) y por tanto de rango $2k+1$.

(C) Un conjunto de aristas conteniendo un ciclo esencial de longitud $2k$ y un ciclo de longitud 6.

Sean $A(H)$, $B(H)$ y $C(H)$ (donde H es $H_{k,m,0}$ ó $H_{k',m',a}$) el número de conjuntos de aristas de H que pertenecen a los grupos A, B y C respectivamente. $A(H)+B(H)+C(H)$ es el número de posibilidades de añadir una arista a los conjuntos de aristas de tamaño $2k+2$ conteniendo un hexágono y tal que la arista añadida no pertenezca a un hexágono fijado.

$$A(H)+B(H)+C(H)=\frac{s_{2k+1}(3k(m+1)-2k-2)}{2k+3-6}$$

Veamos a continuación que $(2k-3)B(H)=\sum_{B\in B(H)}(2k+3-\delta(B))$, donde $\delta(B)$ es el número de aristas de B que no pertenecen a todos los ciclos de longitud 6 de B. Para obtener un conjunto de aristas $B\in B(H)$ añadimos una arista a un conjunto de aristas de tamaño $2k+2$ que contiene un hexágono, formando un nuevo ciclo contractible. Como el ciclo es contractible, podemos considerar que la arista añadida tiene que pertenecer a un nuevo hexágono. Por tanto:

$$B(H)=\frac{\sum_{B\in B(H)}(2k+3-\delta(B))}{2k+3-6}$$

Si aplicamos el Corolario 7.10 para cada $B\in B(H)$, el número de conjuntos normales de aristas de tamaño $2k+3-\delta(B)$ es igual en $H_{k,m,0}$ y $H_{k,m,a}$ entonces:

$$\sum_{B\in B(H_{k,m,0})}(2k+3-\delta(B))=\sum_{B\in B(H_{k',m',a})}(2k+3-\delta(B))$$

A partir de todas las expresiones obtenidas y utilizando que $C(H_{k,m,0})=0$ y $C(H_{k',m',a})\neq 0$ (Figura 7.5) tenemos demostrado que $A(H_{k,m,0})>A(H_{k',m',a})$.

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

(3) En $H_{k,m,0}$, cada ciclo esencial de longitud $2k$ más tres aristas tiene rango $2k+2$ (Figura 7.5a), pero en $H_{k',m',a}$ hay ciclos esenciales en los cuales si añadimos tres aristas obtenemos conjuntos de aristas de rango $2k+1$ (Figura 7.5b). Por hipótesis, ambos grafos tienen el mismo número de ciclos esenciales de longitud mínima por tanto el número de conjuntos de aristas en este caso es mayor en $H_{k,m,0}$ que en $H_{k',m',a}$.

Caso 11. Supongamos $G \cong H_{k',m',a}$, $l_{H_{k,m,0}} = 2(m+1)$, $l_{H_{k',m',a}} = 2(m'+1)$ con $k > m+1$ y $k' > m'+1$.

Como $m+1 = m'+1$ y $k(m+1) = k'(m'+1)$ entonces $k = k'$. Por la igualdad del número de ciclos esenciales de longitud mínima y haciendo uso de la propiedad especificada en el caso 9 tenemos que:

$$k\binom{m+1}{\lfloor (m+1)/2 \rfloor} = 2^{m+1} = \sum_{j=0}^{m+1}\binom{m+1}{j} <$$

$$< (m+1)\binom{m+1}{\lfloor (m+1)/2 \rfloor} < k\binom{m+1}{\lfloor (m+1)/2 \rfloor}$$

Igual que este caso se demuestra el caso en que $G \cong H_{k',m',a}$, $l_{H_{k,m,0}} = 2(m+1)$, $l_{H_{k',m',a}} = 2(m'+1)$ con $k = m+1$ y $k' = m'+1$.

Caso 12. Supongamos $G \cong H_{k',m',b}$, $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',b}} = 2k'$ con $k < m+1$ y $k' < m'+1$.

Probamos que hay más conjuntos de aristas de rango $2m+1$ y tamaño $2m+2$ en $H_{k,m,0}$ que en $H_{k',m',b}$. Como en el caso 1, se demuestra que el número de ciclos esenciales de longitud $2(m+1)$ en $H_{k,m,b}$ es menor que en $H_{k,m,0}$. La biyección φ_r entre:

$$\{A \subseteq A(H_{k,m,0}) | r(A) = 2m+1, |A| = 2m+2, s(A) = \alpha\}$$
$$\{A \subseteq A(H_{k,m,b}) | r(A) = 2m+1, |A| = 2m+2, s(A) = \alpha\}$$

se define de la siguiente forma:

Si $A \subset A(H_{k,m,0})$, $\varphi_r(A) = \cup\{\psi(\{(i,j),(i',j')\}); \{(i,j),(i',j')\} \in A\}$ donde

$$\psi(\{(i,j),(i',j')\}) = \begin{cases} \{(i,j),(i',j')\} & \text{si} \quad r+1 \leq i, i' \leq m \\ \{(i,2k-j),(i',j')\} & \text{si} \quad i = 0 \text{ y } i' = m \\ \{(i,2k-j),(i',2k-j')\} & \text{si} \quad i, i' \in [0,r] \end{cases}$$

Caso 13. Supongamos $G \cong H_{k',m',b}$, $l_{H_{k,m,0}} = 2k$, $l_{H_{k',m',b}} = 2(m'+1)$ con $k < m+1$ y $k' > m'+1$.

Usamos los mismos argumentos que en el caso 10 para probar que $H_{k,m,0}$ tiene más conjuntos de aristas de rango $2k+2$ y tamaño $2k+3$ que $H_{k',m',b}$.

Caso 14. $G \cong H_{k',m',b}$, $l_{H_{k,m,0}} = 2(m+1)$, $l_{H_{k',m',b}} = 2(m'+1)$ con $k > m+1$ y $k' > m'+1$.

Como $m+1 = m'+1$ y $k(m+1) = k'(m'+1)$ entonces $k = k'$. Por la igualdad del número de ciclos esenciales de longitud mínima tenemos que:

$$k\binom{m+1}{(m+1)/2} = 2\binom{m+1}{(m+1)/2} + 4\sum_{j=1}^{\lfloor (m+1)/4 \rfloor}\binom{m+1}{(m+1)/2-2j} <$$

$$< (m+3)\binom{m+1}{(m+1)/2} \leq k\binom{m+1}{(m+1)/2}$$

La última desigualdad es consecuencia de que k es par, m es impar y $k > m+1$, luego $k \geq m+3$. Si $G \cong H_{k',m',b}$ con $k = m+1$ y $k' = m'+1$ se obtiene una contradicción de forma análoga.

Hay tres comparaciones correspondientes a $H_{k',m',h}$ con $k' < m'-1$. De ellas dos son triviales y la tercera, que sería considerar $k < m+1$ se demuestra siguiendo el mismo proceso que en el caso 10, probando que $H_{k,m,0}$ tiene más conjuntos de aristas de rango $2k+2$ y tamaño $2k+3$ que $H_{k',m',h}$. □

A continuación y al igual que hicimos en el Capítulo 5, relativo a los grafos localmente cuadriculados, ofrecemos una tabla en la que el lector podrá ver fácilmente el motivo por el cuál $H_{k,m,0}$ no tiene el mismo polinomio de Tutte que cualquier otra teselación hexagonal no isomorfa a él.

I significa que en los grafos correspondientes no pueden coincidir el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud.

II significa que aunque en los grafos correspondientes puedan coincidir las cantidades anteriores, dichos grafos no tienen el mismo número de conjuntos de aristas de rango s y tamaño $s+1$. En cada casilla especificaremos el valor de s.

III significa que ambos grafos no tienen el mismo número cromático.

Como ejemplo, $H_{10,12,0}$ y $H_{10,11,a}$ no tienen el mismo polinomio de Tutte, porque el elemento de la fila 9, columna 1 es I, lo cuál quiere decir que no pueden coincidir el número de vértices, la longitud mínima de los ciclos esenciales y el número de ciclos esenciales de dicha longitud en ambos grafos.

	$H_{k,m,0}$	$k < m+1$	$k = m+1$	$k > m+1$
$H_{k',m',0}$	$k' < m'+1$	I	I	I
	$k' = m'+1$	I	I	I
	$k' > m'+1$	I	I	I
$H_{k',m',r}$ $r > 0$	$k' < m'+1$	$s = 2m+1$	I	I
	$r < \lfloor (m'+1)/2 \rfloor < \lfloor k'/2 \rfloor$	I	I	I
	$\lfloor (m'+1)/2 \rfloor \leq r \leq \lfloor k'/2 \rfloor$	I	I	I
	$k' = m'+1$	I	I	I
$H_{k',m',a}$	$k' < m'+1$	$s = 2m+1$	I	I
	$k' = m'+1$	I	I	I
	$k' > m'+1$	$s = 2k+2$	I	I
$H_{k',m',b}$	$k' < m'+1$	$s = 2m+1$	I	I
	$k' = m'+1$, $k' > 4$	I	I	I
	$k' > m'+1$	$s = 2k+2$	I	I
	$k' = 4$, $m' = 3$	I	I	I
$H_{k',m',c}$		III	III	III
$H_{k',m',f}$		III	III	III
$H_{k',m',g}$		III	III	III
$H_{k',m',h}$		$s = 2k+2$	I	I

Por último, queremos resaltar que la técnica empleada para demostrar la Tutte unicidad de la teselación hexagonal toroidal se puede aplicar de forma análoga a cualquier otra teselación hexagonal. Además esta técnica se puede reproducir en los grafos localmente C_6 calculando las longitudes mínimas de los ciclos esenciales y el número de ciclos esenciales de dicha longitud en esta familia de grafos. Debido a que desde el punto de vista matemático no aportamos nada nuevo reproduciendo esta técnica hemos iniciado la búsqueda de una técnica distinta para ofrecer una demostración de la Tutte unicidad de los grafos localmente C_6.

CAPÍTULO 8

Conclusiones y Problemas Abiertos

Con la demostración de la Tutte unicidad de los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6, es la primera vez que se prueba que existen grandes familias de grafos Tutte únicas. Como ya hemos comentado, la Tutte unicidad en matroides ha sido muy estudiada y se conocen grandes familias de matroides Tutte únicas. En el caso de los grafos, se ha estudiado la Tutte unicidad de algunas familias como los grafos multipartitos completos, hipercubos etc [31]; pero los grafos localmente cuadriculados, las teselaciones hexagonales y los grafos localmente C_6 se presentan como las primeras grandes familias de grafos Tutte únicas.

Nos parece importante señalar que las dos familias de teselaciones hexagonales construidas a partir de las estructuras definidas como cilindros hexagonales torcidos, han completado el estudio realizado por Thomassen acerca de esta familia de grafos. Además, la clasificación de las teselaciones hexagonales permite la clasificación de los grafos localmente C_6; estableciéndose una relación de menor muy potente entre las tres familias de grafos estudiadas en esta parte de la memoria.

Queremos resaltar que aunque, en el Capítulo 7, hemos demostrado únicamente la Tutte unicidad de la teselación hexagonal toroidal, la técnica desarrollada permite la demostración de la Tutte unicidad del resto de las teselaciones hexagonales. Además, esta técnica se puede reproducir para los grafos localmente C_6, calculando las longitudes mínimas de los ciclos esenciales y el número de ciclos de dicha longitud. Pero nos gustaría ir más allá y demostrar la Tutte unicidad de estos grafos aprovechando la relación de dualidad que mantiene con las teselaciones hexagonales. En otras palabras, pensamos que al igual que en grafos planos, en los cuales se demuestra que $T(G; x, y) = T(G^*; y, x)$, podemos probar un relación

BIBLIOTECA

del tipo:

$$T(G;x,y) = f(x,y)T(G^*,x,y)$$

siendo G un grafo localmente cuadriculado, una teselación hexagonal o un grafo localmente C_6; y f una función en dos variables que no dependa del grafo. Esta creencia se basa en el hecho de que todos los grafos pertenecientes a estas tres familias son localmente planos.

Por otra parte, demostramos que los grafos localmente cuadriculados son Tutte único para $p, q \geq 6$ pero nuestra técnica no se puede aplicar a los casos $p = 3, 4, 5$. Además, sería interesante demostrar que $p' \begin{pmatrix} q' + \delta - 1 \\ \delta \end{pmatrix}$ no es una potencia de 2. Esta demostración nos proporcionaría un resultado más general de la Tutte unicidad de $T^{\delta}_{p,q}$ y $S_{p,q}$.

APÉNDICE A

Instrucciones para el uso de los programas poltutte, poltutte2 y poltutte3

En este apéndice se describe el uso de los programas **poltutte**, **poltutte2** y **poltutte3** que sirven para generar fórmulas recursivas para calcular el polinomio de Tutte asociado a fragmentos de dimensión $m \times n$ de teselaciones del plano mediante cuadrados, triángulos, hexágonos y combinaciones de estos polígonos regulares.

§ **A.0.1. Distribución.**— Los archivos que componen este paquete están almacenados en el archivo **tutte.tar**. Se debe instalar en un ordenador con el Sistema Operativo Linux. Ha sido desarrollado para la versión Ret Hat 7.1 o superior y Fedora versiones 1 y 2. No creemos que haya ningún problema para ejecutarlo en otras versiones.

Para instalarlo se crea un directorio nuevo y se copia el archivo **tutte.tar**. A continuación se ejecuta el comando:

```
tar xf tutte.tar
```

Los archivos más importantes son:

poltutte - Programa para encontrar fórmulas recursivas para calcular el polinomio de Tutte de un fragmento de teselación cuyo grafo base tenga una sola capa. No se toman en cuenta los grafos isomorfos que se presentan durante el proceso de contración y borrado.

poltutte2 - Programa para encontrar fórmulas recursivas para calcular el polinomio de Tutte de un fragmento de teselación cuyo grafo base tenga una sola capa. Se toman en cuenta algunos grafos isomorfos que se presentan al realizar las operaciones de contracción y borrado, de manera que se reduce significativamente el tiempo de evaluación.

poltutte3 - Programa para encontrar fórmulas recursivas para calcular el polinomio de Tutte de un fragmento de teselación cuyo grafo base tenga varias capas.

genegrid - Programa para generar la codificación del grafo base de una malla cuadrada $m \times n$ con un número de bandas dado como parámetro. Un archivo A que se obtenga como salida de este programa se pueden utilizar como entrada para el programa **poltutte2**. Si en A se elimina manualmente la línea 2 (que es donde tenemos en cuenta los grafos isomorfos producidos al borrar y contraer) entonces el archivo resultante puede ser utilizado como entrada al programa **poltutte** (el proceso se vuelve mucho más lento que cuando se usa **poltutte2**).

gene3grid - Programa para generar la codificación del grafo base de una malla triangular $m \times n$ con un número de bandas dado como parámetro, obtenida al subdividir los cuadrados de la malla cuadrada. Un archivo A que se obtenga como salida de este programa puede ser utilizado como entrada para el programa **poltutte** o **poltutte2**.

genegrida - Programa ejecutable que genera la codificación del grafo base de una malla hexagonal $m \times n$. El número de bandas se da como parámetro. Esta estructura también se puede determinar a partir de la estructura de las teselaciones del tipo B generadas por el programa **genegridb**.

genegridb - Programa ejecutable que genera fragmentos de teselaciones de tipo B (el número de bandas se da como parámetro). Las teselaciones de tipo B (Figura 2.4) son aquellas que pueden utilizar los tres tipos de polígonos regulares: cuadrados, triángulos y hexágonos. Un archivo que se obtenga como salida de este programa (o del programa genegrida) puede ser utilizado como entrada para el programa **poltutte3.**

Los archivos fuente de estos programas son poltutte3.c, genegrida.c y genegridb.c

§ **A.0.2. Compilación.**— En principio no es necesario compilar los programas porque ya se están incluyendo los ejecutables; sin embargo, para hacer alguna modificación se deben dar los siguientes comandos (el directorio de trabajo debe ser aquél donde se hizo la instalación):

- Para compilar poltutte:
  gcc -g -I./include -o poltutte poltutte.c pg.a
- Para compilar poltutte2:
  gcc -g -I./include -o poltutte2 poltutte2.c pg.a
- Para compilar poltutte3:
  gcc -g -I./include -o poltutte3 poltutte3.c pg.a
- Para compilar genegrid:
  gcc -g -o genegrid genegrid.c
- Para compilar gene3grid:
  gcc -g -o gene3grid gene3grid.c
- Para compilar genegrida:
  gcc -g -o genegrida genegrida.c
- Para compilar genegridb:
  gcc -g -o genegridb genegridb.c

§ **A.0.3. Ejemplos.**— A continuación se muestran ejemplos sobre el uso de los programas. En todos los casos el número 3 que aparece en "-s 3" puede ser reemplazado por cualquier otro número entero mayor que cero, esta cantidad indica el número de bandas en el fragmento de la teselación.

- Genera las fórmulas de recursión del polinomio de Tutte de una malla cuadrada con tres bandas:

  ```
  ./genegrid -s 3 | ./poltutte2
  ```

- Genera las fórmulas de recursión del polinomio de Tutte de una malla triangular con tres bandas:

  ```
  ./gene3grid -s 3 | ./poltutte
  ```

- Genera las fórmulas de recursión del polinomio de Tutte de una malla hexagonal con tres bandas:

```
./genegrida -s 3 | ./poltutte3
```

- Genera las fórmulas de recursión del polinomio de Tutte de un fragmento de teselación del tipo B con tres bandas:

```
./genegridb -s 3 | ./poltutte3
```

§ A.0.4. Codificación de teselaciones planas.— Los programas **genegrid**, **gene3grid**, **genegrida** y **genegridb** generan algunos fragmentos de las teselaciones que pueden procesar los programas para construir las fórmulas recursivas del Polinomio de Tutte.

Los fragmentos de las teselaciones se pueden codificar de manera manual en un archivo de texto siguiendo el formato que se explica en esta sección o, como en el caso de los programas **genegrid**, **gene3grid**, **genegrida** y **genegridb**, se puede programar la codificación considerando la estructura del grafo base del fragmento de la teselación.

Cada uno de los programas **poltutte**, **poltutte2** y **poltutte3** tiene su propio esquema de codificación, pero en principio, éstos son parecidos entre sí. El esquema de codificación más amplio es el del programa **poltutte3** y es el que presentamos primero.

Para codificar un fragmento de una teselación plana primero tiene que determinarse su grafo base; por ejemplo, la teselación plana que aparece en la Figura 2.4 tiene como base la estructura de capas mostrada en la Figura 2.5. Esta base contiene cuatro capas y las regiones que se han sombreado son las bandas, así la Figura 2.5 contiene 3 bandas.

Lo que hace el programa **poltutte3** es generar las ecuaciones recursivas para evaluar el polinomio de Tutte de un fragmento de teselación plana con un cierto número de capas y un cierto número de bandas. Para codificar dicho fragmento se especifican cada una de las capas de su grafo base de la siguiente forma: (lo que sigue es la información que debe contener el archivo de especificación)

$$V\ C\ P \left\}\begin{array}{l} V = \text{Número máximo de vértices en una capa} \\ P = \text{Número máximo de aristas paralelas en un menor (Cuando no} \\ \text{se sepa qué poner en este parámetro coloque cualquier número (1,2,} \\ \text{3 ó 4 preferentemente) e incremente hasta que el programa funcione)} \\ C = \text{Número de capas} \end{array}\right.$$

- El punto es un separador. Luego se codifica la primera capa

$$\begin{array}{l} E\ S \\ P_1\ P_2\ \ldots\ P_r \\ v_1\ w_1 \\ v_2\ w_2 \\ \vdots \\ v_k\ w_k \end{array} \left\}\begin{array}{l} E = \text{Índice del último vértice en la parte inferior de la capa} \\ S = \text{Índice del primer vértice en la parte superior de la capa} \\ P_1 \ldots P_r = \text{Permutación de índices de los vértices de la capa} \\ (v_1, w_1), (v_2, w_2), \ldots, (v_k, w_k) = \text{aristas en la capa} \end{array}\right.$$

- } Aquí se codifica la segunda capa

- } Aquí se codifica la tercera capa

$\vdots$

Con respecto a la permutación de índices $P_1 \ldots P_r$ queremos aclarar que es una información que indica el movimiento (renombrado) de los vértices durante el proceso de contracción y borrado y que tiene que cumplir que la etiquetación de vértices obtenida mediante la permutación tiene que dar lugar a la misma estructura que teníamos con la etiquetación original. Esta permutación no tiene por qué existir. Por ejemplo, la codificación del grafo base mostrado en la Figura 2.5 con una banda es la siguiente:

$$8\ 3\ 4 \left\}\begin{array}{l} \text{En cada capa a lo más hay 8 vértices} \\ \text{El número máximo de aristas paralelas es menor o igual a 3} \\ \text{El número de capas es cuatro} \end{array}\right.$$

Las capas se cuentan de abajo hacia arriba.

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

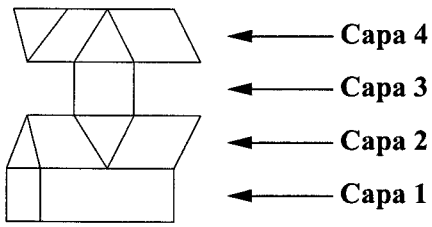


Figura A.1: Capas del grafo base de la Figura 2.5 con una banda

(1) CODIFICACIÓN DE LA CAPA 1.

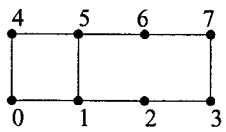


Figura A.2: Capa 1 del grafo base de la Figura 2.5 con una banda

3 4
−1 −1 −1 −1 −1 −1 −1 −1
0 1
1 2
2 3
4 5
5 6
6 7
0 4
1 5
3 7

La primera línea corresponde a las etiquetaciones del último vértice de la parte inferior de la capa y del primer vértice de la parte superior de la capa. Los -1's significan que no hay permutación. El resto de las líneas son la codificación de las aristas de la capa.

De forma análoga codificamos las demás capas.

(2) CODIFICACIÓN DE LA CAPA 2.

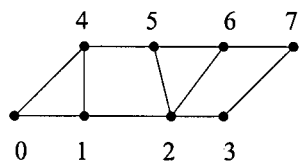


Figura A.3: Capa 2 del grafo base de la Figura 2.5 con una banda

3 4
−1 −1 −1 −1 −1 −1 −1 −1
0 1
1 2
2 3
4 5
5 6
6 7
0 4
1 4
2 5
2 6
3 7

(3) CODIFICACIÓN DE LA CAPA 3.

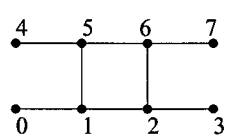


Figura A.4: Capa 3 del grafo base de la Figura 2.5 con una banda

3 4
−1 −1 −1 −1 −1 −1 −1 −1
0 1
1 2
2 3
4 5
5 6
6 7
1 5
2 6
2 5
2 6

(4) CODIFICACIÓN DE LA CAPA 4.

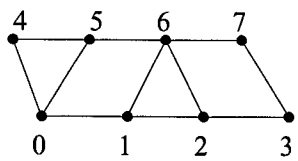


Figura A.5: Capa 4 del grafo base de la Figura 2.5 con una banda

3 4
−1 −1 −1 −1 −1 −1 −1 −1
0 1
1 2
2 3
4 5
5 6
6 7
0 4
0 5
1 6
2 6
3 7

Este archivo puede codificarse manualmente o generarse con el comando

```
./genegridb -s 1
```

El archivo que se obtiene es:

```
8 3 4
.
3 4
-1 -1 -1 -1 -1 -1 -1 -1
0 1
1 2
2 3
4 5
5 6
6 7
0 4
1 5
3 7
.
3 4
-1 -1 -1 -1 -1 -1 -1 -1
0 1
1 2
2 3
4 5
5 6
6 7
0 4
1 4
2 5
2 6
3 7
.
3 4
-1 -1 -1 -1 -1 -1 -1 -1
0 1
1 2
```

```
2 3
4 5
5 6
6 7
1 5
2 6
.
3 4
-1 -1 -1 -1 -1 -1 -1 -1
0 1
1 2
2 3
4 5
5 6
6 7
0 4
0 5
1 6
2 6
3 7
```

Para el programa **poltutte2** el formato es parecido aunque más simple porque sólo trabaja con fragmentos de teselaciones cuyo grafo base tenga una sola capa. El grafo base de la malla cuadrada $L_{3,n}$ (Figura A.6) se codifica así:

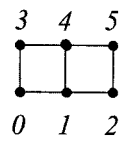


Figura A.6: Grafo base de $L_{3,n}$

```
6 2 2 3
2 1 0 5 4 3
1 0
4 3
1 4
2 1
5 4
```

2 5
0 3

En la primera línea el 6 indica el número de vértices del grafo base, el primer 2 corresponde al número máximo de aristas paralelas en un menor, el segundo 2 se asocia con el último vértice en la parte inferior del grafo base y el 3 con el primer vértice en la parte superior del grafo base. En la línea 2 se codifica una permutación de los vértices de la capa. A partir de la tercer línea se codifican las aristas del grafo base.

Para el programa **poltutte** la codificación es la misma, salvo que la permutación en la línea 2 no debe ser incluida para no tener en cuenta la existencia de grafos isomorfos en el proceso de contracción y borrado.

APÉNDICE B

Listados

A continuación ofrecemos un listado que incluye los siguientes programas:

(1) PROGRAMAS PARA LA GENERACIÓN DE TESELACIONES
genegrid
gene3grid
genegrida
genegridb
(2) PROGRAMAS PARA EL CÁLCULO DEL POLINOMIO DE TUTTE
poltutte
poltutte2
poltutte3
(3) ARCHIVOS NECESARIOS PARA (1) y (2)

A23.H	**ORDEN.H**
AB.H	**UDE.H**
ARCH.H	**LISTAS**$-D1$**.H**
BTREE.C	**GRAF.H**
BCC.H	**PERM.H**
COLA.H	**PGERROR.H**
COMB.H	**PILA.H**
MARCAS.H	**TIEM.C**
LD.H	**UTIL.H**
MC.H	

PROGRAMA genegrid PARA GENERAR UNA MALLA CUADRADA

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
  int opt;
  int size;
  int i;

  while ((opt = getopt(argc, argv, "s:")) != -1)
    switch (opt) {
    case 's':
      size = atoi(optarg);
      if (size < 0) {
    printf("Tamaño de la malla fuera de rango\n");
    exit(1);
      }
      break;
    }

  printf("%d %d %d %d\n", size*2, 3, size-1, size);
  for (i = 1; i < size; i++) {
    printf("%d %d\n",i,i-1);
    printf("%d %d\n",size+i, size+i-1);
    printf("%d %d\n",i,i+size);
    printf("%d %d\n",i,size+i-1);
  }
  printf("0 %d\n",size);
  exit(0);
}

/*
gcc -g -o genegrid genegrid.c
*/
```

PROGRAMA gene3grid PARA GENERAR UNA MALLA TRIANGULAR

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
  int opt;
  int size;
  int i;

  while ((opt = getopt(argc, argv, "s:")) != -1)
    switch (opt) {
    case 's':
      size = atoi(optarg);
      if (size < 0) {
    printf("Tamaño de la malla fuera de rango\n");
    exit(1);
      }
      break;
    }

  printf("%d %d %d %d\n", size*2, 2, size-1, size);
  for (i = size-1; i >= 0; i--)
    printf("%d ",i);
  for (i = 2*size-1; i>= size; i--)
    printf("%d ",i);
  printf("\n");
  for (i = 1; i < size; i++) {
    printf("%d %d\n",i,i-1);
    printf("%d %d\n",size+i, size+i-1);
    printf("%d %d\n",i,i+size);
  }
  printf("0 %d\n",size);
  exit(0);
}
/*
gcc -g -o genegrid genegrid.c
```

PROGRAMA genegrida PARA GENERAR UNA MALLA HEXAGONAL

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
  int opt;
  int size;
  int i;

  while ((opt = getopt(argc, argv, "s:")) != -1)
    switch (opt) {
    case 's':
      size = atoi(optarg);
      if (size < 0) {
    printf("Tamaño de la malla fuera de rango\n");
    exit(1);
      }
      break;
    }

  printf("%d 3 2\n", size*2);
  printf(".\n");
  printf("%d %d\n", size-1, size);
  for (i = 0; i < size*2; i++)
    printf("%d ",-1);
  printf("\n");
  for (i = 1; i < size; i++) {
    printf("%d %d\n",i,i-1);
    printf("%d %d\n",size+i, size+i-1);
    printf("%d %d\n",i+size,i);
    if (i % 2 == 0)
      printf("%d %d\n",i-1+size,i);
  }
  printf("0 %d\n",size);
  printf(".\n");
  printf("%d %d\n", size-1, size);
  for (i = 0; i < size*2; i++)
    printf("%d ",-1);
  printf("\n");
  for (i = 1; i < size; i++) {
    printf("%d %d\n",i,i-1);
    printf("%d %d\n",size+i, size+i-1);
    printf("%d %d\n",i+size,i);
    if (i % 2 == 1)
      printf("%d %d\n",i-1,i+size);
  }
```

ESCUELA TÉCNICA SUPERIOR INGENIEROS BIBLIOTECA

```
  printf("0 %d\n",size);
  exit(0);
}

/*
gcc -g -o genegrida genegrida.c


*/
```

PROGRAMA gnegridb PARA GENERAR TESELACIONES DEL TIPO B

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
  int opt;
  int size;
  int i;

  while ((opt = getopt(argc, argv, "s:")) != -1)
    switch (opt) {
    case 's':
      size = atoi(optarg);
      if (size < 0) {
    printf("Tamaño de la malla fuera de rango\n");
    exit(1);
      }
      break;
    }

  printf("%d %d 4\n", size*8, size*2);

  printf(".\n");
  printf("%d %d\n", size*3, size*3+1);
  for (i = 0; i < size*8; i++)
    printf("%d ",-1);
  printf("\n");
  for (i = 0; i < size; i++) {
    printf("%d %d\n",i*3,          i*3+1);
    printf("%d %d\n",i*3+1,        i*3+2);
    printf("%d %d\n",i*3+2,        i*3+3);
    printf("%d %d\n",(i+size)*3+1, (i+size)*3+2);
    printf("%d %d\n",(i+size)*3+2, (i+size)*3+3);
```

```
  printf("%d %d\n",(i+size)*3+3, (i+size)*3+4);
  printf("%d %d\n",i*3,          (i+size)*3+1);
  printf("%d %d\n",i*3+1,        (i+size)*3+2);
}
printf("%d %d\n",size*3,size*6+1);

printf(".\n");
printf("%d %d\n", size*3, size*3+1);
for (i = 0; i < size*8; i++)
  printf("%d ",-1);
printf("\n");
for (i = 0; i < size; i++) {
  printf("%d %d\n",i*3,          i*3+1);
  printf("%d %d\n",i*3+1,        i*3+2);
  printf("%d %d\n",i*3+2,        i*3+3);
  printf("%d %d\n",(i+size)*3+1, (i+size)*3+2);
  printf("%d %d\n",(i+size)*3+2, (i+size)*3+3);
  printf("%d %d\n",(i+size)*3+3, (i+size)*3+4);
  printf("%d %d\n",i*3,          (i+size)*3+1);
  printf("%d %d\n",i*3+1,        (i+size)*3+1);
  printf("%d %d\n",i*3+2,        (i+size)*3+2);
  printf("%d %d\n",i*3+2,        (i+size)*3+3);
}
printf("%d %d\n",size*3,size*6+1);

printf(".\n");
printf("%d %d\n", size*3, size*3+1);
for (i = 0; i < size*8; i++)
  printf("%d ",-1);
printf("\n");
for (i = 0; i < size; i++) {
  printf("%d %d\n",i*3,          i*3+1);
  printf("%d %d\n",i*3+1,        i*3+2);
  printf("%d %d\n",i*3+2,        i*3+3);
  printf("%d %d\n",(i+size)*3+1, (i+size)*3+2);
  printf("%d %d\n",(i+size)*3+2, (i+size)*3+3);
  printf("%d %d\n",(i+size)*3+3, (i+size)*3+4);
  printf("%d %d\n",i*3+1,        (i+size)*3+2);
  printf("%d %d\n",i*3+2,        (i+size)*3+3);
}

printf(".\n");
printf("%d %d\n", size*3, size*3+1);
for (i = 0; i < size*8; i++)
  printf("%d ",-1);
printf("\n");
for (i = 0; i < size; i++) {
  printf("%d %d\n",i*3,          i*3+1);
  printf("%d %d\n",i*3+1,        i*3+2);
  printf("%d %d\n",i*3+2,        i*3+3);
  printf("%d %d\n",(i+size)*3+1, (i+size)*3+2);
  printf("%d %d\n",(i+size)*3+2, (i+size)*3+3);
  printf("%d %d\n",(i+size)*3+3, (i+size)*3+4);
  printf("%d %d\n",i*3,          (i+size)*3+1);
```

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES BIBLIOTECA

```
    printf("%d %d\n",i*3,          (i+size)*3+2);
    printf("%d %d\n",i*3+1,        (i+size)*3+3);
    printf("%d %d\n",i*3+2,        (i+size)*3+3);
  }
  printf("%d %d\n",size*3,size*6+1);

  exit(0);
}

/*
gcc -g -o genegridb genegridb.c
*/
```

PROGRAMA POLTUTTE PARA EL CÁLCULO DEL POLINOMIO DE TUTTE

```
#include <stdlib.h>
#include <stdio.h>
#include "pgerror.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include "mc.h"
#include "a23.h"
#include "ld.h"

/* Número máximo de vértices en una capa */
int maxvertex;

/* Número máximo de aristas paralelas que pueden aparecer */
int maxparallel;

/* Estructuras de datos fundamentales */
a23 MinDictionary;     /* Diccionario de configuraciones minimales */
ld  waitingList;       /* Lista de espera de los elementos minimales
que
                          esperan el proceso de generación recursiva de
                          sus configuraciones minimales */
ld  printList;         /* Auxiliar para la impresión final del
polinomio */

mc *graphTemplate;     /* Descriptor de las matrices que guardan los
grafos */
unsigned *graphMod;    /* Módulo generador de los grafos */
int endBottom = 2;     /* Índice del último vértice en la parte
inferior
                          del grafo base*/
```

```
int startTop = 3;      /* Índice del primer vértice en la parte
superior del
                          grafo base*/

typedef struct tnode {
  int ord;             /* Etiqueta. Usado para nombrar a los grafos
minimales */
  unsigned *graph;     /* Grafo asociado al nodo */
  int x,y;             /* Potencia de x e y con la que contribuye el
nodo */
  struct tnode *con;   /* Reducción recursiva al contraer la arista
mayor */
  struct tnode *del;   /* Reducción recursiva al borrar la arista mayor
*/
} tnode;               /* Representa un nodo en el árbol de recursión
*/

/* Compara los arreglos de datos de dos matrices comprimidas */
int comp_mc(mc *e, unsigned *d1, unsigned *d2) {
  int r = 0;
  int i;
  for (i = 0; i < e->t_e; i++)
    if ((r = d1[i] - d2[i]) != 0)
      return r;
  return(0);
}

int comp_coefTutte(void *t1, void *t2) {
  tnode **p1, **p2;
  int r;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  if ((r = (*p1)->ord - (*p2)->ord) != 0)
    return(r);
  if ((r = (*p2)->x - (*p1)->x) != 0)
    return(r);
  return (*p2)->y - (*p1)->y;
}

int comp_ord(void *t1, void *t2) {
  tnode **p1, **p2;
  int r;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  return ((*p1)->ord - (*p2)->ord);
}

int comp_aptnode(void *t1, void *t2) {
  tnode **p1, **p2;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  return (comp_mc(graphTemplate, (*p1)->graph, (*p2)->graph));
}
```

```
void imp_mc(mc *template, unsigned *mat) {
  int i,j;

  for (i=0; i<maxvertex; i++) {
    for (j=0; j<maxvertex; j++)
      putchar('0'+elto_mc(template,mat,i,j));
    putchar('\n');
  }
}

void imp_tnode(tnode *t) {
  /* printf("x = %d, y = %d %s\n",t->x, t->y, t->con == NULL ? ",
terminal" : "");*/
  printf("x = %d, y = %d, ord = %d\n",t->x, t->y,t->ord);
  imp_mc(graphTemplate, t->graph);
}

/* Crea un nodo nuevo */
tnode *newtnode(unsigned *model) {
  tnode *t;
  int i,j;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL)
    exit(1);
  t->ord = 0;
  t->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->graph, graphMod);
  t->x = t->y = 0;
  t->con = t->del = NULL;
  if (model != NULL)
    for (i=0; i<=endBottom; i++)
      for (j=0; j<i; j++)
    asg_mc(graphTemplate, t->graph, elto_mc(graphTemplate, model, i,
j), i+startTop, j+startTop);
  return t;
}

/* Construye el simetrico de un nodo */
tnode *simetricNode(unsigned *model) {
  tnode *t;
  int i,j;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL)
    exit(1);
  t->ord = 0;
  t->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->graph, graphMod);
  t->x = t->y = 0;
  t->con = t->del = NULL;
  if (model != NULL)
    for (i=0; i<=endBottom; i++)
      for (j=0; j<i; j++)
    asg_mc(graphTemplate, t->graph, elto_mc(graphTemplate, model, i,
j), maxvertex-1-j, maxvertex-1-i);
```

```
  return t;
}

void dest_tnode(tnode *node) {
  destdat_mc(node->graph);
  free(node);
}

/* Duplica un tnode */
tnode *duptnode(tnode *node) {
  tnode *t;
  unsigned *aux;
  int i,j,k;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL)
    exit(1);
  t->ord = 0;
  t->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->graph, node->graph);
  t->x = t->y = 0;
  t->con = t->del = NULL;
  return t;
}

/* Borra las hojas del grafo */
int delete_pendant_edges(unsigned *graph, int noBottom) {
  int i,j;
  int suma = 0;
  int pendant_edges = 0;

  for (i = maxvertex-1; i > (noBottom ? endBottom : -1); i--) {
    suma = 0;
    for (j = 0; j < i; j++)
      suma += elto_mc(graphTemplate, graph,i,j);
    for (j = i+1; j < maxvertex; j++)
      suma += elto_mc(graphTemplate, graph,j,i);
    if (suma == 1) {
      pendant_edges++;
      for (j = 0; j < i; j++)
        if (elto_mc(graphTemplate, graph,i,j) == 1) {
      asg_mc(graphTemplate, graph,0,i,j);
      i = maxvertex;
      break;
    }
      for (j = i+1; j < maxvertex; j++)
        if (elto_mc(graphTemplate, graph,j,i) == 1) {
      asg_mc(graphTemplate, graph,0,j,i);
      i = maxvertex;
      break;
    }
    }
  }
  return(pendant_edges);
}
```

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

```
/* Borra la arista con extremos i, j */
tnode *delete(tnode *node, int i, int j, int noBottom) {
  int aux;
  tnode *t;

  if (i<j) {
    aux = i; i = j; j = aux;
  }
  t = duptnode(node);
  if (t == NULL)
    exit(1);
  asg_mc(graphTemplate, t->graph, elto_mc(graphTemplate,node-
>graph,i,j)-1,i,j);
  t->x = delete_pendant_edges(t->graph, noBottom);
  /*imp_tnode(t);*/
  return(t);
}

/* Borra los lazos del grafo */
int delete_loops(unsigned *graph) {
  int i,j,k;
  int loops = 0;

  for (i = 0; i< maxvertex; i++)
    if ((k=elto_mc(graphTemplate,graph,i,i)) > 0) {
      loops += k;
      asg_mc(graphTemplate, graph,0,i,i);
    }
  return(loops);
}

/* Contrae la arista con extremos i, j */
tnode *contract(tnode *node, int i, int j, int noBottom) {
  int aux;
  tnode *t;
  int i1,j1;
  int k;

  if (i<j) {
    aux = i; i = j; j = aux;
  }
  t = duptnode(node);
  if (t == NULL)
    exit(1);
  asg_mc(graphTemplate, t->graph, elto_mc(graphTemplate,node-
>graph,i,j)-1,i,j);
  for (j1 = 0; j1 <= j; j1++)
    if ((k = elto_mc(graphTemplate, t->graph,i,j1)) > 0) {
      asg_mc(graphTemplate, t->graph, elto_mc(graphTemplate, node-
>graph,j,j1)+k, j, j1);
      asg_mc(graphTemplate, t->graph, 0, i, j1);
    }
  for (j1 = j+1; j1 < i; j1++)
```

```
    if ((k = elto_mc(graphTemplate, t->graph,i,j1)) > 0) {
      asg_mc(graphTemplate, t->graph, elto_mc(graphTemplate, node-
>graph,j1,j)+k, j1,j);
      asg_mc(graphTemplate, t->graph, 0, i, j1);
    }
  t->y = delete_loops(t->graph);
  t->x = delete_pendant_edges(t->graph, noBottom);
  /*imp_tnode(t);*/
  return t;
}


void process(tnode *raiz) {
  int i,j,print = 0;
  /* Búsqueda de una arista fuera del nivel inferior */
  if (print)
    imp_tnode(raiz);
  for (i = maxvertex-1; i > endBottom; i--)
    for (j = i-1; j >= 0; j--)
      if (elto_mc(graphTemplate, raiz->graph,i,j) > 0) {
    /* Se encontró */
    process(raiz->del = delete(raiz, i, j, 1));
    process(raiz->con = contract(raiz, i, j, 1));
    return;
      }
}

/* Procesa la base de la inducción involucrada en la
   definición del Polinomio de Tutte */
int print = 0;
void baseProcess(tnode *raiz) {
  int i,j;
  /* Búsqueda de una arista */
  if (print)
    imp_tnode(raiz);
  for (i = maxvertex-1; i >= 0; i--)
    for (j = i-1; j >= 0; j--)
      if (elto_mc(graphTemplate, raiz->graph,i,j) > 0) {
    /* Se encontró */
    baseProcess(raiz->del = delete(raiz, i,j,0));
    baseProcess(raiz->con = contract(raiz,i,j,0));
    return;
      }
}

void buildMinimal(tnode *n) {
  tnode *t,*auxt;
  tnode *s,*auxs;
  if (n->del == NULL) {
    auxt = t = newtnode(n->graph);
    if (bus_a23(MinDictionary,&t) == NULL) {
      auxs = s = simetricNode(n->graph);
      if (bus_a23(MinDictionary,&s) == NULL) {
    n->ord = t->ord = card_a23(MinDictionary)+1;
```

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

```
    ins_a23(MinDictionary,&t);
    insdes_ld(waitingList,&t);
      } else {
    n->ord = s->ord;
    dest_tnode(auxt);
      }
      dest_tnode(auxs);
      return;
    }
    n->ord = t->ord;
    dest_tnode(auxt);
  } else {
    buildMinimal(n->del);
    buildMinimal(n->con);
  }
}

/* Calcula las potencias en x e y. Prepara también
   la lista de impresión */
void powersCalc(tnode *raiz, int acumx, int acumy) {
  if (raiz == NULL)
    return;
  raiz->x += acumx;
  raiz->y += acumy;
  powersCalc(raiz->del,raiz->x, raiz->y);
  powersCalc(raiz->con,raiz->x, raiz->y);
  if (raiz->con == NULL)
    insdes_ld(printList,&raiz);
}

/* Imprime un coeficiente del polinomio de Tutte */
void printCoef(int coef, int x, int y) {
  if (coef != 1 || (x == 0 && y == 0))
    printf("%d",coef);
  if (x == 1)
    printf("x");
  if (x > 1)
    printf("x^%d",x);
  if (y== 1)
    printf("y");
  if (y > 1)
    printf("y^%d",y);
}

/* Imprime el contenido de la lista de impresión y
   vacia la lista */
void printTutte(tnode *node) {
  tnode **raiz;
  int antord,antx,anty;
  int coef = 0;
  int quitarPrimerMas = 1;

  antord = antx = anty = -1;
  ord_ld(printList,comp_coefTutte);
```

```
  printf("T_{n+1}(%d) = ",node->ord);
  for (raiz = (tnode **)ini_ld(printList); raiz != NULL; raiz =
sgt_ld(printList)) {
    if (antord == (*raiz)->ord && antx == (*raiz)->x && anty ==
(*raiz)->y) {
      coef++;
      continue;
    }
    if (antord == (*raiz)->ord) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      quitarPrimerMas = 0;
      coef = 1;
      antx = (*raiz)->x;
      anty = (*raiz)->y;
      continue;
      }
    if (antord != -1) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      printf(")T_n(%d) + (", antord);
      quitarPrimerMas = 1;
      coef = 1;
      antord = (*raiz)->ord;
      antx = (*raiz)->x;
      anty = (*raiz)->y;
      continue;
    }
    printf("(");
    quitarPrimerMas = 1;
    coef = 1;
    antord = (*raiz)->ord;
    antx = (*raiz)->x;
    anty = (*raiz)->y;
  }
  printf("%s", quitarPrimerMas ? "" : " + ");
  printCoef(coef, antx, anty);
  printf(")T_n(%d)\n", antord);
  vacia_ld(printList);
}

/* Imprime el contenido de la lista de impresión y
   vacia la lista para los términos base. También
   vacia la lista */
void printTutteBase(tnode *node) {
  tnode **raiz;
  int antx,anty;
  int coef = 0;
  int quitarPrimerMas = 1;

  antx = anty = -1;
  ord_ld(printList,comp_coefTutte);
  printf("T_1(%d) = ",node->ord);
```

```
  for (raiz = (tnode **)ini_ld(printList); raiz != NULL; raiz =
sgt_ld(printList)) {
    if (antx == (*raiz)->x && anty == (*raiz)->y) {
      coef++;
      continue;
    }
    if (antx != -1) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      quitarPrimerMas = 0;
      coef = 1;
      antx = (*raiz)->x;
      anty = (*raiz)->y;
      continue;
    }
    quitarPrimerMas = 1;
    coef = 1;
    antx = (*raiz)->x;
    anty = (*raiz)->y;
  }
  printf("%s", quitarPrimerMas ? "" : " + ");
  printCoef(coef, antx, anty);
  printf("\n");
  vacia_ld(printList);
}

void liberaMemRec(tnode *raiz) {
  if (raiz != NULL) {
    liberaMemRec(raiz->del);
    liberaMemRec(raiz->con);
    destdat_mc(raiz->graph);
    free(raiz);
  }
}

/* Libera la memoria de todos los nodos que estan debajo de la raíz
   de un grafo generador. */
void liberaMem(tnode *raiz) {
  liberaMemRec(raiz->del);
  liberaMemRec(raiz->con);
  raiz->del = raiz->con = NULL;
}

void readInput(char *file) {
  FILE *f;
  int i,j;
  if (file[0] != '\0')
    f = fopen(file,"r");
  else
    f = stdin;
  if (f == NULL) {
    printf("Missing file %s\n",file);
    exit(1);
  }
```

```
  fscanf(f,"%d %d %d %d\n",&maxvertex, &maxparallel, &endBottom,
&startTop);
  graphTemplate = creaenc_mc(0, maxparallel, 2, maxvertex, maxvertex);
  graphMod = creadat_mc(graphTemplate);
  inimin_mc(graphTemplate, graphMod);
  while (fscanf(f,"%d %d\n",&i,&j) != EOF) {
    if (i > j)
      asg_mc(graphTemplate, graphMod, 1, i, j);
    else
      asg_mc(graphTemplate, graphMod, 1, j, i);
    }
  if (file[0] != '\0')
    fclose(f);
}

/* Deja sólo la corona de un módulo */
void crownDown(tnode *raiz) {
  int i,j,k;

  for (i = 0; i < maxvertex; i++)
    for (j = i-1; j>=0; j--)
      if ((k=elto_mc(graphTemplate,raiz->graph,i,j)) != 0) {
    if (i <= endBottom || j <= endBottom)
      asg_mc(graphTemplate,raiz->graph,0,i,j);
    if (i >= startTop && j>=startTop) {
      asg_mc(graphTemplate, raiz->graph, 0,i, j);
      asg_mc(graphTemplate, raiz->graph, k, i-startTop, j-startTop);
    }
      }
}

int main(int argc, char *argv[]) {
  char fileName[250];
  int opt;

  tnode *raiz;        /* Raíz de la recursión */
  tnode **apraiz;

  fileName[0] = '\0';
  while ((opt = getopt(argc, argv, "f:")) != -1) {
    switch (opt) {
    case 'f':
       if (strlen(optarg) > 255) {
          fprintf(stderr,"File name %s too long\n",optarg);
          exit(1);
       }
       strcpy(fileName,optarg);
      break;
    }
  }

  readInput(fileName);
  raiz = newtnode(NULL);
  raiz->ord = 1;
```

```
  /* Construcción del árbol de estructuras mínimas,
     de la cola de espera y de la cola de impresión */
  MinDictionary = crea_a23(sizeof(unsigned *),comp_aptnode,0);
  waitingList = crea_ld(sizeof(unsigned *));
  printList   = crea_ld(sizeof(unsigned *));
  ins_a23(MinDictionary,&raiz);
  insdes_ld(waitingList, &raiz);

  while (card_ld(waitingList) > 0) {
    ord_ld(waitingList,comp_ord);
    ini_ld(waitingList);
    bor_ld(waitingList,&raiz);
    process(raiz);
    buildMinimal(raiz);
    powersCalc(raiz,0,0);
    printTutte(raiz);
    liberaMem(raiz);
  }

  /* Cálculo del Polinomio de Tutte de los grafos generadores */
  printf("\n\n");
  for (apraiz = min_a23(MinDictionary,NULL); apraiz != NULL; apraiz =
sgt_a23(MinDictionary))
    insdes_ld(waitingList, apraiz);
  ord_ld(waitingList, comp_ord);
  for (apraiz = ini_ld(waitingList); apraiz != NULL; apraiz =
sgt_ld(waitingList)) {
    crownDown(*apraiz);
    (*apraiz)->x = delete_pendant_edges((*apraiz)->graph, 0);
    (*apraiz)->y = delete_loops((*apraiz)->graph);
    baseProcess(*apraiz);
    powersCalc(*apraiz,0,0);
    printTutteBase(*apraiz);
    liberaMem(*apraiz);
  }
  exit(0);
}

/*
gcc -g -I/usr/include/pg -o poltutte poltutte.c $HOME/pg/lib/pg.a
*/
```

PROGRAMA POLTUTTE2 PARA EL CÁLCULO DEL POLINOMIO DE TUTTE

```
#include <stdlib.h>
#include <stdio.h>
#include "pgerror.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include "mc.h"
#include "a23.h"
#include "ld.h"

/* Número máximo de vértices en una capa */
int maxvertex;

/* Número máximo de aristas paralelas que pueden aparecer */
int maxparallel;

/* Estructuras de datos fundamentales */
ld  minList;           /* Lista de configuraciones minimales */
a23 SchemeDictionary;  /* Diccionario de grafos para las que se ha
                          hecho el proceso recursivo */
ld  waitingList;       /* Lista de espera de los elementos minimales
que
                          esperan el proceso de generación recursiva de
                          sus configuraciones minimales */
ld  printList;         /* Auxiliar para la impresión final del
polinomio */

mc *graphTemplate;     /* Descriptor de las matrices que guardan los
grafos */
unsigned *graphMod;    /* Módulo generador de los grafos */
int endBottom = 2;     /* Índice del último vértice en la parte
inferior
                          del grafo base */
int startTop = 3;      /* Índice del primer vértice en la parte
superior del
                          grafo base */
int *permutacion;      /* Permutación para construir el simétrico de
configuración dada */

int print = 0;         /* Controla la impresion en process y
baseprocess */


typedef struct tuttenode {
  int ord;             /* Etiqueta. Usado para nombrar a los grafos
generadores */
```

BIBLIOTECA

```
  unsigned *graph;     /* Grafo asociado al nodo */
  struct tnode *con;   /* Reducción recursiva al contraer la arista
mayor */
  struct tnode *del;   /* Reducción recursiva al borrar la arista
mayor*/
  int acumx,acumy;     /* Para calcular potencias */
} tuttenode;

typedef struct tnode {
  int x,y;             /* Potencia de x e y con la que contribuye el
nodo */
  tuttenode *node;
} tnode;               /* Representa un nodo en el árbol de recursión
*/

/* Compara los arreglos de datos de dos matrices comprimidas */
int comp_mc(mc *e, unsigned *d1, unsigned *d2) {
  int r = 0;
  int i;
  for (i = 0; i < e->t_e; i++)
    if ((r = d1[i] - d2[i]) != 0)
      return r;
  return(0);
}

int comp_coefTutte(void *t1, void *t2) {
  tnode **p1, **p2;
  int r;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  if ((r = (*p1)->node->ord - (*p2)->node->ord) != 0)
    return(r);
  if ((r = (*p2)->node->acumx - (*p1)->node->acumx) != 0)
    return(r);
  return (*p2)->node->acumy - (*p1)->node->acumy;
}

int comp_ord(void *t1, void *t2) {
  tnode **p1, **p2;
  int r;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  return ((*p1)->node->ord - (*p2)->node->ord);
}

int comp_aptnode(void *t1, void *t2) {
  tnode **p1, **p2;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  return (comp_mc(graphTemplate, (*p1)->node->graph, (*p2)->node-
>graph));
}

int comp_aptuttenode(void *t1, void *t2) {
```

```
  tuttenode **p1, **p2;
  p1 = (tuttenode **)t1;
  p2 = (tuttenode **)t2;
  return (comp_mc(graphTemplate, (*p1)->graph, (*p2)->graph));
}

void imp_mc(mc *template, unsigned *mat) {
  int i,j;

  for (i=0; i<maxvertex; i++) {
    for (j=0; j<maxvertex; j++)
      putchar('0'+elto_mc(template,mat,i,j));
    putchar('\n');
  }
}

void imp_tnode(tnode *t) {
  printf("x = %d, y = %d, ord = %d, acumx = %d, acumy = %d\n",t->x, t-
>y,t->node->ord,t->node->acumx,t->node->acumy);
  imp_mc(graphTemplate, t->node->graph);
}

/* Crea un nodo nuevo */
tnode *newtnode(unsigned *model) {
  tnode *t;
  int i,j;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL) {
    printf("***ERROR FATAL*** Se acabó la memoria\n");
    exit(1);
  }
  if ((t->node = (tuttenode *) malloc(sizeof(tuttenode))) == NULL) {
    printf("***ERROR FATAL*** Se acabó la memoria\n");
    exit(1);
  }
  t->node->ord = 0;
  t->node->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->node->graph, graphMod);
  t->x = t->y = t->node->acumx = t->node->acumx = 0;
  t->node->con = t->node->del = NULL;
  if (model != NULL)
    for (i=0; i<=endBottom; i++)
      for (j=0; j<i; j++)
    asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,
model, i, j), i+startTop, j+startTop);
  return t;
}

/* Construye la permutación de un nodo */
unsigned *permuta(unsigned *model, int *permutacion) {
  unsigned *t;
  int i,j;
  t = creadat_mc(graphTemplate);
  if (model != NULL)
```

```
    for (i=0; i<maxvertex; i++)
      for (j=0; j<i; j++)
    if (permutacion[i] > permutacion[j])
      asg_mc(graphTemplate, t, elto_mc(graphTemplate, model, i, j),
permutacion[i], permutacion[j]);
    else
      asg_mc(graphTemplate, t, elto_mc(graphTemplate, model, i, j),
permutacion[j], permutacion[i]);
 return t;
}

/* Este código se agregó con el fin de "compactar" el
   grafo, por ejemplo, si el vértice 3 queda aislado y los
   vértices 4, 5 y 6 forman parte del grafo entonces el
   vértice 4 se renombra como 3, el 5 como 4 y el 6 como 5.
   Esto sirve para minimizar el número de esquemas que se
   almacenan en el diccionario SchemeDiccionary */
void compress(unsigned *graph) {
    int *p;
    int i,j;
    int sgt,ult,suma;
    unsigned *t;

    if ((p = (int *)malloc(maxvertex*sizeof(int))) == NULL) {
      printf("***ERROR***, se acabó la memoria\n");
      exit(1);
    }
    for (i = 0; i <= endBottom; i++)
      p[i] = i;
    for (ult = maxvertex-1, sgt = i = endBottom+1; i < maxvertex; i++)
{
      suma = 0;
      for (j = 0; j < i; j++)
    suma += elto_mc(graphTemplate, graph,i,j);
      for (j = i+1; j < maxvertex; j++)
    suma += elto_mc(graphTemplate, graph,j,i);
      if (suma > 0)
    p[i] = sgt++;
      else
    p[i] = ult--;
    }
    t = permuta(graph,p);
    dupdat_mc(graphTemplate,graph,t);
    destdat_mc(t);
    free(p);
}

void dest_tnode(tnode *n) {
  destdat_mc(n->node->graph);
  free(n->node);
  free(n);
}

/* Duplica un tnode */
```

```
tnode *duptnode(tnode *node) {
  tnode *t;
  unsigned *aux;
  int i,j,k;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL ||
      (t->node = (tuttenode *)malloc(sizeof(tuttenode))) == NULL) {
    exit(1);
  }
  t->node->ord = node->node->ord;
  t->node->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->node->graph, node->node->graph);
  t->x = t->y = 0;
  t->node->acumx = node->node->acumx;
  t->node->acumy = node->node->acumy;
  t->node->con = t->node->del = NULL;
  return t;
}

/* Borra las hojas del grafo */
int delete_pendant_edges(unsigned *graph, int noBottom) {
  int i,j;
  int suma = 0;
  int pendant_edges = 0;

  for (i = maxvertex-1; i > (noBottom ? endBottom : -1); i--) {
    suma = 0;
    for (j = 0; j < i; j++)
      suma += elto_mc(graphTemplate, graph,i,j);
    for (j = i+1; j < maxvertex; j++)
      suma += elto_mc(graphTemplate, graph,j,i);
    if (suma == 1) {
      pendant_edges++;
      for (j = 0; j < i; j++)
        if (elto_mc(graphTemplate, graph,i,j) == 1) {
      asg_mc(graphTemplate, graph,0,i,j);
      i = maxvertex;
      break;
    }
      for (j = i+1; j < maxvertex; j++)
        if (elto_mc(graphTemplate, graph,j,i) == 1) {
      asg_mc(graphTemplate, graph,0,j,i);
      i = maxvertex;
      break;
    }
    }
  }

  if (pendant_edges > 0)
    compress(graph);
  return(pendant_edges);
}

/* Borra la arista con extremos i, j */
```

```
tnode *delete(tnode *node, int i, int j, int noBottom) {
  int aux;
  tnode *t;

  if (i<j) {
    aux = i; i = j; j = aux;
  }
  t = duptnode(node);
  t->node->ord = 0;
  if (t == NULL)
    exit(1);
  asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,node-
>node->graph,i,j)-1,i,j);
  t->x = delete_pendant_edges(t->node->graph, noBottom);
  /*imp_tnode(t);*/
  return(t);
}

/* Borra los lazos del grafo */
int delete_loops(unsigned *graph) {
  int i,j,k;
  int loops = 0;

  for (i = 0; i< maxvertex; i++)
    if ((k=elto_mc(graphTemplate,graph,i,i)) > 0) {
      loops += k;
      asg_mc(graphTemplate, graph,0,i,i);
    }
  return(loops);
}

/* Contrae la arista con extremos i, j */
tnode *contract(tnode *node, int i, int j, int noBottom) {
  int aux;
  tnode *t;
  int i1,j1;
  int k;

  if (i<j) {
    aux = i; i = j; j = aux;
  }
  t = duptnode(node);
  t->node->ord = 0;
  if (t == NULL)
    exit(1);
  asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,node-
>node->graph,i,j)-1,i,j);
  for (j1 = 0; j1 <= j; j1++)
    if ((k = elto_mc(graphTemplate, t->node->graph,i,j1)) > 0) {
      asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,
node->node->graph,j,j1)+k, j, j1);
      asg_mc(graphTemplate, t->node->graph, 0, i, j1);
    }
  for (j1 = j+1; j1 < i; j1++)
```

```
    if ((k = elto_mc(graphTemplate, t->node->graph,i,j1)) > 0) {
      asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,
node->node->graph,j1,j)+k, j1,j);
      asg_mc(graphTemplate, t->node->graph, 0, i, j1);
    }
  t->y = delete_loops(t->node->graph);
  t->x = delete_pendant_edges(t->node->graph, noBottom);
  /*imp_tnode(t);*/
  return t;
}

int nord = 0;
void process(tnode *raiz) {
  int i,j;
  tuttenode *aptutte;
  unsigned *sym,*aux;
  int found = 0;
  tnode *t;

  /* Búsqueda de una arista fuera del nivel inferior */
  if (print)
    imp_tnode(raiz);
  aptutte = raiz->node;
  found = (bus_a23(SchemeDictionary, &aptutte) != NULL);
  if (!found) {
    sym = permuta(raiz->node->graph,permutacion);
    compress(sym);
    aux = aptutte->graph;
    aptutte->graph = sym;
    found = (bus_a23(SchemeDictionary, &aptutte) != NULL);
    raiz->node->graph = aux;
    destdat_mc(sym);
  }
  if (found) {
    if (raiz->node->ord > 0)
      aptutte->ord = raiz->node->ord;
    destdat_mc(raiz->node->graph);
    free(raiz->node);
    raiz->node = aptutte;
    return;
  }
  aptutte = raiz->node;
  for (i = maxvertex-1; i > endBottom; i--)
    for (j = i-1; j >= 0; j--)
      if (elto_mc(graphTemplate, raiz->node->graph,i,j) > 0) {
    process(raiz->node->del = delete(raiz, i, j, 1));
    process(raiz->node->con = contract(raiz, i, j, 1));
    ins_a23(SchemeDictionary, &aptutte);
    return;
      }
  ins_a23(SchemeDictionary, &aptutte);
  nord++;
  raiz->node->ord = nord;
  insdes_ld(minList, &raiz);
```

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INFORMÁTICA BIBLIOTECA

```
  if (nord > 1) {
    t = newtnode(raiz->node->graph);
    raiz->node->ord = t->node->ord = nord;
    insdes_ld(waitingList,&t);
  }
}

/* Procesa la base de la inducción involucrada en la
   definición del Polinomio de Tutte */
void baseProcess(tnode *raiz) {
  int i,j;
  /* Búsqueda de una arista */
  if (print)
    imp_tnode(raiz);
  for (i = maxvertex-1; i >= 0; i--)
    for (j = i-1; j >= 0; j--)
      if (elto_mc(graphTemplate, raiz->node->graph,i,j) > 0) {
    /* Se encontró */
    baseProcess(raiz->node->del = delete(raiz, i,j,0));
    baseProcess(raiz->node->con = contract(raiz,i,j,0));
    return;
      }
}

/* Calcula las potencias en x e y. Prepara también
   la lista de impresión */
void powersCalc(tnode *raiz, int acumx, int acumy) {
  tnode *t;
  if (raiz == NULL)
    return;
  /*raiz->x += acumx;
    raiz->y += acumy;*/
  raiz->node->acumx = acumx+raiz->x;
  raiz->node->acumy = acumy+raiz->y;
  if (print)
    imp_tnode(raiz);
  if (raiz->node->con == NULL) {
    t = duptnode(raiz);
    insdes_ld(printList,&t);
  }
  powersCalc(raiz->node->del,raiz->node->acumx, raiz->node->acumy);
  powersCalc(raiz->node->con,raiz->node->acumx, raiz->node->acumy);
}

/* Imprime un coeficiente del polinomio de Tutte */
void printCoef(int coef, int x, int y) {
  if (coef != 1 || (x == 0 && y == 0))
    printf("%d",coef);
  if (x == 1)
    printf("x");
  if (x > 1)
    printf("x^%d",x);
  if (y== 1)
    printf("y");
```

```
  if (y > 1)
    printf("y^%d",y);
}

/* Imprime el contenido de la lista de impresión y
   vacia la lista */
void printTutte(tnode *node) {
  tnode **raiz;
  int antord,antx,anty;
  int coef = 0;
  int quitarPrimerMas = 1;

  antord = antx = anty = -1;
  ord_ld(printList,comp_coefTutte);
  printf("T_{n+1}(%d) = ",node->node->ord);
  for (raiz = (tnode **)ini_ld(printList); raiz != NULL;
dest_tnode(*raiz), raiz = sgt_ld(printList)) {
    if (antord == (*raiz)->node->ord && antx == (*raiz)->node->acumx
&& anty == (*raiz)->node->acumy) {
      coef++;
      continue;
    }
    if (antord == (*raiz)->node->ord) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      quitarPrimerMas = 0;
      coef = 1;
      antx = (*raiz)->node->acumx;
      anty = (*raiz)->node->acumy;
      continue;
      }
    if (antord != -1) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      printf(")T_n(%d) + (", antord);
      quitarPrimerMas = 1;
      coef = 1;
      antord = (*raiz)->node->ord;
      antx = (*raiz)->node->acumx;
      anty = (*raiz)->node->acumy;
      continue;
    }
    printf("(");
    quitarPrimerMas = 1;
    coef = 1;
    antord = (*raiz)->node->ord;
    antx = (*raiz)->node->acumx;
    anty = (*raiz)->node->acumy;
  }
  printf("%s", quitarPrimerMas ? "" : " + ");
  printCoef(coef, antx, anty);
  printf(")T_n(%d)\n", antord);
  vacia_ld(printList);
}
```

```
/* Imprime el contenido de la lista de impresión y
   vacia la lista para los términos base. También
   vacia la lista */
void printTutteBase(tnode *node) {
  tnode **raiz;
  int antx,anty;
  int coef = 0;
  int quitarPrimerMas = 1;

  antx = anty = -1;
  ord_ld(printList,comp_coefTutte);
  printf("T_1(%d) = ",node->node->ord);
  for (raiz = (tnode **)ini_ld(printList); raiz != NULL; raiz =
sgt_ld(printList)) {
    if (antx == (*raiz)->node->acumx && anty == (*raiz)->node->acumy)
{
      coef++;
      continue;
    }
    if (antx != -1) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      quitarPrimerMas = 0;
      coef = 1;
      antx = (*raiz)->node->acumx;
      anty = (*raiz)->node->acumy;
      continue;
    }
    quitarPrimerMas = 1;
    coef = 1;
    antx = (*raiz)->node->acumx;
    anty = (*raiz)->node->acumy;
  }
  printf("%s", quitarPrimerMas ? "" : " + ");
  printCoef(coef, antx, anty);
  printf("\n");
  vacia_ld(printList);
}

void readInput(char *file) {
  FILE *f;
  int i,j;
  if (file[0] != '\0')
    f = fopen(file,"r");
  else
    f = stdin;
  if (f == NULL) {
    printf("Missing file %s\n",file);
    exit(1);
  }
  fscanf(f,"%d %d %d %d\n",&maxvertex, &maxparallel, &endBottom,
&startTop);
  if ((permutacion = (int *) malloc(sizeof(int)*maxvertex)) == NULL) {
```

```
    printf("***ERROR*** Memoria insuficiente\n");
  }
  graphTemplate = creaenc_mc(0, maxparallel, 2, maxvertex, maxvertex);
  graphMod = creadat_mc(graphTemplate);
  inimin_mc(graphTemplate, graphMod);
  for (i = 0; i < maxvertex; i++)
    fscanf(f,"%d",permutacion+i);
  fscanf(f,"\n");
  while (fscanf(f,"%d %d\n",&i,&j) != EOF) {
    if (i > j)
      asg_mc(graphTemplate, graphMod, 1, i, j);
    else
      asg_mc(graphTemplate, graphMod, 1, j, i);
    }
  if (file[0] != '\0')
    fclose(f);
}

int main(int argc, char *argv[]) {
  char fileName[250];
  int opt;
  int complexity = 0;

  tnode *raiz;        /* Raíz de la recursión */
  tnode **apraiz;

  fileName[0] = '\0';
  while ((opt = getopt(argc, argv, "f:c")) != -1) {
    switch (opt) {
    case 'f':
       if (strlen(optarg) > 255) {
          fprintf(stderr,"File name %s too long\n",optarg);
          exit(1);
       }
       strcpy(fileName,optarg);
      break;
    case 'c': complexity = 1;
      break;
    }
  }

  readInput(fileName);
  raiz = newtnode(NULL);
  raiz->node->ord = 1;

  /* Construcción del árbol de estructuras mínimas,
     de la cola de espera y de la cola de impresión */
  minList = crea_ld(sizeof(unsigned *));
  SchemeDictionary = crea_a23(sizeof(unsigned *),comp_aptuttenode,0);
  waitingList = crea_ld(sizeof(unsigned *));
  printList   = crea_ld(sizeof(unsigned *));
  insdes_ld(waitingList, &raiz);

  while (card_ld(waitingList) > 0) {
```

BIBLIOTECA

```
    ord_ld(waitingList,comp_ord);
    ini_ld(waitingList);
    bor_ld(waitingList,&raiz);
    process(raiz);
    /*buildMinimal(raiz);*/
    powersCalc(raiz,0,0);
    printTutte(raiz);
  }

  /* Cálculo del Polinomio de Tutte de los grafos generadores */
  printf("\n\n");
  ord_ld(minList, comp_ord);
  for (apraiz = ini_ld(minList); apraiz != NULL; apraiz =
sgt_ld(minList)) {
    (*apraiz)->x = delete_pendant_edges((*apraiz)->node->graph, 0);
    (*apraiz)->y = delete_loops((*apraiz)->node->graph);
    baseProcess(*apraiz);
    powersCalc(*apraiz,0,0);
    printTutteBase(*apraiz);
  }

  if (complexity = 1)
    printf("El diccionario de esquemas tiene %d
elementos\n",card_a23(SchemeDictionary));
  exit(0);
}

/*
gcc -g -I/usr/include/pg -o poltutte2 poltutte2.c $HOME/pg/lib/pg.a
*/
```

PROGRAMA POLTUTTE3 PARA EL CÁLCULO DEL POLINOMIO DE TUTTE

```
#include <stdlib.h>
#include <stdio.h>
#include "pgerror.h"
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include "mc.h"
#include "a23.h"
#include "ld.h"

int maxvertex;          /* Número máximo de vértices en una capa */
int maxmodule;          /* Número de capas en la teselación */
int maxparallel;        /* Número máximo de aristas paralelas */

/* Estructuras de datos fundamentales */
ld  minList;            /* Lista de configuraciones minimales */
a23 SchemeDictionary;   /* Diccionario de grafos para los que se ha
                           hecho el proceso recursivo */
ld  waitingList;        /* Lista de espera de los grafos generadores que
                           esperan el proceso de generación recursiva de
                           sus configuraciones minimales */
ld  printList;          /* Auxiliar para la impresión final del
polinomio */

mc *graphTemplate;      /* Descriptor de las matrices que guardan los
grafos */

int print = 0;          /* Controla la impresion en process y
baseprocess */

typedef struct tmodule { /* Descriptor de las capas básicas que
componen la teselación */
  unsigned *graphMod;   /* Grafo base del módulo */
  int endBottom;        /* Índice del último vértice en la parte
inferior
             del grafo base */
  int startTop;         /* Índice del primer vértice en la parte
superior del
             grafo base */
  int *permutacion;     /* Permutación para construir el simétrico de
configuración dada */
} tmodule;

tmodule *module;        /* Módulos en la red */
```

```
int *nord;              /* Usado en process para ordenar los polinomios
*/
int nmodule = 0;        /* Módulo actualmente activo */

typedef struct tuttenode {
  int ord;              /* Etiqueta. Usado para nombrar a los grafos
generadores */
  int nmodule;          /* Número de la capa */
  unsigned *graph;      /* Grafo asociado al nodo */
  struct tnode *con;    /* Reducción recursiva al contraer la arista
mayor */
  struct tnode *del;    /* Reducción recursiva al borrar la arista mayor
*/
  int acumx,acumy;      /* Para calcular potencias */
} tuttenode;

typedef struct tnode {
  int x,y;              /* Potencia de x e y con la que contribuye el
nodo */
  tuttenode *node;
} tnode;                /* Representa un nodo en el árbol de recursión
*/

/* Compara los arreglos de datos de dos matrices comprimidas */
int comp_mc(mc *e, unsigned *d1, unsigned *d2) {
  int r = 0;
  int i;
  for (i = 0; i < e->t_e; i++)
    if ((r = d1[i] - d2[i]) != 0)
      return r;
  return(0);
}

int comp_coefTutte(void *t1, void *t2) {
  tnode **p1, **p2;
  int r;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  if ((r = (*p1)->node->ord - (*p2)->node->ord) != 0)
    return(r);
  if ((r = (*p1)->node->nmodule - (*p2)->node->nmodule) != 0)
    return(r);
  if ((r = (*p2)->node->acumx - (*p1)->node->acumx) != 0)
    return(r);
  return (*p2)->node->acumy - (*p1)->node->acumy;
}

int comp_ord(void *t1, void *t2) {
  tnode **p1, **p2;
  int r;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  return ((*p1)->node->ord - (*p2)->node->ord);
}
```

```
int comp_aptnode(void *t1, void *t2) {
  tnode **p1, **p2;
  p1 = (tnode **)t1;
  p2 = (tnode **)t2;
  return (comp_mc(graphTemplate, (*p1)->node->graph, (*p2)->node-
>graph));
}

int comp_aptuttenode(void *t1, void *t2) {
  tuttenode **p1, **p2;
  int r;
  p1 = (tuttenode **)t1;
  p2 = (tuttenode **)t2;
  if ((r = (*p1)->nmodule - (*p2)->nmodule) != 0)
    return r;
  return (comp_mc(graphTemplate, (*p1)->graph, (*p2)->graph));
}

void imp_mc(mc *template, unsigned *mat) {
  int i,j;

  for (i=0; i<maxvertex; i++) {
    for (j=0; j<maxvertex; j++)
      putchar('0'+elto_mc(template,mat,i,j));
    putchar('\n');
  }
}

void imp_tnode(tnode *t) {
  printf("x = %d, y = %d, ord = %d, mod = %d, acumx = %d, acumy =
%d\n",t->x, t->y,t->node->ord,t->node->nmodule, t->node->acumx,t-
>node->acumy);
  imp_mc(graphTemplate, t->node->graph);
}

/* Crea un nodo nuevo */
tnode *newtnode(unsigned *model, int nmod) {
  tnode *t;
  int i,j;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL) {
    printf("***ERROR FATAL*** Se acabó la memoria\n");
    exit(1);
  }
  if ((t->node = (tuttenode *) malloc(sizeof(tuttenode))) == NULL) {
    printf("***ERROR FATAL*** Se acabó la memoria\n");
    exit(1);
  }
  t->node->ord = 0;
  t->node->nmodule = nmod;
  t->node->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->node->graph, module[nmod].graphMod);
  t->x = t->y = t->node->acumx = t->node->acumx = 0;
```

```
  t->node->con = t->node->del = NULL;
  if (model != NULL)
    for (i=0; i<=module[(nmod+maxmodule-1) % maxmodule].endBottom;
i++) /*~Revisar */
      for (j=0; j<i; j++)
    asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,
model, i, j),
           i+module[nmod].startTop, j+module[nmod].startTop);
  return t;
}

/* Construye la permutación de un nodo */
unsigned *permuta(unsigned *model, int *permutacion) {
  unsigned *t;
  int i,j;
  t = creadat_mc(graphTemplate);
  if (model != NULL)
    for (i=0; i<maxvertex; i++)
      for (j=0; j<i; j++)
    if (permutacion[i] > permutacion[j])
      asg_mc(graphTemplate, t, elto_mc(graphTemplate, model, i, j),
permutacion[i], permutacion[j]);
    else
      asg_mc(graphTemplate, t, elto_mc(graphTemplate, model, i, j),
permutacion[j], permutacion[i]);
 return t;
}

/* Este código se agregó con el fin de "compactar" el
   grafo, por ejemplo, si el vértice 3 queda aislado y los
   vértices 4, 5 y 6 forman parte del grafo entonces el
   vértice 4 se renombra como 3, el 5 como 4 y el 6 como 5.
   Esto sirve para minimizar el número de esquemas que se
   almacenan en el diccionario SchemeDiccionary */
void compress(unsigned *graph) {
    int *p;
    int i,j;
    int sgt,ult,suma;
    unsigned *t;

    if ((p = (int *)malloc(maxvertex*sizeof(int))) == NULL) {
      printf("***ERROR***, se acabó la memoria\n");
      exit(1);
    }
    for (i = 0; i <= module[nmodule].endBottom; i++)
      p[i] = i;
    for (ult = maxvertex-1, sgt = i = module[nmodule].endBottom+1; i <
maxvertex; i++) {
      suma = 0;
      for (j = 0; j < i; j++)
    suma += elto_mc(graphTemplate, graph,i,j);
      for (j = i+1; j < maxvertex; j++)
    suma += elto_mc(graphTemplate, graph,j,i);
      if (suma > 0)
```

```
    p[i] = sgt++;
      else
    p[i] = ult--;
    }
    t = permuta(graph,p);
    dupdat_mc(graphTemplate,graph,t);
    destdat_mc(t);
    free(p);
}

void dest_tnode(tnode *n) {
  destdat_mc(n->node->graph);
  free(n->node);
  free(n);
}

/* Duplica un tnode */
tnode *duptnode(tnode *node) {
  tnode *t;
  unsigned *aux;
  int i,j,k;

  if ((t = (tnode *) malloc(sizeof(tnode))) == NULL ||
      (t->node = (tuttenode *)malloc(sizeof(tuttenode))) == NULL) {
    exit(1);
  }
  t->node->ord = node->node->ord;
  t->node->nmodule = node->node->nmodule;
  t->node->graph = creadat_mc(graphTemplate);
  dupdat_mc(graphTemplate, t->node->graph, node->node->graph);
  t->x = t->y = 0;
  t->node->acumx = node->node->acumx;
  t->node->acumy = node->node->acumy;
  t->node->con = t->node->del = NULL;
  return t;
}

/* Borra las hojas del grafo */
int delete_pendant_edges(unsigned *graph, int noBottom) {
  int i,j;
  int suma = 0;
  int pendant_edges = 0;

  for (i = maxvertex-1; i > (noBottom ? module[nmodule].endBottom : -
1); i--) {
    suma = 0;
    for (j = 0; j < i; j++)
      suma += elto_mc(graphTemplate, graph,i,j);
    for (j = i+1; j < maxvertex; j++)
      suma += elto_mc(graphTemplate, graph,j,i);
    if (suma == 1) {
      pendant_edges++;
      for (j = 0; j < i; j++)
        if (elto_mc(graphTemplate, graph,i,j) == 1) {
```

```
      asg_mc(graphTemplate, graph,0,i,j);
      i = maxvertex;
      break;
    }
      for (j = i+1; j < maxvertex; j++)
        if (elto_mc(graphTemplate, graph,j,i) == 1) {
      asg_mc(graphTemplate, graph,0,j,i);
      i = maxvertex;
      break;
    }
    }
  }

  if (pendant_edges > 0)
    compress(graph);
  return(pendant_edges);
}

/* Borra la arista con extremos i, j */
tnode *delete(tnode *node, int i, int j, int noBottom) {
  int aux;
  tnode *t;

  if (i<j) {
    aux = i; i = j; j = aux;
  }
  t = duptnode(node);
  t->node->ord = 0;
  if (t == NULL)
    exit(1);
  asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,node-
>node->graph,i,j)-1,i,j);
  t->x = delete_pendant_edges(t->node->graph, noBottom);
  /*imp_tnode(t);*/
  return(t);
}

/* Borra los lazos del grafo */
int delete_loops(unsigned *graph) {
  int i,j,k;
  int loops = 0;

  for (i = 0; i< maxvertex; i++)
    if ((k=elto_mc(graphTemplate,graph,i,i)) > 0) {
      loops += k;
      asg_mc(graphTemplate, graph,0,i,i);
    }
  return(loops);
}

/* Contrae la arista con extremos i, j */
tnode *contract(tnode *node, int i, int j, int noBottom) {
  int aux;
  tnode *t;
```

```
  int i1,j1;
  int k;

  if (i<j) {
    aux = i; i = j; j = aux;
  }
  t = duptnode(node);
  t->node->ord = 0;
  if (t == NULL)
    exit(1);
  asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,node-
>node->graph,i,j)-1,i,j);
  for (j1 = 0; j1 <= j; j1++)
    if ((k = elto_mc(graphTemplate, t->node->graph,i,j1)) > 0) {
      asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,
node->node->graph,j,j1)+k, j, j1);
      asg_mc(graphTemplate, t->node->graph, 0, i, j1);
    }
  for (j1 = j+1; j1 < i; j1++)
    if ((k = elto_mc(graphTemplate, t->node->graph,i,j1)) > 0) {
      asg_mc(graphTemplate, t->node->graph, elto_mc(graphTemplate,
node->node->graph,j1,j)+k, j1,j);
      asg_mc(graphTemplate, t->node->graph, 0, i, j1);
    }
  t->y = delete_loops(t->node->graph);
  t->x = delete_pendant_edges(t->node->graph, noBottom);
  /*imp_tnode(t);*/
  return t;
}

void process(tnode *raiz) {
  int i,j;
  tuttenode *aptutte;
  unsigned *sym,*aux;
  int found = 0;
  tnode *t;

  /* Búsqueda de una arista fuera del nivel inferior */
  if (print)
    imp_tnode(raiz);
  aptutte = raiz->node;
  found = (bus_a23(SchemeDictionary, &aptutte) != NULL);
  if (!found && module[nmodule].permutacion[0] != -1) {
    sym = permuta(raiz->node->graph,module[nmodule].permutacion);
    compress(sym);
    aux = aptutte->graph;
    aptutte->graph = sym;
    found = (bus_a23(SchemeDictionary, &aptutte) != NULL);
    raiz->node->graph = aux;
    destdat_mc(sym);
  }
  if (found) {
    if (raiz->node->ord > 0)
      aptutte->ord = raiz->node->ord;
```

```
    destdat_mc(raiz->node->graph);
    free(raiz->node);
    raiz->node = aptutte;
    return;
  }
  aptutte = raiz->node;
  for (i = maxvertex-1; i > module[nmodule].endBottom; i--)
    for (j = i-1; j >= 0; j--)
      if (elto_mc(graphTemplate, raiz->node->graph,i,j) > 0) {
    process(raiz->node->del = delete(raiz, i, j, 1));
    process(raiz->node->con = contract(raiz, i, j, 1));
    ins_a23(SchemeDictionary, &aptutte);
    return;
      }
  t = newtnode(raiz->node->graph, (nmodule+1) % maxmodule);
  if (comp_mc(graphTemplate,t->node->graph, module[0].graphMod) == 0
&& nmodule == maxmodule-1) {
    raiz->node->ord = 1;
    dest_tnode(t);
  }
  else {
    raiz->node->ord = t->node->ord = ++(nord[(nmodule+maxmodule-
1)%maxmodule]);
    insdes_ld(waitingList,&t);
  }
  ins_a23(SchemeDictionary, &aptutte);
  insdes_ld(minList, &raiz);
}

/* Procesa la base de la inducción involucrada en la
   definición del Polinomio de Tutte */
void baseProcess(tnode *raiz) {
  int i,j;
  /* Búsqueda de una arista */
  if (print)
    imp_tnode(raiz);
  for (i = maxvertex-1; i >= 0; i--)
    for (j = i-1; j >= 0; j--)
      if (elto_mc(graphTemplate, raiz->node->graph,i,j) > 0) {
    /* Se encontró */
    baseProcess(raiz->node->del = delete(raiz, i,j,0));
    baseProcess(raiz->node->con = contract(raiz,i,j,0));
    return;
      }
}

/* Calcula las potencias en x e y. Prepara también
   la lista de impresión */
void powersCalc(tnode *raiz, int acumx, int acumy) {
  tnode *t;
  if (raiz == NULL)
    return;
  /*raiz->x += acumx;
    raiz->y += acumy;*/
```

```
  raiz->node->acumx = acumx+raiz->x;
  raiz->node->acumy = acumy+raiz->y;
  if (print)
    imp_tnode(raiz);
  if (raiz->node->con == NULL) {
    t = duptnode(raiz);
    insdes_ld(printList,&t);
  }
  powersCalc(raiz->node->del,raiz->node->acumx, raiz->node->acumy);
  powersCalc(raiz->node->con,raiz->node->acumx, raiz->node->acumy);
}

/* Imprime un coeficiente del polinomio de Tutte */
void printCoef(int coef, int x, int y) {
  if (coef != 1 || (x == 0 && y == 0))
    printf("%d",coef);
  if (x == 1)
    printf("x");
  if (x > 1)
    printf("x^%d",x);
  if (y== 1)
    printf("y");
  if (y > 1)
    printf("y^%d",y);
}

/* Imprime el contenido de la lista de impresión y
   vacia la lista */
void printTutte(tnode *node) {
  tnode **raiz;
  int antord, antmod, antx, anty;
  int coef = 0;
  int quitarPrimerMas = 1;

  antord = antmod = antx = anty = -1;
  ord_ld(printList,comp_coefTutte);
  printf("T_{n+1}(%d,%d) = ",node->node->ord, node->node->nmodule);
  for (raiz = (tnode **)ini_ld(printList); raiz != NULL;
dest_tnode(*raiz), raiz = sgt_ld(printList)) {
    if (antord == (*raiz)->node->ord && antmod == (*raiz)->node-
>nmodule &&
    antx == (*raiz)->node->acumx && anty == (*raiz)->node->acumy) {
      coef++;
      continue;
    }
    if (antord == (*raiz)->node->ord) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      quitarPrimerMas = 0;
      coef = 1;
      antx = (*raiz)->node->acumx;
      anty = (*raiz)->node->acumy;
      continue;
      }
```

BIBLIOTECA

```
    if (antord != -1) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      printf(")T_n(%d,%d) + (", antord,(antmod+maxmodule-
1)%maxmodule);
      quitarPrimerMas = 1;
      coef = 1;
      antord = (*raiz)->node->ord;
      antmod = (*raiz)->node->nmodule;
      antx = (*raiz)->node->acumx;
      anty = (*raiz)->node->acumy;
      continue;
    }
    printf("(");
    quitarPrimerMas = 1;
    coef = 1;
    antord = (*raiz)->node->ord;
    antmod = (*raiz)->node->nmodule;
    antx = (*raiz)->node->acumx;
    anty = (*raiz)->node->acumy;
  }
  printf("%s", quitarPrimerMas ? "" : " + ");
  printCoef(coef, antx, anty);
  printf(")T_n(%d,%d)\n", antord, (antmod+maxmodule-1)%maxmodule);
  vacia_ld(printList);
}

/* Imprime el contenido de la lista de impresión y
   vacia la lista para los términos base. También
   vacia la lista */
void printTutteBase(tnode *node) {
  tnode **raiz;
  int antx,anty;
  int coef = 0;
  int quitarPrimerMas = 1;

  antx = anty = -1;
  ord_ld(printList,comp_coefTutte);
  printf("T_1(%d,%d) = ",node->node->ord,node->node->nmodule);
  for (raiz = (tnode **)ini_ld(printList); raiz != NULL; raiz =
sgt_ld(printList)) {
    if (antx == (*raiz)->node->acumx && anty == (*raiz)->node->acumy)
{
      coef++;
      continue;
    }
    if (antx != -1) {
      printf("%s", quitarPrimerMas ? "" : " + ");
      printCoef(coef, antx, anty);
      quitarPrimerMas = 0;
      coef = 1;
      antx = (*raiz)->node->acumx;
      anty = (*raiz)->node->acumy;
      continue;
```

```
    }
    quitarPrimerMas = 1;
    coef = 1;
    antx = (*raiz)->node->acumx;
    anty = (*raiz)->node->acumy;
  }
  printf("%s", quitarPrimerMas ? "" : " + ");
  printCoef(coef, antx, anty);
  printf("\n");
  vacia_ld(printList);
}

/* ~Revisar */
void readInput(char *file) {
  FILE *f;
  int i,j;
  char c;
  if (file[0] != '\0')
    f = fopen(file,"r");
  else
    f = stdin;
  if (f == NULL) {
    printf("Missing file %s\n",file);
    exit(1);
  }
  fscanf(f,"%d %d %d\n",&maxvertex, &maxparallel, &maxmodule);
  if ((module = (tmodule *) malloc(sizeof(tmodule)*maxmodule)) ==
NULL) {
    printf("***ERROR*** Memoria insuficiente en readInput\n");
    exit(1);
  }
  graphTemplate = creaenc_mc(0, maxparallel, 2, maxvertex, maxvertex);
  while ((c = getch(f)) != EOF) {
    scanf("%d %d\n",&module[nmodule].endBottom,
&module[nmodule].startTop);
    if ((module[nmodule].permutacion = (int *)
malloc(sizeof(int)*maxvertex)) == NULL) {
      printf("***ERROR*** Memoria insuficiente en readInput\n");
      exit(1);
    }
    module[nmodule].graphMod = creadat_mc(graphTemplate);
    inimin_mc(graphTemplate, module[nmodule].graphMod);
    for (i = 0; i < maxvertex; i++)
      fscanf(f,"%d",module[nmodule].permutacion+i);
    fscanf(f,"\n");
    while (fscanf(f,"%d %d\n",&i,&j) == 2) {
      if (i > j)
    asg_mc(graphTemplate, module[nmodule].graphMod, 1, i, j);
      else
    asg_mc(graphTemplate, module[nmodule].graphMod, 1, j, i);
    }
    nmodule++;
  }
  nmodule = 0;
```

```
  if (file[0] != '\0')
    fclose(f);
}

int main(int argc, char *argv[]) {
  char fileName[250];
  int i,opt;
  int complexity = 0;

  tnode *raiz;        /* Raíz de la recursión */
  tnode **apraiz;

  fileName[0] = '\0';
  while ((opt = getopt(argc, argv, "f:c")) != -1) {
    switch (opt) {
    case 'f':
       if (strlen(optarg) > 255) {
          fprintf(stderr,"File name %s too long\n",optarg);
          exit(1);
       }
       strcpy(fileName,optarg);
      break;
    case 'c': complexity = 1;
      break;
    }
  }

  readInput(fileName);
  if ((nord = (int *)malloc(sizeof(int)*maxmodule)) == NULL) {
    printf("***ERROR***, Memoria insuficiente en main\n");
    exit(1);
  }
  nord[0] = 1;
  for (i=1; i<maxmodule; i++)
    nord[i] = 0;
  raiz = newtnode(NULL, 0);
  raiz->node->ord = 1;

  /* Construcción del árbol de estructuras mínimas,
     de la cola de espera y de la cola de impresión */
  minList = crea_ld(sizeof(unsigned *));
  SchemeDictionary = crea_a23(sizeof(unsigned *),comp_aptuttenode,0);
  waitingList = crea_ld(sizeof(unsigned *));
  printList   = crea_ld(sizeof(unsigned *));
  insdes_ld(waitingList, &raiz);

  while (card_ld(waitingList) > 0) {
    ord_ld(waitingList,comp_ord);
    ini_ld(waitingList);
    bor_ld(waitingList,&raiz);
    nmodule = raiz->node->nmodule;
    process(raiz);
    /*buildMinimal(raiz);*/
    powersCalc(raiz,0,0);
```

```
    printTutte(raiz);
  }

  /* Cálculo del Polinomio de Tutte de los grafos generadores */
  printf("\n\n");
  ord_ld(minList, comp_ord);
  for (apraiz = ini_ld(minList); apraiz != NULL; apraiz =
sgt_ld(minList)) {
    (*apraiz)->x = delete_pendant_edges((*apraiz)->node->graph, 0);
    (*apraiz)->y = delete_loops((*apraiz)->node->graph);
    baseProcess(*apraiz);
    powersCalc(*apraiz,0,0);
    printTutteBase(*apraiz);
  }

  if (complexity = 1)
    printf("El diccionario de esquemas tiene %d
elementos\n",card_a23(SchemeDictionary));
  exit(0);
}

/*
gcc -g -I/usr/include/pg -o poltutte3 poltutte3.c $HOME/pg/lib/pg.a
*/
```

ARCHIVO A23.H

```
#ifndef _ARB23_
#define _ARB23_

#define NODO   100          /* Para identificar los nodos interiores */
#define HOJA     99           /*  "       "       los nodos hoja */
#define DUP     1           /* Se aceptan duplicados */
#define NO_DUP  0           /* No se aceptan duplicados */

#define ARBOL   1    /* Para el identificar Un  rbol*/
#define COLAC   2           /*  "   "     "        Una Cola
Concatenable */
#define CONJC   3    /*  "   "     "        Un conj. de colas    */

#define a23_vacio(a)    ((a)->raiz == NULL || ((a)->raiz->ident == NODO
&& (a)->raiz->n_hijos == 0))

typedef struct nd230{          /* Estructura para un nodo interno */
   short ident;               /* Tipo de nodo (siempre es igual a NODO)
*/
   struct nd230  *padre;       /* Apuntador al nodo padre */
   void *dato[4];             /* Dato[0]=*L, dato[1]=*M. (Seg£n
referencia) */
                        /* Se anexa el campo dato[2]=*R (m ximo elto.
                         en el sub rbol*/
   struct nd230 *hijos[4];     /* Apuntadores a los descendientes */
                        /* Se agrega el campo hijos[3] para facilitar
                         las operaciones de inserci¢n y borrado */
   int n_hijos;               /* N£mero de descendientes */
}nd23_, *nd23;

typedef struct hoja23_{        /* Estructura para una hoja */
   short ident;               /* Tipo del nodo (siempre es igual a HOJA)
*/
   nd23 padre;                /* Apuntador al nodo padre */
   void *dato;                /* Apuntador al dato */
   struct hoja23_ *sgt;       /* Apuntador a la hoja siguiente */
   struct hoja23_ *ant;       /* Apuntador a la hoja anterior */
}hoja23_, *hoja23;

typedef struct arbol23{        /* Descriptor del  rbol 2-3 */
   short ident;               /* 1 : Arbol 2-3, 2 : Cola Concatenable */
   int t_d;                   /* Tama¤o de los datos */
   int altura;                /* Altura del  rbol */
   nd23 raiz;                 /* Apuntador a la ra¡z del  rbol */
   char dup;                  /* Indicador de la aceptacion de
duplicados */
   int (*comp)();             /* Funci¢n de comparaci¢n de dos datos */
   void *ult_bor;             /* Apuntador al £ltimo dato borrado */
```

```
   hoja23 act;                /* Apuntador a la hoja actual */
   int n_datos;               /* N£mero de datos en el  rbol */
   void* clase;               /* Apuntador al conjunto al que forma
parte */
}a23_, *a23;

typedef struct cc23_{           /* Estructura para un conj. de
                           colas concatenables */
   short ident;                /* CONJC : Conjunto de colas */
   int t_d;             /* Tama¤o de los datos */
   int (*comp)(void*, void*); /* Funci¢n que compara dos datos */
   a23 dicdat;                /* Diccionario de datos */
   a23 diccon;                /* Diccionario de colas concatenables */
}c23_, *c23;

/*-----------------------------------------------------------------
---
   funciones de soporte para este modulo
-----------------------------------------------------------------
---*/
a23     crea_a23(int tam, int (*comp)(), char dup);
void    dest_a23(a23 a);
void   vacia_a23(a23 a);
void     imp_a23(a23 a, void (*imp)());
void    *bus_a23(a23 a, void *dato);
void *busmen_a23(a23 a, void *dato, void* dt);
void    *mod_a23(a23 a, void *dato);
void    *ins_a23(a23 a, void *dato);
int    checa_a23(a23 a);
void    *bor_a23(a23 a, void *dato);
void     rec_a23(a23 a, void (*f)(void *dato));
void    *min_a23(a23 a, void *dato);
void *bormin_a23(a23 a, void *dato);
void implant_a23(a23 a1, a23 a2, a23 a3);
void     div_a23(a23 a1, a23 a2, a23 a3, void* dato);
void*    max_a23(a23 a, void * dt);
void* actual_a23(a23 a, void* dt);
void*    ant_a23(a23 a);
void*    sgt_a23(a23 a);
int     card_a23(a23 a);
#endif
```

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

ARCHIVO AB.H

```
#ifndef _AB_
#define _AB_

typedef void *ab;        /* Tipo  rbol balanceado */
/* Funciones del m¢dulo. */
ab   crea_ab     (int t_d, int (*comp)(), int dup);
void dest_ab     (ab arbol);
ab   vacia_ab    (ab arbol);
int  ab_vacio    (ab arbol);
void imp_ab      (ab arbol, void (*imp)());
void *bus_ab     (ab arbol, void *dato);
void *mod_ab     (ab arbol, void *dato);
void *ins_ab     (ab arbol, void *dato);
void *bor_ab     (ab arbol, void *dato);
int  checa_ab    (ab arbol, int *profundidad, int *nelementos);
void rec_ab_ino (ab arbol, void (*f)());
void rec_ab_pre (ab arbol, void (*f)());
void rec_ab_pos (ab arbol, void (*f)());
void  *min_ab    (ab arbol, void *dato);
#endif
```

ARCHIVO ARCH.H

```
#ifndef _ARCH_
#define _ARCH_

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NOMBRE 80     /*Longitud m xima permitido para el nombres
de archivos */
#define FIN_CAD     '\x0' /*Marca que nos ¡ndica el final de una
cadena. */

/*
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::
```

```
OFFSET          Se refiere al tama¤o en bytes del encabezado del
archivo.
            Apartir del byte NO. OFFSET inician los registros del
            archivo.Esta area contiene:
Del byte 0-3:   El n£mero de registros en el archivo (No_Reg).
Del byte 4-7:   El n£mero de registros no borrados en el archivo
(Borrados).
Del byte 8-9:   El tama¤o de los registros almacenados.
Del byte 12-23: La cadena de autentificaci¢n.
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::
*/

typedef struct archivo{     /* Descriptor del archivo en memoria. */
 long no_reg;               /* N£mero de registros del archivo. */
 long borrados;             /* N£mero de registros borrados a nivel
l¢gico. */
 long reg_actual;     /* N£mero de registros. */
 int  tama;           /* Tama¤o en bytes de los registros. */
 char *inf;             /* Apuntador al buffer de datos. */
 char nombre[MAX_NOMBRE];  /* Nombre del archivo. */
 char ext[4];               /* Extensi¢n del archivo. */
 FILE *desc;                     /* Apuntador al archivo. */
}arch_;


/* Funciones empleados en el modulo */

arch_ *abre_arch(char nombre[MAX_NOMBRE], char extension[4]);
int   cierra_arch(arch_ *a);
void  *lee_arch(arch_ *a,long p, void *buf, char *bor);
void  *ins_arch(arch_ *a,void *b);
void  *mod_arch(arch_ *a,long p, void *b);
int   rec_arch(arch_ *a,void (*fimp)(void *));
int   dest_arch(arch_ *a);
int   checa_arch(arch_ *a);
/*int   crea_arch(char nombre[MAX_NOMBRE], char extension[4], int
t,int (*existe_arch)(char *)); */
int   crea_arch(char nombre[MAX_NOMBRE], char extension[4], int t, int
(existe_arch)(char *));
int   existe_arch(char *nombre);
int   merge_sort(long ini, long fin,arch_ *a,FILE *f, double
(*comp)(void *, void *) );  /*sacar */
int   intercala(long ini,long fin,long mitad,arch_ *a,FILE *f,double
(*comp)(void *, void *) ); /*sacar */


void  *comprime_arch(arch_ *a);
void  *fin_arch(   arch_ *a,    void *b, char *bor);
void  *ini_arch(   arch_ *a,    void *b, char *bor);
void  *ant_arch(   arch_ *a,    void *b, char *bor);
void  *sgte_arch(  arch_ *a,    void *b, char *bor);
void  *actual_arch(arch_ *a,    void *b, char *bor);
```

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

```
int   arch_fin(arch_ *a);
int   arch_ini(arch_ *a);
int   bor_arch(arch_ *a, long p);
int   desbor_arch(arch_ *a, long p);
int   vacia_arch(arch_ *a);
void  *ord_arch(arch_ *a, double (*comp)(void *, void *));
int   tama_arch(arch_ *in);

#endif
```

ARCHIVO BTREE.C

```
#define MAX_BCAM      5
#define ASCENDENTE    1
#define DESCENDENTE -1

/* ---- Estructura para representar una lista de campos en un b- rbol
---- */
typedef struct blist
   {
   int length;              /* Longitud del campo */
   int dom;                 /* Dominio al que pertenece el campo */
   int tipo_orden;          /* Puede ser ASCENDENTE ¢ DESCENDENTE */
   } blist;

void construye_ba(char *name, int nc, blist *c);
int inicializa_ba(char *ndx_name);
int cierra_ba(int tree);
int inserta_ba(int tree, char *x, long ad, int unique);
long busca_ba(int tree, char *k);
int borra_ba(int tree, char *x, long ad);
long primera_llave(int tree);
long ultima_llave(int tree);
long sgte_llave(int tree);
long prev_llave(int tree);
void val_llave(int tree, char *ky);
long actual_llave(int tree);
```

ARCHIVO BCC.H

```
#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>

#ifndef _BCC_
#define _BCC_

typedef void *bcc;

/* Funciones del m¢dulo */
bcc  crea_bcc     (int t_d, int t_e, int np);
void dest_bcc     (bcc b);
void *vacia_bcc   (bcc b);
int  bcc_vacio    (bcc b);
void *actual_bcc  (bcc b, void *dato, int np);
void *ins_bcc     (bcc b, void *dato);
void *bor_bcc     (bcc b, void *dato, int n_proc);
void *ini_bcc     (bcc b, int n_proc);
void rec_bcc      (bcc b, void (*fun)());

/*
Notas: - El par metro entero que aparce en agunas funciones indica:
       Para crea_bcc indica el n£mero de procesos que comparten el
buffer.
       Para actual_bcc, bor_bcc e ini_bcc, indican que la operaci¢n
       correspondiente ser  realizada respecto al proceso np.

       - En la funci¢n ins_bcc el error EST_LLENA no se activa porque
un
       BCC es una estructura est tica circular.
*/
#endif/* --------------- BTREE.H ---------------------- */

#define TURBOC      8   /* Turbo C                          */

#define ERROR -1
#define OK 0

#ifndef TRUE
#define TRUE 1
#define FALSE 0
#endif

#include <errno.h>
#include <fcntl.h>
#include <ctype.h>
#include <sys\types.h>
#include <sys\stat.h>

#define OPENMODE O_RDWR+O_BINARY  /*Modo de lectura y de escritura
simult neo */
#define CMODE S_IWRITE   /*Escritura y lectura permitida en archivos
nuevos */
#define NODE 512        /* Longitud de un nodo de un B-Arbol*/
#define RPTR long       /* Nodo de un b- rbol. Tipo guardado en un
nodo */
```

```
long lseek();
extern int errno;


/* ----------- Valores de error------ */
#define D_NF       1 /* Registro no encontrado          */
#define D_PRIOR    2 /* No hay registro previo          */
#define D_EOF      3 /* Fin de archivo                  */
#define D_BOF      4 /* Principio de archivo            */
#define D_DUPL     5 /* Llave duplicada                 */
#define E_FM       6 /* Memoria agotada                 */
#define D_INDXC    7 /* Indice alterado                 */
#define D_IOERR    8 /* Error de e/s                    */

#define MXKEYLEN 80 /* Longitud maxima de una llave     */
```

ARCHIVO COLA.H

```
#ifndef _COLA_
#define _COLA_
typedef void* cola;      /* Tipo cola    */

/*  Funciones para manipulaci¢n de colas */
cola crea_cola     (int t_c, int t_d);
void dest_cola     (cola c);
int  cola_vacia    (cola c);
cola vacia_cola    (cola c);
void *actual_cola (cola c, void *dato);
void *ins_cola     (cola c, void *dato);
void *bor_cola     (cola c, void *dato);
void rec_cola      (cola c, void (*funcion)());
#endif
```

ARCHIVO COMB.H

```
int comb_itera(int *a, int n, int k, int inicio);
```

ARCHIVO MARCAS.H

```
Este m¢dulo permite el manejo de conjuntos no disjuntos contenidos en
un universo finito dado, suponemos que este universo está formado por
'n' elementos y sus nombres son 0,1,...,n-1. La operaciones del m¢dulo
son: crear el unverso, insertar y borrar un elemento, preguntar si un
elemento pertenece a un conjunto.
*/


#ifndef _CONJUNTO_
#define _CONJUNTO_

extern "C" {
#include "ld.h"
#include "pg.h"
#include "dossim.h"
}

class Conjunto {
private:
   unsigned int *marcas;// Arreglo para las marcas
   int nmarcas;         // N£mero de elementos en el arreglo
   ld  lm;        // Lista de marcas
   void marcaInt(int i);// Función para marcar internamente un
elemento
   void borraInt(int i);// Función para borrar internamente un
elemento
public:
   Conjunto(int n);
   ~Conjunto(void);
   void ins(int i);     // Inserta en el conjunto el i-ésimo elemento
   void insdes(int i);  // Inserta después del elemento actual
   void insant(int i);  // Inserta antes del elemento actual
   void bor();          // Borra del conjunto al i-ésimo elemento
   int pert(int i);     // Predicado, inidica la pertenencia del elto.
i.
   void vacia(void);    // Vacía el conjunto
   int actual(void);   // Regresa el elemento actual
   void inv();            // Invierte la lista de elementos
   int *compacta();     // Guarda en una lista doble los elementos del
conjunto,
                  // la función crea la lista doble
```

```
   void expande(int *l);// Expande las marcas compactadas en una lista
doble
                    // Se destruye la lista doble
   void copia(Conjunto *a); // Copia a *a el conjunto
   void imp(char *s);    // Imprime el contenido del conjunto
   int tamano();         // Regresa el número de elementos en el
universo
   int esvacio();        // Indica si o no el conjunto es vacio
   int card();           // Regresa la cardinalidad del conjunto
   int ini();      // Indice del primer elemento insertado en el
conjunto
   int fin();            // Trae el último elemento marcado
   int sgt();            // Regresa el siguiente elemento marcado
   int ant();            // Regresa el elemento anterior
   };

#endif
#define N_DOM 10              /* Número máximo de dominios */
#define INT    0              /* DEFINICIÓN DE LA TABLA DE DOMINIOS */
#define CHAR   1
#define FLOAT  2
#define STRING 3
#define LONG   4

typedef struct                /* Tipo base de la tabla de dominios */
   {
   int tam;                   /* Número de bytes que ocupa, 0 indica
long. var*/
   void (*string_a_dom)();    /* Convierte de cadena de caracteres al
dominio */
   void (*dom_a_string)();    /* Convierte de dominio a cadena de
caracteres */
   int  (*comp)();            /* Compara dos elementos del dominio */
   } tab_dom;

/* Funciones para la conversi¢n de valores del dominio al que
corresponden, a
   cadenas de caracteres y viceversa.
*/
void string_a_int    (int *i, char *s   );
void string_a_float  (float *f, char *s );
void string_a_char   (char *c, char *s  );
void string_a_string (char *s2, char *s1);
void string_a_long   (long *i, char *s  );
void    int_a_string (char *s, int *i   );
void  float_a_string (char *s, float *f );
void   char_a_string (char *s, char *c  );
void   long_a_string (char *s, long *i  );

/* Tabla de Dominios
*/
extern tab_dom  t_d[N_DOM];
extern int n_dom;#ifdef UNIX
void clrscr(void);
```

```
unsigned coreleft(void);
char getch(void);
void init_keyboard(void);
void close_keyboard(void);
int kbdhit(void);
int readch(void);
#endif
```

ARCHIVO LISTA_D1.H

```
*/

#ifndef LD
#define LD
typedef void *ld;       /* Tipo lista doblemente encadenada. */

#endif
```

ARCHIVO GRAF.H

```
Este archivo contiene la programación de varios algoritmos en Teoría
de Grafos.
*/

int acopla_bipgraf(short **a, int *ren, int *col, int nren, int n, int
inicio);
```

BIBLIOTECA

ARCHIVO LD.H

```
*/

#ifndef _LD_
#define _LD_
typedef void *ld;       /* Tipo lista doblemente encadenada. */

int  ld_vacia    (ld l);
void *ini_ld     (ld l);
void *fin_ld     (ld l);
void *sgt_ld     (ld l);
void *ant_ld     (ld l);
void *actual_ld  (ld l, void *dato);
void rec_ld      (ld l, void (*f)());
ld   crea_ld     (int t_d);
void dest_ld     (ld l);
ld   vacia_ld    (ld l);
void *insant_ld  (ld l, void *dato);
void *insdes_ld  (ld l, void *dato);
void *bor_ld     (ld l, void *dato);
void *bus_ld     (ld l, void *dato, int (*compara)());
void *mod_ld     (ld l, void *dato);
void  ord_ld     (ld l, int (*compara)());
int  card_ld     (ld l);
void  inv_ld     (ld l);
void *ld_fin     (ld l);
void *ld_ini     (ld l);

/*
Notas: - Las funciones 'insant' e 'insdes' son casos particulares de
la operación 'inserta'.
       - La función 'bor_ld' trata de mover el apuntador del nodo
actual al siguiente nodo, si no se puede lo trata de mover al
anterior, y si nuevamente no se puede lo pone apuntando a NULL.
*/
#endif
```

ARCHIVO mc.h

```
#include <stdio.h>

#ifndef _MC_
#define _MC_
typedef struct
   {
     int *dims,        /* Arreglo de dimensiones */
       *aux,           /* Auxiliar, usado en asigna_mc y elto_mc */
       max,            /* Valor maximo en la matriz */
       min,            /* Valor minimo en la matriz */
       rango,          /* max-min+1 */
       dim,            /* Numero de dimensiones */
       por_entero;     /* Numero de elementos por entero */
     long  t_e;        /* Tamano de los arreglos de datos */
     unsigned *pot;    /* Arreglo de potencias de rango */
} mc;

mc       *creaenc_mc (int min, int max, int dim, ...);
void      destenc_mc (mc *e);
unsigned *creadat_mc (mc *e);
void      destdat_mc (unsigned *d);
void       dupdat_mc (mc *e, unsigned *destino, unsigned *fuente);
void          asg_mc (mc *e, unsigned *d, int valor, ...);
int          elto_mc (mc *e, unsigned *d, ...);
void       inimin_mc (mc *e, unsigned *d);
#endif
```

ARCHIVO ORDEN.H

```
*/

#ifndef _ORDEN_
#define _ORDEN_

/* M,todos de ordenamiento: */
void     *insdir_ord (void *a, int n, int t_d, int (*compara)());
void     *seldir_ord (void *a, int n, int t_d, int (*compara)());
void       *mont_ord (void *a, int n, int t_d, int (*compara)());
void        *rap_ord (void *a, int n, int t_d, int (*compara)());
void        *sac_ord (void *a, int n, int t_d, int (*compara)());
void *insdiriter_ord (void *a, int max, int *ndatos, int t_d,
```

BIBLIOTECA

```
                    int (*compara)(), void *dato, void (*dest)(void
*));
int          min_ord (char *a,int n, int t_d, int (*comp)());
void     elimDup_ord (char *a, int *n, int t_d);
/* Estad\'isticas de orden */
#endif
```

ARCHIVO PERM.H

```
#ifndef _PERM_
#define _PERM_
int    perm        (char *a, int n, int t_d, void (*f)(void *,int));
int    perm_itera (char *a, int n, int t_d, int **m);
int subperm_itera (char *a, int n, int t_d, int **m, int *s, int ns);
#endif


#ifdef DOS
#include <mem.h>
#include <alloc.h>
#include <conio.h>
#include <dos.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <io.h>
#include "\pg\pgerror\pgerror.h"
#endif

#ifdef UNIX
#include "pgerror.h"
#include "dossim.h"
#endif
```

ARCHIVO PGERROR.H

```
#ifndef _PGERROR_
#define _PGERROR_

#include <stdarg.h>

/*#define MEDIO_INSEGURO              // Esta definicion se quita*/
                              /* cuando no se quiere hacer
                                 la verificacion de errores */

/*
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:
CODIGOS DE ERROR
*/
#define SIN_DATOS     0001              /* ADVERTENCIAS */
#define FIN_EST       0002
#define INI_EST       0003
#define NO_VACIA      0004
#define ADVERTENCIA   0005
                                        /* ERRORES */
#define NO_HALLADO    1003
#define DATO_DUP      1004
#define ERROR_DISCO   1007

#define EST_LLENA     2001              /* ERRORES FATALES */
#define FALTA_MEM     2002
#define MAL_PAR       2005
#define MAL_AMB       2006
#define FALTA_ARCH    2008
#define ERROR_FATAL   3000


                      /* Manejador de usuario de errores */

extern void *(*pguerror)(int nerror, char *merror, ...);
int  pgnerror   (void);
char *pgmerror  (void);
void *pgerror   (int nerror, char *merror, ...);
void *imp_error (int n, char *m, ...);
void pglimpia   (void);

#endif
```

ESCUELA TECNICA SUPERIOR INGENIERIA INFORMATICA BIBLIOTECA

ARCHIVO PILA.H

```
#ifndef _PILA_
#define _PILA_
typedef void* pila;      /* Tipo pila */

/* Funciones para manipulaci¢n de pilas */
pila crea_pila     (int t_p, int t_d);
void dest_pila     (pila p);
int  pila_vacia    (pila p);
void *actual_pila (pila p, void *dato);
pila vacia_pila    (pila p);
void *ins_pila     (pila p, void *dato);
void *bor_pila     (pila p, void *dato);
void rec_pila      (pila p, void (*f)());
#endif
```

ARCHIVO TIEM.C

```
#ifndef _TIEM_
#define _TIEM_

void inicia_tiem();
void tiem_trans(int *horas, int *minutos, int *segundos, int
*milisegundos);

#endif
```

ARCHIVO ude.h

```
#ifndef _UDE_
#define _UDE_
typedef void* ude;      /* Tipo ude */

ude   creaenc_ude (int t_e, int t_d, int d_p);
void   inidat_ude (ude u, void *a);
int       enc_ude (ude u, void *a, int elto);
void    union_ude (ude u, void *a, int c1, int c2, int c3);
void desunion_ude (ude u, void *a, int c1, int c2);
int      card_ude (ude u, void *a);
void  destenc_ude (ude u);
#endif
```

ARCHIVO UTIL.H

```
#ifndef _UTIL_
#define _UTIL_

/* Funciones del m¢dulo */
void imp_ent(int *i);
void imp_car(char *c);
int  comp_ent(int *e1,  int *e2);
int  comp_car(char *c1, char *c2);
int  existe_arch(char *nombre);
void genera_nombre_aleatorio(char *aleatorio);

#endif
```

Referencias

[1] F. Bartels, J. Mount, D.J.A. Welsh, The polytope of win vectors, *Annals of Combinatorics*, Volume 1, 1-15 (1997).

[2] P. Bachmann, *Die analytische Zahlentheorie.* Teubner, Leipzig (1894).

[3] N. Biggs, Colouring square lattice graphs, *London Math. Soc.* Volume 9, 54-56 (1977).

[4] G.D. Birkhoff, A determinant formula for the number of ways of coloring a map, *Annals of Math.*(2) Volume 14, 42-6 (1912).

[5] B. Bollobas. *Modern Graph Theory.* Graduate texts in Mathematics, Volume 184, SpringerVerlag (1998).

[6] J. Bourgoin, *Théories de l'ornement*, A Lévy, Paris (1873).

[7] J. Bourgoin, *Grammaire élémentaire de l'ornement, pour servir à l'histoire, à la théorie et à la patrique des arts et à l'enseignement*, Delagrave, Paris (1880).

[8] H.J. Brascamp, H. Kunz, F.Y. Wu, Some rigorous results for the vertex model in statistical mechanics, *Math. Phys.* Volume 14, Number 12 (1973).

[9] N. Calkin, C. Merino, S.D. Noble, M. Noy, Improved bounds for the number of forest and acyclic orientations in the square lattice, *The electronic journal of combinatories*, Volume 10(1) (2003).

[10] H. Crapo, The Tutte polynomial, *Aequationes Math.* Volume 3, 211-219 (1969).

[11] L.F. Day, *Pattern Design*, Batsford, London (1903).

[12] D. Garijo, I. Gitler, A. Márquez, M.P. Revuelta, *Hexagonal tilings and Locally C_6 graphs*, Preprint, (2004).

[13] D. Garijo, A. Márquez, A. Mier, M. Noy, M.P. Revuelta, Tutte Unicidad: grafos localmente grid y grafos localmente C_6. En *Actas de las III Jormadas de Matemática Discreta y Algorítmica*, páginas 261-264. Sevilla (2002).

[14] D. Garijo, A. Márquez, M.P. Revuelta, Tutte Uniqueness of Locally Grid Graphs, *Morfismos*, CINVESTAV, Mexico D.F. (2004).

[15] D. Garijo, A. Márquez, M.P. Revuelta, *Hexagonal Tilings: Tutte Uniqueness*, Preprint (2004).

[16] D. Garijo, A. Márquez, M.P. Revuelta, El polinomio de Tutte y sus aplicaciones. En *Actas del II Encuentro Andaluz de Matemática Discreta*, páginas 10-12. Los Molares (Sevilla) (2001).

[17] D. Garijo, A. Márquez, M.P. Revuelta, Teselaciones Hexagonales y Grafos Localmente C_6. En *Actas del III Encuentro Andaluz de Matemática Discreta*, páginas 27-31. Almería (2003).

[18] M.E. Gegúndez, A. Márquez, M.P. Revuelta, The Tutte polynomial on the square lattice. Preprint (2001)

[19] C. Green, Acyclic orientations (Notes), *Higher Combinatorics*, M. Aigner, ed., pp. 65-68. D. Reidel, Dordrecht (1977).

[20] C. Greene, T. Zaslavsky, On the interpretation of Whitney numbers through arrangements of hyperplanes, zonotopes, non-radon partitions, and orientations of graphs, *Trans. Amer. Math. Soc.*, Volume 280, Number 1 (1983).

[21] B. Grünbaum, G.C. Shepard, *Tilings and Patterns.* Editors: W.H. Freeman and Company (1987).

[22] J.I. Hall, Locally petersen graphs, *J. Graph Theory*, Volume 4, 173-187 (1980).

[23] F. Harary, *Graph Theory.* Addison Wesley Publishing Company (1972).

[24] I. Keppler, *Harmonice Mundi*, Lincii (1619).

[25] E. Khan, D. Slock, A polynomial time algorithm for exact maximum likelihood decoding of mimo channels: a discrete geometric approach. Preprint (2004).

[26] D.E. Knuth, Big omicron and big omega and big theta. *SIGACT News*, Volume 8(2),18-24 (1976).

[27] K. Kuratowsky, Sur le problème des courbes gauches en topologie. *Fund. Math.*, Volume 15, 271-283 (1930).

[28] J.M. Marín, A. Márquez, M.P. Revuelta, Chromatic Polynomials, Bell Numbers and Chromatic Zeros on the square lattice. Preprint (2001).

[29] A. Márquez, A. de Mier, M. Noy, M.P. Revuelta, Locally grid graphs: clasification and Tutte uniqueness, *Discrete Mathematics*, Volume 266, 327-352 (2003).

[30] C. Merino, D.J.A. Welsh, Forest, colourings and aciclic orientations of the square lattice, *Annals of Combinatorics*, Volume 3, 417-419 (1999).

[31] A. de Mier, M. Noy, On graphs determined by their Tutte polynomials, *Graphs Combin*, Volume 20, no.1, 105-119 (2004).

[32] R.C. Read, An introduction to chromatic polynomial, *J. Comb. Theory*, Volume 4, 52-71 (1968).

[33] R.C. Read, W.T. Tutte, Chromatic Polynomials, *Selected Topics in Graph Theory 3*, Academic Press, London (1988).

[34] R. Stanley, Acyclic orientations of graphs, *Discrete Math.*, Volume 5, 171-178 (1973).

[35] C. Thomassen, Tilings of the Torus and the Klein Bottle, and vertex-transitive graphs on a fixed surface, *Trans. Amer. Math. Soc.*, Volume 323, Number 2 (1991).

[36] W.T. Tutte, A contribution to the theory of chromatic polynomials, *Cand. J. Math*, Volume 6, 80-91 (1954).

[37] W.T. Tutte, *Graph Theory*, Addison Wesley, California (1984).

[38] D.J.A. Welsh, Percolation and the random cluster model: combinatorial and algorithmic problems, *Probabilistic Methods for Algorithmic Discrete Mathematics.* Editors: M. Habib, C.McDiarmid, J.Ramirez-Alfonsin, B.Reed. (1991).

[39] D.J.A. Welsh, *Complexity: Knots, Colourings and Counting*, Cambridge University Press, Cambridge (1993).

[40] W. von Wersin, *Das elementare Ornament und seine Gesetzlichkeit*, Maier, Ravensburg (1953).

[41] T. Zaslavsky, Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes. *Mem. Amer. Math. Soc.*, Volume 1, Issue 1; Number 154 (1975).

BIBLIOTECA

DELIA GARIJO 2040
POLINOMIO DE TUTTE DE TESELACIONES
REGULARES

SOBRESALIENTE CUM LAUDE

POR UNANIMIDAD 14 OCTUBRE 2004

El Presidente,