LA COMPLEJIDAD COMPUTACIONAL DE CONTAR POLÍMEROS

FRANCISCO JAVIER GUTIÉRREZ LIZARAZO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICAS
BUCARAMANGA
2013

LA COMPLEJIDAD COMPUTACIONAL DE CONTAR POLÍMEROS

FRANCISCO JAVIER GUTIÉRREZ LIZARAZO

Trabajo de grado presentado como requisito para optar al título de Magister en Matemáticas

Director
RAFAEL ISAACS GIRALDO
Ph.D. en Matemáticas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICAS
BUCARAMANGA
2013

"A mi madre Luz Marina, en conmemoración de sus primeros cincuenta años de vida".

Agradecimientos

Quiero expresar mis mas sinceros agradecimientos (me disculparán si se me olvido a alguien)

A mis padres, Reyes Ignacio Gutiérrez Salcedo y Luz Marina Lizarazo Sánchez; mis hermanos, Manuel Ignacio y Ronald Mauricio y a todos mis familiares cercanos, por toda su colaboración, apoyo y paciencia durante toda la maestría.

A Edinson Archila, Jose Luis Puello García, Óscary Ávila Hernández, Sergio Montoya, Duwang Prada, Rosana Martínez, Arturo Castro, Jáiver Rodríguez, por su sincera amistad y por todos los momentos de academia y de no academia. Y a mi querida Jenifer Tatiana Delgado, quien apoyó esta tarea con su compañía, comprensión, fuerza y amor.
A el profesor Juan Andrés Montoya A. por su apoyo académico y económico, gracias a él conocí parte del hermoso mundo de la teoría de la computación, pero muy especialmente a Rafael Isaacs Giraldo mi profesor y mecenas, uno de los culpables de haber empezado y terminado esta maestría.

Índice general

Índice de figuras

TÍTULO: LA COMPLEJIDAD COMPUTACIONAL DE CONTAR POLÍMEROS[1]
AUTOR: FRANCISCO JAVIER GUTIÉRREZ LIZARAZO[2]

PALABRAS CLAVES: Teoría de la computación, Complejidad computacional, problemas de conteo, matchings, caminos hamiltonianos, algoritmos de aproximación, cadenas de Markov, Monte Carlo, Método de Shutzenberger, autómatas, lenguajes regulares.

DESCRIPCIÓN:

La teoría de la computación es el estudio de aquellos problemas que pueden ser resueltos algorítmicamente. Pero aunque podamos resolver un problema algorítmicamente esto no garantiza que podamos conocer su solución en un tiempo razonable para todas las instancias de dicho problema. La Complejidad Computacional se encarga de responder cuales problemas son computacionalmente tratables y cuales no, clasificándolos en lo que se conocen como clases de complejidad. Para los problemas de conteo algunas de estas son las clases FP, $\#P$ y $\#P$-completos. En este trabajo estudiaremos la complejidad computacional de algunos problemas de conteo entre los que se destacan el conteo de emparejamientos perfectos en grafos planos, el de emparejamientos en general y el de caminos que se autoevitan, que están relacionados con dos modelos de la Mecánica Estadística: El modelo monómero-dímero y el modelo SAW de la termodinámica de polímeros. Mostraremos que contar los árboles generadores de un grafo conexo y los matchings perfectos en grafos planos es computacionalmente tratable gracias al ingenioso teorema de Kasteleyn, y que contar matchings y caminos que se autoevitan son problemas en la clase computacional de los problemas $\#P$-completos. Evidencia suficiente para creer que estos dos problemas son computacionalmente intratables. Después de esto estudiaremos dos algoritmos de aproximación probabilística que nos permitirá aproximar la solución de dichos problemas, para finalizar con el estudio de dos subproblemas tratables acerca de caminos que se autoevitan utilizando el método de Shutzenberger: el problema de contar caminos sin descenso en el retículo bidimensional y el problema de contar caminos hamiltonianos en retículos rectangulares de altura fija.

[1]Trabajo de grado
[2]Facultad de Ciencias, Escuela de Matemáticas. Director: Ph.D. Rafael Isaacs Giraldo.

TITLE: THE COMPUTACIONAL COMPLEXITY OF COUNTING POLIMERS[1]
AUTHOR: FRANCISCO JAVIER GUTIÉRREZ LIZARAZO [2]

KEY WORDS: Theory of computation, computational complexity, counting problems, hamiltonian walks, approximation algorithms, markov chains, Monte Carlo, Shutzenberger method, automata, regular language.

DESCRIPTION:

Theory of computation studies which problems can be resolved with algorithms. But even if we can solve a problem with any algorithm this is not guarantee that we can know its solution in short time for any instance of such a problem. Computational complexity answers which problems are computational tractable and which are not, classifying them in complexity classes. For counting problems some of such classes are FP, $\#P$ y $\#PC$ classes. In this work we're going to study the computational complexity of some problems such as the counting of perfect matchings, the counting of general matchings and self avoiding walks. These problems are related two models of the Statistical Mechanics: the monomer-dimer SAW model (self avoiding walks model). We will see that the counting of spanning trees in connected graphs and perfect matchings in planar graphs (on account of Kasteleyn Theorem) are tractable problems; and counting matchings and self avoiding walks are problems that belong to the $\#PC$ computational complexity class. When a counting problem is in $\#PC$ this means strong evidence to think that it is a intractable counting problem. Then we're going to study two probabilistic approximation algorithms to solve in approximation way the problems of counting matching and walks in lattices of any dimension. We finish this work with the study of two tractable subproblems of the counting of self avoiding walks using the Shutzenberger method, namely the problem of counting up-side walks in a bidimensional lattices and the problem of counting hamiltonian walks in rectangular lattices whit a fixed high.

[1]Paper work

[2]Faculty of Sciences, School of Mathematics. Supervisor: Ph.D. Rafael Isaacs Giraldo

Introducción

En muchas ocasiones los problemas de conteo han jugado un rol importante para la ciencia, solo para citar un ejemplo, en 1857 Sir Arthur Cayley estaba interesado en enumerar ciertos compuestos orgánicos; los isómeros de hidrocarburos saturados, cuya fórmula es C_kH_{2k+2}: los átomos de carbono tienen valencia 4, mientras que los átomos de hidrógeno tienen valencia 1. Cayley represento estos compuestos como grafos conexos con k vértices de grado 4 y $2k+2$ vértices de grado 1 que representaban los átomos de carbono e hidrógeno respectivamente, es decir grafos con $3k+2$ vértices, cuya suma de grados es $6k+2$ y por lo tanto con $3k+1$ aristas. Observe que en estos grafos el número de vértices es uno más que el de sus aristas, es decir los grafos que representan estos compuestos son árboles. Si somos capaces de contar este tipo de árboles, contaríamos también el número de hidrocarburos saturados. En el capítulo 2 estudiamos la fórmula de Cayley que cuenta el número de árboles generadores de un grafo completo. Otros ejemplos muy estudiados los podemos ver en los modelos de llamada Mecánica Estadística. En el capítulo 3 veremos dos modelos donde las partículas se pueden pensar están ubicadas en los vértices de retículos rectangulares bidimensionales o tridimensionales. El primero es el modelo M-DIM donde las partículas interactuan formando dímeros que se pueden ver como emparejamientos o matchings entre las partículas y el segundo es el modelo SAW donde las partículas forman caminos que se autoevitan (self avoiding walks, SAW's) larguísimos que se pueden ver como polímeros formados por una gran cantidad de partículas. Contar matchings y SAW's, es decir dímeros y polímeros es indispensable para definir la función de partición de estos modelos; funciones que revelan muchas propiedades termodinámicas de dichos modelos. Es por esto, entre otras cosas que este trabajo recibe el nombre de: *LA COMPLEJIDAD COMPUTACIONAL DE CONTAR POLÍMEROS*.

Cuando un problema de conteo se puede resolver por medio de un algoritmo cuyo tiempo de computo está acotado superiormente por un polinomio en el tamaño de las instancias se acostumbra a decir que el problema es tratable o que esta la clase de complejidad FP. Por ejemplo veremos en el capitulo 2 que contar árboles generadores y matchings perfectos en grafos planos son dos problemas de conteo tratables.

Existen otras clases de complejidad a saber: $\#P$ y $\#PC$, que nos permitirán reunir suficiente evidencia para pensar que algunos problemas no pueden se tratables. Por ejemplo contar matchings de cualquier tamaño y SAW's se cree son problemas no tratables. En el capítulo 4 usaremos algoritmos probabilísticos de aproximación para calcular el número de mathings y de SAW's en el retículo rectangular infinito $(\mathbb{Z} \times \mathbb{Z})$ de manera aproximada. Finalmente en el capítulo 5, estudiaremos el método de Shutzenberger que nos permitirá resolver dos subproblemas tratables a cerca de SAW's: el problema de contar caminos sin descenso en el retículo bidimensional y el problema de contar caminos hamiltonianos en retículos rectangulares de altura fija.

Capítulo 1

Problemas de conteo delgados

Solucionar un problema de conteo es equivalente a calcular el cardinal del conjunto de todos los elementos que se desean contar. Por ejemplo (*problema de los números primos*), si lo que deseamos contar son todos los números primos menores que un millón, algo que podemos hacer es calcular el cardinal del siguiente conjunto:

$$P(10^6) = \{m : m \text{ es primo y } m < 1{,}000{,}000\}$$

De la misma manera podemos formular infinidad de problemas de la misma naturaleza tan solo cambiando el tamaño de la cota, es decir ¿cuántos numeros primos hay menores que n (para todo $n \in \mathbb{N}$)?, o sea cuál es el cardinal de $P(n) = \{m : m \text{ es primo y } m < n\}$ (para todo $n \in \mathbb{N}$). Si observamos por cada valor de n (la cota) va existir un único $|P(n)|$. Haciendo unos primeros cálculos $|P(1)| = 0$, $|P(2)| = 0$, $|P(3)| = 1, \cdots$, $|P(10)| = 4, \cdots, |P(20)| = 8, \cdots, |P(256)| = ?, \cdots$. Implícitamente lo que hemos definido es una función (llamada función de conteo del problema) $f : \mathbb{N} \to \mathbb{N}$, donde $f(n) = |P(n)|$. Habremos solucionado el *problema de los números primos* si podemos calcular la función f. De manera general tenemos la siguiente

Definición 1.0.1. sea $\mathcal{A} = \{A_i\}_{i \geq 1}$ una sucesión de conjuntos finitos, la *función de conteo* asociada a $\mathcal{A}$ es la función $f_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ definida por:

$$f_{\mathcal{A}}(n) = |A_n|$$

Muchos de los objetos intensamente estudiados en combinatoria son funciones de conteo. Considere los siguientes ejemplos:

Ejemplo 1.0.1.

1. Sea $\mathcal{G} = \{G_i\}_{i \geq 1}$ la secuencia definida por: dado $i \geq 1$ el conjunto G_i es el conjunto de los tipos de isomorfismo de grupos de orden i. La función de conteo $f_{\mathcal{G}}$ asociada es:

$$f_{\mathcal{G}}(n) = \text{ número de grupos de orden } n \text{ (salvo isomorfismo)}$$

2. Sea $\mathcal{P}_r = \{Pr_i\}_{i\geq 1}$ la secuencia definida por: dado $i \geq 1$ el conjunto Pr_i es el conjunto de los números menores o iguales que i que son primos relativos con i. La función de conteo asociada es el famoso *Tociente de Euler* definido por:

$$\phi(i) = |\{n \leq i : mcd(i, n) = 1\}|$$

Dada una función de conteo de un problema, podemos asociar a f el problema algorítmico definido por :

Problema($\#f$, calculo de f)
• Input: 1^n, donde n es un entero positivo.
• Problema: Calcule $f(n)$.

Observe que en el problema anterior al codificar el input de esa forma, para cada n entero positivo hay solo un input de tamaño n. A este tipo de problemas algorítmicos los llamaremos **problemas delgados**.
El tipo de problemas algorítmicos que consideraremos en este trabajo, son los problemas de la forma $\#f$, donde f es una función de conteo de un problema delgado.

Capítulo 2

Problemas de conteo tratables

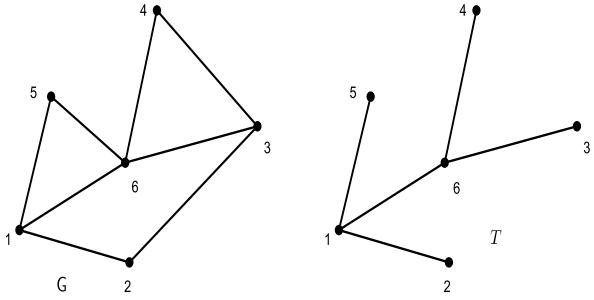
Diremos que un problema de conteo es tratable cuando existe un algoritmo que calcula la función de conteo asociada en tiempo polinomial; es decir el tiempo de computo del algoritmo esta acotado superiormente por un polinomio en el tamaño del input. De estos problemas hay muy pocos no triviales que poseen soluciones combinatorias; es decir que su función de conteo sea una fórmula cerrada que depende del tamaño de sus instancias. En este capítulo presentaremos dos problemas no triviales tratables, tomados de [4] . El primero es el conteo de subárboles generadores de un grafo completo, el segundo el conteo de matchings (emparejamientos) perfectos. Los dos serán solucionados de manera ingeniosa utilizando herramientas del algebra lineal, y curiosamente se reducen a calcular el determinante de una matriz.

2.1. Número de árboles generadores

Sea $G = (V, E)$ un grafo con conjunto de vértices $V = \{1, 2, \cdots n\}$ y conjunto de aristas E. Un *árbol generador* de G es un subgrafo de G que es un árbol y cuyo conjunto de vértices es exactamente V o equivalentemente son los subgrafos conexos de G con $n - 1$ aristas y cuyo conjunto de vértices es V (ver figura 2.1). *La matriz de adyacencia* de G es la matriz A de tamaño $n \times n$, donde $a_{i,j}$ es 1 si $\{i, j\} \in E$ y es 0 en caso contrario. *La matriz de grado* de el grafo G es la matriz triangular $D = diag(grado(1), grado(2), \cdots, grado(n))$. El *Laplaciano* de G es la matriz $L = D - A$. $L_{i,i}$ denotará la matriz resultante de eliminar la i-ésima fila y la i-ésima columnna de la matriz L, y es llamada el *i-ésimo Laplaciano reducido* de el grafo G. El siguiente teorema relaciona el número de árboles generados de un grafo con su laplaciano reducido.

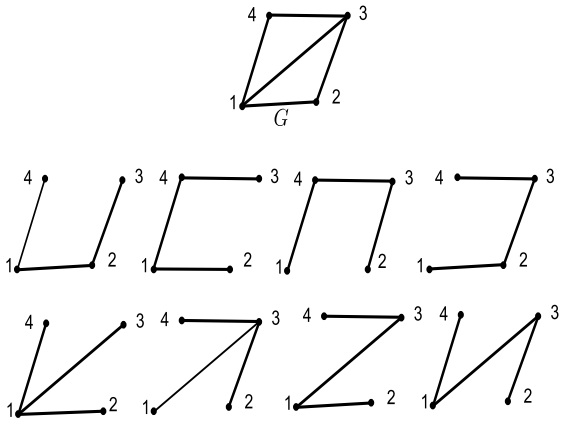
Teorema 2.1.1 (Kirchhoff)**.** *Sea $G = (V, E)$ un grafo conexo con n vértices, entonces para cualquier i, tal que $1 \leq i \leq n$, se cumple que el número de árboles generadores de G es igual a $det(L_{i,i})$.*

La figura 2.2 muestra un grafo G y ocho sus árboles generadores. Verifique por



Grafo conexo G y árbol generador T.

Figura 2.1:



Subárboles generadores de G

Figura 2.2:

ejemplo que $\det(L_1) = 8$.

Gustav Robert Kirchhoff (1824-1887) uno de los padres de la teoría de grafos y quien era además físico, estudió intensamente la teoría de las redes eléctricas (circuitos). Kirchhoff representó una red eléctrica como un grafo conexo. Él estaba interesado en la *fiabilidad* de sus redes, esto es, si la red puede ser capaz de soportar la pérdida de algunos tramos (aristas) sin que el fluido se vea interrumpido. El que el grafo abtraído tenga muchos árboles generadores sugiere que el sistema será bastante robusto, y que podrá sobreponerse a estas pérdidas. Estas cuestiones le interesaban, y mucho, a Kirchhoff, quien en 1847 enunció y demostró el teorema 2.1.1. Por otro lado es un hecho conocido que el determinante de una matriz $n \times n$ se puede calcular utilizando eliminación Gaussiana en tiempo $O(n^3)$, por lo tanto existe un algoritmo de tiempo polinomial que resuelve el problema de contar el numero de árboles generadores de cualquier grafo conexo. Utilizaremos el teorema de Kirchhoff para

probar que el numero de árboles generadores de el grafo completo de orden n, es exactamente n^{n-2}, a este resultado usualmente se le conoce como el teorema de Cayley.

Antes de probar el teorema 2.1.1 enunciaremos y probaremos algunos lemas necesarios.

Lema 2.1.1 (Binet-Cauchy)**.** *Sean A y B matrices $(r\times m)$ y $(m\times r)$ respectivamente, con $r \leq m$, entonces*

$$det(AB) = \sum_{S\subseteq[m],|S|=r} detA_S \cdot detB_S,$$

donde $[m] = \{1, 2, \cdots, m\}$, A_S es la matriz cuadrada que resulta de eliminar las columnas de A cuyos indices no están en S, y de manera similar B_S es la matriz cuadrada que resulta de eliminar las filas de B cuyos indices no están en S.

Demostración.
Si probamos que

$$det(A \triangle B) = \sum_{S\subseteq[m],|S|=r} detA_S \cdot detB_S \cdot \prod_{i\in S} e_i,$$

donde $\triangle = diag(e_1, e_2, \cdots e_m)$, el lema 2.1.1 será un caso particular haciendo $e_i = 1$, para $i = 1, 2, \cdots, m$. Las entradas de $A \triangle B$ son de la forma $\sum_{k=1}^{m} a_{ik}b_{kj}e_k$, por lo tanto $det(A \triangle B)$ será un polinomio homogéneo de grado r en $e_1, e_2, \cdots e_m$ (si las vemos como variables). Observemos que en dicho polinomio cada término tiene exactamente r variables diferentes. Para ver esto consideremos un término con menos de r variables diferentes, haciendo 1 estas variables y 0 el resto obtenemos el coeficiente de dicho término que viene a ser igual a $det(A\triangle B)$. Como $rank(A\triangle B) \leq rank\triangle < r$ por lo tanto $det(A \triangle B) = 0$. Ahora consideremos un término con exactamente r variables diferentes digamos e_i, con $i \in S$ y $|S| = r$. El coeficiente de $\prod_{i\in S} e_i$ es exactamente $det(A \triangle B)$ haciendo 1 estas variables con índice en S y 0 el resto, pero observemos que $(A \triangle B)_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}e_k = \sum_{k\in S} a_{ik}b_{kj} = (A_S B_S)_{ij}$, para $i, j \in \{1, 2, \cdots, r\}$, por lo tanto $A \triangle B = A_S B_S$ y $det(A \triangle B) = det(A_S B_S) = detA_S \cdot detB_S$. ☑

Sea H un grafo dirigido con n vértices y m aristas, la *matriz de incidencia* de H es la matriz N de tamaño $(n \times m)$ definida así:

$$(N)_{ve} = \begin{cases} 1, & \text{si } v \text{ es la cabeza de } e; \\ -1, & \text{si } v \text{ es la cola de } e; \\ 0, & \text{en otro caso.} \end{cases}$$

Las *componentes débilmente conexas* de H son las componentes conexas de H (ignora las orientaciones de la aristas). Dado H denotaremos a la familia de subconjuntos de $V(H)$ que son precisamente los vertices de alguna componente débilmente conexa de H como $\mathcal{C}(H)$. Note que $|\mathcal{C}(H)|$ es el numero de estas componentes.

Lema 2.1.2. *Si N es la matriz de incidencia de un grafo dirigido H con n vértices, entonces $rank\, N = n - |\mathcal{C}(H)|$.*

Demostración.
Sea N^T la matriz transpuesta de N y $v \in \mathbb{R}^n$. Es fácil ver que $N^T v = 0$ si y sólo si v es constante en cada uno de los elementos de $\mathcal{C}(H)$. Por lo tanto $dim(Ker\, N^T) = |\mathcal{C}(H)|$ y esto implica que $rank\, N = rank\, N^T = n - ker\, N^T = n - |\mathcal{C}(H)|$. ☑

Lema 2.1.3. *Sea B una matriz cuadrada cuyas entradas están en $\{-1, 0, 1\}$ y tal que en cada columna hay a lo mas un -1 y a lo mas un 1, entonces $detB \in \{-1, 0, 1\}$.*

Demostración.
Procederemos por casos. En el caso de que una columna de B solo tenga ceros es claro que su determinantes es cero. En el caso de que B tenga exactamente un -1 y un 1 en cada columna, entonces el sistema asociado a B^T tiene una solución no trivial (cualquier vector componentes iguales diferente de cero) y por lo tanto el determinante de B es igual a de B^T que es exactamente cero. Por último el caso en que existe una columna con un -1 o un 1, procederemos por inducción sobre el tamaño de B. Es claro que si B tiene tamaño 1×1 el lema se tiene. Ahora sea B con tamaño $n \times n$ y j la columna de B con $(B)_{ij} \neq 0$ y el resto de elementos cero. Desarrollando el determinante de B en la j-ésima columna, tenemos que $detB = (B)_{ij} detB_{ij}$, donde B_{ij} es una menor de B, pero por alguno de los tres casos $detB_{ij} \in \{-1, 0, 1\}$. ☑

Es el momento de demostrar el teorema 2.1.1

Demostración.
Sea $\overrightarrow{G}$ una orientación arbitraria de G, N su matriz de incidencia y $S \subseteq E$ tal que $|S| = n - 1$, entonces por el lema 2.1.2 $rank(N_S) = n - 1$ si y solo si S define un árbol generador en G (ignorando la orientación de sus aristas). Además si N' es la matriz que resulta de borrar una fila de N, entonces $rank(N'_S) = rankN$, porque la fila borrada es combinación lineal de las demás. Combinando todo lo anterior y el lema 2.1.3 tenemos que

$$det(N'_S) = \begin{cases} \pm 1, & \text{si S es un arbol generado de G;} \\ 0, & \text{en otro caso.} \end{cases}$$

Por otra parte es fácil ver que $L = D - A = NN^T$ y de esto que $L_{ii} = (D - A)_{ii} = N'(N')^T$ borrando la i-ésima fila de N. Por lo tanto aplicando el lema 2.1.1 tenemos

$$\begin{aligned} det(L_{ii}) &= det(N'(N')^T) \\ &= \sum_{|S|=n-1} detN'_S \cdot det((N')^T)_S \\ &= \sum_{|S|=n-1} detN'_S \cdot det((N')_S)^T) \\ &= \text{Número de arboles generadores de } G. \end{aligned}$$

☑

2.1.1. Un caso particular: grafos completos

Para el caso particular de grafos completos existe una formula cerrada muy sencilla que nos permite calcular el numero de árboles generadores conocida como la formula de Cayley. Es necesario para efecto de deducir dicha formula probar el siguiente resultado.

Teorema 2.1.2.

$$det\begin{pmatrix} x & y & y & \cdots & y \\ y & x & y & \cdots & y \\ \vdots & & \ddots & & \vdots \\ y & y & y & \cdots & x \end{pmatrix} = (x-y)^{m-1}(x+(m-1)y),$$

donde m es el tamaño de la matriz.

Demostración.
Si $x = y = 0$ es claro que el teorema se cumple. En otro caso sumando cada columna a la ultima columna, y sacando el factor común $x + (m-1)y$ de ella tenemos,

$$(x+(m-1)y)det\begin{pmatrix} x & y & y & \cdots & 1 \\ y & x & y & \cdots & 1 \\ \vdots & & \ddots & & \vdots \\ y & y & y & \cdots & 1 \end{pmatrix},$$

ahora multiplicando por y la última columna y restándola con el resto obtenemos,

$$(x+(m-1)y)det\begin{pmatrix} x-y & 0 & 0 & \cdots & 1 \\ 0 & x-y & 0 & \cdots & 1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix},$$

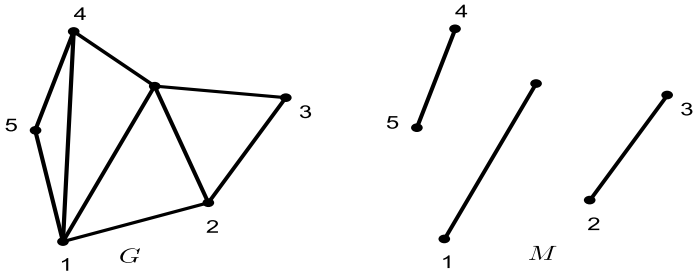
y por lo tanto el determinante de la matriz original es $(x-y)^{m-1}(x+(m-1)y)$, dado que la ultima matriz es triangular. ☑

Sabemos por el teorema 2.1.1 que el numero de árboles generadores del grafo completo K_n para $n \in \mathbb{N}$ es el determinante del laplaciano reducido,

$$\begin{pmatrix} n-1 & -1 & -1 & \cdots & -1 \\ -1 & n-1 & -1 & \cdots & -1 \\ \vdots & & \ddots & & \vdots \\ -1 & -1 & -1 & \cdots & n-1 \end{pmatrix},$$

en este caso eliminando la primera fila y la primera columna. Note que la anterior matriz tiene tamaño $(n-1)\times(n-1)$. Aplicando el teorema 2.1.2 haciendo $x = n-1$, $y = -1$ y $m = n-1$ tenemos el siguiente

Teorema 2.1.3 (Fórmula de Cayley)**.** *El numero de árboles generadores del grafo completo K_n para $n \in \mathbb{N}$ es n^{n-2}.*



Grafo G y matching perfecto M

Figura 2.3:

2.2. Número de matchings perfectos

Sea $G = (V, E)$ un grafo. Un *matching (emparejamiento) perfecto* de G es un subconjunto $M \subseteq E$ donde para todo $v \in V$, existe un único $e \in M$ tal que $v \in e$ (ver figura 2.3).
Note que para que un matching perfecto exista en un grafo, es necesario que el número de vértices sea par.
Nuestro objetivo en esta sección es probar el teorema de Kasteleyn, el cual implica automáticamente que existe un algoritmo de tiempo polinomial que resuelve el problema de contar el número de matchings perfectos en grafos planos. Para lograr esto como es costumbre introduciremos algunas definiciones, notaciones y resultados previos.

Sea $G = (V, E)$ un grafo y $E' \subseteq E$, diremos que $e \in E$ es una *arista aislada* en E' si $e \in E'$ y para toda otra arista $e' \in E$ se tiene que $e \cap e' = \emptyset$. Y dado un ciclo C en G diremos que es un *ciclo par* si C tiene longitud par.

Teorema 2.2.1. *Sean M y M' dos matchings perfectos de un grafo G. $M \cup M'$ consta de aristas aisladas en $M \cup M'$ y ciclos pares.*

Demostración.
Pensamos en M y M' como subgrafos de G. Observemos que todo vértice de G tiene grado uno o dos en $M \cup M'$. Esto no es difícil de deducir, ya que si existiera un vértice de $v \in G$ con $deg_{M \cup M'}(v) = d$ y $d > 2$, entonces existirían también d aristas diferentes $e_1, e_2, \cdots, e_d \in E(M \cup M')$, donde $v \in e_i$, para $i = 1, 2, \cdots, d$. Podemos suponer sin pérdida de generalidad que $e_1 \in M$ y $e_2 \in M'$, entonces e_d debe pertenecer a M o a M' contradiciendo que alguno de los dos es un matching perfecto de G. Procederemos ahora por casos sobre las aristas de $M \cup M'$. Si $e = \{v_1, v_2\} \in M \cap M'$ entonces, $deg(v_1)_{M \cup M'} = deg(v_2)_{M \cup M'} = 1$ si no fuera así existiría $e' \neq e$, tal que $v_1 \in e'$ o $v_2 \in e'$ contradiciendo el hecho que M o M' son matchings perfectos. Ahora si $e_1 = \{v_0, v_1\} \notin M \cap M'$, sin pérdida de generalidad podemos suponer que $e_1 \in M$, entonces $deg_{M \cup M'}(v_1) = 2$, (porque si no fuera así $e_1 \in M'$

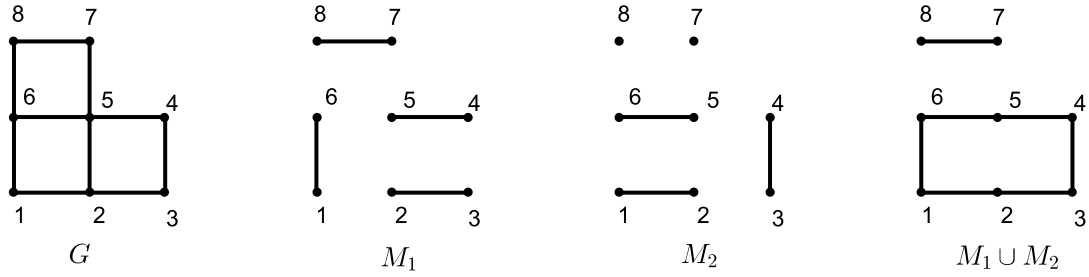


Figura 2.4:

que es una clara contradicción). Y por lo tanto existe $e_2 = \{v_1, v_2\} \in M'$, para algún $v_2 \in G$ con $deg_{M\cup M'}(v_2) = 2$. De igual forma existe $e_3 = \{v_2, v_3\} \in M$, para algún $v_3 \in G$ con $deg(v_3) = 2$. Seguimos de forma análoga hasta que encontremos el primer vértice v_k que se repita. Este procedimiento debe parar inevitablemente en algún momento dado que tenemos un número finito de vértices. Al finalizar habremos generado una secuencia de vértices $v_1, v_2, \cdots, v_k$ que forman un recorrido en $M \cup M'$. Donde $e_i = \{v_{i-1}, v_i\} \in M$, si $i \leq k$ es impar y $e_i \in M'$, si $i \leq k$ es par positivo. Afirmamos que $v_k = v_1$, si no fuera así $deg_{M\cup M'}(v_k) > 2$ que es una contradicción. Además k debe ser par, en caso contrario v_1 pertenecería a dos aristas distintas en M, contradiciendo el hecho de que es un matching de G. Por todo lo anterior concluimos que toda arista en $M \cup M'$ es aislada o pertenece a un ciclo par. ☑

La figura 2.4 muestra un grafo G, M_1, M_2 dos matchings perfectos de G y la unión de ellos.
Sea $G = (V, E)$ un grafo, C un ciclo par y $\overrightarrow{G}$ una orientación de G. Diremos que C es *imparmente orientado* por $\overrightarrow{G}$ si al recorrer C en cualquiera de los dos sentidos posibles el número de aristas en C que coinciden con la orientación $\overrightarrow{G}$ es impar. Una orientación $\overrightarrow{G}$ de G la llamaremos *Pfaffiana*[1] si para todo par M y M' de matchings perfectos de G, todos los ciclos de $M \cup M'$ son imparmente orientados.
Sea $\overrightarrow{G}$ una orientación de $G = (V, E)$, donde $V = [n]$. La *matriz de adyacencia* de $\overrightarrow{G}$ es la matriz $A(\overrightarrow{G})$ de tamaño $n \times n$, definida como

$$A(\overrightarrow{G})_{i,j} = \begin{cases} 1, & \text{si } (i,j) \in E(\overrightarrow{G}); \\ -1, & \text{si } (j,i) \in E(\overrightarrow{G}); \\ 0, & \text{en otro caso.} \end{cases}$$

para $0 \leq i, j \leq n-1$.

Con el símbolo $\overline{G}$ denotaremos el grafo dirigido obtenido del grafo $G = (V, E)$, al reemplazar cada arista $\{i, j\} \in E$, por el par de aristas dirigidas (i, j) y (j, i). Un *cubrimiento por ciclos pares* de $\overline{G}$ es una colección de ciclos pares dirigidos tales que

[1]Johann Friedrich Pfaff (1765-1825) matemático alemán, fue advisor de Carl Friedrich Gauss.

cada vértice de G está contenido en exactamente uno de dichos ciclos. El siguiente lema será de mucha ayuda a la hora de probar el teorema de Kasteleyn.

Lema 2.2.1. *Sea G un grafo. Existe una biyección entre los pares ordenados de matchings en G y los cubrimientos por ciclos pares en $\overline{G}$.*

Demostración.
Por conveniencia sea $V(G) = [n]$, para algún $n \in \mathbb{N}$ y sea (M, M') un par ordenado de matchings perfectos en G. Por el teorema 2.2.1 $M \cup M'$ es una colección de aristas aisladas y ciclos pares (las aristas aisladas pertenecen a $M \cap M'$). Construyamos un cubrimiento por ciclos pares de $\overline{G}$ de la siguiente manera, si $\{i, j\}$ es una arista aislada en $M \cup M'$ la reemplazamos por (i, j) y (j, i) obteniendo así un ciclo dirigido de longitud 2. Para cada ciclo par en $M \cup M'$ seleccionamos el vértice menor y orientamos la arista de M que contiene dicho vértice alejándose de él, y el resto de aristas en el ciclo en el mismo sentido. Obteniendo así un cubrimiento por ciclos pares de $\overline{G}$. El anterior procedimiento se puede revertir construyendo para cada recubrimiento por ciclos pares de $\overline{G}$ dos matchings perfectos M y M' así, si hay un ciclo dirigido de longitud 2, entonces incluimos dicha arista de G, en M y en M'. Y para cada ciclo par dirigido de longitud mayor que 2 en $\overline{G}$ vamos alternando cada arista de G, una pertenecerá a M y la siguiente a M', empezando por la arista que contiene el menor vértice y que además se aleja de él. Note que hemos definido implícitamente una biyección entre los pares de matchings perfectos en G y los cubrimientos por ciclos pares de $\overline{G}$. ☑

Una consecuencia inmediata del anterior lema es el siguiente corolario.

Corolario 2.2.1. *Sea G un grafo, $\mathcal{M}(G)$ el conjunto de matchings perfectos de G y $\mathcal{C}(G)$ el conjunto de cubrimientos por ciclos pares de G. Se tiene que $|\mathcal{M}(G)| = \sqrt{|\mathcal{C}(G)|}$.*

Estamos ahora preparados para demostrar el teorema de Kasteleyn

Teorema 2.2.2 (Kasteleyn)**.** *Sea G un grafo. Si $\overrightarrow{G}$ es una orientación Pfaffiana de G entonces,*

$$\#\textit{matchings perfectos en } G = \sqrt{det A(\overrightarrow{G})}$$

Demostración.
Por el anterior corolario bastará con probar que $|\mathcal{C}(G)| = det(A(\overrightarrow{G})$. Recordemos que

$$det(A(\overrightarrow{G})) = \sum_{\pi \in S_n} sgn(\pi) \prod_{i=0}^{n-1} a_{i,\pi(i)} \tag{2.1}$$

donde S_n es el conjunto de permutaciones sobre $V(G) = [n]$ y $sgn(\pi)$ es el signo de π. Como es sabido toda permutación se puede escribir como el producto único de ciclos ajenos (a dicho producto se le llama la descomposición por ciclos ajenos de la permutación). Es decir si $\pi \in S_n$, entonces $\pi = \gamma_0 \gamma_1 \cdots \gamma_k$, donde cada γ_i es un ciclo

ajeno en $[n]$, es claro que cada γ_i actúa sobre un subconjunto V_i de $V(G)$. Note que $\{(i, \pi(i)) : i \in V_i\}$ es un ciclo en G si y solo si $\prod_{i \in V_i} a_{i,\pi(i)} \neq 0$. Es decir hay una correspondencia biunívoca entre las permutaciones $\pi \in S_n$ con $\prod_{i=0}^{n-1} a_{i,\pi(i)} \neq 0$ y los cubrimientos por ciclos (no necesariamente pares) de $\overline{G}$. Si $S_{n(par)}$ es el conjunto de las permutaciones sobre $[n]$ cuya descomposición por ciclos ajenos consta solo de ciclos pares, no es difícil ver que

$$\sum_{\pi \in S_n} sgn(\pi) \prod_{i=0}^{n-1} a_{i,\pi(i)} = \sum_{\pi \in S_n(par)} sgn(\pi) \prod_{i=0}^{n-1} a_{i,\pi(i)} \tag{2.2}$$

Para ver esto sean $\pi = \gamma_0 \gamma_1 \cdots \gamma_i \cdots \gamma_k \notin S_{n(par)}$ y $\pi^{inv} = \gamma_0 \gamma_1 \cdots \gamma_i^{-1} \cdots \gamma_k$, donde γ_i es el primer ciclo ajeno de longitud impar y γ_i^{-1} el ciclo inverso de γ_i, que también tendría longitud impar. Es claro que $\prod_{i=0}^{n-1} a_{i,\pi(i)} = -\prod_{i=0}^{n-1} a_{i,\pi^{inv}(i)}$ y además $sgn(\pi) = sgn(\pi^{inv})$, por lo tanto los términos correspondientes a estas dos permutaciones se anulan en la suma 2.1 y en consecuencia la igualdad 2.2 se cumple.
Dado que hay una correspondencia biunívoca entre las permutaciones $\pi \in S_{n(par)}$ tales que $\prod_{i=0}^{n-1} a_{i,\pi(i)} \neq 0$ y los cubrimientos por ciclos pares de $\overline{G}$, si $\overline{G}$ es pfaffiana, $\pi = \gamma_0 \gamma_1 \cdots \gamma_k \in S_{n(par)}$ y V_i es el conjunto de vértices donde actúa γ_i, cada $\prod_{i \in V_i} a_{i,\pi(i)} = -1$ y $sgn(\pi) = (-1)^k$, por lo tanto π aporta 1 a la suma 2.1 sin importar si k es par o impar y por lo tanto el número de cubrimientos por ciclos pares coincide con $det(A(\overrightarrow{G}))$. ☑

Un ingrediente clave del anterior teorema es la orientación pfaffiana del grafo, si de antemano tenemos dicha orientación el teorema de Kasteleyn nos permitirá calcular en tiempo polinomial el número de matchings perfectos en el grafo. Pero cómo saber si un grafo dado posee una orientación Pfaffiana y tal vez más importante cómo construir dicha orientación sobre el grafo? Los siguientes resultados nos permiten probar que la clase de los grafos planos poseen orientaciones Pfafianas y además nos muestran un método para construir dichas orientaciones en tiempo polinomial probando con ello que el problema de contar matchings perfectos en grafos planos es computacionalmente tratable.

Lema 2.2.2. *Sea G un grafo planar conexo y $\overrightarrow{G}$ una orientación sobre G. Suponga que cada cara de G excepto la cara exterior, tiene un numero impar de aristas orientadas en el sentido de las manecillas del reloj. Entonces, en cualquier ciclo C, el numero de aristas orientadas en el sentido de las manecillas de reloj según $\overrightarrow{G}$ es de paridad opuesta a el número de vértices de $\overrightarrow{G}$ dentro de C. En particular, $\overrightarrow{G}$ es Pfaffiana (ver figura 2.5).*

Demostración.
Sean C un ciclo simple en G y
$v =:$ número de vértices adentro de C,
$k =:$ número de aristas en $C =$ número de vertices en C,
$c =:$ numero de aristas en C orientadas en el sentido de las manecillas del reloj por

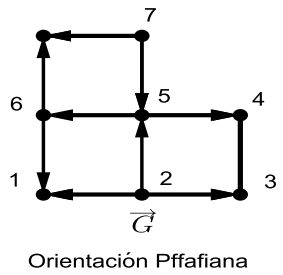


Figura 2.5:

la orientación $\overrightarrow{G}$,
$f =:$ número de caras adentro de C,
$e =:$ número de aristas dentro de C,
$c_i =:$ número de aristas en la frontera de la cara i orientadas en el sentido de las manecillas del reloj por la orientación $\overrightarrow{G}$, para $i = 0, 1, \cdots f-1$.
Según la fórmula de Euler para grafos planos ($\#vertices + \#caras - \#aristas = 2$) aplicada a el ciclo C tenemos que $(v+k)+(f+1)-(e+k) = 2$, lo cual implica que

$$e = v + f - 1 \tag{2.3}$$

Además por hipótesis se tiene que $c_i \equiv 1(mod2)$, para $i = 0, 1, \cdots f-1$. Por lo tanto $f \equiv \sum_{i=0}^{f-1} c_i(mod2)$, pero por otro lado tenemos que $\sum_{i=0}^{f-1} c_i = c+e$, esto se cumple porque cada arista interna de C pertenece a dos caras contiguas y está orientada en el sentido de las manecillas del reloj solo en una de dichas caras. De esta manera $f \equiv c + e(mod2)$, es decir $f \equiv c + v + f - 1(mod2)$ (usando la ecuación) 2.3 y por lo tanto $c + v$ es impar.
Para finalizar la prueba resta probar que $\overrightarrow{G}$ define una orientación Pfaffiana en G. Para esto sea C un ciclo formado por la unión de dos matchings perfectos de G. El número de vértices adentro de C debe ser par, si no fuera así, un vértice dentro de C debería ser adyacente en la unión de los matchings con un vértice afuera de C y por lo tanto esto contradice el hecho de que G es plano. Es decir el número de vértices dentro del ciclo C es par y por lo tanto el número de aristas orientadas en el sentido de las manecillas del reloj es impar, es decir $\overrightarrow{G}$ es una orientación Pfaffiana de G. ☑

Teorema 2.2.3. *Todo grafo plano tiene una orientación Pfaffiana.*

Demostración.
Sea G un grafo plano conexo, con n vértices y m aristas (sin pérdida de generalidad asumiremos que G es conexo, de otra forma podríamos tratar cada componente conexa por separado). Probaremos el teorema usando inducción sobre m (el número de aristas). Como base de nuestra inducción tomaremos el caso en que $m = n - 1$, es decir G es un árbol. En este caso cualquier orientación es Pfaffiana dado que

no hay ciclos en los árboles. Ahora para el caso en el que G es tal que $m \geq n$, tomamos una arista e de G en el borde de G, es decir una arista que pertenezca a la cara infinita de G. Por la hipótesis de inducción el grafo $G \setminus e$ debe tener una orientación Pfaffiana. Adicionando la arista e a esta orientación, se creará solo una nueva cara que la podemos orientar de tal manera que la nueva cara creada tenga un número impar de aristas orientadas en el sentido de las manecillas del reloj. Este procedimiento permitirá crear una orientación de G (en tiempo polinomial) donde todas sus caras tienen un número impar de aristas orientadas en el sentido de las manecillas del reloj y por el lema 2.2.2 dicha orientación es Pfaffiana. ☑

Corolario 2.2.2. *El problema de contar matchings perfectos en grafos planos es computacionalmente tratable.*

Capítulo 3

Problemas de conteo intratables

La Teoría de la complejidad computacional es la rama de la Teoría de la computación que se encarga entre otras cosas de clasificar los problemas computacionales entre aquellos que son tratables y aquellos que no. Hay dos tipos de problemas computacionales que se destacan, los problemas de decisión y los problemas de conteo. Dos Problemas de decisión son por ejemplo:

1. Dado un numero entero positivo n, decida si n es primo.

2. Dado un grafo G no dirigido, decida si G es Hamiltoniano[1].

El primer problema se sabe que es tratable por el algoritmo de primalidad AKS (ver [1]). Todos los problemas de decisión tratables se dice que están en la clase de complejidad P [2]. El segundo problema no sabe si está en la clase P, pero si en una clase de complejidad computacional llamada la clase NP[3] y además en una subclase de esta muy especial, la clase de los problemas $NP-$completos. Cuando un problema de decisión es $NP-$completo esto es evidencia suficiente para creer que no existe un algoritmo de tiempo polinomial que lo resuelve debido a que lo contrario implicaría que $P = NP$ y esto a su vez quiere decir que la gran cantidad de problemas difíciles en NP que se conocen desde hace desde hace ya varias décadas (como el segundo citado arriba) tendrán una solución en tiempo polinomial lo cual para la mayoría de los científicos de la computación es imposible[4]. Leslie Valiant en [2] definió para los problemas de conteo, clases que desempeñan el mismo rol que las clases anteriormente citadas para los problemas de decisión. Para una profundización de los temas de este capítulo ver [18].

[1] Un grafo G es hamiltoniano si posee un ciclo que visita todos los vertices de G exactamente una vez

[2] P es la clase de complejidad de los problemas de decisión que se pueden resolver con máquinas de Turing determinísticas de tiempo polinomial

[3] NP es la clase de complejidad de los problemas de decisión que se pueden resolver con máquinas de Turing no determinísticas de tiempo polinomial

[4] $P = NP$ o $P \neq NP$ es uno de los problemas del milenio del instituto Clay.

3.1. Las clases FP, $\#P$, $\#P_1$ y completez

La clase FP es la clase de las funciones de la forma $f : \{0,1\}^* \to \mathbb{N}$ que se pueden computar con máquinas de Turing determinísticas de tiempo polinomial. Es decir coincide con la clase de los problemas de conteo tratables. Como vimos en el capítulo 2 el problema de contar matchings perfectos en grafos planos es un problema que está en FP. A su vez $\#P$ es la clase de funciones $f : \{0,1\}^* \to \mathbb{N}$ tales que existe una MTND (máquina de Turing no determinística) M de tiempo polinomial donde $f(x)$ es igual a el número de caminos desde la configuración inicial a una configuración aceptante (caminos aceptantes) en el árbol de configuraciones de M en el input $x \in \{0,1\}^*$.

Teorema 3.1.1. *Una función $f : \{0,1\}^* \to \mathbb{N}$ está en $\#P$ si y solo si existe un polinomio $p : \mathbb{N} \to \mathbb{N}$ y una máquina de Turing M determinística de tiempo polinomial tal que para todo $x \in \{0,1\}^*$ se tiene que:*

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} : M \text{ acepta la palabra } xy\}|$$

donde $|x|$ es el numero de bits (ceros y unos) en x.

La palabra y se conoce como testigo o certificado de x, por lo tanto $f(x)$ calcula el número de certificados de x de tamaño polinomial respecto al tamaño de x. Para una prueba del anterior teorema ver [18]. A continuación tres problemas que están en $\#P$

Ejemplo 3.1.1.

1. *Problema*: $\#HAM$
   - input: G, grafo no dirigido.
   - output: Número de ciclos Hamiltonianos en G.

   $\#HAM \in P$ dado que cada certificado es un ciclo hamiltoniano en G, que tiene longitud n, donde n es el número de vértices en G. Además verificar que un ciclo en G es hamiltoniano o no se puede realizar utilizando una MTD de tiempo polinomial en el tamaño de G.

2. *Problema*: $\#MATCHP$
   - input: G, grafo no dirigido.
   - output: Número de matchings perfectos en G.

   $\#MATCHP \in \#P$ debido a que cada certificado es precisamente un matching perfecto de G que tiene tamaño a lo más $n/2$, donde n es el número de vértices de G. Además no es difícil diseñar una MTD (máquina de Turing determinística) que decida en tiempo polinomial si el matching dado es o no un matching perfecto en G.

3. Sea A una matriz $n \times n$. La permanente de A se define como:

$$perm(A) = \sum_{\delta \in S_n} \prod_{i=1}^{n} A_{i\delta(i)}$$

donde S_n es el conjunto de todas las permutaciones sobre $\{1, 2, \cdots, n\}$ y $A_{i\delta(i)}$ es la componente $(i, \delta(i))$ de la matriz A.

Problema: $perm_{0,1}$
• input: A, una matriz cuadrada cuyas componentes son ceros o unos.
• output: $perm_{0,1}(A)$.

Nota 1. Observe que si A es una matriz cuyas componentes son ceros o unos es claro que $perm_{0,1}(A)$ coincide con el número de $\delta \in S_n$ tales que $\prod_{i=1}^{n} A_{i\delta(i)} = 1$.

Por lo tanto $perm_{0,1} \in \#P$, porque en este caso los certificados son las $\delta \in S_n$ tales que $\prod_{i=1}^{n} A_{i\delta(i)} = 1$, que tienen tamaño n, donde el tamaño de A es $n \times n$. Además no es difícil diseñar una MTD que decida en tiempo polinomial si la permutación dada tiene o no esta propiedad en A.

Los anteriores tres problemas además de estar en $\#P$ pertenecen a una subclase muy especial de problemas llamados problemas $\#P-$completos (L. Valiant probó en [2] que $perm_{0,1}$ es uno de ellos, nosotros en este capítulo haremos lo propio con $\#MATCHP$ y para una prueba de la completez de $\#HAM$ puede ver [18]). Los problemas $\#P-$completos se cree son intratables debido a que si existiera un algoritmo de tiempo polinomial que los solucionara esto implicaría que $FP = \#P$ y esto a su vez que $P = NP$ cosa que es poco probable. Por lo tanto probar que un problema es $\#P-$completo es para nosotros evidencia suficiente para clasificarlo como intratable. Para entender mejor esto y para definir los problemas $\#P-$completos primero tenemos que definir qué es una reducción entre funciones (problemas) de conteo.

Definición 3.1.1. Sean f y $g : \{0,1\}^* \to \mathbb{N}$ dos funciones (problemas) de conteo. Diremos que f es *reducible* en tiempo polinomial a g y lo denotaremos $f \prec g$ si y sólo si existe un algoritmo que resuelve a f en tiempo polinomial utilizando un oráculo[5] para g.

Para ilustrar mejor nuestra definición de reducción damos el siguiente ejemplo.

Teorema 3.1.2. $perm_{0,1} \prec \#MATCHP$

[5]Un oráculo es un algoritmo hipotético que resuelve a g en una unidad de tiempo.

Antes de probar el teorema observe que si A es una matriz de ceros o unos de tamaño $n \times n$, podemos a partir de A construir un grafo bipartito G_A tal que $V(G_A) = X \cup Y$ donde $X = \{x_1, x_2, \cdots, x_n\}$, $Y = \{y_1, y_2, \cdots, y_n\}$ y además $\{x_i, y_i\} \in E(G_A)$ si y sólo si $A_{i,j} = 1$. Es claro a partir de esto la siguiente

Proposición 3.1.1. *Sea A una matriz de ceros y unos de tamaño $n \times n$ y $\delta \in S_n$. Entonces $\prod_{i=1}^{n} A_{i\delta(i)} = 1$ si y sólo si $M = \{\{x_i, y_{\delta(i)}\} : i = 1, 2, \cdots, n\}$ es un matching perfecto de G_A*

Y parafraseando la proposición 3.1.1 tenemos el

Corolario 3.1.1. *Sea A una matriz de ceros y unos de tamaño $n \times n$. Hay tantas δ permutaciones en S_n como matchings perfectos en G_A. Es decir $perm_{0,1}(A) = \#MATCHP(G_A)$.*

Demostración.
Combine la proposición 3.1.1 y la nota 1. ☑

Tenemos todo listo para probar el teorema 3.1.2

Demostración ***Teorema 3.1.2.***
Considere el siguiente algoritmo que resuelve $0, 1 - perm$

• Input: A matriz de ceros o unos.
1. Construya a partir de A el grafo G_A.
2. Calcule $\#MATCHP(G_A)$ utilizando un oráculo.
• output: $perm_{0,1} = \#MATCHP(G_A)$.

Es claro por el corolario 3.1.1 que el anterior algoritmo resuelve el problema $perm_{0,1}$, además es de tiempo polinomial dado que el paso 1 toma tiempo polinomial en el tamaño de A y el paso 2 toma una unidad de tiempo. ☑

Ahora si podemos definir lo que es un problema $\#P-$completo.

Definición 3.1.2. Sea $f : \{0,1\}^* \to \mathbb{N}$ una función (problema) de conteo. Diremos que f es $\#P-$completo si:

1. $f \in \#P$
2. Todo problema en $\#P$ es reducible a f.[6]

A partir de la anterior definición se tiene la siguiente

Proposición 3.1.2. *Sean f, g y $h : \{0,1\}^* \to \mathbb{N}$ tres funciones (problemas) de conteo.*

1. Si $f \prec g$ y g se puede resolver en tiempo polinomial, entonces f también.

[6]Si sólo se cumple la condición 2 diremos que el problema es $\#P-$duro.

2. *Si $f \prec g$ y $g \prec h$, entonces $f \prec h$.*

3. *Si f es $\#P-$completo, $g \in \#P$ y $f \prec g$, entonces g es $\#P-$completo.*

4. *Si f es $\#P-$completo y $f \in FP$, entonces $FP = \#P$.*

Demostración.

1. En vez de utilizar en el algoritmo que resuelve a f el oráculo para resolver a g, utilizamos el algoritmo de tiempo polinomial que lo resuelve, como una subrutina. Esto nos da un algoritmo de tiempo polinomial que resuelve a f, dado que es la composición de algoritmos de tiempo polinomial.

2. Sea $\mathcal{N}$ el algoritmo de tiempo polinomial que resuelve a f utilizando un oráculo $\mathcal{O}$ para g. Si en lugar de utilizar el oráculo $\mathcal{O}$ para resolver g utilizamos el algoritmo $\mathcal{N}'$ que resuelve a g en tiempo polinomial y que tiene un oráculo $\mathcal{O}'$ para h, tendremos un nuevo algoritmo $\mathcal{M}$ de tiempo polinomial que resuelve a f con oráculo $\mathcal{O}'$ para h.

3. Combine el inciso 2 de esta proposición (sabiendo que $g \in \#P$) y la definición de $\#P-$completo.

4. Combine la definición de $\#P-$completo y la parte 1 de esta proposición.

☑

Nota 2. Si $FP = \#P$ esto implicaría que $\#HAM$ se podría resolver en tiempo polinomial y a su vez el problema de decidir si un grafo no dirigido dado tiene a no un ciclo hamiltoniano, que es un problema $NP-$completo. Concluyendo que $P = NP$. Por esta razón decimos que probar que un problema es $\#P-$completo es evidencia suficiente de su intratabilidad debido a la parte 4 de la anterior proposición.

Y como lo prometido es deuda tenemos el siguiente

Teorema 3.1.3. *$\#MATCHP$ es $\#P-$completo.*

Demostración.
Aplicando la parte 3 de la proposición 3.1.2 sabiendo que $\#MATCHP \in \#P$, $perm_{0,1}$ es $\#P-$completo y que $perm_{0,1} \prec \#MATCHP$. ☑

Terminamos esta sección estudiando de manera breve la clase donde muchos problemas de conteo delgados[7] pertenecen, la clase de complejidad computacional $\#P_1$. En ella hay problemas que nos ocuparán en adelante y que han sido estudiados por décadas no solo por matemáticos y científicos de la computación sino incluso por físicos, químicos y biólogos. Empecemos con la siguiente

[7] ver capítulo 1 de este trabajo.

Definición 3.1.3. Una *relación delgada p-balanceada* es una relación $R \subset \{1\}^* \times \{0,1\}^*$ tal que:

1. Existe un polinomio p tal que si $(1^n, x) \in R$, entonces $|x| \leq p(n)$.

2. El conjunto R es decidible en tiempo polinomial.[8]

Dada una R relación delgada p-balanceada, R define un problema de conteo delgado de la siguiente manera:

Problema: $\#R$
• input: 1^n, donde $n \in \mathbb{N}$
• problema: compute $|\{x : (1^n, x) \in R\}|$.

Definiremos $\#P_1$ utilizando la misma noción de reducibilidad utilizada para problemas en $\#P$ (definición 3.1.1), pero con la diferencia que esta es sobre problemas delgados y que utilizaremos el símbolo $\prec_1$ para denotarla.

Definición 3.1.4. Sea $f : \{1\}^* \to \mathbb{N}$ un problema delgado de conteo. Diremos que $f \in \#P_1$ si y sólo si existe una relación R delgada p-balanceada tal que $f \prec_1 \#R$.

Ejemplo 3.1.2. Sean $n \in \mathbb{N}$ y $\mathcal{L}_n^2$ el grafo con conjunto de vértices $V(\mathcal{L}_n^2) = [n] \times [n]$, donde $[n] = \{0, 1, \cdots n-1\}$ y conjunto de aristas $E(\mathcal{L}_n^2) = \{\{(a,b),(c,d)\} : |a-c| + |b-d| = 1\}$. Definamos el siguiente problema de conteo

Problema: $\#MATCHP_1$
• input: 1^n, donde $n \in \mathbb{N}$
• problema: compute el número de matchings perfectos en $\mathcal{L}_n^2$.

Consideremos la relación $R = \{(1^n, < M >)\}$ donde M es un matching perfecto en $\mathcal{L}_n^2$ y $< M >$ es una codificación binaria de M. No es difícil ver que:

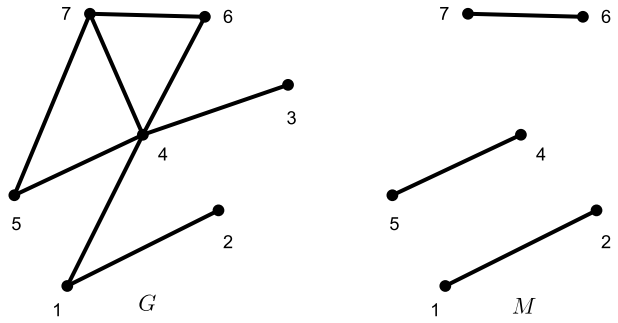
1. Si M es un matching perfecto de $\mathcal{L}_n^2$, entonces $| < M > | \leq cn^3$, donde c es una constante positiva adecuada.

2. R es decidible en tiempo polinomial ya que podemos verificar en tiempo polinomial si una cadena de ceros y unos es precisamente una codificación de un matching perfecto de $\mathcal{L}_n^2$.

3. $\#MATCHP_1(1^n) = \#R(1^n)$ y por lo tanto $\#MATCHP_1(1^n) \prec_1 \#R(1^n)$.

Es decir $\#MATCHP_1 \in \#P_1$.

[8] R es reconocido por una MTD de tiempo polinomial.



Grafo G y matching M de tamaño 3

Figura 3.1:

Note que como $\mathcal{L}_n^2$ es un grafo plano, el teorema de Kasteleyn (teorema 2.2.2)nos dice que $\#MATCHP_1 \in FP$. En $\#P_1$ también hay problemas completos[9], L. Valiant prueba en [5] que existen problemas $\#P_1$-completos y da ejemplos concretos. De igual forma que los problemas $\#P$-completos, si un problema $\#P_1$-completo puede ser resuelto en tiempo polinomial entonces $\#P_1 \subseteq FP$. Sin embargo J. Goldsmith en [6] da razones fuertes para creer que esto no puede suceder. Por lo tanto probar que un problema es $\#P_1$-completo es evidencia suficiente para creer que no existe un algoritmo de tiempo polinomial que lo resuelva.

En las siguientes dos secciones definiremos dos problemas en $\#P_1$ que nos ocuparan en adelante.

3.2. Contando Emparejamientos (Matchings)

Sea $G(V, E)$ un grafo. Un matching en G es un subconjunto M de E con la particularidad de que ningún par de aristas en M tengan vértices en común. El tamaño de un matching es precisamente el número de aristas que contiene (ver figura 3.1). Lo anterior nos permite definir los siguientes problemas de conteo.
Problema: $\#MATCH$
• input: G, grafo no dirigido.
• problema: computar el número de matchings en G de todos los tamaños.

Problema: $\#MATCH^*$
• input: (G, k), G grafo no dirigido y $k \in \mathbb{N}$.
• problema: computar el número de matchings en G de tamaño k.

[9]la definición es practicamente la misma que para el caso $\#P$-completo solo cambie $\#P$ por $\#P_1$ en la definición 3.1.2 y listo.

No es difícil probar así como se hizo en el ejemplo 3.1.1 que $\#MATCH$ y $\#MATCH^*$ están en $\#P$. Dado que el problema de computar el numero de matchings perfectos de grafos planos está en FP (gracias al ingenioso algoritmo de Kasteleyn visto en el capítulo 2) es apenas natural preguntarse qué ocurre si consideramos:

1. grafos en general.

2. matchings de cualquier tamaño.

La primera pregunta ya la respondimos en este capítulo. Según el teorema 3.1.3 $\#MATCHP$ es $\#P-$completo y por lo tanto evidencia suficiente de su intratabilidad. La segunda pregunta fue respondida por M. Jerrum. En [3] prueba que incluso para grafos planos el problema $\#MATCH$ es $\#P-$completo y por lo tanto $\#MATCH^*$ también. Todo esto explica por qué las técnicas utilizadas por Kasteleyn no pudieron ser extendidas para computar en cualquier grafo el número de matchings de un tamaño dado.

Cuando restringimos $\#MATCH$ a los retículos bidimensionales $\mathcal{L}_n^2$ tenemos un problema de conteo delgado que podemos enunciar así:

Problema: $\#MATCH_1$
• input: 1^n, donde $n \in \mathbb{N}$.
• problema: Computar el número de matchings en $\mathcal{L}_n^2$ de todos los tamaños.

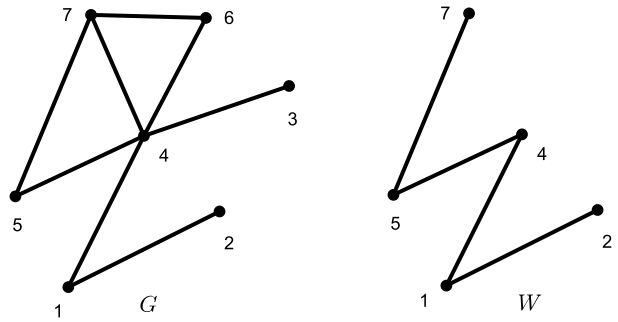
No es difícil ver así como se hizo con $\#MATCHP_1$ (ejemplo 3.1.2) que $\#MATCH_1 \in \#P_1$. Sin embargo a la fecha no sabemos si $\#MATCH_1 \in FP$ o si es un problema $\#P_1-$completo o duro. Nada sabemos a cerca de su complejidad computacional. A pesar de décadas de intenso estudio el problema sigue abierto.

3.3. Contando Caminos que se Autoevitan (SAWs)

Sea G un grafo. Un SAW (*self avoiding walk*) en G es un camino simple en G. La longitud de un SAW es el número de aristas que intervienen en él (ver figura 3.2). Definamos el siguiente problema de conteo:

Problema: $\#SAW$
• input: (G, k, v), grafo no dirigido, $k \in \mathbb{N}$ y $v \in V(G)$.
• problema: computar el número de SAWs en G de longitud k que empiezan en v.

$\#SAW \in \#P$ dado que cada certificado es un SAW en G, que tiene longitud k y $k+1$ vértices diferentes. Si n es el número de vértices en G, entonces $k+1 \leq n$. Además verificar que un camino simple en G es un SAW de longitud k que empieza en $v \in V(G)$ se puede hacer utilizando una MTD de tiempo polinomial en n.



Grafo G y SAW W de longitud 4.

Figura 3.2:

Consideremos el problema de contar los caminos hamiltonianos[10] en un grafo dado

Problema: $\#WHAM$
• input: G, v, grafo no dirigido y $v \in G$.
• problema: computar el número de caminos hamiltonianos en G que empiezan en v.

Observe que $\#WHAM \prec \#SAW$. Además es sabido que $\#WHAM$ es $\#P-$completo[11] y por lo tanto $\#SAW$ también es $\#P-$completo. Evidencia suficiente de que $\#SAW$ no se puede resolver en tiempo polinomial.

Cuando consideramos el problema $\#SAW$ solo en los grafos $\mathcal{L}_n^2$ tenemos el problema delgado:

Problema: $\#SAW_1$
• input: 1^n, donde $n \in \mathbb{N}$.
• problema: computar el número de SAWs en $\{-n, \cdots, -1, 0, 1, \cdots, n\}^2$ de longitud n que empiezan en el vértice $(0, 0)$.

La figura 3.3 muestra un SAW de longitud 15 en $\{-15, \cdots, -1, 0, 1, \cdots, 15\}^2$. Como en el ejemplo 3.1.2 no es difícil ver que $\#SAW_1 \in \#P_1$. L. Valiant pregunta en [2] por la complejidad computacional de $\#SAW_1$. Sin embargo hasta el momento no tenemos la respuesta, aunque M. Liskiewicz et al. en [8] ha hecho algunos avances en ese sentido.

[10]Un camino hamiltoniano que empieza en un vértice dado de un grafo es un camino simple que pasa por cada uno de los vértices del grafo.
[11]Porque el problema de decisión correspondiente es NP-completo ver [7].

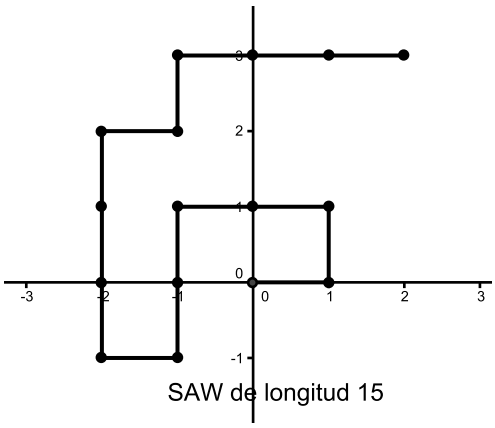


Figura 3.3:

3.4. Dos modelos de la Mecánica Estadística

La Mecánica Estadística es el estudio de cantidades tales como energía, temperatura, etc. de sistemas con un gran número de partículas utilizando métodos estadísticos. Dependiendo de cómo estas partículas interactuan entre sí en el sistema se proponen diversas configuraciones de las mismas y a su vez diversos modelos. Las partículas en dichos modelos se supone están ubicadas en los vértices de retículos rectangulares bidimensionales o tridimensionales. Conociendo el número de configuraciones posibles en un modelo propuesto podemos conocer sus propiedades termodinámicas. Por tal motivo encontrar un método eficiente de computar estas configuraciones es una tarea de gran interés en el estudio de dichos sistemas. Muchos modelos han sido propuestos y estudiados intensamente, por ejemplo el modelo MDIM y el modelo SAW.

3.4.1. Modelo MDIM

El modelo MDIM (monómero-dímero) es un modelo donde los vértices del retículo están cubiertos por un arreglo no sobrepuesto de *dímeros* y *monómeros*. Un dímero es una molécula diátomica que une dos vértices adyacentes en el retículo y un monómero cubre cada vértice no cubierto por un dímero. En términos de la teoría de grafos un arreglo de monómeros y dimeros[12] con k dímeros coincide con un matching en el retículo de tamaño k y viceversa. El modelo bidimensional MDIM sirve como un modelo para la absorción de las moléculas diátomicas en la superficie de un cristal. Donde los monómeros corresponden a lugares vacíos. El modelo tridimensional ocurre generalmente en la teoría de mezclas de moléculas de diferente tamaño y en la teoría de la agrupación celular del estado líquido.

[12]Cuando solo consideramos dímeros el modelo se conoce como el modelo DIM.

3.4.2. Modelo SAW

El modelo SAW es un modelo donde las partículas interactúan de tal manera de se forman caminos autoevitables entre algunas y otras permanecen aisladas. Un camino autoevitable puede ser visto como un polímero que forma una cadena larga con la restricción física de que dos moléculas no pueden ocupar la misma posición. Tiene múltiples aplicaciones en la teoría de polímeros, la teoría de la percolación, entre otras (ver [17]).

Capítulo 4

Monte Carlo y Conteo Aproximado

Según el capítulo anterior hay suficiente evidencia para creer que los problemas de conteo $\#MATCH$ y $\#SAW$ no se pueden resolver exactamente en tiempo polinomial. Por lo tanto en este capítulo los aproximaremos con un error tan pequeño como queramos utilizando el Método de Monte Carlo para crear algoritmos de aproximación probabilísticos.

4.1. Algoritmos probabilísticos de aproximación

Definición 4.1.1. Sea $f : \Sigma^* \to \mathbb{N}$ un problema de conteo. Un *algoritmo de aproximación probabilístico* para f es un algoritmo que toma como input una palabra $x \in \Sigma^n$ y $\varepsilon > 0$ y produce como output un número Y (una variable aleatoria) tal que

$$Pr\left((1-\varepsilon)f(x) \leq Y \leq (1+\varepsilon)f(x)\right) \geq 3/4$$

En realidad no hay nada especial en la constante 3/4 de la definición anterior. Bastaría tomar en vez de 3/4 un número estrictamente entre 1/2 y 1. Cualquier suceso con probabilidad más grande que 1/2 puede ser amplificado a $1-\delta$, para cualquier $\delta > 0$, llevando a cabo un pequeño número de ensayos y tomando el promedio de los resultados; el número de ensayos requerido es $O(ln\delta^{-1})$. Un *algoritmo de aproximación probabilístico completamente polinomial* o para resumir FPRAS (fully polynomial random approximation scheme) es un algoritmo de aproximación probabilístico que corre en tiempo polinomial en el tamaño del input y en ε^{-1}. Precisamente el objetivo de este capítulo es presentar FPRAS para los problemas $\#MATCH$ y $\#SAW$, extraídos de los artículos [13] y [14] respectivamente.

4.2. Cadenas de Markov y El método de Monte Carlo

En nuestro camino para construir los FPRAS deseados eventualmente necesitaremos responder la siguiente pregunta: Dado un conjunto finito $S = \{s_1, s_2, \cdots, s_k\}$ y una distribución de probabilidad π en S, cómo escoger elementos de S con dicha probabilidad? Responder esta pregunta será muy importante para el estudio de este capítulo, y para hacerlo necesitamos antes conocer algunas nociones básicas de las cadenas de Markov (ver [12] para un estudio más detallado).

Definición 4.2.1. Sea P una matriz de tamaño $k \times k$, una sucesión de variables aleatorias $(X_0, X_1, \cdots)$ sobre el espacio de estados $S = \{s_1, s_2, \cdots, s_k\}$ se llama una *cadena de Markov* con matriz de transición P, si para todo n, todo $i, j \in \{1, 2, \cdots, k\}$ y todo $i_o, i_1, \cdots, i_{n-1} \in \{1, 2, \cdots, k\}$ se tiene que:

$$\begin{aligned} P(X_{n+1} = s_j \,|\, X_0 = s_{i_o}, X_1 &= s_{i_1}, \cdots, X_{n-1} = s_{i_{n-1}}, X_n = s_{i_n}) \\ &= P(X_{n+1} = s_j | X_n = s_{i_n}) \\ &= P_{i,j} \end{aligned}$$

Los elementos de la matriz de transición son llamados *probabilidades de transición*. La probabilidad de transición $P_{i,j}$ es la probabilidad de estar en el estado s_j mañana dado que hoy se está en el estado s_i. La *distribución inicial* de una cadena de Markov es la distribución de probabilidad $\mu^{(0)}$ de la variable aleatoria X_0 es decir, $\mu^0 = (P(X_0 = s_1), P(X_0 = s_2), \cdots, P(X_0 = s_k))$ y en general $\mu^{(n)} = (P(X_n = s_1), P(X_n = s_2), \cdots, P(X_n = s_k))$ es la distribución de probabilidad de la variable X_n, que nos proporciona la probabilidad de que en el n-ésimo paso estemos en cualquiera de los k estados. En [12] se prueba que $\mu^{(n)} = \mu^0 P^n$, para todo $n \in \mathbb{N}$ y si la cadena empieza en un estado especifico s_i, entonces $\mu^0(j) = 1$ si $i = j$ y $\mu^0(j) = 0$ si $i \neq j$ y por lo tanto $\mu^{(n)} = \mu^0 P^n = P_i^n$ es decir la $i-$ésima fila de P^n. Si $\pi = (\pi_1, \pi_2, \cdots, \pi_k)$ es una distribución de probabilidad sobre S diremos que π es una *distribución estacionaria* si $\pi = \pi P^n$, para todo $n \in \mathbb{N}$ y que es *reversible* si $\pi_i P_{i,j} = \pi_j P_{j,i}$. En [12] se prueba que toda distribución reversible es estacionaria. Si para todo $s_i, s_j \in S$, se tiene que $(P^n)_{i,j} > 0$ para algún $n \in \mathbb{N}$, diremos que la cadena de Markov es *irreducible*. Además el *periodo* $d(s_i)$ del estado $s_i \in S$ se define como $d(s_i) = mcd\{n \geq 1 : (P^n)_{i,i} > 0\}$. Si en la cadena de Markov todos sus estados tienen período uno entonces diremos que la cadena es *aperiódica*. Y es claro que si $P_{i,i} > 0$ para todo $s_i \in S$ entonces la cadena es aperiódica, a una cadena de Markov irreducible y periódica se le llama *ergódica*.
La anterior terminología se relaciona en los siguientes teoremas demostrados en [12].

Teorema 4.2.1 (Existencia y unicidad de distribuciones estacionarias)**.** *Toda cadena de Markov ergódica tiene una y sola una distribución estacionaria.*

4.2.1. Convergencia de una cadena de Markov

Definición 4.2.2. Sea $\mathfrak{M}$ una cadena de Markov, con conjunto de estados S, matriz de transición P y distribución de probabilidad estacionaria π. La *variación de distancia* de $\mathfrak{M}$ en el paso t dado que la cadena empieza en el estado $s_i \in S$ es $\triangle_{s_i}(t) = (1/2)\sum_{y\in S}|P^t(s_i, y) - \pi(y)|$ y el tiempo para que $\mathfrak{M}$ se acerque una distancia menor que $\epsilon > 0$ dado que empieza en s_i es la cantidad $\tau_{s_i}(\epsilon) = \min\{t : \triangle_{s_i}(t') < \epsilon$ para todo $t' \geq t\}$. Diremos que la cadena de Markov *converge* a π empezando en el estado s_i si para todo $\epsilon > 0$, $\tau_{s_i}(\epsilon)$ existe.

Teorema 4.2.2 (Convergencia)**.** *Sea $\mathfrak{M}$ una cadena de Markov ergódica con distribución estacionaria π. Entonces $\mathfrak{M}$ converge a π sin importar en que estado empiece.*

El anterior teorema responde de manera aproximada la pregunta planteada al inicio de esta sección. Es decir si tenemos un conjunto $S = \{s_1, s_2, \cdots, s_k\}$ y queremos seleccionar sus elementos con una distribución de probabilidad $\pi = (\pi_1, \pi_2, \cdots, \pi_k)$, donde π_i es la probabilidad de seleccionar el elemento s_i, para $i = 1, 2, \cdots, k$; una manera de hacer esto seria construyendo (si tenemos suerte) una cadena de Markov $\mathfrak{M} = (X_0, X_1, \cdots)$ con conjunto de estados S, irreducible, aperiódica y cuya distribución estacionaria sea precisamente π (lo ultimo se podría obtener si π es reversible respecto la matriz de transición de $\mathfrak{M}$). Esto nos permitirá seleccionar elementos de S con una distribución de probabilidad tan cercana a π como queramos. Solo basta correr la cadena de Markov un número de pasos n suficientemente grande para que la variación de distancia entre la distribución de X_n y π sea tan pequeña como deseemos, sin importar en que elemento inicial de S empecemos (esto se conoce como el *mixing time* de la cadena). Este método de aproximar distribuciones de probabilidad se conoce como el método de *Monte Carlo*.

4.2.2. Caminos canónicos y el mixing time de un cadena de Markov

Es claro que el método de Monte Carlo será útil solo si la cadena de Markov invierte un número pequeño de pasos para acercarse tanto como se quiera a la distribución estacionaria (es decir si su mixing time es pequeño comparado con el tamaño del conjunto estados). Una técnica para analizar el mixing time de una cadena de Markov $\mathfrak{M}$ (ergódica, reversible, con conjunto de estados Ω, matriz de transicion P y distribución estacionaria π) consiste en ver a $\mathfrak{M}$ como un grafo no dirigido con conjunto de vértices Ω y conjunto de aristas $E = \{\{x, y\} \in \Omega^{(2)} : Q(x, y) > 0\}$, donde $\Omega^{(2)}$ es el conjunto de subconjuntos de tamaño dos de Ω y $Q(x, y) =: \pi(x)P(x, y) = \pi(y)P(y, x)$. Esto tiene sentido porque $\mathfrak{M}$ es reversible. Un *camino canónico* γ_{xy} del vértice x al vértice y en el grafo (Ω, E) corresponde a una secuencia de transiciones legales en $\mathfrak{M}$ del estado x al estado y. Sea $\Gamma = \{\gamma_{xy} : x, y \in \Omega\}$ un conjunto de

caminos canónicos uno por cada par ordenado $(x, y) \in \Omega^2$. La cantidad

$$\overline{\rho} = \overline{\rho}(\Gamma) = max_e \frac{1}{Q(e)} \sum_{e \in \gamma_{xy}} \pi(x)\pi(y)|\gamma_{xy}|$$

Donde el máximo es sobre las aristas orientadas e de (Ω, E), y $|\gamma_{xy}|$ denota la la longitud del camino γ_{xy}.

La cantidad $\overline{\rho}$ juega un papel importante para acotar el mixing time de $\mathfrak{M}$ según el siguiente teorema (ver una prueba en [9])

Teorema 4.2.3. *Sea $\mathfrak{M}$ una cadena de Markov ergódica, reversible, con matriz de transición P tal que $P(x, x) > 0$ para todo estado x y sea Γ un conjunto de caminos canónicos. Entonces $\mathfrak{M}$ satisface*

$$\tau_x(\epsilon) \leq \overline{\rho}(ln\pi(x))^{-1} + ln(\epsilon^{-1}))$$

Esta técnica de los caminos canónicos es la que utilizaremos para acotar el mixing time de las cadenas de Markov involucradas en los FPRAS definidos en las siguientes dos secciones.

4.3. Aproximando probabilísticamente $\#MATCH$

Sea G un grafo y $\lambda \geq 0$. La función de partición de G se define como:

$$Z_G(\lambda) = \sum_{k=0}^{\lfloor n/2 \rfloor} m_k \lambda^k$$

donde n es el numero de vertices de G y m_k es el numero de $k-$matchings en G. Es claro que $Z_G(1) = \#MATCH(G)$ y por lo tanto un FPRAS que aproxima la función Z_λ también aproxima $\#MATCH(G)$. Construiremos un FPRAS utilizando el método de Monte Carlo para aproximar probabilísticamente Z_G. El algoritmo que estudiamos en esta sección se encuentra en [4].

Sea G un grafo con n vértices, $\lambda > 0$ y $\mathfrak{M}(\lambda)$ la cadena de Markov con conjunto de estados el conjunto de matchings en G denotado por Ω. Definamos sobre Ω la distribución de probabilidad π_λ como:

$$\pi_\lambda(M) = \frac{\lambda^{|M|}}{Z(\lambda)}$$

para todo $M \in \Omega$.

Si $M \in \Omega$ entonces el mecanismo de transición de $\mathfrak{M}(\lambda)$ es el siguiente:

1. con probabilidad 1/2 sea $M' = M$; en otro caso

2. Selecciones una arista de $e = \{u, v\} \in E(G)$ uniformemente al azar y sea:

$$M' = \begin{cases} M - e, & \text{si } e \in M \text{ (transición tipo 1);} \\ M + e, & \text{si } u \text{ y } v \text{ no están en } M \text{ (transición tipo 2);} \\ M + e - e', & \text{si exactamente uno } u \text{ o } v \text{ está en } M \\ & \text{y } e' \text{ es la arista en } M \text{ (transición tipo 0);} \\ M, & \text{en otro caso;} \end{cases}$$

3. Vaya a M' con probabilidad $min\{1, \pi_\lambda(M')/\pi_\lambda(M)\}$.

Es claro que $\mathfrak{M}(\lambda)$ es ergódica. Es irreducible porque cualquier matching en Ω se puede alcanzar a partir del matching vacío con probabilidad mayor que cero. Es aperiódica porque según el mecanismo de transición, la probabilidad de que un matching cualquiera siga siendo el mismo después de un paso en la cadena es mayor que cero. Algo que no es tan obvio es que π_λ es la distribución estacionaria de $\mathfrak{M}(\lambda)$. Para ver esto es fácil probar que π_λ es reversible. Es decir tenemos el siguiente resultado.

Proposición 4.3.1. *La distribución de probabilidad π_λ es reversible en $\mathfrak{M}(\lambda)$.*

Observe ahora que si se quiere estimar el valor de $Z(\lambda)$ en el valor específico $\lambda = \widehat{\lambda}$ podemos expresar $Z(\widehat{\lambda})$ como el producto:

$$Z(\widehat{\lambda}) = \frac{Z(\lambda_r)}{Z(\lambda_{r-1})} \times \frac{Z(\lambda_{r-1})}{Z(\lambda_{r-2})} \times \cdots \times \frac{Z(\lambda_2)}{Z(\lambda_1)} \times \frac{Z(\lambda_1)}{Z(\lambda_0)} \times Z(\lambda_0).$$

donde $\lambda_0 = 0$, $\lambda_1 = |E(G)|^{-1}$ y $\lambda_i = (1 + 1/n)^{i-1}\lambda_1$ para $1 < i < r$, con r el menor entero tal que $(1 + 1/n)^{r-1}\lambda_1 \geq \widehat{\lambda}$ y $\lambda_r = \widehat{\lambda}$.[1] Además observe que:

1. $Z(0) = 1$,

2. $r \leq \lceil 2n(ln\widehat{\lambda}) + ln|E| \rceil + 1$, y

3. si $f_i = (M) = (\lambda_{i-1}/\lambda_i)^{|M|}$ es una variable aleatoria donde M es un matching en Ω escogido con la distribución de probabilidad π_{λ_i}, entonces el valor esperado de f_i es:

$$\rho_i = Ef_i = \sum_{M \in \Omega} \left(\frac{\lambda_{i-1}}{\lambda_i}\right)^{|M|} \frac{\lambda_i^{|M|}}{Z(\lambda_i)} = \frac{1}{Z(\lambda_i)} \sum_{M \in \Omega} \lambda_{i-1}^{|M|} = \frac{Z(\lambda_{i-1})}{Z(\lambda_i)}.$$

Por lo tanto ρ_i puede ser estimado escogiendo matchings con probabilidad π_{λ_i} y computando la media de f_i. Seguidamente $Z(\widehat{\lambda})$ se puede estimar calculando el producto de los recíprocos de los ρ_i's estimados. Todo esto lo resumimos en el

[1] La razón por la cual λ_i se define de esta manera se dará mas adelante.

siguiente algoritmo $\mathcal{A}$:

Algoritmo $\mathcal{A}$:

1) Haga $\lambda_0 = 0$, $\lambda_r = \widehat{\lambda}$, y compute λ_i para $1 < i < r$.

2) Para cada λ_i, donde $i = 1, 2, \cdots r$ compute un estimador X_i de la razón ρ_i de la siguiente manera:

   a) Haga S independientes simulaciones de la cadena de Markov $\mathfrak{M}(\lambda_i)$, cada uno de longitud T_i. Obteniendo así una muestra de tamaño S con distribución cercana a π_{λ_i}.

   b) Sea X_i la muestra media de la cantidad $(\lambda_{i-1}/\lambda_i)^{|M|}$.

3) Salida: $Y = \prod_{i=1}^{r} X_i^{-1}$.

Lo único que falta para que nuestro algoritmo $\mathcal{A}$ quede totalmente definido es especificar el tamaño S de la muestra y la longitud T_i, para cada $i = 1, 2, \cdots r$ de tal manera que $\mathcal{A}$ sea una $FPRAS$ para $Z(\lambda)$. Es decir que la cantidad $\sum_{i=1}^{r} ST_i$ este acotada superiormente por un polinomio en n. Antes de esto observe que f_i toma valores en el intervalo $[0, 1]$ y por lo tanto $Varf_i \leq Ef_i = \rho_i$ y así $Varf_i/Ef_i \leq \rho^{-1}$. Además por la definición de $Z(\lambda)$ y de π_i se tiene que:

$$\rho_i^{-1} = \frac{Z(\lambda_i)}{Z(\lambda_{i-1})} = \frac{\sum_k m_k \lambda_i^k}{\sum_k m_k \lambda_{i-1}^k} = \frac{\sum_k m_k (\lambda_i^n/\lambda_i^{n-k})}{\sum_k m_k (\lambda_{i-1}^n/\lambda_{i-1}^{n-k})} \leq \left(\frac{\lambda_i}{\lambda_{i-1}}\right)^n = \left(1 + \frac{1}{n}\right)^n \leq e,^2 \tag{4.1}$$

para $1 < i \leq r$. Como caso aparte

$\rho_1^{-1} = Z(\lambda_1) = Z(|E(G)|^{-1}) = \sum_k m_k (|E(G)|^{-1})^k \leq \sum_k \binom{|E|}{k} (|E(G)|^{-1})^k \leq \sum_k \frac{1}{k!} \leq e$, la primera desigualdad se tiene porque todo matching de tamaño k es un subconjunto de $E(G)$ de tamaño k. Por lo tanto para todo $1 \leq i \leq r$, $\rho_i^{-1} \leq e$. Esto quiere decir intuitivamente que S es pequeño y es la razón por la cual se definieron así los λ_i's. Más formalmente tenemos la siguiente

Proposición 4.3.2. *En el algoritmo $\mathcal{A}$, suponga que el tamaño de la muestra $S = \lceil 130 e \epsilon^{-2} r \rceil$ y que la longitud T_i es suficientemente larga para que la variación de la distancia de $\mathfrak{M}(\lambda_i)$ a su distribución estacionara π_i sea a lo sumo $\epsilon/5er$, para todo $i = 1, 2, \cdots, r$. Entonces la variable aleatoria Y satisface*

$$Pr((1-\epsilon)Z(\widehat{\lambda}) \leq Y \leq (1+\epsilon)Z(\widehat{\lambda})) \geq \frac{3}{4}.$$

Demostración.
Sea $\widehat{\pi}_{\lambda_i}$ la distribución obtenida después de una simulación finita de la cadena de Markov $\mathfrak{M}(\lambda_i)$, de tal manera que:

$$\|\widehat{\pi}_i - \pi_i\| \leq \frac{\epsilon}{5er}, \tag{4.2}$$

[2] e es la constante Euleriana

para $i \in \{1, 2, \cdots, r\}$. Sea también $\widehat{f_i}$, definida análogamente que f_i excepto que el matching M es seleccionado con la distribución $\widehat{\pi}_{\lambda_i}$ en vez de π_{λ_i}. Como $\widehat{f_i}$ toma valores en $(0, 1]$, su valor esperado $E\widehat{f_i} =: \widehat{\rho_i}$ satisface que

$$|\widehat{\rho_i} - \rho_i| = \left| \sum_{M \in \Omega} \left(\frac{\lambda_{i-1}}{\lambda_i} \right)^{|M|} (\widehat{\pi}_{\lambda_i}(M) - \pi_{\lambda_i}(M)) \right| \leq \left| \sum_{M \in \Omega} \widehat{\pi}_{\lambda_i}(M) - \pi_{\lambda_i}(M) \right|$$

$$\leq \|\widehat{\pi}_i - \pi_i\| \leq \frac{\epsilon}{5er}.$$

Lo cual implica por 4.2 que

$$\left(1 - \frac{\epsilon}{5r}\right) \rho_i \leq \widehat{\rho_i} \leq \left(1 + \frac{\epsilon}{5r}\right) \rho_i \tag{4.3}$$

Además dado que $\widehat{f_i}$ toma valores en $(0, 1]$, por 4.1 se tiene que

$$\frac{Var\widehat{f_i}}{E\widehat{f_i}} \leq \widehat{\rho_i}^{-1} \leq 2\rho_i^{-1} \leq 2e \tag{4.4}$$

donde la segunda desigualdad se cumple gracias a 4.3.

Ahora calcularemos el tamaño de una muestra suficientemente grande para asegurar una aproximación de $Z(\widehat{\lambda})$. Sean $X_i^{(1)}, X_i^{(2)}, \cdots, X_i^{(S)}$, S independientes copias de la variable aleatoria $\widehat{f_i}$ obtenidas seleccionando S matchings con la distribución de probabilidad $\widehat{\pi}_i$, y sean $\overline{X_i} = \left(\sum_{j=1}^{S} X_i^{(j)}\right)/S$ la media de la muestra. Es claro que $E\overline{X_i} = E\widehat{f_i} = \widehat{\rho_i}$ y que $Var\overline{X_i} = (\widehat{Varf_i})/S$. Nuestro estimador de $\rho = Z(\widehat{\lambda})^{-1}$ es la variable aleatoria $X = \prod_{i=1}^{r} \overline{X_i}$. No es difícil ver utilizando 4.3 que $\widehat{\rho} =: EX = \prod_{i=1}^{r} \widehat{\rho_i}$ satisface:

$$\left(1 - \frac{\epsilon}{4}\right) \rho \leq \widehat{\rho} \leq \left(1 + \frac{\epsilon}{4}\right) \rho \tag{4.5}$$

y aplicando 4.4 que

$$\frac{VarX}{(EX)^2} = \prod_{i=1}^{r} \left(1 + \frac{Var\overline{X_i}}{(E\overline{X_i})^2}\right) - 1 \leq \left(1 + \frac{2e}{S}\right)^r - 1 \leq e^{(2er/S)} - 1 \leq \epsilon^2/64 \tag{4.6}$$

Lo anterior, usando el hecho de que $e^{(x/65)} \leq 1 + (x/64)$, si $0 \leq x \leq 1$ y haciendo $S = \lceil 130e\epsilon^{-2}r \rceil$. Además utilizando 4.6 y aplicando la desigualdad de Chebyshev a X obtenemos que:

$$Pr\left(|X - \widehat{\rho}| > \left(\frac{\epsilon}{4}\right)\widehat{\rho}\right) \leq \frac{16}{\epsilon^2} \frac{VarX}{(EX)^2} \leq \frac{1}{4}$$

y por lo tanto que:

$$Pr\left(|X - \widehat{\rho}| \leq \left(\frac{\epsilon}{4}\right)\widehat{\rho}\right) \geq \frac{3}{4},$$

es decir con probabilidad mayor a 3/4

$$\left(1-\frac{\epsilon}{4}\right)\widehat{\rho} \leq X \leq \left(1+\frac{\epsilon}{4}\right)\widehat{\rho}, \tag{4.7}$$

aplicando la desigualdad 4.5 obtenemos con probabilidad mayor a 3/4 que

$$(1-\epsilon)\,\rho \leq X \leq (1+\epsilon)\,\rho, \tag{4.8}$$

y por lo tanto con probabilidad mayor que 3/4

$$(1-\epsilon)\,Z(\widehat{\lambda}) \leq Y \leq (1+\epsilon)\,Z(\widehat{\lambda}).$$

☑

La siguiente proposición nos permitirá calcular los T_i's.

Proposición 4.3.3. *La cadena de Markov $\mathfrak{M}(\lambda)$ satisface que*

$$\tau_X(\epsilon) \leq 4|E|n(\lambda')^2(n(\ln n + \ln \lambda') + \ln \epsilon^{-1}),$$

donde $\lambda = \text{máx}\{1, \lambda\}$.

Demostración.
Observe que si primero $\ln \pi_\lambda(x)^{-1} \leq n(\ln n + \ln \lambda')$ y segundo $\overline{\rho} \leq 4|E|(\lambda')^2$ para algún conjunto de caminos canónicos Γ adecuado, entonces usando el teorema 4.2.3 la proposición estaría probada. Para ver lo primero, es claro que $\sum_k M_k \leq n!$ (dado que todo matching en G se puede ver como una función biyectiva de $V(G)$ en $V(G)$) y que $\ln n! \leq \ln n^n \leq n \ln n$ y por lo tanto

$$\pi_\lambda(X) = \frac{\lambda^{|X|}}{Z(\lambda)} = \frac{\lambda^{|X|}}{\sum_k M_k \lambda^k} \geq \frac{\lambda^{|X|}}{\sum_k M_k \lambda^n} = \frac{1}{\sum_k M_k \lambda^{n-|X|}} = \frac{1}{\lambda^{n-|X|}\sum_k M_k}$$
$$\geq \frac{1}{\lambda^{n-|X|} n!} \geq \frac{1}{(\lambda')^{n-|X|} n!} \geq \frac{1}{n!(\lambda')^n}$$

es decir,
$\pi_\lambda(X)^{-1} \leq n!(\lambda')^n$ y $\ln \pi_\lambda(X)^{-1} \leq \ln n!(\lambda')^n \leq \ln(n\lambda')^n = n(\ln n + \ln \lambda')$
Para probar lo segundo debemos encontrar un conjunto de caminos canónicos Γ en $\mathfrak{M}(\lambda)$ de tal manera que $\overline{\rho} \leq 4|E|(\lambda')^2$. Sean X y Y matchings en G, definimos el camino canónico γ_{XY} así:
Consideremos la diferencia simétrica $X \bigoplus Y$ que tal como se discutió forma un conjunto de caminos simples y/o ciclos simples de longitud par cada uno de los cuales tiene aristas que se alternan en X y en Y. Designemos un orden sobre todos los caminos simples y los ciclos simples de longitud par en G y escojamos un vértice en cada uno de ellos como el vértice inicial, que para los caminos será uno de los dos vértices finales y para los ciclos puede ser cualquier vértice. Este orden inducirá a su vez un orden sobre los caminos y ciclos en $X \bigoplus Y$, digamos $P_1, P_2, \cdots P_m$. El camino canónico γ_{XY} de X a Y involucra *detrás del telón* cada uno de los $P_i's$ ordenadamente de la siguiente manera. Consideremos dos casos:

1. P_i es un camino simple: Sea P_i la secuencia de vértices $(v_0, v_1, \cdots, v_l)$, con v_0 el vértice inicial. Si $(v_0, v_1) \in Y$, realice una secuencia de transiciones de tipo cero reemplazando (v_{2j+1}, v_{2j+2}) por (v_{2j}, v_{2j+1}) para $j = 0, 1, \cdots$, y finalice con una transición de tipo dos si l es impar. Si $(v_0, v_1) \in X$, empiece con una transición de tipo uno eliminando (v_0, v_1) y después se procede como antes en el camino reducido $(v_1, v_2, \cdots, v_l)$.

2. P_i es un ciclo simple: Sea P_i la secuencia de vértices $(v_0, v_1, \cdots, v_{2l+1})$ donde $l \geq 1$, v_0 es el vértice inicial, $(v_{2j}, v_{2j+1}) \in X$ para $0 \leq j \leq l$, y el resto de aristas pertenecen a Y. Entonces empezamos con una transición de tipo uno para remover (v_0, v_1). Quedamos con un camino simple O con vértices finales v_0 y v_1, uno de los cuales será el vértice inicial de O. Finalmente procedemos como en el primer caso con O.

Gracias a los items 1) y 2) podemos utilizar los P_i,s como guías para pasar de X a Y por medio de un camino γ_{XY} en el grafo $\mathfrak{M}(\lambda)$. Cuando estemos usando la guía P_i, esto quiere decir que $P_0, P_1, \cdots, P_{i-1}$ ya han sido procesadas y que $P_{i+1}, P_{i+2}, \cdots, P_m$ aún no. Llamaremos a el conjunto de caminos asi creados Γ.
Ahora calculemos $\rho(\Gamma)$, para hacer esto sea t una arista en $\mathfrak{M}(\lambda)$ es decir una transición de M a M' (con M y M' matchings diferentes en G) y sea $cp(t) = \{(X, Y) : t \in \gamma_{XY}\}$ el conjunto de caminos canónicos en Γ que usan la arista t. Observe que si la arista t ocurre en la guía P_i, entonces $X \bigoplus Y \bigoplus (M \cup M')$ coincide con X en las posiciones $1, 2, \cdots, i-1$ y con Y en $i+1, i+2, \cdots, m$. La única forma para que $X \bigoplus Y \bigoplus (M \cup M')$ no sea un matching en G es que P_i sea un ciclo y la transición de M a M' sea del tipo cero, entonces las dos aristas una de X y la otra de Y pertenecerán a $X \bigoplus Y \bigoplus (M \cup M')$. Sin embargo si e_{XYt} es la arista de X que tiene el vértice inicial de P_i entonces $X \bigoplus Y \bigoplus (M \cup M') \setminus e_{XYt}$ es un matching en G. Sea

$$\eta_t(X, Y) = \begin{cases} X \bigoplus Y \bigoplus (M \cup M') \setminus e_{XYt}, & \text{si } t \text{ es de tipo cero y el } P_i \\ & \text{actual es un ciclo;} \\ X \bigoplus Y \bigoplus (M \cup M'), & \text{en otro caso.} \end{cases}$$

Observe que $\eta_t(X, Y)$ es inyectiva porque

$$X \bigoplus Y = \begin{cases} \eta_t(X, Y) \bigoplus (M \cup M') + e_{XYt}, & \text{si } t \text{ es de tipo cero y el } P_i \\ & \text{actual es un ciclo;} \\ \eta_t(X, Y) \bigoplus (M \cup M'), & \text{en otro caso.} \end{cases}$$

Y dado $X \bigoplus Y$, tenemos la secuencia $P_1, P_2, \cdots, P_m$ de caminos simples y/o ciclos simples, que utilizaremos como guía para ir de X a Y, y la transición nos dice cual de estos P_i es la guía actual. Observe que X coincide con $\eta_t(X, Y)$ en las guías $P_1, P_2, \cdots, P_{i-1}$ y con M en $P_{i+1}, P_{i+2}, \cdots, P_m$. Para las aristas en X y en Y es claro que $X \cap Y = M \setminus (X \bigoplus Y)$, por lo tanto X y Y puede ser recuperado de manera única a partir de $\eta(X, Y)$, y por lo tanto es inyectiva. Finalmente note que si

$$\pi_\lambda(X)\pi_\lambda(Y) \leq 2|E|(\lambda')^2 Q(t)\pi_\lambda(\eta_t(X, Y)), \tag{4.9}$$

entonces,

$$\overline{\rho} = \overline{\rho}(\Gamma) \leq \frac{1}{Q(t)} \sum_{\gamma_{XY} \ni t} \pi_\lambda(X)\pi_\lambda(Y)|\gamma_{XY}| \leq 2|E|(\lambda')^2 \sum_{\gamma_{XY} \ni t} \pi_\lambda(\eta(X,Y))|\gamma_{XY}|$$

$$\leq 2|E|n(\lambda')^2 \sum_{\gamma_{XY} \ni t} \pi_\lambda(\eta_t(X,Y)) \leq 2|E|n(\lambda')^2,$$

y con esto terminamos la demostración de la proposición. Donde la tercera desigualdad se tiene porque $|\gamma_{XY}| \leq n$ y la cuarta porque $\sum_{\gamma_{XY} \ni t} \pi_\lambda(\eta_t(X,Y)) \leq 1$, dado que η_t es inyectiva y π_λ es una distribución de probabilidad.
Para verificar 4.9 distinguiremos cuatro posibles casos (antes de esto note que $Q(t) = (2|E|)^{-1} \min\{\pi_\lambda(M), \pi_\lambda(M')\}$)

1. t **es una transición tipo 1**: $M' = M - e$, entonces $\eta_t(X,Y) = X \bigoplus Y \bigoplus M$, y asi $M \cup \eta_t(X,Y) = X \cup Y$ incluso vistos como multiconjuntos, por lo tanto

$$\begin{aligned}
\pi_\lambda(X)\pi_\lambda(Y) &= \pi_\lambda(M)\pi_\lambda(\eta_t(X,Y)) \\
&= \frac{2|E|Q(t)}{\min\{\pi_\lambda(M), \pi_\lambda(M')\}} \times \pi_\lambda(M)\pi_\lambda(\eta_t(X,Y)) \\
&= 2|E|Q(t) \max\left\{1, \frac{\pi_\lambda(M)}{\pi_\lambda(M')}\right\} \pi_\lambda(\eta_t(X,Y)) \\
&= 2|E|Q(t)\lambda'\pi_\lambda(\eta_t(X,Y)) \leq 2|E|Q(t)(\lambda')^2\pi_\lambda(\eta_t(X,Y)).
\end{aligned}$$

2. t **es una transición tipo 2**: Este caso es similar al caso 1) solo debemos intercambiar M y M', dado que $M' = M + e$.

3. t **es una transición tipo 0 y el camino actual es un ciclo**: Suponga que $M' = M + e - e'$, y considere el multiconjunto $M \cup \eta_t(X,Y)$. Entonces $\eta_t(X,Y) = X \bigoplus Y \bigoplus (M + e) - e_{XYt}$, así el multiconjunto difiere de $X \cup Y$ solo en el caso que e y e_{XYt} sean disyuntos con él. Es decir

$$\begin{aligned}
\pi_\lambda(X)\pi_\lambda(Y) &\leq (\lambda')^2\pi_\lambda(M)\pi_\lambda(\eta_t(X,Y)) \\
&= 2|E|(\lambda')^2Q(t)\pi_\lambda(\eta_t(X,Y)),
\end{aligned}$$

   como en este caso $\pi_\lambda(M) = \pi_\lambda(M')$, y así $Q(t) = (2|E|)^{-1}\pi_\lambda(M)$. Por lo tanto 4.9 se vuelve a cumplir.

4. t **es una transición tipo 0 y el camino actual no es un ciclo**: Este caso es idéntico a el caso 3), excepto que la arista e_{XYt} no aparece en el análisis, por lo tanto 4.9 se cumple nuevamente.

☑

Teorema 4.3.1. *El algoritmo $\mathcal{A}$ es un FPRAS para la función de partición de un sistema monómero-dímero arbitrario.*

Demostración.
Debido a la proposición 4.3.2 nos falta probar que el tiempo de computo del algoritmo $\mathcal{A}$ crece polinomialmente en n, $\widehat{\lambda'}$ y ϵ^{-1}. Sabemos también que $r \leq \lceil 2n(ln\widehat{\lambda}) + ln|E|\rceil + 1$, que $S = \lceil 130e\epsilon^{-2}r\rceil$ y de acuerdo a las proposiciones 4.3.2 y 4.3.3 es necesario tomar $T_r = \lceil 2|E|n(\lambda_i')^2(n(\ln n + \ln\lambda_i') + \ln(5er/\epsilon))\rceil$ para que la variación de distancia de la cadena de Markov $\mathfrak{M}(\lambda_i)$ sea a lo sumo $\varepsilon/5er$. El tiempo total del FPRAS $\mathcal{A}$ sería $T = \sum_{i=0}^{r} ST_i \leq rST_r$ dado que $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_r$. Por lo tanto $r = O(n\ln n\widehat{\lambda})$, $S = O(n\epsilon^{-2}\ln n\widehat{\lambda})$ y sin perdida de generalidad podríamos asumir que $\ln(\epsilon^{-1}) = O(n\ln n)$, esta suposición es justificada porque el problema puede siempre ser resuelto exactamente por fuerza bruta en tiempo $O(n^{2n})$, lo cual es $O(\epsilon^{-2})$ si $\ln(\epsilon^{-1})$ excede la cota supuesta arriba. Bajo ésta suposición $T_r = O(|E|n^2\widehat{\lambda_i'}\ln(n\widehat{\lambda_i'}))$ y por lo tanto el algoritmo $\mathcal{A}$ tiene tiempo de cómputo $T = O(n^4|E|(\ln n\widehat{\lambda'})^3\epsilon^{-2})$, el cual crece solo polinomialmente en n, $\widehat{\lambda'}$ y ϵ^{-1}. ☑

Corolario 4.3.1. *Existe un FPRAS que aproxima el número de matchings en un grafo arbitrario.*

4.4. Aproximando probabilísticamente $\#SAW$

En esta sección construiremos un FPRAS que nos permitirá aproximar probabilisticamente el numero de caminos de una longitud dada que se autoevitan (SAWs) en el retículo de dimensión d (es decir en $\mathbb{Z}^d$) y que empiezan en el origen de coordenadas. Antes de esto definamos la cantidad α_n que jugará un papel importante en nuestro algoritmo. Para una dimensión fija d, se define

$$\alpha_n = \min_{i,j;\, j+k=n} \frac{c_{j+k}}{c_j c_k}, \tag{4.10}$$

para j y k fijos, $\frac{c_{j+k}}{c_j c_k}$ se puede interpretar como la probabilidad de escoger al azar un SAW w_1 de longitud j y uno w_2 de longitud k de tal manera que al trasladar w_2 poniendo el vértice inicial de este en el lugar del vértice final de w_1 el nuevo camino denotado $w_1 \circ w_2$ es un SAW de longitud $j + k$.

Sea S_i el conjunto de SAWs de longitud i y $c_i = |S_i|$. $\mathfrak{M}_n$ denotará la cadena de Markov con conjunto de estados $X_n = \bigcup_{i=0}^{n} S_i$, es decir X_n es el conjunto de SAWs con longitud a lo más n.
Las transiciones en $\mathfrak{M}_n$ se definen así: En el estado $w \in X_n$, con longitud $|w| = i \leq n$ se escoge uniformemente al azar una de las $2d$ aristas incidentes en el vértice final de w. Si la última arista coincide con la última arista de w, dicha arista se elimina o borra de w. Si la arista escogida extiende a w con una longitud a lo mas n, entonces la arista se adiciona con probabilidad β_{i+1}[3]. En cualquier otro caso deje a w como

[3]Es un parámetro que se discutirá mas adelante.

estaba.

Definamos ahora el orden parcial $\prec$ en el conjunto de todos los SAWs así: $w \prec w'$ si y solo si $|w| < |w'|$ y los primero $|w|$ pasos de w' coinciden con w. Además escribiremos $w \prec_1 w'$ si y solo si $w \prec w'$ y $|w'| = |w| + 1$, es decir, si w' se obtiene extendiendo w en un paso. La probabilidad de transición P_n de $\mathfrak{M}_n$ se define como:

$$P_n(w, w') = \begin{cases} \beta_{|w'|}/2d, & \text{si } w \prec_1 w'; \\ 1/2d, & \text{si } w' \prec_1 w; \\ r(w), & \text{si } w = w'; \\ 0, & \text{en otro caso.} \end{cases}$$

donde $r(w)$ se escoge de tal manera que las probabilidades sumen 1, y $w, w' \in X_n$.
Observe que $\mathfrak{M}_n$ se puede ver como un árbol definido por el orden parcial $\prec$. Este árbol tiene como raíz el camino trivial de longitud cero, y los hijos de un camino w son los caminos w' tales que $w \prec_1 w'$. Es decir este árbol tiene $n + 1$ niveles $0, 1, \cdots, n$; en el nivel i están todos los caminos de longitud i. La probabilidad de ir de cualquier estado a su padre es $1/2d$ y la probabilidad de ir de cualquier estado en el nivel i a cualquiera de sus hijos es $\beta_{i+1}/2d$.
Por conveniencia modificaremos un poco $\mathfrak{M}_n$ de la siguiente manera: con probabilidad $1/2$ haremos la transición de un estado a otro utilizando P_n como se definió antes y con probabilidad $1/2$ el estado actual permanece igual. Esto lo hacemos para asegurar la aperiodicidad de la cadena. Además es claro que la cadena es irreducible porque todos los estados se comunican entre sí por ejemplo a través de el SAW trivial. Por lo tanto $\mathfrak{M}_n$ es ergódica. Esto implica por el teorema 4.2.2 que la cadena de Markov $\mathfrak{M}_n$ converge a una única distribución estacionaria sobre X_n

Proposición 4.4.1. *La distribución estacionaria π_n de la cadena de Markov $\mathfrak{M}_n$ está dada por*

$$\pi_n(w) = \frac{1}{Z_n} \prod_{i=1}^{|w|} \beta_i,$$

para $w \in X_n$, donde Z_n es un factor normalizador.

Demostración.
Es suficiente con mostrar que π_n es reversible. Es decir que se cumpla la siguiente condición

$$\pi_n(w)P_n(w, w') = \pi_n(w')P_n(w', w); \ \forall w, w' \in X_n.$$

Basta con considerar los siguientes tres casos: si $|w| - |w'| > 1$, entonces $P_n(w, w') = P_n(w', w) = 0$. Si $|w| - |w'| = 0$ es posible que $w \neq w'$ o que $w = w'$; en el primero de los casos $P_n(w, w') = P_n(w', w) = 0$ y en el segundo $P_n(w, w') = P_n(w', w) = r(w)$.

Ahora si $|w| - |w'| = 1$, entonces

$$\pi_n(w)P_n(w,w') = \left(\frac{1}{Z_n}\prod_{i=1}^{|w|}\beta_i\right)\cdot\frac{\beta_{w'}}{2d} = \left(\frac{1}{Z_n}\prod_{i=1}^{|w'|}\beta_i\right)\cdot\frac{1}{2d} = \pi_n(w')P_n(w',w).$$

Es decir la condición se cumple en cualquiera de los posibles casos y por lo tanto π_n es la distribución estacionaria de la cadena de Markov. ☑

Si hacemos $\beta_i = c_{i-1}/c_i$ tendremos por la anterior definición que

$$\pi_n(w) = \frac{1}{Z_n}\prod_{i=1}^{|w|} c_{i-1}/c_i = \frac{1}{Z_n c_{|w|}} = \frac{1}{(n+1)c_{|w|}}$$

Es decir la probabilidad de que w sea un camino de longitud i es $\frac{1}{n+1}$, para cada $i \leq n$.
Aunque los valores β_i (tal y como los acabamos de definir) son desconocidos para nosotros, más adelante mostraremos una forma de aproximarlos que de paso nos servirá para aproximar los c_i's que es nuestro objetivo principal. Pero cuántos pasos tiene que dar $\mathfrak{M}_n$ para que su distribución de probabilidad este tan cercana a π_n como queramos? La respuesta se resume en el siguiente teorema.

Teorema 4.4.1. *Para la cadena de Markov $\mathfrak{M}_n$, empezando en el camino trivial $\mathcal{O}$, tenemos que*

$$\tau_{\mathcal{O}} \leq Kdn^2\alpha_n^{-1}(\ln n + \ln \epsilon^{-1}),$$

para alguna constante K.

Demostración.
Sabemos que $\pi_n(\mathcal{O}) = 1/n+1$. También como el grafo correspondiente a la cadena de Markov $\mathfrak{M}_n$ es un árbol, esto implica que hay un único camino simple γ_{XY} entre cualquier par de vértices X, Y. Denotaremos esta colección de caminos como $\Gamma = \{\gamma_{XY} : X, Y \text{ son estados de } \mathfrak{M}_n\}$. Como la altura del árbol es n, entonces la longitud del máximo camino estará acotada superiormente por $2n$. Observe además que el teorema estará probado si logramos probar que $\rho(\Gamma) \leq K'dn\alpha_n^{-1}$, para alguna constante K', aplicando el teorema 4.2.3. Note además que si e es cualquier arista del árbol, suponiendo que los vértices de e son un camino w de longitud k y un camino w' de longitud $k+1$. Si S es el subárbol con raíz w', como e es una arista de corte, es claro que

$$\sum_{\gamma_{XY}\ni e}\pi_n(X)\pi_n(Y) = \sum_{\widetilde{w}\succeq w',\,\widehat{w}\preceq w}\pi_n(\widetilde{w})\pi_n(\widehat{w}) = \sum_{\widetilde{w}\succeq w'}\pi_n(\widetilde{w})\sum_{\widehat{w}\preceq w}\pi_n(\widehat{w}) = \pi_n(S)\pi_n(\overline{S}),$$

donde $\overline{S} = X_n \setminus S$. Por lo tanto $\rho(\Gamma) = \max_e Q(e)^{-1}\pi_n(S)\pi_n(\overline{S})$.
Un hecho importante que utilizaremos en esta prueba, es que el árbol determinado por $\mathfrak{M}_n$ tiene la llamada propiedad sub-Cayley, es decir el número de vértices en el

nivel l de cualquier subárbol está acotado superiormente por el número del vértices en el nivel l del árbol completo. Esto es así en nuestro caso dado que cualquier segmento final de un SAW también es un SAW. Tenemos que

$$Q(e) = \pi_n(w')P_n(w', w) = \frac{1}{4dZ_n c_{k+1}},$$

y que

$$\pi_n(S) = \sum_{\widetilde{w} \succeq w'} \pi_n(\widetilde{w}) = \sum_{j=k+1}^{n} \frac{1}{Z_n c_j} |\{\widetilde{w} \succeq w' : |\widetilde{w}| = j\}| =$$

$$\frac{1}{Z_n c_{k+1}} \sum_{j=k+1}^{n} \frac{c_{k+1}}{c_j} |\{\widetilde{w} \succeq w' : |\widetilde{w}| = j\}| \leq \frac{1}{Z_n c_{k+1}} \sum_{j=k+1}^{n} \frac{c_{k+1} c_{j-(k+1)}}{c_j} \leq \frac{1}{Z_n c_{k+1} \alpha_n},$$

donde la primera desigualdad se sigue de la propiedad sub-Cayley del árbol (porque todo camino de longitud j que pasa por w' está en el nivel $c_{j-(k+1)}$ del subárbol cuya raíz es w', dado que $|w'| = k+1$) y la segunda de la definición de α. Además como $\pi_n(\overline{S}) \leq 1$, entonces $Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}) \leq Q(e)^{-1}\pi_n(S) \leq 4dn\alpha_n^{-1}$. Y como e es cualquier arista entonces $\rho(\Gamma) \leq 4dn\alpha_n^{-1}$. ☑

Para construir nuestro FPRAS debemos ensamblar la secuencia de cadenas de Markov $\mathfrak{M}_1, \mathfrak{M}_2 \cdots, \mathfrak{M}_n$ en un algoritmo que produce una secuencia de números $\widetilde{c}_1, \widetilde{c}_2, \cdots \widetilde{c}_n$, cada uno de los cuales es una buena estimación de $c_1, c_2, \cdots, c_n$ respectivamente. La precisión de estas estimaciones estará controlada por los parametros, ϵ y δ, exactamente como en la definición de FPRAS. Los principales ingredientes de nuestro algoritmo son dos procedimientos para estimar la cantidad α_n, la cual aparece en el mixing time de la cadena $\mathfrak{M}_n$ y los parámetros β_n que gobiernan las probabilidades de transición de las cadenas de Markov. Recordemos que hemos asumido hasta el momento que conocemos α_n y $\beta_n = c_{n-1}/c_n$ para todo n. Sin embargo estas cantidades no están disponibles para nosotros; de hecho, calcular las cantidades c_n es nuestro objetivo. En vez de esto nuestro algoritmo calculará buenos estimados de las cantidades α_n y c_{n-1}/c_n para cada n, usando la anterior cadena de Markov $\mathfrak{M}_{n-1}$. En el teorema 4.4.1 determinamos el mixing time de $\mathfrak{M}_n$ como una función de n, ϵ, δ y de la cantidad desconocida α_n. Es decir necesitamos saber una cota superior razonable para determinar el número de pasos en la simulación de $\mathfrak{M}_n$. Lo que haremos será calcular una estimación $\widetilde{\alpha}$ de α_n, de tal manera que con muy alta probabilidad $\alpha_n/4 \leq \widetilde{\alpha}_n \leq \alpha_n$. Y por lo tanto por el teorema 4.4.1, será suficiente simular $\mathfrak{M}_n$ en $Kdn^2\widetilde{\alpha}_n^{-1}(\ln n + \ln \epsilon^{-1})$ pasos para generar SAW's con una probabilidad muy cercana a su distribución estacionaria π_n. Además como $\widetilde{\alpha}_{n-1}^{-1} \leq 4\alpha_{n-1}^{-1}$, el tiempo de simulación sigue siendo lineal en α_n^{-1}.
También recordemos que $\alpha_n = \text{mín}_{j+k \leq n} \frac{c_{j+k}}{c_j c_k}$, y que para j y k fijos esta razón es la probabilidad de que dos SAW's de longitudes j y k puedan ser concatenados para formar un nuevo SAW de longitud $j+k$. Asumiendo que ya sabemos una cota superior para el mixing time de la cadena de Markov $\mathfrak{M}_{n-1}$, podemos usar esta para generar caminos de longitud $i \leq n-1$ (casi) uniformemente. Generando

estos caminos pequeños podemos calcular una estimación de α_n, de esta forma determinando el número de pasos que se debe simular la siguiente cadena de Markov $\mathfrak{M}_n$ para asegurar que su distribución de probabilidad este cercana a su distribución estacionaria. Antes de describir el procedimiento para calcular el estimador $\widetilde{\alpha}_n$, enunciaremos un teorema y una conjetura altamente creíble (dado que mucha información experimental la respalda) de gran ayuda para dicha tarea. La demostración del teorema se puede ver por ejemplo en [10] o en [11].

Teorema 4.4.2 (Teorema generalizado del estimador zero-uno)**.** *Sean $Z_1, Z_2, \cdots, Z_N$, N variables aleatorias independientes e identicamente distribuidas de acuerdo a Z, variable aleatoria en $[0,1]$.*
Si $N = 4\lambda \ln(2/\delta) \cdot \rho_Z/(\epsilon\mu_Z)^2$, entonces

$$Pr\left[(1+\epsilon)^{-1}\mu_Z \leq \sum_{i=1}^{N} Z_i/N \leq (1+\epsilon)\mu_Z\right] > 1-\delta,$$

donde $\lambda = e-2 \approx 0,72$ y $\rho_Z = \max\{Var[Z], \epsilon\mu_Z\}$.

Conjetura 4.4.1. *Para una dimensión d dada, existe un polinomio fijo g, tal que $c_j c_k \leq g(i+k)$, para todo i, j. En particular se tiene que $\alpha_n^{-1} \leq g(n)$, para todo n.*

El siguiente algoritmo $\mathcal{A}_1$ nos permite calcular con alta probabilidad la estimación $\widetilde{\alpha}_n$ de tal manera que $\alpha_n/4 \leq \widetilde{\alpha}_n \leq \alpha_n$, para todo n.

Algoritmo $\mathcal{A}_1(\alpha_{n-1}, \beta_1, \cdots, \beta_{n-1})$
$\widetilde{\alpha}_n = \widetilde{\alpha}_{n-1}$,
Para $i = 1, 2, \cdots, \lfloor n/2 \rfloor$ **haga**:
use $\mathfrak{M}_{n-1}$, genere t_n caminos $u_j \in S_i$ y t_n caminos $v_j \in S_{n-i}$
Para $j = 1, 2, \cdots, t_n$ **haga**

$$X_j = \begin{cases} 1, & \text{si } u_j \circ v_j \in S_n; \\ 0, & \text{en otro caso.} \end{cases}$$

$q_{n,i} = \sum_j X_j/t_n$
$\widetilde{\alpha}_n = \min\{\widetilde{\alpha}_n, q_{n,i}/2\}$
return $\widetilde{\alpha}_n$.

El algoritmo descrito arriba, funciona de la siguiente manera: fijamos una dimensión d. Asumimos primero que podemos generar caminos de cualquier longitud dada $i < n$ (casi) uniformemente al azar en tiempo polinomial en n, α_{n-1} y $\ln \epsilon^{-1}$, usando la cadena de Markov $\mathfrak{M}_{n-1}$. Esto se sigue del teorema 4.4.1 y del hecho que la distribución estacionaria π_{n-1} de $\mathfrak{M}_{n-1}$ asigna igual probabilidad a caminos de igual longitud $i < n$ y es (casi) uniforme sobre longitudes. Es decir para obtener t caminos independientes de longitud i, es suficiente generar $2nt$ caminos independientes a partir de π_{n-1}; con alta probabilidad, al menos t de estos caminos tendrán longitud i.

Para que el algoritmo este bien definido, asumiremos que nuestro algoritmo aborta si en algún punto el falla en recolectar la cantidad suficiente de caminos adecuados. La muy pequeña probabilidad de este evento puede fácilmente ser absorbida en el parámetro δ. Asumiremos también que hemos calculado previamente $\widetilde{\alpha}_{n-1}$, de tal manera que $\alpha_n/4 \leq \widetilde{\alpha}_n \leq \alpha_n$ con alta probabilidad. El valor $\widetilde{\alpha}_{n-1}$ es un estimado de la probabilidad de que la concatenación de dos SAW's sea otro SAW de longitud estrictamente menor que n, de esta manera necesitaremos solo estimar esta probabilidad para caminos cuya longitud total sea n. Para cada $0 < i < n$, generamos t_n pares independientes de caminos de longitud i y $n-i$; sea $q_{n,i}$ la proporción de estos pares cuya concatenación también es un SAW. Es claro que $q_{n,i}$ podría ser un estimador de la razón $\frac{c_i c_{n-i}}{c_n}$. Ahora el teorema 4.4.2 nos dice que $t_n = O(\alpha_n^{-1} \ln(n/\delta))$ caminos son suficientes para que $q_{n,i}$ sea tal que

$$Pr\left[\frac{1}{2}\alpha_n \leq q_{n,i} \leq 2\alpha_n\right] > 1 - \frac{\delta}{n^2},$$

haciendo $\epsilon = 1$. Si $\widetilde{\alpha}_n = \text{mín}\{\widetilde{\alpha}_{n-1}, \text{mín}_i\, q_{n,i}/2\}$, obtenemos una estimación en el rango $[\alpha_n/4, \alpha_n]$ con probabilidad al menos $1-\delta$.
El tiempo de ejecución de $\mathcal{A}_1$ esta dominado por el tiempo requerido para producir t_n SAW´s cada uno de longitud $i < n$, usando la cadena de Markov $\mathfrak{M}_{n-1}$. El número de caminos será con alta probabilidad a lo sumo $O(t_n n^2)$, $O(n^2 \alpha_n^{-1}(\ln n + \ln \delta^{-1}))$. El tiempo requerido por camino es el mixing time de $\mathfrak{M}_{n-1}$, se sabe que es $O(n^2 \alpha_{n-1}^{-1}(\ln n + \ln \epsilon^{-1}))$, para cualquier dimensión d. Por lo tanto el tiempo de ejecución del algoritmo $\mathcal{A}_1$ es $O(n^4 \alpha_n^{-2}(\ln n + \ln \delta^{-1})(\ln n + \epsilon^{-1}))$.

Ahora es el turno para la cantidad β_n. Recordemos que el valor ideal para el parámetro β_n es la razón c_{n-1}/c_n. Como α_n, ésta cantidad puede ser calculada por un algoritmo $\mathcal{A}_2$ basado en la cadena de Markov $\mathfrak{M}_{n-1}$. Note que esta razón es precisamente lo que necesitamos para calcular c_n, si de antemano conocemos el valor de c_{n-1}. Por lo tanto una estimación para β_n inmediatamente nos proporciona una estimación para c_n también. El Algoritmo $\mathcal{A}_2$ se describe como sigue:

Algoritmo $\mathcal{A}_2$:
Haga $\beta_1 = 1/2d$, $\widetilde{c}_1 = 2d$ y $\alpha_1 = 1$,
Para $n = 1, 2, \cdots$, **haga**:
usando $\mathfrak{M}_{n-1}$, genere T_n caminos aleatorios $w_j \in S_{n-1}$,
sea $E = \sum_j |\{w \in S_n : w_j \prec_1 w\}|$ y $\beta_n = T_n/E$
Salida: $\widetilde{c} = \widetilde{c}_{n-1}/\beta_n$
Haga $\alpha = \mathcal{A}_1(\alpha_{n-1}, \beta_1, \cdots, \beta_{n-1})$.

El algoritmo $\mathcal{A}_2$ funciona en una secuencia de etapas, una para cada valor sucesivo de n, correspondientes a las iteraciones del bucle *para*. Diremos que la etapa n es *buena*, si en ella se computa el valor β_n de tal manera que:

$$\frac{c_{n-1}}{c_n}(1 + \frac{\epsilon}{4n^2})^{-1} \leq \beta_n \leq \frac{c_{n-1}}{c_n}(1 + \frac{\epsilon}{4n^2}),$$

con $\epsilon > 0$.
Para calcular una buena aproximación de β_n, producimos caminos de longitud $n-1$ usando la cadena de Markov $\mathfrak{M}_{n-1}$ y estimando el número promedio de extensiones de un camino de un sólo paso. Podemos calcular el número de estas extensiones de un camino dado inspeccionando explícitamente cada una de las $2d-1$ posibilidades. Note que para un camino aleatorio el número de extensiones en un solo paso define una variable que toma valores en $[0, 2d-1]$, con media al menos 1, dado que $c_n \geq c_{n-1}$. El número de caminos generado por $\mathcal{A}_2$ es controlado por el parámetro T_n. Con una pequeña variación del teorema 4.4.2 para variables acotadas no negativas se tiene la siguiente proposición

Proposición 4.4.2. *En el algoritmo $\mathcal{A}_2$, si las etapas $1, 2, \cdots n-1$ son buenas etapas, entonces la etapa n es buena también con probabilidad al menos $1-(\delta/2n^2)$, para T_n al menos $cn^4\epsilon^{-2}(\ln n + \ln \delta^{-1})$, donde c es una constante que depende de la dimensión d.*

El algoritmo $\mathcal{A}_2$ es diseñado de tal manera que si asumimos que $1, 2, \cdots, n-1$ son buenas etapas, entonces la etapa n será también buena con probabilidad al menos $1-(\delta/2n^2)$. La razón para esto es la siguiente: si las etapas $1, 2, \cdots, n-1$ son buenas, entonces el valor $\widetilde{c}_n = \prod_{i=1}^{n} \beta_i^{-1}$, (salida del algoritmo $\mathcal{A}_2$ al final de la etapa n) aproxima la cantidad $\widetilde{c}_n = \prod_{i=1}^{n} c_i/c_{i-i}$, de tal manera que

$$Pr\left[(1+\epsilon)^{-1}c_n \leq \widetilde{c}_n \leq (1+\epsilon)c_n\right] > 1-\delta,$$

dado que $\prod_{i=1}^{n}(1+\epsilon/4i^2) \leq 1+\epsilon$ y que $\prod_{i=1}^{n}(1-(\delta/2i^2)) > 1-\delta$. El tiempo de ejecución de $\mathcal{A}_2$ en la etapa n es dominada por el tiempo necesario para producir T_n SAW´s de longitud $n-1$ de manera (casi) uniformemente al azar usando la cadena de Markov $\mathfrak{M}_{n-1}$. Es decir el tiempo total de $\mathcal{A}_2$ está acotado por $O(T_n n^2 \alpha_{n-1}^2(\ln n + \ln \epsilon^{-1}))$ para cualquier dimensión fija d. Por la proposición anterior , deducimos que en la etapa n, el tiempo de ejecución del algoritmo $\mathcal{A}_2$ esta acotado superiormente por un polinomio en n, ϵ^{-1}, $\ln \delta^{-1}$ y $\alpha_{n\alpha-1}^{-1}$.

Corolario 4.4.1. *Sea d un entero positivo fijo. Si aceptamos la conjetura 4.4.1, entonces el algoritmo $\mathcal{A}_2$ en la etapa n, es una FPRAS que aproxima el número de SAW's de longitud n, en el retículo rectangular entero de dimensión d.*

Capítulo 5

Schutzenberger y dos subproblemas

En este capítulo estudiaremos una técnica para resolver exactamente y en tiempo polinomial algunos problemas de conteo, ésta técnica es conocida como el *método de Schutzenberger* [1] (ver [16]). Dado un problema de conteo delgado el método funciona si podemos identificar de manera única las estructuras que queremos contar de un tamaño dado n (por ejemplo caminos o matchings) con las palabras de longitud n de un lenguaje regular[2] adecuado. En este caso contar dichas estructuras será equivalente a contar palabras.
Por todo lo anterior estaríamos tentados a solucionar por ejemplo $\#SAW$ utilizando el método de Schutzenberger codificando de manera natural los *self avoiding walks* como palabras en el alfabeto $\Sigma = \{U, D, L, R\}$ donde U significaría movimiento hacia arriba, D movimiento hacia abajo y L y R movimientos hacía la izquierda y la derecha respectivamente (observe que si un SAW tiene longitud n su codificación también tiene longitud n). Desafortunadamente esta codificación (no lo probaremos aquí)[3] y todas las propuestas hasta el momento no producen un lenguaje regular. Por lo tanto el método de Schutzenberger parece ineficaz para solucionar $\#SAW$. Sin embargo hay subproblemas no triviales de $\#SAW$ que si se pueden resolver eficientemente utilizando el método. Precisamente terminaremos este capítulo probando que el problema de contar *up-side caminos* (es decir SAWs que no tienen movimientos hacia abajo) y el problema de contar caminos hamiltonianos en retículos de altura constante se pueden resolver en tiempo polinomial haciendo uso del método de Shutzenberger. Aunque haber resuelto $\#SAW$ en tiempo polinomial hubiera sido una increíble proeza, la solución de subproblemas es también importante porque entre otras cosas éstos nos ayudan a entender y a apreciar la dificultad inherente en $\#SAW$.

[1] Marcel-Paul Schutzenberger (1920-1996) fue un matemático francés y doctor en medicina. Su trabajo tiene especial impacto en los campos del lenguaje formal, combinatoria y teoría de la información.

[2] Un lenguaje es regular si es el lenguaje reconocido por un algún autómata finito.

[3] Utilice el teorema de Myhill-Nerode (vea por ejemplo [19]).

5.1. El método de Schutzenberger

Dado L un lenguaje formal, su *función de censo* es la función $f_L : \mathbb{N} \to \mathbb{N}$ definida como:

$$f_l(n) = |\{w \in L : |w| = n\}|$$

Sea $G = (V, E)$ un digrafo y sean A y B subconjuntos de V. Un A-B *camino* en G es un camino en G cuyo extremo inicial pertenece a A y cuyo extremo final pertenece a B. Considere el siguiente problema de conteo

Problema($\#paths_{G,A,B}$: conteo de A-B caminos)
• Input: 1^n: donde n es un entero positivo.
• Problema: Calcule $\#paths_{G,A,B}(n)$, el número de A-B caminos de longitud n en G.

Lema 5.1.1. *Dados $G = (V, E)$ un digrafo y A, $B \subseteq V$, el problema $\#paths_{G,A,B}$ pertenece a FP.*

Demostración.
La matriz de adyacencia de G es la matriz $A_G = (a_{ij})_{i,j \in V}$ definida por:

$$a_{ij} = \begin{cases} 1, & \text{si } (i,j) \in E \\ 0, & \text{si } (i,j) \notin E. \end{cases}$$

Sean $n \in \mathbb{N}$ y $A_G^n = (r_{ij})$ la potencia n-ésima de A_G y $i, j \in V$, es un hecho conocido de la teoría de grafos que la entrada r_{ij} es el número de caminos en G de longitud n cuyo vértice inicial es i y vértice final j. Es decir tenemos que

$$\#paths_{G,A,B}(n) = \sum_{(i,j) \in A \times B} r_{ij}.$$

Y por lo tanto $\#paths_{G,A,B}(n)$ puede calcularse en tiempo polinomial respecto a n. ☑

Proposición 5.1.1. *Las funciones de censo de lenguajes regulares pueden ser calculadas en tiempo polinomial.*

Demostración.
Sea L un lenguaje regular y $\mathcal{M} = (\Sigma, Q, \delta, q_0, F)$ un autómata regular determinístico[4] que reconoce a L. Sea $G(\mathcal{M})$ el digrafo de transiciones de $\mathcal{M}$ y $A_{G(\mathcal{M})}$ la matriz de adyacencia de $G(\mathcal{M})$. Es claro que cada camino en $G(\mathcal{M})$ de longitud n cuyo vértice inicial es q_0 y cuyo vértice final pertenece a F, se puede ver como una palabra de longitud n en L. El recíproco también se cumple. Esta correspondencia biunívoca entre caminos y palabras implica que $\#paths_{G,\{q_0\},F}(n) = f_L(n)$, para todo $n \in \mathbb{N}$. Y por lo tanto según el lema anterior f_L puede ser calculada en tiempo polinomial. ☑

[4]Esto se supone sin pérdida de generalidad

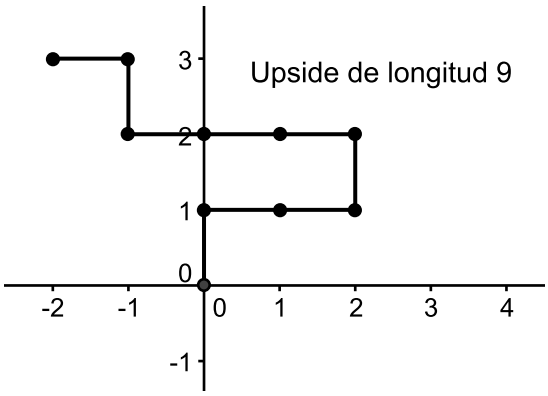


Figura 5.1: Up-side de longitud 15

El *método de Schutzenberguer* se basa en la siguiente observación:
Sea $\mathcal{A} = \{A_n\}_{n\in\mathbb{N}}$ una sucesión de conjuntos finitos y $f_{\mathcal{A}}$ la función de conteo asociada a $\mathcal{A}$ tal como en la definición 1.0.1. Si existe un lenguaje regular L y una función $\phi : \bigcup_{i\in\mathbb{N}} A_i \to L$ que satisface:

1. ϕ es una biyección.
2. Para todo $n \in \mathbb{N}$ y para todo $w \in A_n$ se tiene que $\phi(w) \in f_L(n)$.

Es claro por 1) y 2) que $f_{\mathcal{A}}(n) = |A_n| = f_L(n)$ y por la proposición 5.1.1 la función $f_{\mathcal{A}}$ puede ser calculada en tiempo polinomial.

La biyección ϕ definida arriba no es otra cosa que una codificación de las estructuras que se quieren contar en $\mathcal{A}$ como palabras en un lenguaje adecuado. Una tal codificación debe satisfacer las siguientes condiciones:

I) La codificación preserva longitudes, es decir dado $n \in \mathbb{N}$ y $\gamma \in A_n$, el código de γ es una palabra de longitud n.

II) El conjunto de códigos es un lenguaje regular.

En las siguientes dos secciones veremos el método de Schutzenberger en acción resolviendo dos subproblemas relacionados con self avoiding walks.

5.2. Contando up-side caminos

Sea $\mathcal{L}_n^2$ el grafo definido en el ejemplo 3.1.2. Un up-side camino en $\mathcal{L}_n^2$ es un camino simple en $\mathcal{L}_n^2$ al cual se le prohíben movimientos hacia abajo (ver figura 5.1).
Probaremos en esta sección que el problema $\#up - side$ definido como:

Problema: $\#up - side$
• *input*: 1^n

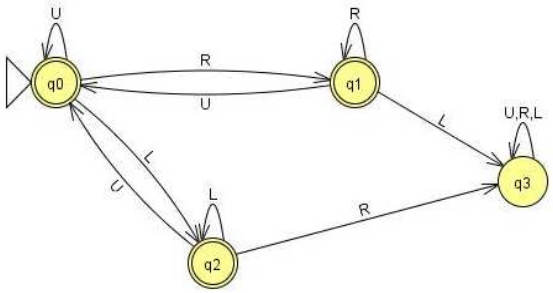


Figura 5.2:

• *output*: número de up-sides en $\mathcal{L}_n^2$.

se puede resolver en tiempo polinomial. Incluso presentaremos una fórmula cerrada que lo resuelve (extraído de [8]).

Observe que todo up-side camino en $\mathcal{L}_n^2$ puede ser codificado de manera única como una palabra sobre el alfabeto $\Sigma = \{U, L, R\}$. Donde la letra U significa movimiento hacia arriba y, L y R movimientos hacia la izquierda y derecha respectivamente. Denotaremos como $L_{up-side}$ al lenguaje de las palabras sobre Σ que codifican un up-side camino en $\mathcal{L}_n^2$. Note que el lenguaje $L_{up-side}$ es reconocido por el autómata finito determinístico definido con el digrafo o diagrama de estados de la figura 5.2. Y por lo tanto $L_{up-side}$ es un lenguaje regular. Entonces por el método de Schutzenberguer el problema $\#up - side$ se puede resolver en tiempo polinomial.

Dado que toda palabra aceptada por el automata describe un camino en el diagrama de estados que nunca utiliza el vértice q_3, podemos olvidarnos de él junto con todas las aristas incidentes. Y por lo tanto la matriz de adyacencia de nuestro nuevo grafo es:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \text{ y definamos } \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix} = A^n \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

No es difícil probar (e.g. por inducción) que $b_n = c_n$, para toda n. Es claro que a_n es el numero de palabras de longitud n aceptadas por el automata y por lo tanto también el numero de up-side caminos de longitud n. Observe que

$$\begin{pmatrix} a_{n+1} \\ b_{n+1} \\ b_{n+1} \end{pmatrix} = A^{n+1} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = A \cdot \left(A^n \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) = A \cdot \begin{pmatrix} a_n \\ b_n \\ b_n \end{pmatrix} = \begin{pmatrix} a_n + 2b_n \\ a_n + b_n \\ a_n + b_n \end{pmatrix}$$

Por lo tanto $a_{n+1} = a_n + 2b_n$ y $b_{n+1} = a_n + b_n$, para todo n, con $a_0 = 1$ y $b_0 = 1$. Si

$A(z) = \sum_{n=0}^{\infty} a_n z^n$ es la función generatriz de a_n, entonces

$$\sum_{n=0}^{\infty} a_{n+1} z^{n+1} = \sum_{n=0}^{\infty} a_n z^{n+1} + 2\sum_{n=0}^{\infty} b_n z^{n+1}, \text{ y}$$

$$A(z) - 1 = \sum_{n=1}^{\infty} a_n z^n = z\sum_{n=0}^{\infty} a_n z^n + 2z\sum_{n=0}^{\infty} b_n z^n = zA(z) + 2zB(z),$$

y por lo tanto

$$(z-1)A(z) + 2zB(z) + 1 = 0. \tag{5.1}$$

Además

$$\sum_{n=0}^{\infty} b_{n+1} z^{n+1} = \sum_{n=0}^{\infty} a_n z^{n+1} + \sum_{n=0}^{\infty} b_n z^{n+1}, \text{ y}$$

$$B(z) - 1 = \sum_{n=1}^{\infty} b_n z^n = z\sum_{n=0}^{\infty} a_n z^n + z\sum_{n=0}^{\infty} b_n z^n = zA(z) + zB(z),$$

y por lo tanto

$$zA(z) + (z-1)B(z) + 1 = 0. \tag{5.2}$$

A partir de 5.1 y 5.2 tenemos que

$$A(z) = \frac{z+1}{-z^2 - 2z + 1} = \frac{a}{z - (1+\sqrt{2})} + \frac{b}{z - (1-\sqrt{2})},$$

donde $a = \frac{1}{2}(1-\sqrt{2})$ y $b = \frac{1}{2}(1+\sqrt{2})$, y como $\sum_{n=0}^{\infty} r^n z^n = 1/(z-r)$, para r cualquier constante. Entonces

$$A(z) = \sum_{n=0}^{\infty} a_n z^n = a\sum_{n=0}^{\infty} (1+\sqrt{2})^n z^n + b\sum_{n=0}^{\infty} (1-\sqrt{2})^n z^n$$

$$= \sum_{n=0}^{\infty} \left(\frac{1}{2}(1+\sqrt{2})^{n+1} + \frac{1}{2}(1-\sqrt{2})^{n+1} \right) z^n, \text{ por lo tanto}$$

$$a_n = \frac{1}{2}\left[(1+\sqrt{2})^{n+1} + (1-\sqrt{2})^{n+1} \right].$$

Por ejemplo el número de up-side caminos de longitud 2 en $\mathcal{L}_n^2$, es $a_2 = \frac{1}{2}\left[(1+\sqrt{2})^3 + (1-\sqrt{2})^3\right] = 7$ (ver figura 5.3).

5.3. Contando caminos hamiltonianos en retículos rectangulares

Sea r un número entero positivo, usaremos el símbolo $\#ham[r]$ para denotar el problema de conteo delgado definido como:

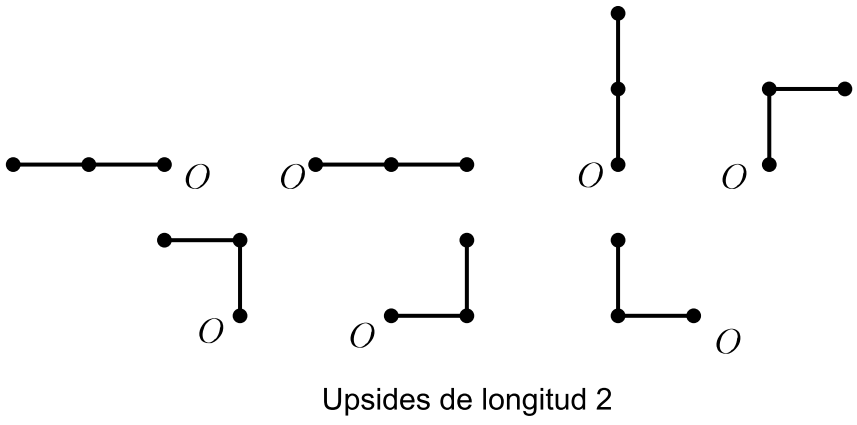


Figura 5.3:

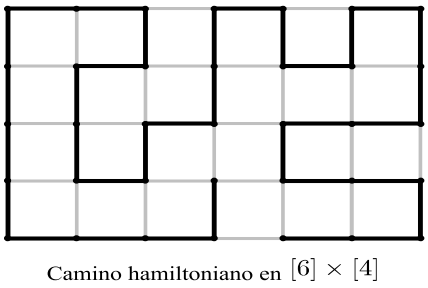


Figura 5.4:

Problema ($\#ham[r]$: conteo de caminos hamiltonianos en el retículo rectangular $[n] \times [r]$)
Input: 1^n (entero positivo)
Problema: Compute $\#ham[r](n)$ el número de caminos hamiltonianos (caminos que pasan por todos los vértices) en $[n] \times [r]$.

La figura 5.4 muestra un camino hamiltoniano en el retículo $[6] \times [4]$. Probaremos en esta sección que el problema $\#ham[r]$ es un problema de conteo tratable haciendo uso nuevamente del Método de Schutzenberguer (este resultado fue publicado en [15]). Observe que para esto es suficiente probar que si r es un entero positivo fijo, podemos computar en tiempo $2^{O(r)}$ un autómata finito $\mathcal{M}_r$ de tal manera que $\#ham[r](n) = f_{\mathcal{M}_r}(n)$. Donde $f_{\mathcal{M}_r}$ es la función de censo del lenguaje $L_{\mathcal{M}_r}$, el lenguaje reconocido por el automata $\mathcal{M}_r$ y es el lenguaje que codifica cada camino hamiltoniano en $[n] \times [r]$ como una palabra en un alfabeto adecuado de longitud n, para todo n entero positivo.

Estamos listos para describir nuestra codificación. De ahora en adelante fijaremos r

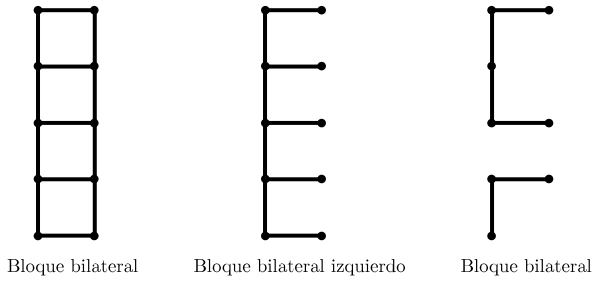


Figura 5.5:

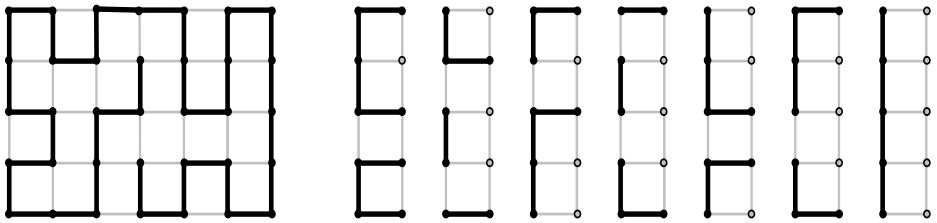


Figura 5.6:

un entero positivo. Un *bloque bilateral* es un grafo isomorfo a $\{1,2\} \times [r]$. Un *bloque izquierdo* es un subgrafo de un bloque bilateral que se obtiene eliminando todas las aristas de la forma $\{(2,x),(2,x+1)\}$ (ver figura 5.5). Es claro que $[n] \times [r]$ puede ser particionado en un número finito de bloques izquierdos y un bloque bilateral. Un *patrón* es un subgrafo de un bloque izquierdo que se obtiene eliminando aristas del bloque. Note que dado γ, un camino hamiltoniano en el retículo rectangular $[n] \times [r]$, γ es la concatenación de una secuencia $p_1, p_2, \cdots, p_n$ de n patrones tales que p_n no tiene aristas horizontales. Es decir cada camino hamiltoniano pude ser visto o codificado como una palabra en el alfabeto de los patrones denotado como $\mathcal{P}_L$. Observe la figura 5.6.
Note también que dado que un patron es un subgrafo de un bloque izquierdo tenemos la siguiente

Proposición 5.3.1. *Dado r entero positivo mayor que 1, existe a lo sumo 2^{2r} patrones.*

La anterior proposición implica que podemos enumerar el conjunto de todos los patrones en tiempo $2^{O(r)}$.
Dados p y q dos patrones, podemos ver a pq (la concatenación de p con q) como un subgrafo de $\{1,2,3\} \times [r]$. El conjunto de los vértices de pq localizados en $\{2\} \times [r]$ sera llamado la *2-fibra* de pq, y lo denotaremos con el símbolo $\varsigma(pq)$. Dado i en $\{0,1,2,3,4\}$ definimos la función $\alpha_i : \mathcal{P}_L \times \mathcal{P}_L \to \mathbb{N}$ de la siguiente manera

$$\alpha_i(p,q) = |\{v \in \varsigma(pq) : deg_{pq}(v) = i\}|$$

donde $deg_{pq}(v)$ denota el grado de v como un vértice del grafo pq. Definimos análogamente las tres funciones $\beta_0, \beta_1, \beta_3 : \mathcal{P}_L \to \mathbb{N}$ las cuales cuentan el número de vértices con grado cero, uno y tres respectivamente localizados en la *1-fibra* del patrón evaluado. Note que podemos computar en tiempo $2^{O(r)}$ las tablas de cada una de las funciones α_0, α_1, α_3, α_4, β_0, β_1 y β_3.
Dados p y q dos patrones, diremos que el par (p, q) es *compatible* si y sólo si se cumple que $\alpha_1(p, q) \leq 2$ y $\alpha_0(p, q) = \alpha_3(p, q) = \alpha_4(p, q) = 0$. Usaremos el símbolo $\mathcal{I}$ para denotar el conjunto de todos los pares compatibles. Note también que $\mathcal{I}$ puede ser computado en tiempo $2^{O(r)}$.

La siguiente caracterización de los caminos hamiltonianos es clave para definición del automata $\mathcal{M}$.

Lema 5.3.1. *γ es un camino hamiltoniano en $[n] \times [r]$ si y solo si γ es un subgrafo generador de $[n] \times [r]$ sin ciclos, donde exactamente dos vértices tienen grado uno y el resto de vértices tienen grado dos.*

El anterior lema nos permite codificar los caminos hamiltonianos en $[n] \times [r]$ en terminos de los patrones.

Proposición 5.3.2. *Una secuencia de patrones $p_1 p_2 \cdots p_n$ codifica un camino hamiltoniano en $[n] \times [r]$ si y solo si:*

1. *Todas las aristas en p_n son verticales y están localizadas en la columna izquierda. Además para todo $i \leq n-1$, el par (p_i, p_{i+1}) es compatible.*
2. *$\beta_1(p_1) + \sum_{i \leq n-1} \alpha_1(p_i, p_{i+1}) = 2$ y $\beta_1(p_1) = \beta_3(p_1) = 0$.*
3. *El grafo $p_1 p_2 \cdots p_n$ no tiene ciclos.*

Dado $n \geq 2$, una *$n-$cadena* es una secuencia de n patrones $p_1 p_2 \cdots p_n$ que satisface las condiciones impuestas en la anterior proposición. También es claro que hay una biyección entre el conjunto de $n-$cadenas y el conjunto de caminos hamiltonianos en $[n] \times [r]$.

Ahora sea Σ el alfabeto $\{p : p \in \mathcal{P}_L\}$ y L_r el lenguaje $\{p_1 p_2 \cdots p_n \in \Sigma^* : p_1 p_2 \cdots p_n$ es una $n-$cadena$\}$

Teorema 5.3.1. *L_r es un lenguaje regular. Además, podemos computar en tiempo $2^{O(r)}$ un autómata finito $\mathcal{M}_r$ que reconoce L_r.*

Demostración.
Queremos probar que el lenguaje

$$L_r = \{p_1 p_2 \cdots p_n \in \Sigma^* : \Psi\}$$

es regular, donde Ψ es la conjunción de los items 1,2 y 3, como en la anterior proposición. Podemos escribir Ψ como $\Psi_1 \wedge \Psi_2$, donde Ψ_1 es la conjunción de los

items 1 y 2, y Ψ_2 es la condición 3. Podemos entonces definir los lenguajes $L_r^1 = \{p_1p_2\cdots p_n \in \Sigma^* : \Psi_1\}$ y $L_r^2 = \{p_1p_2\cdots p_n \in \Sigma^* : \Psi_2\}$, es claro que $L_r = L_r^1 \cap L_r^2$. Y dado que la intersección de lenguajes regulares es regular, será suficiente probar que L_1 y L_2 son lenguajes regulares y que los respectivos autómatas finitos que los reconocen pueden ser ambos computados en tiempo $2^{O(r)}$.
Sea $\mathcal{M}_1 = (\Sigma, Q, \delta, q_0, F)$ el autómata finito definido como:

I) $Q = \{(q, i) : q \in \Sigma \text{ y } 0 \leq i \leq 2\} \cup \{q_0, q_r, q_a\}$,

II) $F = Q \setminus \{q_r\}$,

III) Sea $p_1p_2\cdots p_n$ un input de $\mathcal{M}_1$. La función de transición se define de la siguiente forma:

a) $\delta(q_0, p_1) = (p_1, \beta_1(p_1))$, si $\beta_0(p_1) = \beta_3(p_1) = 0$ y $\beta_1(p_1) \leq 2$, en otro caso $\delta(q_0, p_1 = q_r)$.

b) Dado $i \leq n-1$, tenemos que $\delta((p_i, k), p_{i+1}) = q_r$, siempre que las siguientes condiciones se cumpla:

- $(p_i, p_{i+1}) \in \mathcal{I}$,
- $k + \alpha_1(p_i, p_{i+1}) \geq 3$.

c) Sea $i \leq n-1$. Si $(p_i, p_{i+1}) \in \mathcal{I}$ y $k + \alpha_1(p_i, p_{i+1}) \leq 2$, entonces $\delta((p_i, k), p_{i+1}) = (p_{i+1}, k + \alpha_1(p_i, p_{i+1}))$.

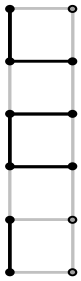
d) Si $\beta_1(p_n) + k = 2$ y p_n no contiene aristas horizontales, entonces $\delta((p_n, k), \square) = q_a$.

e) Si $\beta_1(p_n) + k \neq 2$ o p_n contiene aristas horizontales, entonces $\delta((p_n, k), \square) = q_r$.

Note que el autómata $\mathcal{M}_1$ simplemente verifica que $p_1p_2\cdots p_n$ es una secuencia de patrones compatibles, tales que el número de vértices de grado uno sea dos y tales que todas las aristas que ocurren en p_n sean aristas verticales localizadas en su columna izquierda. Es claro que el autómata finito $\mathcal{M}_1$ reconoce el lenguaje L_r^1. El tiempo requerido para construir el autómata $\mathcal{M}_1$ está acotado superiormente por $2^{O(r)}$ porque las tablas de las funciones involucradas en la función de transición pueden ser computadas en tiempo $2^{O(r)}$ y $|Q| \in 2^{O(r)}$.

Para finalizar la prueba mostraremos que L_r^2 es un lenguaje regular. Daremos una breve (pero suficiente) descripción de un autómata $\mathcal{M}_2$ que reconoce L_r^2. Mostraremos que si fijamos $n \geq 1$, podemos usar un autómata $\mathcal{M}_2$ que reconoce los subgrafos cíclicos de $[n] \times [r]$.

Una *c−estructura* es un subgrafo de $[n] \times [r]$ conformado por un conjunto (o arreglo) conexo de aristas verticales en la izquierda, y dos aristas horizontales apuntando a la derecha, una de ellas localizada en el fondo y la otra en la cima del arreglo vertical (ver figura 5.7). Observemos que:



Patrón con una C-estructura.

Figura 5.7:

I) Cualquier ciclo comienza con una $c-$*estructura*, es decir: el patrón que esta más a la izquierda del ciclo necesariamente contiene al menos una $c-$*estructura*.

II) cualquier $c-$*estructura* puede ser detectada por $\mathcal{M}_2$, cuando el autómata está leyendo el correspondiente patrón.

Una *c-estructura* es como una alerta, que nos avisa de la posible aparición de un ciclo en el futuro. Debemos guardar información a cerca de las *c-estructuras* que hemos observado hasta el momento. Para lograr esto, seguiremos la evolución de las trayectorias que se originan de los puntos finales de dichas *c-estructuras*. Es posible hacer esto dado que:

I) Si w es un input de $\mathcal{M}_2$ y $t \leq |w|$, hay a lo sumo $r/2$ pares de trayectorias en el instante t, originadas por *c-estructuras* previamente vistas.

II) La única información que tenemos que guardar, para controlar la evolución de un par de trayectorias *peligrosas*,[5] son las y-coordenadas de los vértices donde aquellas trayectorias alcanzan la columna derecha del patrón que está siendo leído.

Dado i un entero positivo menor o igual a $r/2$, usaremos el símbolo P_i para denotar el conjunto

$$\left\{((a_1, b_1), \cdots, (a_i, b_i)) : i \leq \frac{r}{2} \ \& \ \Phi\right\},$$

donde Φ son las condiciones $a_1, \cdots, a_i, b_1, \cdots, b_i \in \{1, \cdots, r\}$, $(a_1, b_1), \cdots, (a_i, b_i)$ son diferentes componente a componente y $a_1 \lneqq b_1$; $a_2 \lneqq b_2$;$\cdots$;$a_i \lneqq b_i$.
Sea P igual al conjunto de los estados del autómata $\mathcal{M}_2$, el conjunto P es esencialmente igual a $\bigcup_{0 \leq \frac{r}{2}} P_i$. Sea $p = ((a_1, b_1), \cdots, (a_i, b_i))$ un elemento de P, y suponga que p es el estado de $\mathcal{M}_2$ en el instante t. El estado p nos informa los puntos finales de los i pares de trayectorias peligrosas que se han originado hasta el patrón que está siendo leído en el instante t. La función de transición de $\mathcal{M}_2$, denotada por

[5]Es decir trayectorias que pueden desembocar en un ciclo.

ρ, se define de tal manera que nos permitirá seguir la evolución de las trayectorias peligrosas, de tal manera que si un ciclo es encontrado entonces el autómata pasará a un estado de rechazo. Consideremos, como un ejemplo la siguiente situación:

1. El estado del autómata $\mathcal{M}_2$, en el instante t, es igual a $((a_1, b_1), (a_2, b_2), (a_3, b_3))$.
2. $a_1 \lneqq b_1 \lneqq a_2 \lneqq b_2 \lneqq a_3 \lneqq b_3$.
3. La trayectoria cuya y-coordenada es igual a $a_1 \geq 2$, se extiende con una arista vertical hacia abajo y después con una arista horizontal.
4. La trayectoria cuya y-coordenada es igual a b_1 se extiende con una arista vertical hacia arriba y después con una arista horizontal.
5. La trayectoria cuya y-coordenada es igual a a_2, se extiende con una arista vertical hacia abajo y después con una arista horizontal. Además supongamos que $a_2 - b_1 = 2$.
6. La trayectoria cuya y-coordenada es igual a b_2 se extiende con una arista vertical hacia arriba y después con una arista horizontal.
7. Las otras trayectorias se extienden con aristas horizontales. Además suponga que $a_3 - b_2 \geq 2$.
8. Una nueva *c-estructura* es observada, constituida por dos aristas horizontales, cuyas y-coordenadas son iguales a $k + 3$ y $k \gneqq b_i$, y por tres aristas verticales en la izquierda uniendo los vértices localizados en las alturas k y $k + 3$.

Entonces, dada toda esta información, tenemos que el estado interno del automata $\mathcal{M}_2$, en el instante $t + 1$, es igual a

$$((a_1 - 1, b_2 + 1), (a_3, b_3), (k, k + 3))$$

Suponga que el autómata $\mathcal{M}_2$ está leyendo la palabra w y que el par (a, b) pertenece al estado interno del autómata, en el instante t. Hay tres posibilidades para el par (a, b), en el instante $t + 1$: el par (a, b) se *fusiona* con otro par que ocurrió también en el instante t, el par (a, b) *sobrevive* o el par (a, b) se *desvanece*. Diremos que el par (a, b) se desvanece en el instante $t + 1$ si y sólo si el caracter w_{i+1}, el $t + 1$-ésimo patrón de la palabra que está siendo leída, contiene una cadena de aristas verticales que unen los vértices localizados en las alturas a y b. Si un par se desvanece, un ciclo ha sido creado (detectado). Es también posible que dos pares (a, b) y (c, d) se fusionen, esta fusión origina un ciclo si y sólo si los pares están anidados en el instante t, es decir si $a \lneqq c \lneqq d \lneqq b$ y en el instante $t + 1$, el patrón w_{i+1} une con aristas verticales las alturas a, c y b, d, creando de esta forma un ciclo. Usaremos el término 2-*vectores* para denotar los estados de $\mathcal{M}_2$, note que la evolución de pares define un álgebra de 2-*vectores* la cual, por ser finita, puede ser precomputada en

tiempo $2^{O(r)}$, para ser utilizada en la construcción de la función de transición de $\mathcal{M}_2$.
Hemos discutido las principales características de $\mathcal{M}_2$. En este punto, el lector debería estar completamente convencido que $\mathcal{M}_2$ reconoce el lenguaje L_r^2, es decir que L_r^2 es regular, y que $\mathcal{M}_2$ puede ser computado en tiempo $2^{O(r)}$.
Podemos ahora afirmar que un autómata finito que reconoce L_r puede ser computado en tiempo $2^{O(r)}$. Es decir, hemos terminado con la prueba del teorema. ☑

Corolario 5.3.1. *Si r es un entero positivo fijo, entonces el número de caminos hamiltonianos en $[n] \times [r]$, puede ser computado en tiempo polinomial.*

Bibliografía

[1] Agrawal M., Kayal N., Saxena N. *PRIMES is in P.* Annals of Mathematics 160 (2). 2004.

[2] Valiant L., *The complexity of computing the permanent.* Theoretical Computer Science, 1979.

[3] Jerrum M., *Two dimensional monomer-dimer system are computationally intractable.* Journal of Statistical Physics, vol 48, 1987.

[4] Karp M., Liptak Z., *Counting, sampling and integrating: Algorithms and complexity.* Notas de lectura.

[5] Valiant L., *The complexity of enumeration and reliability problems.* SIAM J. COMPUT, vol 8, 1979.

[6] Goldsmith J., *Tally NP set and easy census functions.*

[7] Garey M., *Computers and intractability.* W.H. Freeman and Company. New York, 1979.

[8] Liskiewicz M., *The complexity of counting self avoiding walks in two-dimensional grids graphs and hypercube graphs.* Electronic Colloquium on Computational Complexity, 2001.

[9] Sinclair A., *Improve bounds for mixing rates of Markov chains and multicommodity flow.* Combinatorics, probability an computing 1. 1992.

[10] Karp R. (et. all), *An optimal algorithm for Monte Carlo estimation.*

[11] Karp R. (et all), *Monte Carlo approximation algorithms por enumeration problems.* Journal for algorithms, 1989, vol. 10, p.429-448.

[12] Haggstrom O., *Finite Markov Chains and Algorithmic Applications.* London Mathematical Society. Cambridge University Press, 2002.

[13] Kenyon C., Randall D., Sinclair A. *Approximating the Number of Monomer-Dimer Coverings of a lattice.*, 1997.

[14] Randall D., Sinclair A. *Self-testings Algorithms for Self Avoidings Walks.* Journal for algorithms, 1989, vol. 10, p.429-448.

[15] Gutierrez F., Montoya J. A. y Zambrano L. *Applications of Schuzenberger-Bertoni method: Counting Hamiltonian Structures.* Submitted LAGOS 2011.

[16] Bertoni A., Goldwurm M. y Sabadini N. *The complexity of computing the number of strings of given length in context-free languages.* Theoretical Computer Sciencie. 86(1991) 325-342.

[17] Madras N., Gordon S. *The Self-Avoiding walk.* Birkhauser Boston. 1996.

[18] Arora S., Barak B. *Computacional Complexity: A Modern Approach.* Princeton University.

[19] Sipser M., *introduction to the theory of computation.* 2da edición, Thomson. 2006.