Universidad Nacional Autónoma de México

Facultad de Ciencias

Método de enumeración estocástica

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
MATEMÁTICO

PRESENTA:
Jorge Vasquez Arriaga

DIRECTOR DE TESIS:
Dr. Luis Antonio Rincón Solís

Cd. Mx. 2024

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

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

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

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

II

1. Datos del alumno	1. Datos del alumno
Apellido paterno	Vásquez
Apellido materno	Arriaga
Nombre	Jorge
Teléfono	95 13 49 33 64
Universidad Nacional Autónoma de México	Universidad Nacional Autónoma de México
Facultad de Ciencias	Facultad de Ciencias
Carrera	Matemáticas
Número de cuenta	419004633
2. Datos del tutor	2. Datos del tutor
Grado	Dr
Nombres	Luis Antonio
Apellido Paterno	Rincón
Apellido Materno	Sólis
3. Datos del sinodal 1	3. Datos del sinodal 1
Grado	Dr
Nombre	Germán
Apellido Paterno	Benítez
Apellido Materno	Bobadilla
4. Datos del sinodal 2	4. Datos del sinodal 2
Grado	L en CC
Nombres	Pedro Ulises
Apellido Paterno	Cervantes
Apellido Materno	Gonzáles
5. Datos del sinodal 3	5. Datos del sinodal 3
Grado	Dr
Nombre	Gonzalo
Apellido Paterno	Pérez
Apellido Materno	de la Cruz
6. Datos del sinodal 4	6. Datos del sinodal 4
Grado	Dr
Nombre	Yuri
Apellido Paterno	Salazar
Apellido Materno	Flores

7. Datos del trabajo escrito	7. Datos del trabajo escrito
Título	Método de enumeración estocástica
Número de páginas	114
Año	2024

Agradecimientos

Quisiera agradecer a mis padres, por el esfuerzo y apoyo que me brindan día con día. Mis logros son más de ustedes que míos.

A mi tía Silvia que me animó a estudiar una carrera en ciencias y me brindó las facilidades para afrontar los primeros semestres.

A Milka que me abrió las puertas de su casa y cuidó de mí. A Marta y Alejandro que desde el primer momento me tendieron una mano para ayudarme en lo que necesitara.

A mi amiga Cristhian, por todos los buenos momentos. A Ricardo y Gerardo que me ayudaron en la pandemia y en mi momento más difícil en la carrera. A mis compañeros, Luis, Hugo, Adrián, Miguel y Odalys, que hicieron más amenos mis primeros semestres.

Al Dr. Luis Rincón, mi asesor de tesis, por toda la dedicación, los comentarios y la paciencia.

Índice general

Introducción

En el campo de las ciencias de la computación, existe un importante pero difícil problema: estimar el costo de un árbol. El problema consiste en tener un árbol donde a cada vértice tiene asociado un costo; el costo del árbol se calcula como la suma de los costos de cada vértice. Si se logra encontrar una solución eficiente para este problema, se tendría un gran impacto en el análisis de la complejidad computacional.

Un enfoque inicial para resolver este problema consiste en generar trayectorias aleatorias a través del árbol, lo cual fue propuesto por Knuth. Sin embargo, como Knuth y otros investigadores señalaron, este estimador puede tener una gran variación y volverse ineficiente al subestimar el costo del árbol.

Recientemente, se ha propuesto un nuevo algoritmo llamado: método de enumeración estocástica desarrollado por Rubinstein y su equipo. Los autores han demostrado mediante pruebas numéricas que este algoritmo sencillo puede ser muy eficiente para resolver problemas específicos de conteo, como contar el número de asignaciones de satisfacción (*satisfiability*) o el número de emparejamientos perfectos en gráficas bipartitas, sin embargo, una de sus aplicaciones más destacadas se encuentra en el campo de la confiabilidad de redes.

El objetivo principal de este trabajo es presentar métodos de conteo de árboles, centrándonos especialmente en el análisis del método de enumeración estocástica, y demostrar que este método produce una significativa reducción en la varianza en comparación con el estimador de Knuth. Además, se mostrará cómo funciona el método de enumeración y se explorarán aplicaciones prácticas.

La estructura del documento es la siguiente: en el primer capítulo, se presenta una rama de las matemáticas llamada teoría de gráficas. Aquí se explicará qué es un árbol y cuál es el problema asociado a calcular su costo. También se abordarán

conceptos clave, como qué es una trayectoria y qué es una gráfica bipartita, los cuales serán utilizados en los capítulos siguientes. En resumen, este capítulo es una base fundamental para entender el resto del documento.

En el segundo capítulo, se exploran conceptos esenciales para comprender el problema de conteo de árboles. Se introduce el concepto de complejidad computacional y se explican algunas estructuras de datos. Se pone un enfoque especial en los árboles, detallando la estructura de datos asociada a ellos, la cual nos permitirá abordar el cálculo del costo de un árbol. También se presentan otras estructuras de datos que serán empleadas en los métodos de conteo de árboles. En resumen, este capítulo proporciona las bases técnicas necesarias para el estudio y desarrollo de los métodos de conteo de árboles.

En el tercer capítulo, se presentan las herramientas probabilísticas necesarias para analizar algoritmos aleatorios. Estas herramientas serán especialmente útiles para entender y analizar dos algoritmos en particular: el algoritmo de Knuth y el método de enumeración estocástica. Con este enfoque, se podrá comprender cómo funcionan estos algoritmos y cómo afecta la aleatoriedad en sus resultados.

En el cuarto capítulo, se examinan métodos para resolver el problema de calcular el costo de un árbol. Se presta especial atención al algoritmo de Knuth, destacando sus limitaciones, y al método de enumeración estocástica, resaltando cómo mejora el algoritmo de Knuth. De esta manera, podremos comprender las fortalezas y debilidades de cada método, y apreciar cómo el enfoque de enumeración estocástica ofrece una mejora significativa sobre el algoritmo de Knuth.

En el quinto capítulo, se muestran diversas aplicaciones del problema de conteo de árboles. Se presentan ejemplos prácticos de cómo utilizar el método de enumeración estocástica para abordar estos problemas. En concreto, se exploran cuatro situaciones: el conteo de trayectorias, el conteo de emparejamientos en gráficas bipartitas, el problema de satisfacción (*satisfiability problem*) y la confiabilidad en redes. Estos ejemplos ayudarán a entender cómo el método de enumeración estocástica puede ser empleado para resolver diferentes problemas.

Capítulo 1

Teoría de gráficas

La teoría de gráficas es una rama de las matemáticas que estudia las relaciones entre objetos. En lugar de enfocarse en los objetos en sí mismos, la teoría de gráficas se centra en cómo esos objetos están conectados entre sí, esta teoría analiza patrones, estructuras y propiedades que surgen de estas conexiones.

Además de su utilidad práctica, la teoría de gráficas también tiene profundas implicaciones teóricas y se encuentra en la intersección de diversas disciplinas como las matemáticas, las ciencias de computación y la física.

El objetivo de este capítulo es presentar una breve introducción a teoría de gráficas, el tema es muy amplio y si se quiere profundizar se recomienda consultar [6] y [8], mismos en los que está basado el material que se presenta en este capítulo.

1.1. Definiciones básicas

Definición 1.1 *Una gráfica G consiste en un conjunto finito y no vacío V de objetos llamados vértices, así como de un conjunto E de pares de dos elementos pertenecientes a V, conocidos como aristas. Cada arista establece una conexión entre dos vértices distintos, y se garantiza que no exista más de una arista conectando un mismo par de vértices.*

La notación G es por la palabra gráfica, el término V es por la palabra vértice y la notación E es debido al término en inglés, *edges*, que significa aristas.

Ejemplo 1.1 Consideremos la gráfica G con conjunto de vértices $V = \{a,b,c,d\}$ y conjuntos de aristas $E = \{\{a,b\},\{a,c\},\{c,d\}\}$. Véase la Figura 1.1. Dada una gráfica, es posible darle una representación geométrica de la siguiente manera: A cada vértice de G se le asocia un nodo o punto, y se dibuja un segmento de recta o curva entre dos nodos si los vértices asociados a dichos nodos pertenecen a una arista en G.

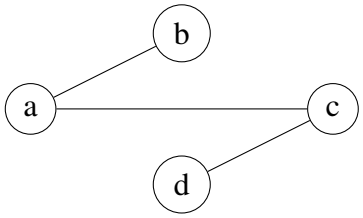


Figura 1.1: Representación geométrica de una gráfica G.

•

Los vértices también son llamados nodos o puntos, mientras que las aristas también son llamadas líneas, vínculos o enlaces. Para indicar que una gráfica G tiene un conjunto de vértices V y un conjunto de aristas E, se escribe $G = (V,E)$. Para enfatizar que ambos conjuntos pertenecen a la gráfica G se escribe $V(G)$ y $E(G)$, respectivamente. Cada arista $\{u,v\}$ perteneciente a G es usualmente denotada como uv o vu.

Los siguientes conceptos son importantes en la teoría de gráficas.

Definición 1.2 *Sea G una gráfica. Se define el orden de G como el número de vértices. El tamaño de G como el número de aristas. Los números enteros $\boldsymbol{n}$ y $\boldsymbol{m}$ denotan el orden y tamaño de la gráfica en cuestión, si se está trabajando con dos o más gráficas es usual usar la notación n_G y m_G.*

Definición 1.3 *Sea G una gráfica. Sean u y v dos vértices cualesquiera pertenecientes a $V(G)$ y sean a y b dos aristas cualesquiera pertenecientes a $E(G)$.*

1. *Dos vértices u y v son adyacentes si $uv \in E(G)$.*
2. *Dos aristas a y b son adyacentes si a y b tienen un vértice en común.*
3. *Una arista a y un vértice u son incidentes si u es un vértice perteneciente a la arista a.*

Definición 1.4 *Sea G una gráfica, v en $V(G)$ y S un subconjunto de $V(G)$.*

1. *El conjunto de vecinos de v, denotado por $N(v)$ o $N_G(v)$, se define como $\{w \in V(G) : vw \in E(G)\}$.*
2. *El conjunto de vecinos de S, denotado por $N(S)$ o $N_G(S)$, se define como $\{w \in V(G) : su \in E(G)$ para algún $s \in S\}$.*

La notación N hace referencia al término en inglés *neighborhood* que significa vecindad. En este trabajo, para referirse a la cardinalidad de algún conjunto A, se usa el símbolo $|A|$.

Definición 1.5 *Sea G una gráfica, v en $V(G)$ y S un subconjunto de $V(G)$.*

1. *El grado de un vértice v, denotado por $\delta(v)$ o $\delta_G(v)$, se define como $|N(v)|$.*
2. *El grado máximo de G, denotado por $\Delta(G)$, se define como* $\text{máx}\{\delta(v) : v \in V(G)\}$.
3. *El grado mínimo de G, denotado por $\delta(G)$, se define como* $\text{mín}\{\delta(v) : v \in V(G)\}$.

Teorema 1.1 *En toda gráfica G,*

$$2\boldsymbol{m} = \sum_{v \in V(G)} \delta(v).$$

Demostración: Sea $e \in E(G)$, e es de la forma $e = \{uw\}$ con $u, w \in V(G)$. Al considerar $\delta(w)$ se cuenta la arista e, y lo mismo si se considera $\delta(u)$, puesto que $\delta(u)$ y $\delta(w)$ son sumandos de $\sum_{v \in V(G)} \delta(v)$ y dado que e es una arista arbitraria, se sigue que cada arista se cuenta dos veces, de lo cual se concluye que $2\mathbf{m} = \sum_{v \in V(G)} \delta(v)$. ■

Corolario 1.1 *Toda gráfica G tiene un número par de vértices con grado impar.*

Demostración: Sea $X_1 = \{v \in V(G) : \delta(v)$ es par$\}$ y $X_2 = \{v \in V(G) : \delta(v)$ es impar$\}$.

Dado que

$$2\mathbf{m} = \sum_{v \in V(G)} \delta(v) = \sum_{u \in X_1} \delta(u) + \sum_{w \in X_2} \delta(w)$$

y

$$\sum_{u \in X_1} \delta(u) = 2k,$$

para algún k en $\mathbb{N}$.

Despejando el término $\sum_{w \in X_2} \delta(w)$, se tiene que

$$\sum_{w \in X_2} \delta(w) = 2\mathbf{m} - 2k = 2(\mathbf{m} - k).$$

La expresión anterior implica que $\sum_{w \in X_2} \delta(w)$ es un número par, puesto que los sumandos son impares y la suma total es par. Se sigue que debe existir un número par de sumandos. Por lo tanto, $|X_2|$ es par. ■

Definición 1.6 *Sea G una gráfica. Una gráfica G' es subgráfica de G si*

1. $V(G') \subseteq V(G)$
2. $E(G') \subseteq E(G)$.

Definición 1.7 *Sea $A \subseteq V(G)$. Una subgráfica inducida por A se obtiene al considerar todos los vértices en A y a las aristas de G que son incidentes con ellos. A tal subgráfica se le denota por $G[A]$.*

Definición 1.8 *Dos gráficas G y H son isomorfas si existe una función biyectiva $\varphi : V(G) \to V(H)$ que preserva las adyacencias, es decir*

$$uv \in E(G) \text{ si y solo si } \varphi(u)\varphi(v) \in E(H).$$

A la función φ se le conoce como isomorfismo.

Ejemplo 1.2 En la Figura 1.2 se muestra un ejemplo de un isomorfismo entre dos gráficas. La Gráfica G consiste de los vértices U, V, W, Z y la gráfica H consta de los vértices A, B, C, D. El isomorfismo consta de la función $\varphi : V(G) \to V(H)$ dado por $\varphi(U) = B, \varphi(V) = A, \varphi(W) = C, \varphi(Z) = D$. Puede observarse que conserva las adyacencias ya que $UV \in E(G), \varphi(U)\varphi(V) = BA \in E(H)$; $VW \in E(G), \varphi(V)\varphi(W) = AC \in E(H)$; $WZ \in E(G), \varphi(W)\varphi(Z) = CD \in E(H)$.

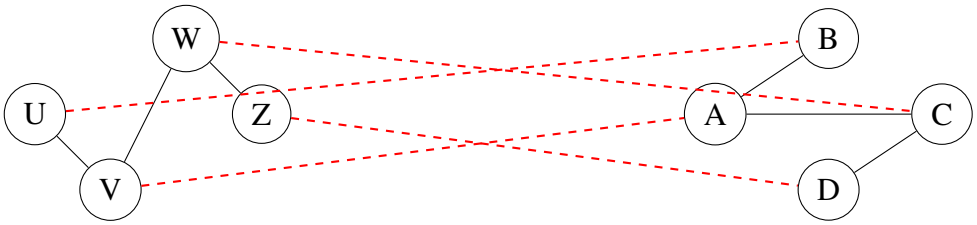


Figura 1.2: Representación geométrica de un isomorfismo entre dos gráficas.

1.2. Caminos y trayectorias

En teoría de gráficas, los caminos y trayectorias son conceptos fundamentales que permiten analizar cómo se conectan los nodos. Los caminos y trayectorias son esenciales para comprender y resolver problemas complejos en sistemas interconectados como la optimización de trayectorias.

Definición 1.9 *Dada una gráfica G, un camino es una sucesión de vértices $(v_1, v_2, \ldots, v_l)$, en donde $v_i v_{i+1} \in E(G)$ para todo $i \in \{1, 2, \ldots, l-1\}$. Si $v_1 = v_l$ entonces el camino es cerrado. En un camino algunos vértices pueden aparecer repetidos en la sucesión.*

Ejemplo 1.3 Considere una gráfica G con $V(G) = \{v_1, v_2, v_3\}$, $E(G) = \{v_1v_2, v_2v_3\}$. Las siguientes sucesiones son ejemplos de caminos, (v_1, v_2, v_3), (v_1, v_2, v_1, v_2), (v_1). Se observa que se pueden repetir vértices, o inclusive se puede tener un camino que solo contiene un vértice. •

Definición 1.10 *Un paseo es un camino que no repite aristas. Una trayectoria es un camino que no repite vértices.*

Un paseo al igual que una trayectoria es un camino. Una trayectoria al no repetir vértices implica que no repite aristas, por lo cual, toda trayectoria es un paseo.
Si el camino, paseo o trayectoria empieza en u y termina en v se dice que es un camino uv, un paseo uv, una trayectoria uv, según sea el caso.

Teorema 1.2 *Si u y v son dos vértices distintos de G, entonces todo camino uv contiene una trayectoria uv.*

Demostración: Sea $C = (u = x_0, x_1, \ldots, x_k = v)$ un camino uv, si C no repite vértices se sigue el resultado. Suponiendo que C contiene al menos un vértice que se repite. Sea x_i el vértice que se repite, es decir existe para $i \neq j$ con $0 \leq i < j \leq k$, $x_i = x_j$. Para no tener vértices repetidos se ajusta el camino quitando el segmento $x_{i+1}, \ldots, x_j$. De esta forma se obtiene un nuevo camino uv que repite menos vértices. Este proceso se repite cuantas veces sea necesario para que el camino uv no repita vértices, y así obtener una trayectoria uv. ■

Definición 1.11 *Un ciclo es una camino cerrado en el que todos los vértices son distintos a excepción del vértice inicial y el vértice final y tiene longitud al menos 3.*

Definición 1.12 *La longitud de un camino C es el número de aristas que tiene dicho camino. De forma análoga se define la longitud de un paseo y la longitud de una trayectoria, y la longitud de un ciclo.*

Definición 1.13 *La longitud del ciclo más pequeño de la gráfica se llama cuello. Si la gráfica no contiene ciclos, se dice que la gráfica es acíclica, y que su cuello es infinito.*

Una vez conocido el concepto de longitud se puede introducir la definición de distancia.

Definición 1.14 *La distancia entre dos vértices u y v, es la trayectoria de menor longitud. Si dicha trayectoria no existe entonces la distancia es infinito. Dicha distancia entre u y v se denota como* $d(u,v)$.

Esta distancia $d(u,v)$, es una distancia en el sentido matemático y se demuestra en el siguiente teorema.

Teorema 1.3 *Dada una gráfica G, se cumplen las siguientes propiedades:*

1. $d(u,v) \geq 0$*, para todo* u,v *en* $V(G)$*. Además* $d(u,v) = 0$ *si y solo si* $u = v$.
2. $d(u,v) = d(v,u)$*, para todo* u,v *en* $V(G)$.
3. $d(u,w) + d(w,v) \geq d(u,v)$*, para todo* $u,v,w \in V(G)$.

Demostración: La primera propiedad se sigue de la definición de distancia entre vértices, ya que esta siempre es mayor o igual a cero y solo es igual a cero si una trayectoria tiene longitud cero, esto es, si el vértice inicial es igual al vértice final.
Por definición de $d(u,v)$, se tiene la trayectoria de menor longitud entre u y v, que también es la mínima trayectoria entre v y u, de lo cual se cumple la segunda propiedad.
Consideremos la trayectoria de menor longitud entre u y w llamándole t_1 y la trayectoria de menor longitud entre w y v llamándole t_2. Consideremos la trayectoria que resulta de concatenar las trayectorias t_1, t_2, esta es una trayectoria entre u, v. Por definición de $d(u,v)$, la trayectoria t_1, t_2 es mayor o igual que $d(u,v)$, de lo cual, $d(u,w)+d(w,v) \geq d(u,v)$ y se cumple la propiedad 3. ■

Definición 1.15 *El diámetro de una gráfica es la distancia máxima entre cualquier par de vértices de la gráfica.*

1.3. Gráficas conexas

Las gráficas conexas son de gran importancia ya que representan sistemas o redes completamente conectadas. Las gráficas conexas son ampliamente utilizadas en diversas aplicaciones, como redes de comunicación, sistemas de transporte, análisis de redes sociales, distribución de energía, entre otros.

Definición 1.16 *Una gŕafica G es conexa si para todo par de vértices u y v de G, existe una trayectoria uv. En otro caso se dice que G es inconexa.*

Lema 1.1 *Toda gráfica conexa tiene al menos $n-1$ aristas.*

Puede consultar la demostración del lema anterior en [8].

Definición 1.17 *Si H es una subgráfica de G, decimos que H es una componente conexa de G si H no está contenida en ninguna subgráfica conexa de G con más vértices o más aristas que H.*

Las gráficas son susceptibles a operaciones como eliminar un vértice o arista produciendo nuevas gráficas.

Definición 1.18 *Si $G = (V,E)$ es una gráfica, entonces la gráfica $G \setminus \{e\} = (V, E \setminus \{e\})$, donde $E \setminus \{e\}$ representa la exclusión de la arista e del conjunto de aristas original.*

Definición 1.19 *Si $G = (V,E)$ es una gráfica, entonces la gráfica $G \setminus \{v\} = (V \setminus \{v\}, E \setminus \{(vu) \mid u \in V\})$, donde $V \setminus \{v\}$ representa la exclusión del vértice v del conjunto de vértices original, y $E \setminus \{(vu) \mid u \in V\}$ implica la eliminación de todas las aristas adyacentes a v.*

Una vez se cuenta con estos conceptos se pueden dar las siguientes dos definiciones.

Definición 1.20 *Dada una gráfica conexa G, se dice que $e \in E(G)$ es un puente si la gráfica $G \setminus \{e\}$ no es conexa.*

Teorema 1.4 *Sean G una gráfica conexa. La arista e es un puente de G si y solo si e no pertenece a ningún ciclo de G.*

Puede consultar la demostración del teorema anterior en [8].

Definición 1.21 *Dada una gráfica conexa G, se dice que $v \in V(G)$ es un vértice de corte si la gráfica $G \setminus \{v\}$ no es conexa.*

1.4. Gráficas bipartitas y apareamientos

Las graficas bipartitas proporcionan una forma efectiva de modelar y representar relaciones entre dos conjuntos de elementos. Son ampliamente utilizados en problemas de asignación, como asignar parejas en citas o designar tareas a trabajadores.

Definición 1.22 *Una gráfica G es bipartita si existen dos conjuntos A y B, subconjuntos de los vértices de G, tal que:*

1. $A \cup B = V(G)$.
2. $A \cap B = \emptyset$.
3. *Si $uv \in E(G)$, entonces $u \in A$ y $v \in B$.*

Es importante señalar que los conjuntos A y B no son necesariamente únicos y que algún conjunto A ó B puede ser vacío. Debido a esto se tiene que una gráfica que tiene un solo vértice es bipartita.

De la condición 3 se sigue que en las gráficas inducidas por los conjuntos A y B no existen aristas.

Teorema 1.5 *Una gráfica G es bipartita si y solo si G no contiene ciclos de longitud impar.*

Demostración:
$\Rightarrow]$
Sea G una gráfica bipartita, con A y B subconjuntos de $V(G)$ tal que cumplen con la Definición 1.22. Sea $C = (v_1, v_2, \ldots, v_k, v_1)$ un ciclo en G. Sin pérdida de generalidad $v_1 \in A$, de lo cual $v_2 \in B$, y así sucesivamente. Note que $v_k \in B$, puesto que $v_1 v_k$ es una arista. El vértice v_j $j \in \{1, 2, \ldots, k\}$ pertenecerá a B si y solo si j es par. Por lo tanto, k es un número par, y entonces la longitud del ciclo es par.

$\Leftarrow]$
Sea G una gráfica que no tiene ciclos de longitud impar. La gráfica G será bipartita si y solo si cada componente conexa lo es, entonces consideraremos a G conexa. Sea u un vértice de G. Se define los conjuntos A y B de la siguiente forma:

$$A = \{v \in V(G) : d(u,v) \text{ es impar.}\}, \qquad B = V(G) \setminus A.$$

Dados los conjuntos A y B no puede existir una arista e entre elementos ya sea de A o B debido a que si existiera e, con $e = wz$ se tendría un ciclo de longitud impar, ya que:

a) Si $w \in A$ y $z \in A$ entonces consideremos el camino cerrado $a = (u, \ldots, w, z, \ldots u)$ donde $(u, \ldots, w)$ y $(z, \ldots, u)$ tienen distancia mínima. Note que al considerar dos caminos con longitud impar y una arista el camino cerrado a tiene longitud impar.

Del camino cerrado a eliminamos las aristas que estén repetidas, es decir las aristas que comparten los caminos $(u, \ldots, w)$ y $(z, \ldots, u)$. Al eliminar las aristas compartidas del camino a se tienen $n - 2k$ aristas, donde n es el número de aristas en el camino a, y k es el número de aristas compartidas, $0 \leq k < n$.

Observe que al eliminar aristas compartidas de este camino cerrado se forman ciclos.

Teniendo en cuenta que n es impar, entonces $n-2k$ es impar, por lo cual al menos uno de los ciclos que se generaron tendrá longitud impar, contradiciendo la hipótesis.

b) Si w pertenece al conjunto B y z al conjunto B, consideremos el ciclo $(u,\ldots,w,u)$, donde $(u,\ldots,w)$ tiene la distancia mínima. Dado que la distancia $d(u,w)$ es par, se genera un ciclo de longitud impar, lo que conduce a una contradicción.

Note que $A\cap B=\emptyset$ y que $A\cup B$ es el conjunto de todos los vértices en la componente conexa.

De lo anterior se puede comprobar que A y B cumplen con la Definición 1.22, por lo tanto, G es bipartita. ■

Definición 1.23 *Dado una gráfica $G=(V,E)$, una descomposición de vértices es una partición $\mathcal{V}=\{V_1,V_2,\ldots,V_k\}$ de los vértices de G en conjuntos disjuntos, es decir, $V_i\cap V_j=\emptyset$ para $i\neq j$, tal que para cada par de vértices distintos u y v en V_i, la arista uv no está en E.*

Definición 1.24 *Se dice que una gráfica bipartita G con descomposición A,B es completa si todo vértice en A es adyacente a todo vértice en B. Si $|A|=n$ y $|B|=m$ se denota a G como $K_{n,m}$.*

En teoría de gráficas, los apareamientos se refieren a conjuntos de aristas en una gráfica que no comparten vértices en común. Es decir, en un apareamiento, ningún par de aristas comparte un extremo. Los apareamientos pueden ser de interés en diversos contextos, como en la asignación de tareas o la resolución de problemas de emparejamiento, como el emparejamiento máximo o el emparejamiento perfecto.

Definición 1.25 *Sea G una gráfica, un apareamiento M en G es un conjunto de aristas no adyacentes entre sí. Un vértice está apareado si es incidente con una arista en el apareamiento. Un emparejamiento perfecto es aquel que cubre todos los vértices de la gráfica.*

Ejemplo 1.4 En la Figura 1.3 se muestra una gráfica que tiene seis vértices etiquetados como A,B,C,D,E y F. Las aristas se dibujan entre los vértices correspondientes. El emparejamiento perfecto se muestra en color rojo y consiste en las aristas AD,BE y CF. Este emparejamiento se denota como $(A-D,B-E,C-F)$.

Este emparejamiento es perfecto porque cada uno de los vértices de la gráfica está cubierto por una arista del apareamiento, es decir, no hay vértices sin aparear.

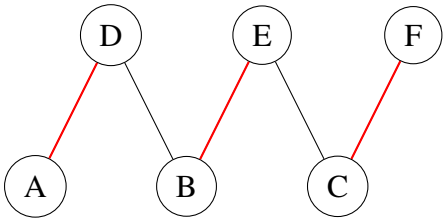


Figura 1.3: Apareamiento perfecto en color rojo, (A-D, B-E, C-F).

1.5. Gráficas y matrices

Existen varias formas de representar una gráfica, anteriormente se mencionó la representación geométrica. Una nueva forma de representar gráficas es a través de matrices, lo cual es muy útil ya que permite establecer una conexión entre la teoría de gráficas y el álgebra lineal.

Matriz de adyacencia

En la matriz de adyacencia cada renglón y columna corresponden a un vértice. Esta matriz refleja la conectividad entre los vértices de la gráfica. Si en una entrada de la matriz aparece un "1", significa que hay una arista que conecta los vértices correspondientes del renglón y de la columna. Si aparece un 0, significa que no hay arista entre estos vértices. Lo anterior se plasma en la siguiente definición.

Definición 1.26 *Sea $G = (V,E)$ una gráfica con $V = \{v_1, v_2, \ldots, v_n\}$. La matriz de adyacencia asociada a la gráfica G es una matriz cuadrada de $n \times n$, denotada por $A(G)$, en donde el elemento (i, j) es*

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

Ejemplo 1.5 Sea G una gráfica con $V = \{A, B, C\}$ y $E = \{AC, AB, CB\}$. La matriz de adyacencia de G es la matriz $A(G)$.

$$A(G) = \begin{array}{c} \\ \text{A} \\ \text{B} \\ \text{C} \end{array} \begin{array}{c} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}.$$

En la Figura 1.4 se muestra la representación geométrica de G.

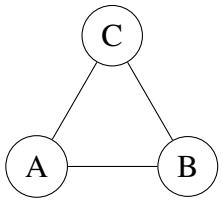


Figura 1.4: Gráfica asociada a la matriz de adyacencia $A(G)$.

•

La matriz de adyacencia es simétrica, lo que significa que cuando hay una arista entre los vértices i y j, esto se refleja en la matriz con valores en las posiciones a_{ij} y a_{ji}.

Esta matriz se puede utilizar para contar el número total de aristas de un vértice, sumando todos los valores del renglón o columna correspondiente a este vértice. De igual forma se puede obtener el número total de aristas de toda la gráfica sumando sobre todos los renglones, o en su caso sobre todas las columnas. Estas son algunas de las utilidades de este tipo de gráfica, en el Capítulo 2 se verá que esta matriz es útil para representar una gráfica como estructura de dato.

Teorema 1.6 *Si G una gráfica y su matriz de adyacencia es $A(G)$, entonces el número de caminos, de longitud ℓ entre los vértices v_i y v_j es la entrada a_{ij} de la matriz $(A(G))^\ell$, donde $(A(G))^\ell$ representa la matriz de adyacencia de G multiplicada ℓ veces.*

Para demostrar el teorema anterior se requieren herramientas de álgebra lineal, y se recomienda consultar [2]. La utilidad de dicho teorema en este trabajo es mostrar la fuerte conexión entre la teoría de gráficas y el álgebra lineal. Una consecuencia inmediata es el siguiente corolario.

Corolario 1.2 *Sea G una gráfica conexa con matriz de adyacencia $A(G)$. La distancia entre dos vértices v_i y v_j de G es el menor entero ℓ tal que la entrada a_{ij} de la matriz $(A(G))^\ell$ es diferente de cero.*

Matriz de incidencia

La matriz de incidencia se puede entender como una matriz donde cada renglón corresponde a un vértice y en ese renglón aparece un "1" en aquellas aristas que involucran dicho vértice.

Definición 1.27 *Sea G una gráfica con $V(G) = \{v_1, v_2, \ldots, v_m\}$ y $E(G) = \{e_1, e_2, \ldots, e_n\}$. La matriz de incidencia $C(G)$ es una matriz de $n \times m$, en donde la entrada (i, j) está dada por:*

$$c_{ij} = \begin{cases} 1 & \text{si } v_i \text{ incide con la arista } e_j. \\ 0 & \text{en otro caso.} \end{cases}$$

Ejemplo 1.6 Sea G una gráfica con $V = \{A, B, C\}$, $E = \{e_1 = AC, e_2 = AB, e_3 = CB\}$. La matriz de incidencia de G es la matriz

$$C(G) = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \end{array}.$$

•

La matriz de incidencia ocupa un espacio proporcional a $n \times m$, con n el número de vértices y m el número de aristas. Este espacio puede ser más eficiente que el espacio utilizado por la matriz de adyacencia en graficas grandes o dispersas, ya que solo se almacenan las conexiones entre vértices y aristas.

El grado de un vértice se puede determinar contando el número de aristas incidentes a ese vértice en la matriz de incidencia. Esto se logra sumando los elementos del renglón correspondiente al vértice. Igualmente, algunos algoritmos utilizan la información contenida en la matriz de incidencia para bajar sus tiempos de ejecución.

1.6. Árboles

Los árboles son esenciales en teoría de gráficas debido a su estructura, propiedades y su amplia aplicabilidad en diversas áreas. Proporcionan una forma efectiva de organizar y representar datos, relaciones entre elementos, y permiten una manipulación y búsqueda eficiente de información.

Definición 1.28 *Un árbol es una gráfica conexa y sin ciclos.*

Ejemplo 1.7 En la Figura 1.5 se muestra el ejemplo de un árbol T donde $V(T) = \{a,b,c,d\}$ y $E(G) = \{ab,ac,ad\}$. La notación T hace referencia al término en inglés *tree* que significa árbol.

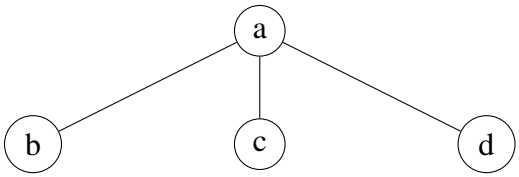


Figura 1.5: Representación geométrica de un árbol T.

Definición 1.29 *Un bosque es una gráfica sin ciclos. A las gráficas sin ciclos se les llama acíclicas.*

Se puede notar que todo árbol es un bosque y que las componentes conexas de un bosque son árboles.

Teorema 1.7 *Si G es un árbol no trivial (|V(G)|>1), entonces G tiene al menos dos vértices de grado uno.*

Demostración: Sea $P = (v_1, \ldots, v_k)$ una trayectoria de longitud máxima. Comprobaremos que los vértices v_1 y v_k son de grado 1.

Suponiendo que el grado de v_k es diferente de 1, entonces existe un vértice w distinto de v_{k-1} tal que $v_k w$ está en $E(G)$.

Si $w \notin \{v_1, \ldots, v_k\}$, entonces $P \cup (v_k, w)$ es una trayectoria de longitud $k+1$, contradiciendo así que P sea de longitud máxima.
Si $w \in V(P)$, entonces $w = v_i$ para algún $i \in \{1, 2, \ldots, k-2\}$ entonces $(w = v_i, v_{i+1}, \ldots, v_k) \cup (v_k, w)$, es un ciclo en G, lo cual no es posible ya que G es un árbol. Por lo tanto, v_k tiene grado 1. De forma análoga se demuestra que v_1 tiene grado 1. ■

Definición 1.30 *A los vértices de un árbol que tienen grado 1 se les llama hojas.*

Existen diversas formas de caracterizar árboles. A continuación se presentan algunas de ellas.

Teorema 1.8 *Sea T un árbol, de orden $\boldsymbol{n}$ y tamaño $\boldsymbol{m}$. Los siguientes enunciados son equivalentes :*

1. *T es un árbol.*
2. *T es conexo y $\boldsymbol{m} = \boldsymbol{n} - 1$.*
3. *T no contiene ciclos y $\boldsymbol{m} = \boldsymbol{n} - 1$.*
4. *Existe una única trayectoria que une a cualesquiera dos vértices de T.*

Demostración:
(1) $\Rightarrow$ (2)]
Por inducción. Caso base.
Si $\mathbf{n} = 1$ se tiene la gráfica trivial, y se cumple que es conexo y $\mathbf{m} = 0 = 1 - 1 = \mathbf{n} - 1$.
Hipótesis de inducción.
Suponga cierto el resultado para todo árbol con k-vértices, con $k \in \mathbb{N}$.
Paso inductivo.
Sea $e = uv$ con $u, v \in V(T)$, $e \in E(T)$. Considere $T \setminus \{e\}$ la cual es una gráfica inconexa ya que si fuera conexa implicaría que existe una trayectoria uv en $T \setminus \{e\}$ y si la unimos con e, obtenemos un ciclo, lo cual es una contradicción. Por lo tanto, $T \setminus \{e\}$ es inconexa, y de hecho tiene 2 componentes conexas pues T es conexa y solo se quitó una arista. Sean T_1, T_2 dichas componentes conexas. Aplicando la hipótesis de inducción se tiene

$$|E(T_1)| = |V(T_1)| - 1,$$
$$|E(T_2)| = |V(T_2)| - 1.$$

Note que $E(T) = E(T_1) \cup E(T_2) \cup \{e\}$, por lo cual,

$$\begin{aligned} |E(T)| =& |E(T_1)| + |E(T_2)| + 1 \\ =& |V(T_1)| - 1 + |V(T_2)| - 1 + 1 \\ =& |V(T_1)| + |V(T_2)| - 1 \\ =& \mathbf{n} - 1. \end{aligned}$$

(2) $\Rightarrow$ (3)]
Por contradicción, consideremos que T tiene un ciclo y llamémoslo C. Al eliminar una arista de dicho ciclo C, la gráfica resultante aún es conexa por el Teorema 1.4. No obstante, la nueva gráfica tiene $n-2$ aristas por el Teorema 1.1. Sin embargo, esto genera una contradicción. En consecuencia, se concluye que T no puede contener ciclos, y por lo tanto $m = n-1$.

(3) $\Rightarrow$ (4)]
Por contradicción. Si existe más de una trayectoria entre dos vértices, $T_1 = \{v_1, \ldots v_k\}$, $T_2 = \{v_1, \ldots v_k\}$ con $T_1 \neq T_2$, entonces la unión de T_1 con T_2 aún cuando compartan vértices forma un ciclo, contradiciendo una hipótesis.
Si no existe ninguna trayectoria entre algún par de vértices entonces T no es conexa, lo que implica que al menos hay k componentes conexas, $k > 1$, pero T no tiene ciclos, por lo cual cada componente conexo es un árbol. Entonces $\mathbf{n} - 1 = \mathbf{m} = \sum_{i=1}^{k} \mathbf{m}_{T_i} = \sum_{i=1}^{k} (\mathbf{n}_{T_i} - 1) = \mathbf{n} - k$, donde cada T_i es una componente conexa de T y $\mathbf{n}_{T_i}, \mathbf{m}_{T_i}$ son el orden y tamaño de T_i, respectivamente. De lo anterior, $k = 1$. Teniendo así una contradicción. Por lo tanto, existe una única trayectoria que une a cualesquiera dos vértices de T.

(4) $\Rightarrow$ (1)]
Del hecho de que exista una trayectoria entre cualquiera dos vértices de T se sigue que la gráfica es conexa, y de que sea única se sigue que no tiene ciclos, ya que si tuviera, implicaría que existen por lo menos dos trayectorias diferentes entre un par de vértices. De lo anterior se concluye que T es un árbol. ■

Árboles generadores

Definición 1.31 *Un árbol generador de una gráfica G es una subgráfica acíclica y conexa que contiene todos los vértices de G.*

Se puede notar que el árbol generador no necesariamente es único.

Teorema 1.9 *Toda gráfica conexa contiene al menos un árbol generador.*

Demostración: Si G no contiene ciclos, entonces G es un árbol y este es generador.
Si G contiene algún ciclo $c = (v_1, \ldots, v_k, v_1)$ considere T_1 como la gráfica G sin la arista $v_k v_1$, se sigue teniendo que para cada par de vértices existe una trayectoria, además $V(T_1) = V(G)$. Repitiendo este proceso para cada ciclo existente, se llega a un punto donde para alguna $i \in \mathbb{N}$, la gráfica T_i no tiene ciclos y es conexa, por lo tanto, T_i es un árbol, tal que $V(T_i) = V(C)$. De lo anterior se concluye que la gráfica G tiene un árbol generador.

■

Árboles con costos

Definición 1.32 *Se dice que una gráfica G es ponderada si a cada arista $e \in E(G)$ se le asocia un número $c(e)$ al cual se le conoce como costo o peso de la arista*

Definición 1.33 *Un árbol generador de peso mínimo es aquel árbol generador donde la suma de los pesos de las aristas es la más pequeña posible. A un árbol generador de peso mínimo se le conoce como árbol óptimo.*

El problema de encontrar el árbol óptimo en una gráfica se puede resolver utilizando el Algoritmo de Kruskal o el Algoritmo de Prim, dos de los algoritmos más conocidos en teoría de gráficas. Se pueden consultar estos resultado en [6] ó [8].

Definición 1.34 *Sea T un árbol tal que para $v \in V(T)$ se tiene asociado un costo o peso no negativo $c(v)$, El costo del árbol denotado por* $\text{Costo}(T)$ *está dado por*

$$\text{Costo}(T) = \sum_{v \in V} c(v).$$

Ejemplo 1.8 Considere el árbol que se tiene en la Figura 1.6, los costos de los vértices son: cinco, dos, tres, dos, uno, cero, uno, cero y uno. El costo total del árbol es quince. Observe que se permiten costos igual a cero y que dos vértices pueden tener el mismo costo.

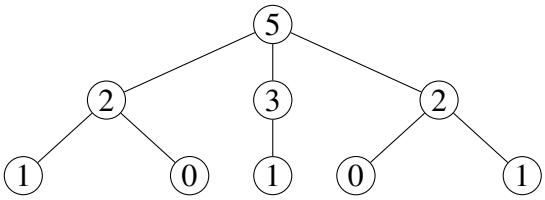


Figura 1.6: Árbol con costo total igual a 15.

Muchos problemas de conteo se pueden reformular en términos de calcular el costo de árboles. En el Capítulo 5 se mostrarán ejemplos y se resolverán problemas de costos de árboles.

1.7. Gráficas dirigidas

Se puede entender una gráfica dirigida o también llamada digráfica, como una gráfica en la que las aristas tienen direcciones. En las gráficas dirigidas si se tiene una arista que conecte al vértice u con el vértice v, no necesariamente existirá una arista que conecte v con u. Esto se expresa mejor en la siguiente definición.

Definición 1.35 *Una gráfica dirigida consiste en un conjunto finito y no vacío V de vértices, y un conjunto E de pares ordenados de dos elementos contenidos en V llamadas aristas.*

Ejemplo 1.9 Sea G una digráfica con $V(G) = \{a,b,c\}$, y $E(G) = \{(a,b),(b,c),(c,a)\}$. La representación geométrica de esta gráfica se puede ver en la Figura 1.7. Usualmente la aristas se denotan igual que como en una gráfica simple, es decir la arista (a,b) se denota como ab, debido a que esto puede conllevar a confusiones otra notación que se suele ocupar para denotar la arista (a,b) es $\overrightarrow{ab}$.

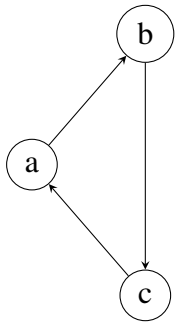


Figura 1.7: Ejemplo de una gráfica dirigida.

La mayoría de las definiciones anteriormente mencionadas para gráficas simples son aplicables a gráficas dirigidas. Para profundizar en el tema se recomienda consultar [1].

El concepto de gráfica dirigida tiene un papel muy importante ya que existen múltiples situaciones que se pueden modelar con esta estructura, y además se preservan muchas características de una gráfica simple.

Glosario

A continuación se presentan los conceptos clave explorados en este capítulo sobre teoría de gráficas. Estos términos fundamentales proporcionan una comprensión sólida de los elementos esenciales de la teoría de gráficas y muchos serán utilizados en los siguientes capítulos en especial cuando se vean las aplicaciones del método de enumeración estocástica.

G	Una gráfica G.
$V(G)$	Vértices de la gráfica G.
$E(G)$	Aristas de la gráfica G.
$N_G(v)$	Vecinos del vértice v.
$N_G(S)$	Vecinos del conjunto S.
$\delta_G(v)$	Grado de un vértice.
$\Delta(G)$	Grado máximo de la gráfica.
$\delta(G)$	Grado mínimo de la gráfica.
$\mathbf{n}_G$	Orden de la gráfica G.
$\mathbf{m}_G$	Tamaño de la gráfica G.
$K_{n,m}$	Gráfica bipartita completa, con descomposición $\|A\| = n$, $\|B\| = m$.
$(A-D, B-C)$	Indica un apareamiento.
$d(u,v)$	Distancia entre vértices.
$G \setminus \{e\}$	Operación que elimina la arista e de la gráfica.
$G \setminus \{v\}$	Operación que elimina el vértice v de la gráfica.
$A(G)$	Matriz de adyacencia asociada a la gráfica G.
$B(G)$	Matriz de incidencia asociada a la gráfica G.

Capítulo 2

Estructura de datos

Uno de los objetivos en la programación de computadoras es hacer códigos eficientes. Se busca, además, crear códigos fáciles de comprender y modificar. Para esto es útil analizar el problema y ver las operaciones que se deben ejecutar, además analizar si hay requerimientos extras. Para todo lo anterior es útil tener una estructura de datos adecuada.

Las estructuras de datos son un tema extenso en el área de las ciencias de la computación, tanto de forma teórica como práctica. Hay libros enteros dedicados sólo al estudio de este tema. Al lector interesado en el tema se le recomienda consultar [5]. En este capítulo se presentan las estructuras de datos necesarias para implementar de manera adecuada los algoritmos de conteo de árboles. El material de este capítulo está basado en [4], [7] y [16].

Definición 2.1 *Un tipo de dato es la especificación del conjunto de valores que puede tomar una variable y de las operaciones que pueden efectuarse sobre estos valores.*

Definición 2.2 *Un tipo de dato abstracto es un modelo con operaciones definidas pero sin especificar su implementación.*

Definición 2.3 *Una estructura de dato es una implementación de un tipo de dato abstracto.*

Es fundamental entender que no existe una estructura de datos sin defectos que sea adecuada para aplicar en todas las situaciones. Dependiendo de los problemas

que se busquen resolver y las prioridades que se tengan conllevará a elegir una estructura sobre otra. Por ejemplo, en este capítulo se muestran diferentes opciones para implementar una gráfica, cada una con sus propias cualidades.

2.1. Complejidad computacional

Una parte fundamental de las ciencias de la computación es el análisis de algoritmos. Un algoritmo es una secuencia de pasos bien definidos y no ambiguos que se utilizan para resolver un problema o llevar a cabo una tarea específica.

Los datos de entrada son la información que se utiliza como punto de partida para un algoritmo. Estos datos pueden ser variables, constantes, estructuras de datos, colecciones o cualquier otro tipo de información necesaria para el algoritmo. Por otro lado, los datos de salida en un algoritmo son los resultados generados después de aplicar el algoritmo a los datos de entrada. Cuando se habla de una entrada de tamaño n, se refiere a la cantidad de elementos que componen los datos de entrada del algoritmo, por ejemplo, puede representar la cantidad de elementos en una lista o arreglo, el número de vértices en una gráfica o la longitud de una cadena de caracteres.

Existen diversas técnicas para resolver un mismo problema, es decir, existen varios algoritmos que se pueden implementar que dan la solución correcta a un problema. Sin embargo, se tiene que tener en cuenta dos factores muy importantes: el tiempo y espacio de ejecución. De poco puede servir resolver un problema si el tiempo en el que se soluciona es demasiado grande. El espacio es otro tema a tener en cuenta, a pesar de que las computadoras cada vez aumentan más su capacidad, no dejan de ser un objeto físico, por lo que un algoritmo que tome mucho espacio puede no ser viable.

Por lo general, existe una negociación entre optimizar espacio y tiempo, es decir, si se mejora el tiempo, se pierde en espacio y viceversa. Sin embargo, dependiendo del problema, se debe enfocar en mejorar más uno que el otro. En la mayoría de los casos, lo que se busca mejorar es el tiempo de ejecución. Esta sección se centrará en el estudio de tiempos de ejecución, aunque como se verá es fácil extender esta noción a espacios de ejecución.

Modelo *RAM*

Un modelo de computadora para estudiar tiempos de ejecución es la así llamada computadora *RAM* por sus siglas en inglés *Random Access Machine*. Este modelo no debe confundirse con la memoria RAM, *Random Access Memory* de una computadora.

En el modelo RAM:

- Cada operación simple toma exactamente una unidad de tiempo. La suma, resta, multiplicación y división sobre números reales, al igual que asignar o comparar variables son consideradas operaciones simples.
- Ciclos y subrutinas no son consideradas operaciones simples. Estas son composiciones de varias operaciones simples. El tiempo que tarda en correr los ciclos o subrutinas depende del número de iteraciones o de la naturaleza propia de la subrutina.
- Cada acceso a memoria toma exactamente una unidad de tiempo. Aun más, se tiene acceso a tanta memoria como se necesite.

Observe que el modelo *RAM* es un concepto teórico. Este modelo no considera los componentes específicos de una computadora ni los tipos de datos que un lenguaje de programación pueda manejar. Esta característica permite enfocarse por completo en el análisis del algoritmo.

Se usará el modelo *RAM* para medir el tiempo de ejecución contando las unidades de tiempo que tarda el algoritmo en ejecutarse. Aunque pueda parecer simple, este modelo es muy útil ya que permite comparar algoritmos de manera precisa.

Mejor, peor y caso promedio

Aunque el modelo *RAM* nos dice cuántas unidades de tiempo tarda en ejecutarse un algoritmo, para entender qué tan bueno o malo es un algoritmo dado, se deben conocer los mejores y peores casos.

Peor caso

El peor caso de un algoritmo se refiere a la situación en la cual el algoritmo requiere la máxima cantidad de tiempo posible para ejecutarse en comparación con cualquier otra instancia del programa con una entrada de tamaño n.

En otras palabras, es la entrada que provoca el rendimiento más deficiente del algoritmo en términos de tiempo.

Mejor caso

El mejo caso de un algoritmo se refiere a la situación en la cual el algoritmo requiere la menor cantidad de tiempo posible para ejecutarse en comparación con cualquier otra instancia del programa con una entrada de tamaño n.

Caso promedio

El caso promedio de un algoritmo se refiere al escenario en el cual se evalúa el rendimiento del algoritmo considerando todas las posibles entradas de tamaño n y calculando el tiempo de ejecución promedio. A diferencia del peor caso, que se enfoca en la entrada que maximiza el tiempo de ejecución, el caso promedio proporciona una visión más realista del rendimiento del algoritmo.

Definición 2.4 *La complejidad temporal de un algoritmo, denotada como $T(n)$, se define como el número de operaciones realizadas por el algoritmo en función del tamaño de los datos de entrada n.*

Notaciones asintóticas

El peor caso generalmente es el más útil para medir la eficacia de un algoritmo, ya que, como su nombre lo indica, nos proporciona un panorama de lo peor que podría pasar.

Uno de los problemas de trabajar contando unidades de tiempo es que se requiere ser demasiado preciso. Contar el número exacto de instrucciones *RAM* puede variar para cada persona debido a pequeñas diferencias, como el considerar o no una asignación. Sin embargo, en términos generales las complejidades deben ser las mismas o al menos muy parecidas.

Las notaciones asintóticas ignoran la diferencia que se produce al multiplicar constantes. Por ejemplo las funciones $f(n) = n$ y $g(n) = 2n$ son idénticas ante las notaciones asintóticas. A continuación se presentan definiciones formales de tres de estas notaciones.

Definición 2.5 *Notaciones asintóticas.*

- $f(n) \in O(g(n))$ *significa que la expresión* $f(n)$ *puede acotarse superiormente por la expresión* $c \cdot g(n)$*, es decir* $f(n) \leq c \cdot g(n)$*, para n suficientemente grande, en donde c es una constante.*
- $f(n) \in \Omega(g(n))$ *significa que la expresión* $f(n)$ *puede acotarse inferiormente por la expresión* $c \cdot g(n)$*, es decir* $f(n) \geq c \cdot g(n)$*, para n suficientemente grande, en donde c es una constante.*
- $f(n) \in \Theta(g(n))$ *significa que la expresión* $f(n)$ *puede acotarse superiormente por la expresión* $c_1 \cdot g(n)$*, e inferiormente por la expresión* $c_2 \cdot g(n)$*, es decir* $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$*, para n suficientemente grande, con* c_1 *y* c_2 *constantes.*

Las notaciones asintóticas son una herramienta para comparar la eficiencia de los algoritmos ya que ignora los detalles y se enfoca en el panorama completo. Existen otras notaciones asintóticas, sin embargo, las tres presentadas anteriormente son las más utilizadas en el cálculo de complejidades. Usualmente se hace el abuso de notación $f(n) = O(g(n))$ que significa $f(n) \in O(g(n))$, con el fin de simplificar este enfoque.

La complejidad temporal nos ayuda a comprender cómo el tiempo de ejecución de un algoritmo aumenta a medida que se procesan conjuntos de datos más grandes. Usualmente, se representa mediante la notación $O(f(n))$, y se dice que el algoritmo tiene una complejidad temporal $f(n)$. Esta notación nos permite estimar cómo crece el tiempo de ejecución en función del tamaño de los datos de entrada, lo que nos ayuda a evaluar la eficiencia del algoritmo para distintos escenarios.

2.2. Pilas y colas

Las pilas y las colas son conjuntos dinámicos, es decir, conjuntos donde la cantidad de elementos puede variar durante la ejecución del algoritmo. Ademas, las operaciones eliminar e insertar un elemento del conjunto están preespecificadas.

Pilas

En una pila, el elemento eliminado del conjunto es el más reciente insertado. Comúnmente esto es llamado *LIFO*, por sus siglas en inglés *Last In, First Out*. En-

tonces, en este conjunto debe existir un orden para que siempre se cumpla esta condición.

Las pilas son fáciles de implementar y muy eficientes. Sus usos son variados y van desde evaluadores de expresiones hasta la administración de trabajos.
Las operaciones insertar y borrar elementos usualmente son llamadas *push* y *pop*.

- $Push(x,s)$: Insertar el elemento x en el tope de la pila s.
- $Pop(s)$: Elimina el elemento en el tope de la pila s. Tiene como dato de salida el elemento eliminado.

Ejemplo 2.1 Considere una pila a la que se le insertan los siguientes números, uno, dos, tres y cuatro. Si se quiere ejecutar la operación eliminar, el elemento eliminado será el número cuatro, ya que fue el último elemento que se insertó. Si se quiere insertar algún elemento, será insertado en el tope de la pila, así este elemento será el primero en ser eliminado si se vuelve a ejecutar la operación eliminar. En la figura 2.1 se muestra una representación visual de este ejemplo.

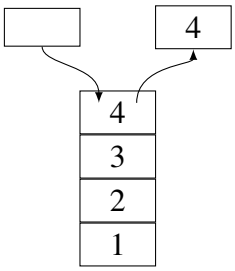


Figura 2.1: Funcionamiento de una pila.

Las pilas pueden soportar más operaciones como comprobar si la pila está vacía, o devolver el último elemento insertado, etc. implementarlas o no depende del contexto donde se este trabajando, Sin embargo, las operaciones *push* y *pop* son indispensables al momento de implementar una pila.

Colas

En una cola, el elemento eliminado del conjunto es el más antiguo insertado. Al igual que en las pilas, en el conjunto debe existir un orden para que siempre se cumpla esta condición.
Las operaciones insertar y borrar elementos usualmente son llamadas *Enqueue* y *Dequeue*.

- *Enqueue*(x,q): Inserta el elemento x en la cola q.
- *Dequeue*(q): Borra el último elemento insertado en la cola q. Tiene como dato de salida el elemento eliminado.

Ejemplo 2.2 Considere una cola a la que se le insertan los siguientes números, uno, dos, tres y cuatro. Si se quiere ejecutar la operación eliminar, el elemento eliminado será el número uno, ya que fue el primer elemento que se insertó. Si se quiere insertar algún elemento, será insertado en el tope de la cola, así para poder eliminar este elemento, se tendrá que eliminar todos los elementos que fueron insertados antes. En la figura 2.2 se muestra una representación visual de este ejemplo.

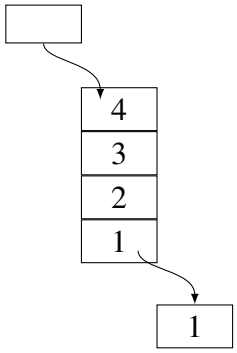


Figura 2.2: Ejemplo del funcionamiento de una cola.

De nueva cuenta, las colas pueden soportar más operaciones pero éstas dependen del problema que se esté resolviendo.

2.3. Gráficas

Hay algoritmos que utilizan gráficas para resolver ciertos problemas. Por lo tanto, es útil tener una manera de representar las características y propiedades de estas gráficas en una estructura de datos. Como se mencionó en este capítulo, la forma de implementación dependerá del problema que esté siendo resuelto.

Implementaciones de una gráfica

Existen diversas formas de implementar una gráfica. A continuación se enlistan algunas de ellas.

Lista de aristas

Se tienen todas las aristas de una gráfica en una lista. Por ejemplo, sea $e_1, e_2, \ldots, e_k$ las aristas de una gráfica G. Entonces la implementación de la gráfica mediante lista de aristas consiste en tener una lista que contenga las aristas $e_1, e_2, \ldots, e_k$. Dependiendo de si es dirigida o no, será necesario o no tener las parejas simétricas. Esta implementación es útil cuando la gráfica tiene relativamente pocas aristas en comparación con el número de vértices, almacenar solo las aristas existentes puede ahorrar mucho espacio en memoria y hacer que las operaciones sean más eficientes.

Matriz de adyacencia

Debido a que en la mayoría de lenguajes de programación es fácil implementar una matriz, se puede recurrir a la Definición 1.5 en el Capítulo 1, donde se establece que cada gráfica tiene asociada una matriz. Un punto importante es que en la mayoría de los lenguajes el consultar el valor de una entrada de la matriz toma tiempo constante. Por lo cual, en esta estructura, el conocer si una arista pertenece a la gráfica toma tiempo constante, es decir, sin importar el tamaño de la matriz esta operación siempre se ejecuta en la mismas unidades de tiempo.

Lista de adyacencias

La implementación consiste en que para cada vértice se tiene una lista de los nodos con los que esté conectado.
Con base en [16], la lista de adyacencias es la implementación que menor complejidad temporal tiene para la mayoría de problemas.

2.4. Árboles

Al igual que pasa con las gráficas, es muy útil tener una forma de representar árboles en la computadora. Como un árbol es una gráfica, podemos utilizar la implementación de gráficas. Sin embargo, en computación los árboles se suelen entender como estructuras donde se distingue a un nodo llamado raíz, y suelen representarse de arriba a abajo empezando por la raíz, sin perder las características que los hacen árboles, es por eso que es útil tener una estructura de datos particular.

Se utilizará la palabra nodo en lugar de vértice cuando se hable de un árbol como estructura de dato.

Árboles binarios

Definición 2.6 *Los árboles binarios son gráficas conexas sin ciclos donde todos los nodos tienen a lo más grado 3.*

La Definición 2.6 es correcta y muestra que, efectivamente, los árboles binarios son un tipo de gráfica, sin embargo, no ayuda mucho a la implementación, por lo cual, se utilizará una nueva definición.

Definición 2.7 *Un nodo de un árbol binario es:*

- *El nodo vacío.*
- *Un nodo padre, un nodo hijo izquierdo o un nodo hijo derecho.*

Para cualquier par de nodos a, b, se tiene que a es el nodo padre de b si y solo si b es el nodo hijo izquierdo o nodo hijo derecho de a.

Definición 2.8 *Los árboles binarios cuentan con un nodo especial que se llama raíz y cumplen que:*

- *El árbol no tiene elementos si y solo si el nodo raíz es vacío.*
- *Si la raíz es distinta del vacío, entonces no tiene padre.*
- *Todo elemento del árbol está en un nodo v tal que existe una secuencia única de nodos $v_0, v_1, \ldots, v_k = v$, donde v_0 es la raíz del árbol y v_i es el nodo padre de v_{i+1} con $i = \{1, \ldots, k-1\}$.*

Definición 2.9 *Un nodo se considera una hoja en un árbol si no tiene nodos hijos.*

En la Figura 2.3 se muestra un ejemplo de un árbol binario. El elemento A es el nodo raíz, el elemento B es el nodo hijo izquierdo de A mientras que el elemento C es el nodo hijo derecho de A, y tiene como nodo hijo izquierdo al elemento F. Los nodos D, E y F son hojas del árbol.

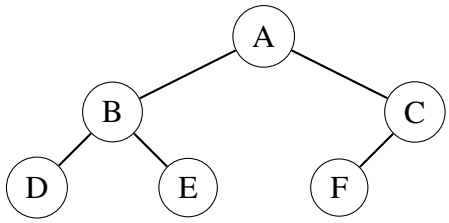


Figura 2.3: Ejemplo de un árbol binario.

Definición 2.10 *Un subárbol con raíz en v de un árbol T es un árbol formado por el nodo v de T y todos sus descendientes en T.*

La definición anterior enfatiza que todo lo que es descendiente de un nodo es un árbol también, y es un subconjunto de un árbol más grande. El subárbol con raíz en v se denota por T_v. Esta definición es válida para cualquier tipo de árbol que cuente con un nodo raíz.

A continuación, se presentan conceptos básicos sobre árboles como estructura de datos. Estos conceptos serán útiles más adelante cuando se aborde la teoría y las aplicaciones del tema central de la tesis.

Definición 2.11 *Sea v un nodo de un árbol binario. La altura del nodo se denota por $h(v)$ y se define de la siguiente manera recursiva:*

- *Si v es una hoja, $h(v) := 0$.*
- *Si v_i y v_d son los hijos de v, entonces $h(v) := 1 + \max\{h(v_i), h(v_d)\}$.*

La altura del árbol es la altura de la raíz.

Definición 2.12 *Sea v un nodo de un árbol binario. La profundidad del nodo se denota por $d(v)$ y se define de la siguiente manera recursiva:*

- *Si v es la raíz, entonces $d(v) := 0$.*

- *Si p es el padre de v, entonces* $d(v) := 1 + d(p)$.

Definición 2.13 *El i-ésimo nivel de profundidad en un árbol binario T, donde* $0 \leq i \leq h(T)$ *son todos los nodos tales que* $d(v) = i$*. La profundidad de un árbol es la cantidad de niveles de profundidad que éste posee.*

Si se está hablando de un árbol, usualmente, se utiliza el término, nivel *i*, para referirse al nivel de profundidad *i* del árbol.

Árboles con *k* descendientes

Una generalización de los árboles binarios son los árboles con *k* descendientes, donde cada nodo tiene a lo más *k* subárboles en lugar de dos, esto se plasma en la siguiente definición.

Definición 2.14 *Un árbol con k descendientes es un árbol con una raíz donde cada nodo no tiene más de k hijos.*

Las propiedades se extienden directamente de las de un árbol binario. Se añaden definiciones propias de árboles con *k* descendientes, éstas serán útiles en próximos algoritmos que se presentarán más adelante.

Definición 2.15 *Un árbol con k descendientes completo es un tipo especial de árbol en el que cada nodo, excepto los del penúltimo nivel, tiene exactamente k hijos (está completo). En el último nivel, los nodos pueden tener menos de k hijos, pero se deben colocar lo más a la izquierda posible.*

Definición 2.16 *Un árbol con k descendientes perfecto es un árbol con k descendientes en donde la profundidad de las hojas es la misma.*

Recorridos

Un recorrido se usa para explorar todos los nodos de un árbol, existen varias formas de hacerlo, pero debe cumplir que todos los nodos deben ser visitados solo una vez. A continuación se presentan dos de los recorridos más conocidos.

Algoritmo BFS

El algoritmo llamado búsqueda por amplitud, o *BFS* por sus siglas en inglés *Breadth-First Search*, es un recorrido donde los nodos se visitan por niveles. En el Algoritmo 2.4 se proporciona el pseudocódigo del algoritmo BFS.

Algoritmo 1: Búsqueda por amplitud

```
    Entrada Un árbol T con raíz u.
    Salida Lista V de todos los nodos del árbol.
V ← () ;                          //Inicializar la lista de nodos
Q ← (u) ;                                  //Inicializar la cola
mientras |Q| > 0 hacer
    v ← Q(1) ;              //Tomar el último elemento de la cola
    Q ← Q(2 : end) ;  //Eliminar el último elemento de la cola
    si v ∉ V entonces
        V ← (V, v) ;  //Agregar v a la lista de nodos visitados
        para w ∈ N(v) hacer
            Q ← (Q, w) ;  //Agregar todos los vecinos de v a la
             cla
devolver V
```

Figura 2.4: Algoritmo BFS.

Ejemplo 2.3 En la Figura 2.5 se proporciona un ejemplo de la manera en la que el algoritmo BFS visita lo nodos de un árbol, empezando en el nodo raíz *A*, después visitando los nodos *B, C, D, E, F, G, H, I*.

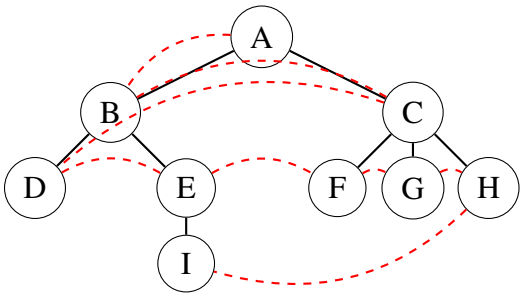


Figura 2.5: Ejemplo de recorrido *BFS*.

Algoritmo DFS

El algoritmo llamado búsqueda por profundidad, o *DFS* por sus siglas en inglés *Depth-First Search*, es un recorrido donde, como su nombre lo indica, se visitan los nodos con base en la profundidad. En la Figura 2.6 se proporciona el pseudo-código del algoritmo DFS.

Algoritmo 2: Búsqueda por profundidad

```
    Entrada Un árbol T con raíz u.
    Salida Lista V de todos los nodos del árbol.
V ← () ;                          //Inicializar la lista de nodos
S ← (u) ;                                  //Inicializar la pila
mientras |S| > 0 hacer
    v ← S(end) ;            //Tomar el último elemento de la pila
    S ← S(1 : end − 1) ;      //Eliminar el último elemento de la
     pila
    si v ∉ V entonces
        V ← (V, v) ;  //Agregar v a la lista de nodos visitados
        para w ∈ N(v) hacer
            S ← (S, w) ;    //Agregar todos los vecinos de v a la
             pila
devolver V
```

Figura 2.6: Algoritmo DFS.

Ejemplo 2.4 En la Figura 2.7 se proporciona un ejemplo de la forma como el algoritmo DFS visita lo nodos del árbol, empezando en el nodo raíz *A*, después visitando los nodos *C, H, G, F, B, E, I, D*.

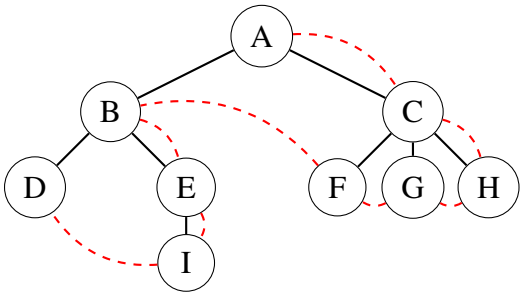


Figura 2.7: Ejemplo de recorrido *DFS*.

Árboles balanceados

Uno de los problemas que surgen en un árbol, es la posibilidad de que cada nodo solo tenga un hijo, de esta forma un árbol se degenera a una lista. Lo ideal sería que todos las hojas del árbol tuvieran la misma profundidad, por lo cual surge una nueva definición.

Definición 2.17 *Un árbol T es k balanceado si y solo si para todo nodo del árbol T se cumple que la distancia máxima del nodo a una de sus hojas es a lo más k veces la distancia mínima del nodo a alguna de sus hojas. Con k una constante fija.*

Ejemplo 2.5 En el árbol que se presenta en la Figura 2.8 es 3 balanceado, ya que la distancia del nodo *A* al nodo *B* es uno, mientras que la distancia del nodo *A* al nodo *I* es tres.

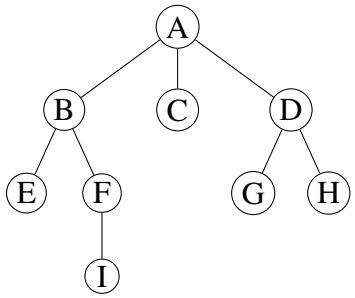


Figura 2.8: Árbol 3 balanceado.

Mientras más se reduzca k, la mayoría de los algoritmos para árboles reducen su complejidad temporal. Los árboles balanceados se suelen ocupar sobretodo en árboles binarios de búsqueda, la cual es otra estructura de árbol. Existen estructuras específicas para balancear árboles binarios de búsqueda, como lo son árboles rojinegros y árboles AVL. Puede consultarse más información en [7].

2.5. Algoritmos aleatorios

La mayoría del diseño de algoritmos se centra en abordar el peor escenario posible, pero en muchos problemas, este peor caso puede tener una complejidad temporal muy alta, llegando a ser exponencial o similar. Por esta razón, se han desarrollado algoritmos aleatorios, donde un mal rendimiento se debe más a una mala suerte con los números aleatorios generados, en lugar de tener datos de entrada adversos. Es decir, en lugar de enfrentar siempre el peor caso, estos algoritmos utilizan aleatoriedad para lograr un mejor rendimiento promedio.

Definición 2.18 *Un algoritmo aleatorio es aquel en el que la salida no depende únicamente de los datos de entrada, sino que también se ve influenciada por la generación de números aleatorios.*

En otras palabras, en lugar de producir siempre el mismo resultado para un conjunto de datos dado, un algoritmo aleatorio puede dar resultados diferentes cada vez que se ejecuta debido a la influencia de los números aleatorios generados durante su funcionamiento. Esto puede hacer que el algoritmo sea más flexible y útil para ciertos problemas.

Definición 2.19 *La exactitud (correctness) de un algoritmo se refiere a su capacidad para obtener la respuesta correcta en todas las situaciones válidas del problema que está diseñado para resolver.*

En resumen, el algoritmo es considerado correcto si siempre entrega la respuesta adecuada cuando se le presentan datos o situaciones válidas para el problema en cuestión.

Definición 2.20 *Un algoritmo eficiente es aquel que resuelve un problema en un tiempo de ejecución razonable y utiliza recursos computacionales de manera óptima.*

Un buen tiempo de ejecución depende del problema que se está resolviendo, pero en general, se considera que un buen tiempo de ejecución es aquel que crece de forma manejable a medida que el tamaño del problema aumenta, es decir, que su complejidad temporal es a lo más polinomial.

Los algoritmos aleatorios se suelen clasificar en dos tipos, estos dependen de la garantía de obtener el resultado correcto y de la eficiencia.

- Algoritmos *Las Vegas*
  Este tipo de algoritmos aleatorios aseguran que siempre el resultado sea correcto, y usualmente pero no siempre sea eficiente. Un ejemplo de esta variabilidad se encuentra en una versión de *quicksort*, un algoritmo de ordenamiento. En este caso, se introduce un reajuste aleatorio en la entrada inicial. A pesar de la naturaleza aleatoria, el algoritmo siempre devuelve la colección ordenada. No obstante, en algunas ocasiones lo hará en complejidad temporal $O(n \log n)$ y en otras en $O(n^2)$.

- Algoritmos *Monte Carlo*
  Los algoritmos Monte Carlo son eficientes, y usualmente pero no siempre proporcionan una respuesta correcta o al menos lo suficientemente cercana a ésta. Un ejemplo de este tipo de algoritmo es *bootstrap* el cual se basa en muestras para estimar una cantidad de interés, sin embargo, en algunas ocasiones la estimación no será adecuada.

Se puede encontrar más información del algoritmo *quicksort* en [7], y del algoritmo *bootstrap* en [10].

Para analizar algoritmos aleatorios es indispensable hacerlo a través de la teoría de la probabilidad. Este enfoque permite establecer los resultados de manera rigurosa.

Capítulo 3

Probabilidad

Se pueden agrupar los experimentos en dos tipos: deterministas y aleatorios. Los experimentos deterministas, como su nombre indica, tienen resultados predefinidos, es decir, desde el principio se conoce el resultado y, sin importar cuántas veces se repitan, siempre se obtendrá el mismo resultado. Por otro lado, los experimentos aleatorios son aquellos en los que no se puede predecir el resultado desde el principio. Un ejemplo típico es lanzar una moneda, donde no se puede prever el resultado.

La probabilidad es una herramienta matemática que nos permite estudiar fenómenos aleatorios. En este capítulo, presentaremos los conceptos básicos de la teoría de probabilidad, los cuales serán fundamentales para analizar algoritmos que involucran aleatoriedad. La información de este capítulo se basa en las referencias [13] y [14], donde se encuentran ampliamente desarrollados estos temas.

Definición 3.1 *El espacio muestral se define como el conjunto de todos los posibles resultados de un experimento y se denota por* Ω*. A un resultado en particular se le denota por* ω*.*

Inicialmente, se entenderá como evento a cualquier subconjunto del espacio muestral; posteriormente, se dará una definición más precisa. La probabilidad de un evento A se denota por $P(A)$.

3.1. Axiomas de probabilidad

A continuación se presentan los axiomas que fueron establecidos por Andrey Nikolaevich Kolmogorov, matemático ruso.

Definición 3.2 ***Axiomas de probabilidad***

1. *Para cualquier evento A,*
$$P(A) \geq 0.$$
2. $P(\Omega) = 1$.
3. *Sean $A_1, A_2 \ldots$ eventos ajenos dos a dos. Entonces*
$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

Un axioma es una proposición o declaración que se acepta como verdadera, sirviendo como punto de partida o fundamento para construir una teoría matemática, en este caso la teoría de la probabilidad.

En los axiomas de probabilidad no se menciona la manera de calcular probabilidades solo las reglas que ésta debe cumplir.

Definición 3.3 *Una medida de probabilidad es cualquier función P definida sobre una colección de eventos que cumple con los axiomas de probabilidad.*

Propiedades

A continuación se presentan algunas propiedades que se derivan de los axiomas de probabilidad.

Proposición 3.1 *Sean A y B dos eventos. Se cumplen las siguientes propiedades.*

1. $P(\emptyset) = 0$.
2. *Sea $A_1, A_2, \ldots, A_n$ una colección finita de eventos ajenos dos a dos. Entonces*
$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i).$$

3. *Sea A^c el complemento del conjunto A. Entonces $P(A^c) = 1 - P(A)$.*
4. *Si $A \subseteq B$ entonces $P(A) \leq P(B)$.*
5. $0 \leq P(A) \leq 1$.
6. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

3.2. Sigma Álgebras

Una sigma álgebra denotada por σ-álgebra, proporciona una estructura precisa para definir lo que constituye un evento y cómo asignarles probabilidades, garantizando consistencia en los cálculos de probabilidad.

Definición 3.4 *Una colección $\mathscr{F}$ de subconjuntos de un espacio muestral es una σ-álgebra si cumple:*

1. $\Omega \in \mathscr{F}$.
2. *Si $A \in \mathscr{F}$ entonces $A^c \in \mathscr{F}$.*
3. *Si $A_1, A_2, \ldots \in \mathscr{F}$ entonces $\bigcup_{k=1}^{\infty} A_k \in \mathscr{F}$.*

A los elementos de $\mathscr{F}$ se les llama eventos.

Es importante destacar que no todos los elementos o subconjuntos del espacio muestral son eventos; solamente aquellos que pertenecen a la σ-álgebra asociada al espacio muestral son llamados eventos.

Con lo anterior se puede definir un espacio de probabilidad que es la estructura que se ocupa para modelar y estudiar fenómenos aleatorios.

Definición 3.5 *Un espacio de probabilidad es una terna $(\Omega, \mathscr{F}, P)$, en donde Ω es un conjunto que representa el espacio muestral, $\mathscr{F}$ es una σ-álgebra de subconjuntos de Ω y P es una medida de probabilidad.*

Sigma álgebra de Borel

Una σ-álgebra importante es la sigma álgebra de Borel, donde se parte de los números reales como espacio muestral. Existen diversas formas de definir esta sigma álgebra, a continuación se presenta una de las más comunes.

Definición 3.6 *La σ-álgebra de Borel de $\mathbb{R}$ se define como la mínima σ-álgebra de subconjuntos de $\mathbb{R}$ que contiene a todos los intervalos de la forma $(-\infty, x]$. La σ-álgebra de Borel se denota como $\mathscr{B}(\mathbb{R})$, es decir,*

$$\mathscr{B}(\mathbb{R}) = \sigma\{(-\infty, x] : x \in \mathbb{R}\}.$$

La definición anterior hace referencia a que si existe $\mathscr{F}$ una σ-álgebra que contiene a la colección $\{(-\infty, x] : x \in \mathbb{R}\}$, entonces $\mathscr{B}(\mathbb{R}) \subseteq \mathscr{F}$. A los elementos en $\mathscr{B}(\mathbb{R})$ se les suele llamar borelianos o conjuntos borel medibles.

3.3. Probabilidad condicional

La probabilidad condicional es una medida de probabilidad que se calcula cuando se tiene información adicional o conocimiento previo sobre un evento. Se puede entender como la probabilidad de que ocurra un evento A, dado que otro evento B ya ha ocurrido.

Definición 3.7 *Sean A y B dos eventos con $P(B) > 0$. La probabilidad de A dado B, denotada por $P(A|B)$, se define como*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Definición 3.8 *Sea Ω el espacio muestral asociado a un experimento aleatorio. Una colección de eventos $\{B_1, B_2, \ldots, B_n\}$ es una partición finita de Ω si cumple:*

1. $B_i \neq \emptyset$, para $i = 1, 2, \ldots n$.
2. $B_i \cap B_j \neq \emptyset$, con $i \neq j$.
3. $\bigcup_{i=1}^{n} B_i = \Omega$.

La probabilidad condicional, en muchos casos, permite realizar cálculos más sencillos en situaciones donde se tiene información adicional sobre el sistema o evento en estudio. Para prueba de esto, están los dos siguientes teoremas.

Teorema 3.1 ***(Teorema de probabilidad total)*** *Si $B_1, B_2, \ldots, B_n$ es una partición de Ω tal que $P(B_i) \neq 0$ con $i = 1, 2, \ldots n$, entonces para cualquier evento A se*

cumple que:

$$P(A) = \sum_{i=1}^{n} P(A|B_i)P(B_i).$$

Si se tiene un evento B tal que $0 < P(B) < 1$. Entonces el teorema de probabilidad total se puede expresar de la siguiente manera:

$$P(A) = P(A|B)P(B) + P(A|B^c)P(B^c).$$

El teorema de probabilidad total es importante porque permite calcular la probabilidad de un evento A, incluso si no conocemos directamente $P(A)$, al descomponer el espacio en eventos más manejables cuyas probabilidades sí conocemos (los B_i). En muchos casos, es más fácil calcular las probabilidades condicionales $P(A|B_i)$ y las probabilidades de los eventos B_i que obtener directamente la probabilidad de A.

Teorema 3.2 ***(Teorema de Bayes)*** *Sea $B_1, B_2, \ldots B_n$ una partición de Ω tal que $P(B_i) \neq 0$ con $i = 1, 2, \ldots n$. Sea A tal que $P(A) > 0$. Se cumple que*

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{i=1}^{n} P(A|B_i)P(B_i)},$$

con $j = 1, 2, \ldots, n$.

Si la partición de Ω consta de dos elementos B y B^c, el teorema de Bayes adquiere la forma

$$P(B|A) = \frac{P(A|B)P(B)}{P(A|B)P(B) + P(A|B^c)P(B^c)}.$$

El teorema de Bayes es especialmente útil cuando se desea actualizar una probabilidad a medida que se obtienen nuevos datos o información. Puede consultarse más detalles en [14].

3.4. Independencia de eventos

La independencia hace referencia a que la ocurrencia de un evento no afecta la probabilidad que ocurra otro evento. En caso contrario se tendría que los eventos dependen el uno del otro.

Definición 3.9 *Sean A,B dos eventos, A y B son independientes si*

$$P(A\cap B)=P(A)P(B).$$

Definición 3.10 *Sean $B_1,B_2,\ldots B_n$ eventos, estos son independientes si se cumplen las siguientes condiciones:*

$$P(B_i\cap B_j)=P(B_i)P(B_j) \text{ con } i\neq j,\quad i,j=1,2,\ldots,n.$$

$$P(B_i\cap B_j\cap B_k)=P(B_i)P(B_j)P(B_k) \text{ con } \ i\neq j\neq k,\quad i,j,k=1,2,\ldots,n.$$

$$\vdots$$

$$P(B_1\cap\cdots\cap B_n)=P(B_i)\cdots P(B_n).$$

Definición 3.11 *Una colección infinita de eventos es independiente si cualquier subconjunto finito de esta colección lo es.*

3.5. Variables aleatorias

A partir de esta sección consideraremos que se tiene un espacio de probabilidad $(\Omega,\mathscr{F},P)$.

Una variable aleatoria es una función que asigna un valor numérico a cada resultado posible de un experimento aleatorio. Lo anterior se plasma en la siguiente definición.

Definición 3.12 *Una variable aleatoria es una transformación $X:\Omega\to\mathbb{R}$ tal que, para cualquier número real x, se tiene que*

$$\{\omega\in\Omega: X(\omega)\leq x\}\in\mathscr{F}.$$

De este punto en adelante, para A un boreliano de $\mathbb{R}$, se usará la notación $(X\in A)$, para referirse conjunto de elementos ω del espacio muestral Ω tales que, al aplicarles la función X toman un valor dentro del conjunto A, esto es,

$$(X\in A)=\{\omega\in\Omega: X(\omega)\in A\}.$$

Variables aleatorias discretas y continuas

Dependiendo de los posible valores que toma una variable aleatoria se clasificará ésta como discreta o continua. Existen variables aleatorias que no son ni discretas ni continuas; sin embargo, por simplicidad, solo se trabajará con estos dos tipos.

Una variable aleatoria discreta toma valores en un conjunto discreto, es decir un conjunto finito o numerable. Como ejemplo, si una variable aleatoria toma valores en los números naturales, ésta será discreta, pues aunque los números naturales no son finitos, si son un conjunto numerable.

Cuando se vea el concepto de función de distribución se definirá lo que es una variable aleatoria continua. Por el momento se considerará una variable aleatoria continua si toma valores en algún intervalo $(a,b) \subseteq \mathbb{R}$.

3.6. Funciones de probabilidad y distribución

De manera intuitiva se puede entender a la función de probabilidad como una función que asigna probabilidades a los diferentes valores que puede tomar una variable aleatoria.

Definición 3.13 *Sea X una variable aleatoria discreta con valores $x_0, x_1, \cdots$. La función de probabilidad denotada por $f(x) : \mathbb{R} \to \mathbb{R}$ es*

$$f(x) = \begin{cases} P(X = x) & \text{si } x = x_0, x_1, \ldots \\ 0 & \text{en otro caso.} \end{cases}$$

Definición 3.14 *Sea X una variable aleatoria continua. Sea $f(x) : \mathbb{R} \to \mathbb{R}$ una función integrable y no negativa. Se dice que f es la función de densidad de X si para cualquier intervalo $[a,b]$ de $\mathbb{R}$ se cumple:*

$$P(X \in [a,b]) = \int_a^b f(x)dx.$$

La función de distribución evaluada en un valor $x \in \mathbb{R}$ es la probabilidad de que la variable aleatoria tome un valor menor o igual que x.

Definición 3.15 *Sea X una variable aleatoria cualquiera. La función de distribución de X, denotada por* $F(x) : \mathbb{R} \to \mathbb{R}$*, se define como*

$$F(x) = P(X \leq x).$$

La función de distribución de X se denota por $F_X(x)$, pero se omitirá el subíndice cuando no exista riesgo de confusión.

Una vez que se tiene el concepto de función de distribución se puede dar una definición más acertada acerca de las variables aleatorias continuas.

Definición 3.16 *Una variable aleatoria X es continua si su función de distribución* $F(x)$ *es una función continua.*

3.7. Independencia de variables aleatorias

El concepto de independencia en eventos se puede extender a independencia de variables aleatorias.

Definición 3.17 *Sean X y Y dos variables aleatorias definidas en el mismo espacio de probabilidad. Las variables aleatorias* X, Y *son independientes si los eventos* $(X \leq x)$ *y* $(Y \leq y)$ *son independientes para cualesquiera valores de* x, y*, es decir si se cumple la igualdad*

$$P[(X \leq x) \cap (Y \leq y)] = P(X \leq x)P(Y \leq y).$$

Observe que $P[(X \leq x) \cap (Y \leq y)] = P(X \leq x, Y \leq y)$. A esto se le llama función de distribución conjunta, la cual está evaluada en (x, y) y se suele denotar como $F_{X,Y}(x, y)$, así tenemos que dos variables aleatorias son independientes si

$$F_{X,Y}(x, y) = F_X(x)F_Y(y).$$

3.8. Esperanza

La esperanza es uno de los conceptos más importantes en probabilidad. La esperanza de X se denota por $E(X)$ y está definida de la siguiente manera.

Definición 3.18 *Sea X una variable aleatoria discreta con función de probabilidad $f(x)$. La esperanza de X se define como*

$$E(X) = \sum_x xf(x).$$

Es importante mencionar que la suma de los valores absolutos de los sumandos debe converger para que la esperanza exista.

Definición 3.19 *Sea X una variable aleatoria continua con función de probabilidad $f(x)$, la esperanza de X se define como*

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx.$$

Igualmente, la integral tiene que ser absolutamente convergente para que la esperanza exista.

En otras palabras, la esperanza de X se puede entender de manera intuitiva como un promedio ponderado de los posibles valores que X puede tomar, siendo cada valor ponderado por la probabilidad de que X lo asuma.

Proposición 3.2 ***(Propiedades de la esperanza)*** *Sean X y Y dos variables aleatorias con esperanza finita y c una constante. Se cumple que*

1. $E(c) = 0$.
2. $E(cX) = cE(X)$.
3. *Si* $X \geq 0$ *entonces* $E(X) \geq 0$.
4. $E(X+Y) = E(X)+E(Y)$.
5. *Si X, Y son independientes y tienen esperanza finita, entonces*

$$E(XY) = E(X)E(Y).$$

Las siguientes dos proposiciones son conocidas como la ley de estadístico inconsciente. Consiste en expresar el valor esperado de una función $g(X)$ de una variable aleatoria X en términos de la función g y de la distribución de probabilidad de X.

Proposición 3.3 ***(Ley del estadístico inconsciente)*** *Sea X una variable aleatoria discreta y $g : \mathbb{R} \to \mathbb{R}$ una función tal que $g(X)$ es una variable aleatoria con esperanza finita. Se cumple que*

$$E[g(X)] = \sum_x g(x) f_X(x).$$

Proposición 3.4 ***(Ley del estadístico inconsciente)*** *Sea X una variable aleatoria continua y $g : \mathbb{R} \to \mathbb{R}$ una función tal que $g(X)$ es una variable aleatoria con esperanza finita. Se cumple que,*

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x).$$

La esperanza condicional es un concepto avanzado en la teoría de probabilidad, en este trabajo solo se presentarán los conceptos básicos, para más información se puede consultar [14].

Definición 3.20 *Sea X una variable aleatoria con esperanza finita. La esperanza condicional de X dado Y, denotada por $E(X|Y)$, se define como aquella función de Y que cuando $Y = y$ es igual a*

$$E(X|Y=y) = \begin{cases} \sum_x x P(X = x|Y = y), \\ \quad \text{si } X \text{ y } Y \text{ son variables aleatorias discretas.} \\ \\ \int x f_{X|Y}(x|y)dx, \\ \quad \text{si } X, Y \text{ tiene función de probabilidad conjunta } f. \end{cases}$$

donde

$$f_{X|Y}(x|y) = \frac{f(x,y)}{\int f(x,y)dx} = \frac{f(x,y)}{f_Y(y)}.$$

Un resultado importante es la ley de la esperanza total, también llamada propiedad de torre. La ley de la esperanza total es útil cuando se quiere calcular el valor esperado de una variable aleatoria, y se cuenta con información adicional sobre otra variable con el mismo espacio de probabilidad.

Teorema 3.3 ***(Ley de la esperanza total)*** *Sea X una variable aleatoria con esperanza finita y una variable aleatoria Y con el mismo espacio de probabilidad que X, se cumple que*

$$E(X) = E(E(X|Y)).$$

3.9. Varianza

La varianza es una medida importante ya que proporciona información sobre la distribución y la dispersión de los datos. La varianza de X se denota por $\text{Var}(X)$.

Definición 3.21 *Sea X una variable aleatoria,* $\text{Var}(X) = E^2(X - E(X))$.

Si X es una variable aleatoria discreta con función de probabilidad $f(x)$. La varianza de X se calcula como

$$\text{Var}(X) = \sum_x (x - E(X))^2 f(x).$$

Cuando la suma es convergente.

Si X una variable aleatoria continua con función de probabilidad $f(x)$. La varianza de X se calcula como

$$\text{Var}(X) = \int_{-\infty}^{\infty} (x - E(X))^2 f(x) dx.$$

Cuando la integral es convergente.

Proposición 3.5 ***(Propiedades de la varianza)*** *Sean X y Y dos variables aleatorias con varianza finita y sea c una constante. Entonces*

1. $\text{Var}(X) \geq 0$.
2. $\text{Var}(c) = 0$.
3. $\text{Var}(cX) = c^2 \text{Var}(X)$.
4. $\text{Var}(X + c) = \text{Var(X)}$.
5. $\text{Var}(X) = E(X^2) - E^2(X)$.

Al igual que con la esperanza condicional, se puede definir la varianza condicional. Igualmente, la varianza condicional es un concepto avanzado en la teoría de probabilidad, para más detalles se puede consultar [14].

Definición 3.22 *Sea X una variable aleatoria con esperanza finita. La varianza condicional de una variable aleatoria X dada una variable aleatoria Y se denota por $Var(X|Y)$ y se calcula de la siguiente manera:*

$$\text{Var}(X|Y) = E[(X - E(X|Y))^2 |Y]$$

La varianza condicional tiene un resultado muy importante; el cual es la ley de la varianza total.

Teorema 3.4 ***(Ley de la varianza total).*** *Sea X una variable aleatoria con varianza finita, sea Y una variable aleatoria en el mismo espacio de probabilidad que X, se cumple que*

$$\text{Var}(X) = E[\text{Var}(X|Y)] + \text{Var}(E[X|Y]).$$

3.10. Covarianza

Así como el valor esperado y la varianza de una sola variable aleatoria nos proporcionan información sobre esa variable aleatoria, la covarianza entre dos variables aleatorias nos brinda información sobre la relación entre las variables aleatorias. La covarianza entre X y Y se denota por $\text{Cov}(X,Y)$.

Definición 3.23 *La covarianza entre dos variables aleatorias X y Y con esperanza finita, se define como*

$$\text{Cov}(X,Y) = E[(X - E(X))(Y - E(Y))].$$

De la definición anterior se tiene que cuando X y Y son variables aleatorias discretas, la covarianza se calcula como

$$\sum_x \sum_y (x - E(X))(y - E(Y)) f(x,y).$$

En el caso donde se tienen ambas variables aleatorias continuas, la covarianza tiene la siguiente expresión

$$\text{Cov}(X,Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - E(X))(y - E(Y)) f(x,y) dx dy.$$

Proposición 3.6 ***(Propiedades de la covarianza)*** *Sean X, X_1, X_2, Y variables aleatorias con varianza finita y c una constante. Entonces*

1. $\text{Cov}(X,Y) = E(XY) - E(X)E(Y)$.
2. $\text{Cov}(X,Y) = \text{Cov}(Y,X)$.
3. $\text{Cov}(X,X) = \text{Var}(X)$

4. $\text{Cov}(X,c) = \text{Cov}(c,X) = 0$.
5. $\text{Cov}(cX,Y) = \text{Cov}(X,cY) = c\text{Cov}(X,Y)$.
6. $\text{Cov}(X+c,Y) = \text{Cov}(X,Y+c) = \text{Cov}(X,Y)$.
7. $\text{Cov}(X_1+X_2,Y) = \text{Cov}(X_1,Y) + \text{Cov}(X_2,Y)$.
8. $\text{Var}(X+Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X,Y)$.
9. *Si X y Y son independientes entonces* $\text{Cov}(X,Y) = 0$.

La propiedad 2 hace ver que la covarianza es simétrica. Por su parte, las propiedades 4 y 6 muestran que la covarianza es una función lineal en cada variable.

3.11. Dos distribuciones de probabilidad

En esta sección se verán dos distribuciones discretas que son muy sencillas pero a la vez de mucha utilidad. Igualmente, estas distribuciones se utilizarán más adelante en las aplicaciones de los métodos de conteo. Existen muchas distribuciones más, si se desean revisar otras distribuciones se recomienda consultar [13].

Distribución uniforme discreta

Una variable aleatoria X tiene una distribución uniforme discreta si su función de probabilidad es

$$f(x) = \begin{cases} 1/n & \text{si } x = x_1, x_2, \ldots, x_n, \\ 0 & \text{en otro caso.} \end{cases}$$

De lo anterior se sigue que:

- $E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i = \mu$
- $\text{Var}(X) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2$

Esta distribución se utiliza cuando los posibles resultados del fenómeno aleatorio tienen la misma posibilidad de ocurrir.

Distribución Bernoulli

Un ensayo Bernoulli, en honor al matemático suizo Jacob Bernoulli, es un experimento donde sólo hay dos opciones, usualmente éxito y fracaso, cuyas probabilidades son p y $q = 1 - p$, donde $p \in [0,1]$. Una variable aleatoria X tiene distribución Bernoulli si su función de probabilidad es

$$f(x) = \begin{cases} 1-p & \text{si } x = 0, \\ p & \text{si } x = 1, \\ 0 & \text{en otro caso.} \end{cases}$$

De lo anterior se sigue que:

- $E(X) = p$.
- $\text{Var}(X) = p(1-p)$.

Aunque la distribución es sencilla es ampliamente utilizada. Por ejemplo, en un fenómeno aleatorio uno siempre puede preguntarse por la ocurrencia o no de un evento. Esto genera una distribución Bernoulli.

Capítulo 4

Métodos de conteo en árboles

En el Capítulo 1 se mostró el problema de calcular el costo de un árbol, sin embargo, no se proporcionó una manera de resolver el problema. El objetivo de este capítulo es mostrar algunos algoritmos que permiten solucionar el problema. La información de este capítulo está basada en [15] y [17] .

4.1. Algoritmos deterministas

Un algoritmo determinista es aquel que, al recibir una entrada específica, siempre producirá el mismo resultado. En el contexto de calcular el costo total de un árbol, esto significa que un algoritmo determinista siempre dará como resultado el costo exacto del árbol.

En esta sección, se presentan dos algoritmos deterministas que se basan en los recorridos de árboles que se han visto en el Capítulo 2, pero con la diferencia de que también consideran el costo asociado a cada vértice. Las Figuras 4.1 y 4.2 muestran los algoritmos BFS y DFS para el conteo de árboles, respectivamente.

Algoritmo 3: BFS para conteo en árboles.

```
1   Entrada Un árbol T con raíz u.
2   Salida Entero C con el costo total del árbol.
3  V ← () ;                          //Inicializar la lista de vértices
4  Q ← (u) ;                                  //Inicializar la fila
5  C ← 0 ;                                   //Inicializar el costo
6  mientras |Q| > 0 hacer
7      v ← Q(1) ;                 //Tomar el último elemento de la fila
8      Q ← Q(2 : end) ;  //Eliminar el último elemento de la fila
9      si v ∉ V entonces
10         C ← C + C(v) ;                          //Añadir el costo de v
11         V ← (V, v) ;              //Agregar v a la lista de vértices
               visitados
12         para w ∈ N(v) hacer
13             Q ← (Q, w) ;   //Agregar todos los vecinos de v a la
                   fila
14 devolver C
```

Figura 4.1: Algoritmo BFS para conteo de árboles.

La primera modificación del algoritmo BFS para contar árboles, en comparación con el recorrido BFS regular, consiste en añadir una variable para guardar el costo. La segunda modificación ocurre en la línea 10 del pseudocódigo, véase Figura 4.2, donde para cada nodo visitado v, se suma su costo a la variable C, que representa el costo total. Estas modificaciones se aplican de manera análoga en el algoritmo DFS para el conteo de árboles.

Además de los algoritmos mencionados anteriormente, hay otras opciones deterministas para resolver el problema de calcular el costo de un árbol. Por ejemplo, hay diferentes tipos de recorridos que, al igual que los recorridos BFS y DFS, pueden ser ajustados para obtener el costo total del árbol. Si desea obtener más información sobre estos algoritmos, puede consultar [16].

Algoritmo 4: DFS para conteo en árboles.

```
1  Entrada Un árbol T con raíz u.
2  Salida Entero C con el costo total del árbol.
3  V ← () ;                          //Inicializar la lista de vértices
4  S ← (u) ;                                  //Inicializar la pila
5  C ← 0 ;                                   //Inicializar el costo
6  mientras |S| > 0 hacer
7      v ← S(end) ;                 //Tomar el último elemento de la pila
8      S ← S(1 : end − 1) ;          //Eliminar el último elemento de la
         pila
9      si v ∉ V entonces
10         C ← C + C(v) ;                          //Añadir el costo de v
11         V ← (V, v) ;                 //Agregar v a la lista de vértices
             visitados
12         para w ∈ N(v) hacer
13             S ← (S, w) ;          //Agregar todos los vecinos de v a la
                 pila
14 devolver C.
```

Figura 4.2: Algoritmo DFS para conteo de árboles.

4.2. Algoritmo de Knuth

A continuación, se presentan algunas definiciones y un análisis que ayudará a deducir el algoritmo de Knuth.

Análisis previo

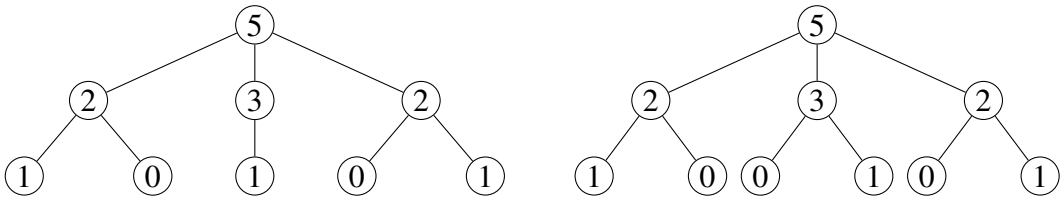
Sea un T árbol con profundidad n. En este árbol, se elige el nodo v_t del nivel t que tiene la mayor cantidad de nodos hijos, donde $0 \leq t \leq n$. En decir, si u_t es un nodo en el nivel t, entonces el número de hijos de u_t no supera al número de hijos de v_t.

Para construir el árbol extendido T' de costo cero a partir de T, se agrega un conjunto de nodos con costo cero a T para asegurar que todos los nodos en el nivel t tengan la misma cantidad de hijos que el nodo v_t.

En otras palabras, el árbol extendido T' de costo cero es simplemente el árbol original T al cual se le añaden los nodos con costo cero, para igualar la cantidad de hijos en cada nivel.

De la definición anterior, se sigue que cada trayectoria de la raíz a una hoja tiene una longitud de n. Véase que el costo del árbol extendido T' sigue siendo igual que el costo de T, ya que solo se añaden nodos con costo cero. A partir de este momento, en esta subsección se considerará que se trabajará con un árbol extendido.

Ejemplo 4.1 En la Figura 4.3 se muestra un ejemplo de un árbol extendido T' con respecto a un árbol T. Para construir el árbol T', solo fue necesario agregar un nodo adicional, para asegurar que todos los nodos de un nivel dado tengan la misma cantidad de hijos.



(a) Árbol T. (b) Árbol T' extendido.

Figura 4.3: Extensión de un árbol T.

•

Considere V_t el conjunto de todos los nodos en el t-ésimo nivel del árbol extendido. Por ejemplo, V_0 solo tiene un nodo, el cual es la raíz del árbol, y V_n tiene todas las hojas del árbol.

Definición 4.1 *Para cada $t = 0,1,\ldots,n$ se define una densidad g_t en el conjunto $\mathscr{X}_t$, de todos las trayectorias $\overline{x_t} = (x_0, x_1, \ldots, x_t)$ desde la raíz hasta el nivel t.*

Específicamente,

$$g_t(\overline{x}_t) = \frac{1}{\delta(x_0)} \frac{1}{\delta(x_1)} \cdots \frac{1}{\delta(x_{t-1})} := \frac{1}{D(\overline{x}_t)}.$$

Donde $\delta(x)$ se refiere al grado del vértice x.

La densidad anterior, efectivamente es una función de probabilidad y esto se demostrará más adelante.

La expresión $\sum_{\overline{x_t} \in \mathscr{X}_t} c(x_t)$ representa la suma de los costos asociados a los vértices situados exclusivamente en el nivel t, tomando en cuenta todas las trayectorias desde la raíz hasta dicho nivel. La expresión $\sum_{v \in V_t} c(v)$ significa sumar los costos de todos los nodos que se encuentran en el nivel t.

Tomando en consideración estas expresiones se tiene el siguiente lema.

Lema 4.1 *Sea T un árbol y t un nivel de profundidad. Se cumple que*

$$\sum_{\overline{x_t} \in \mathscr{X}_t} c(x_t) = \sum_{v \in V_t} c(v),$$

donde si u es un vértice, $c(u)$ es el costo del vértice u.

Demostración: Sea $v \in V_t$, existe una única trayectoria que empieza en la raíz y termina en el vértice v, esto se debe a las propiedades a la Definición 2.8 del Capítulo 2. De igual forma, para cada trayectoria $\overline{x}_t \in \mathscr{X}$, $\overline{x}_t$ termina en un sólo vértice $x_t \in V_t$. Por lo tanto, a cada vértice en el nivel t se le asocia una única trayectoria que inicia en la raíz. De lo anterior, se sigue que

$$\sum_{\overline{x_t} \in \mathscr{X}_t} c(x_t) = \sum_{v \in V_t} c(v).$$

■

Lema 4.2 *La función g_t es de probabilidad.*

Demostración: Para probar esto hace falta ver que, $g_t(\overline{x}_t) > 0$ para $\overline{x}_t \in \mathscr{X}_t$ y $\sum_{\overline{x}_t \in \mathscr{X}_t} g_t(\overline{x}_t) = 1$.

Para probar que $g_t(\overline{x}_t) > 0$ para $\overline{x}_t \in \mathscr{X}_t$, observe que $\delta(x_i) > 0$ para $0 \leq i \leq t$, por lo que

$$g_t(\overline{x}_t) = \frac{1}{\delta(x_0)} \cdot \frac{1}{\delta(x_1)} \cdots \frac{1}{\delta(x_{t-1})} > 0.$$

Para probar que $\sum_{\overline{x}_t \in \mathscr{X}_t} g_t(\overline{x}_t) = 1$, observe que al estar trabajando con un árbol extendido se tiene que cada trayectoria $\overline{x}_t \in \mathscr{X}_t$ tiene la misma densidad,

$\frac{1}{\delta(x_0)} \cdot \frac{1}{\delta(x_1)} \cdots \frac{1}{\delta(x_{t-1})}$. También, note que se tienen exactamente $\delta(x_0) \cdot \delta(x_1) \cdots \delta(x_{t-1})$ nodos en el nivel t. De lo anterior, se concluye que

$$\sum_{\overline{x}_t \in \mathscr{X}_t} g_t(\overline{x}_t) = (\delta(x_0) \cdot \delta(x_1) \cdots \delta(x_{t-1})) \left(\frac{1}{\delta(x_0)} \cdot \frac{1}{\delta(x_{t-1})} \cdots \frac{1}{\delta(x_{t-1})} \right) = 1.$$

■

Teorema 4.1 *Sea T un árbol. El estimador $\sum_{t=0}^{n} c(X_t)D(\overline{X}_t)$ es insesgado para el costo del árbol.*

Demostración: El costo total de los vértices en el nivel de profundidad t puede ser expresado como,

$$\sum_{v \in V_t} c(v) = \sum_{\overline{x}_t \in \mathscr{X}_t} \frac{c(x_t)}{g_t(\overline{x}_t)} g_t(\overline{x}_t) = \mathbb{E}_{g_t} \left[\frac{c(X_t)}{g_t(\overline{X}_t)} \right] = \mathbb{E}_{g_t} \left[c(X_t)D(\overline{X}_t) \right],$$

por lo que $c(X_t)D(\overline{X}_t)$ es un estimador insesgado del costo en el nivel de profundidad t, y se sigue que la suma sobre todos los niveles es un estimador del costo total del árbol. Por lo tanto. $\sum_{t=0}^{n} c(X_t)D(\overline{X}_t)$ es un estimador insesgado del costo total del árbol. ■

Algoritmo de Knuth

En la Figura 4.4 se muestra el algoritmo de Knuth que se deriva del análisis anterior.

Algoritmo 5: Algoritmo de Knuth

Entrada Árbol T con raíz u.
Salida Estimador C del costo total de T.
$D \leftarrow 1$; //Representa el valor $D(\overline{X})$
$X \leftarrow u$; //Representa el nodo seleccionado en cada iteración
$C \leftarrow c(u)$; //Representa el valor $C(X)$
mientras $|S(X)| > 0$ **hacer**
  $D \leftarrow |S(X)|D$; //Estima el valor $D(\overline{X}_t)$
  $X \leftarrow$ sucesor uniformemente elegido en $S(X)$;
  $C \leftarrow C + c(X)D$; //Estima el costo del nivel t y lo añade al costo total
devolver C

Figura 4.4: Pseudocódigo del algoritmo de Knuth.

La expresión $|S(X)|$ en el algoritmo de Knuth hace referencia a la cardinalidad de los sucesores del nodo X.

La salida del algoritmo de Knuth es:

$$\begin{aligned} C =& C(X_0) + |S(X_0)| \cdot C(X_1) + |S(X_0)| \cdot |S(X_1)| \cdot C(X_2) + \cdots \\ &+ |S(X_0)| \cdots |S(x_\tau)| \cdot C(X_\tau). \end{aligned}$$

Donde $\tau \leq h$, con h la altura del árbol T. En el algoritmo, el árbol al que se le estimará su costo no es necesariamente un árbol extendido, por eso $\tau \leq h$. La expresión X_i, $i = 0, \ldots, n$ hace referencia al nodo X en la iteración i.

Ejemplo 4.2 Considere el árbol que se muestra en la Figura 4.5. El objetivo es calcular el costo total del árbol el cual es quince. Los valores C, D, hacen referencia a los mismos valores que en el algoritmo de Knuth. En gris se muestran los nodos utilizados para estimar el costo total del árbol.

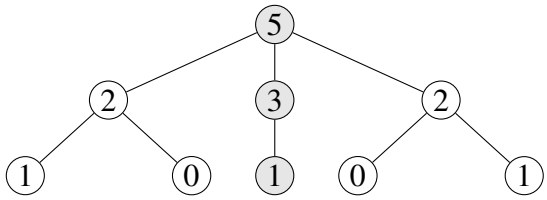


Figura 4.5: Árbol con costo total igual a 15.

En el primer paso se tiene la raíz, $D = 1$, y el costo $C = 5$. Se tienen tres posibles nodos sucesores, entonces, se actualiza el valor $D = (3)(1) = 3$, suponga que se elige el vértice con costo 3, entonces, el costo se actualiza, $C = 5 + (3)(3) = 14$.

El vértice con costo 3, tiene un sucesor, entonces, se actualiza el valor $D = 1(3) = 3$. Se elige el vértice con costo 1. El costo se actualiza, $C = 14 + (1)(3) = 17$.

Ahora estamos en el vértice con costo 1, éste no tiene sucesores, por lo cual, el algoritmo da un costo estimado de 17. El costo estimado difiere en 2 del costo total del árbol. •

Limitaciones

El algoritmo de Knuth hace una estimación del número total de nodos en cada nivel del árbol. Luego, utiliza esta información para estimar el costo asociado a cada nivel del árbol. Finalmente, suma los costos de todos los niveles para obtener el costo total del árbol. En resumen, el algoritmo de Knuth utiliza estimaciones para calcular el costo completo del árbol considerando la distribución de nodos en cada nivel. Por lo cual, el algoritmo de Knuth es efectivo si los costos están distribuidos a lo largo del árbol.
Si se tiene un árbol con poca simetría el algoritmo tiene muy poca eficacia. Un ejemplo extremo es el llamado árbol *Hairbrush*, que es un término en inglés que significa cepillo de cabello, el árbol es llamado así debido a su forma, véase la Figura 4.6. Éste es un árbol binario con profundidad n, tiene un nodo raíz y cada nodo derecho tiene dos nodos hijos, desde la raíz hasta el penúltimo nivel del árbol. Suponga que todos los vértices tienen un costo cero a excepción del nodo derecho en el último nivel de profundidad.

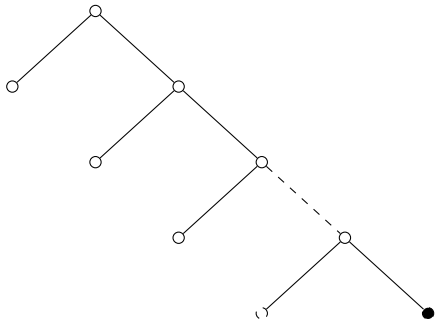


Figura 4.6: Representación del árbol *Hairbrush*.

El algoritmo de Knuth encuentra el vértice de costo 1, con probabilidad $\frac{1}{2^n}$ y con $D = C = 2^n$. En el resto de casos el algoritmo termina con un costo estimado de cero.
La esperanza y varianza del costo en este árbol son las siguientes:

$$E[C] = \frac{1}{2^n} \cdot 2^n$$

y

$$\text{Var}(C) = \frac{1}{2^n} \cdot (2^n)^2 - 1 = 2^n - 1.$$

Por lo que, aunque el estimador es insesgado la varianza crece de forma exponencial en relación con la profundidad del árbol.

4.3. Método de Enumeración estocástica

El método de enumeración estocástica es una generalización del algoritmo de Knuth, donde se simulan múltiples recorridos en el árbol al mismo tiempo. En lugar de depender de una sola trayectoria, este método calcula el costo de cada nivel promediando los costos de todas las trayectorias simuladas. Esto resulta en una estimación más precisa del costo total del árbol, ya que toma en cuenta varias posibles rutas en lugar de basarse en una sola.

Hipernodos, hiperhijos y costos

Al generalizar el algoritmo Knuth también se generalizan conceptos los cuales se presentan en las siguientes definiciones.

Considere un árbol T, con V el conjunto de nodos y E el conjunto de aristas. La raíz del árbol se denota como v_o, y para cualquier nodo v que pertenezca al conjunto V, el subárbol con v como raíz se denota por T_v.

Definición 4.2 *Sean $\{v_1, v_2, \ldots v_r\} \subseteq V$ nodos del árbol.*
Un hipernodo es la colección $\overline{\boldsymbol{v}} = \{v_1, v_2, \ldots, v_r\}$ de distintos nodos en el mismo nivel de profundidad de cardinalidad $|\overline{\boldsymbol{v}}| = r$. El costo del hipernodo se define como

$$c(\overline{\boldsymbol{v}}) = \sum_{v \in \overline{\boldsymbol{v}}} c(v).$$

Ejemplo 4.3 En la Figura 4.7 en color gris se muestran los nodos v_1, v_2, v_4, estos representan a un posible hipernodo en el primer nivel del árbol. El costo del hipernodo será la suma de los costos de los nodos, v_1, v_2, v_4. •

Definición 4.3 *Sea $\overline{\boldsymbol{v}}$ un hipernodo. El conjunto de sucesores de $\overline{\boldsymbol{v}}$ se define como*

$$S(\overline{\boldsymbol{v}}) = \bigcup_{v \in \text{`}\overline{\boldsymbol{v}}} S(v).$$

Definición 4.4 *Sea $\overline{\boldsymbol{v}}$ un hipernodo y considere $B \geq 1$, $B \in \mathbb{N}$, los hiperhijos dados por la función H, se definen de la siguiente manera:*

$$H(\overline{\boldsymbol{v}}) = \begin{cases} \{S(\overline{\boldsymbol{v}})\} & \text{si } |S(\overline{\boldsymbol{v}})| \leq B \\ \{\overline{\boldsymbol{w}} | \overline{\boldsymbol{w}} \subseteq S(\overline{\boldsymbol{v}}), |\overline{\boldsymbol{w}}| = B\} & \text{si } |S(\overline{\boldsymbol{v}})| > B. \end{cases}$$

Definición 4.5 *Para cada hipernodo $\overline{\boldsymbol{v}}$, se define el bosque de árboles con raíz en $\overline{\boldsymbol{v}}$ como*

$$T_{\overline{\boldsymbol{v}}} = \bigcup_{v \in \overline{\boldsymbol{v}}} T_v.$$

Ejemplo 4.4 En la Figura 4.7 el bosque correspondiente al hipernodo $\overline{\mathbf{v}} = \{v_1, v_2, v_4\}$ es $T_{\overline{\mathbf{v}}} = \{T_{v_1}, T_{v_2}, T_{v_4}\}$. •

Definición 4.6 *Sea $T_{\overline{\boldsymbol{v}}}$ el bosque de árboles con raíz en $\overline{\boldsymbol{v}}$, se define el conjunto de vértices al nivel m como*

$$S^{(m)}(\overline{\boldsymbol{v}}) = \underbrace{S(S(S \ldots (\overline{\boldsymbol{v}}) \ldots))}_{m \text{ veces}}.$$

Donde $S^{(0)}(\overline{\boldsymbol{v}}) = \overline{\boldsymbol{v}}$.

Definición 4.7 *Para cada bosque con raíz en el hipernodo* $\overline{\boldsymbol{v}}$ *se define su costo total como,*

$$\text{Costo}(T_{\overline{\boldsymbol{v}}}) = \sum_{v \in \overline{\boldsymbol{v}}} \text{Costo}(T_v). \tag{4.1}$$

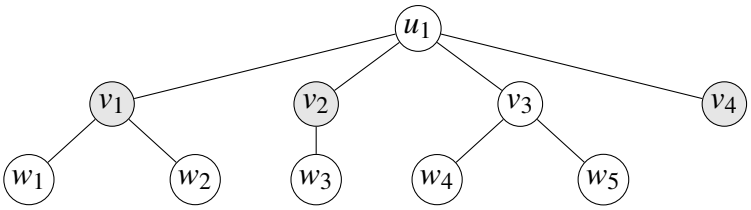


Figura 4.7: Ejemplo de hipernodos.

Una vez que se cuenta con estos conceptos se puede presentar el método de enumeración estocástica. El cual se muestra en la Figura 4.8.

Algoritmo

Algoritmo 6: Método de enumeración estocástica

Entrada Un bosque $T_{\overline{\mathbf{v}}}$ con raíz en el hipernodo $\overline{\mathbf{v}}$ y un presupuesto $B \geq 1$.

Salida Estimador $|\overline{\mathbf{v}}|C_{\mathrm{SE}}$ del costo total del bosque $T_{\overline{\mathbf{v}}}$.

$K \leftarrow 0$; //Representa las iteraciones

$D \leftarrow 1$; //Representa el valor $D(\overline{X}_t)$

$\mathbf{X}_0 \leftarrow \overline{\mathbf{v}}$; //Representa los nodos seleccionados en la iteración k

$C_{\mathrm{SE}} \leftarrow c(\mathbf{X}_0)/|\mathbf{X}_0|$; //Representa el valor $C(X_t)$

mientras $|S(X_k)| > 0$ **hacer**

  $\mathbf{X}_{k+1} \leftarrow$ sucesor (Hiperhijo) uniformemente elegido en $H(\mathbf{X}_k)$;

  $D_k \leftarrow \frac{|S(\mathbf{X}_k)|}{|\mathbf{X}_k|}$; //Estima el total de nodos en el nivel t

  $D \leftarrow D_k D$; //Estima el valor $D(\overline{X}_t)$

  $C_{\mathrm{SE}} \leftarrow C_{\mathrm{SE}} + \left(\frac{c(\mathbf{X}_{k+1})}{|\mathbf{X}_{k+1}|}\right) D$; //Estima el valor $D(\overline{X}_t)C(X_t)$ y lo añade al costo

  $k \leftarrow k+1$;

devolver $|\overline{\mathbf{v}}|C_{\mathrm{SE}}$

Figura 4.8: Pseudocódigo del algoritmo de enumeración estocástica.

El valor B hace referencia al termino en inglés *budget*, que significa presupuesto. Comparando el Algoritmo de enumeración estocástica con el algoritmo de Knuth, la primera diferencia que se encuentra son los datos de entrada del algoritmo, mientras que en el algoritmo de Knuth se inicia con un solo vértice, el cual es la raíz del árbol, el método de enumeración estocástica es más flexible y permite iniciar con múltiples vértices, esto se mantiene en cada paso del algoritmo.

Al considerar $B = 1$ todos los hiperhijos tienen cardinalidad igual a 1, de lo cual, se obtiene el estimador propuesto en el algoritmo de Knuth, es por esto que se considera al método de enumeración estocástica como una extensión del algoritmo de Knuth.

De manera similar al algorimto de Knuth, la salida del método de enumeración

estocástica es:

$$\begin{aligned} C_{\mathrm{SE}} =& \frac{c(|\mathbf{X}_0)|}{|\mathbf{X}_0|} + \frac{|S(\mathbf{X}_0)|}{\mathbf{X}_0}\frac{c(\mathbf{X}_1)}{|\mathbf{X}_1|} + \frac{|S(|\mathbf{X}_0|)|}{|\mathbf{X}_0|}\frac{|S(\mathbf{X}_1)|}{|\mathbf{X}_1|}\frac{c(\mathbf{X}_2)}{|\mathbf{X}_2|} \\ & + \cdots + \left(\prod_{0\le j\le \tau-1}\frac{|S(\mathbf{X}_j)|}{|\mathbf{X}_j|}\right)\frac{c(\mathbf{X}_\tau)}{|\mathbf{X}_\tau|}, \\ =& \frac{c(\mathbf{X}_0)}{|\mathbf{X}_0|} + D_0\frac{c(\mathbf{X}_1)}{|\mathbf{X}_1|} + D_1\frac{c(\mathbf{X}_2)}{|\mathbf{X}_2|} + \cdots + D_{\tau-1}\frac{c(\mathbf{X}_\tau)}{|\mathbf{X}_\tau|}, \end{aligned}$$

donde $\tau \le h$, con h la altura del árbol T.

Ejemplo 4.5 Considere el árbol que se muestra en la Figura 4.9. El objetivo es calcular el costo total del árbol el cual es quince. Se considerará $B = 2$. Los valores C, D hacen referencia a los mismos valores en el método de enumeración estocástica. En gris se muestran los nodos utilizados para estimar el costo total del árbol.

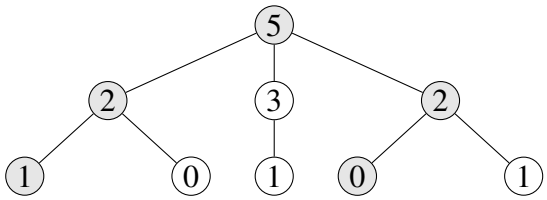


Figura 4.9: Árbol con costo total igual a 15.

En el primer paso se tiene la raíz, $D = 1$ y $C = 5$. Se tienen tres posibles nodos sucesores, entonces se actualiza el valor $D = 3(1) = 3$. Suponga que se eligen los vértices con costo dos, entonces, el costo estimado, $C = 5/1 + (3)(4/2) = 11$.

Los vértices con costo 2, tiene ambos dos sucesores, entonces, se actualiza el valor $D = (4/2)(3) = 6$. Se elige un vértice con costo 1 y un vértice con costo 0, el costo estimado se actualiza, $C = 5/1 + (3)(4/2) + (6)(1/2) = 11 + (6)(1/2) = 14$.

Ahora estamos en el último nivel del árbol, por lo tanto, no hay sucesores. El algoritmo da un costo estimado de 14, el cual difiere en 1 del costo real del árbol.

•

Análisis

En esta subsección se presenta un análisis matemático del por qué el estimador es insesgado, además se presenta su varianza y se muestra que si se escoge un valor de B adecuado el estimador tiene varianza cero.

Preliminares

Antes de analizar el método de enumeración estocástica se probará unos enunciados que se ocuparán para demostrar que la salida del método de enumeración estocástica produce un estimador insesgado del costo del árbol.

Lema 4.3 ***(Suma de k-subconjuntos)*** *Sea $A = \{r_1, \ldots, r_n\}$ un conjunto de escalares. Se define el conjunto:*

$$J = \{R | R \subseteq A, |R| = u\}, \quad u = 1, \ldots, n.$$

Para el cual se cumple que

$$\frac{1}{u} \sum_{R \in J} \sum_{r \in R} r = \frac{1}{n} \binom{n}{u} \sum_{j=1}^{n} r_j.$$

Demostración: Existen exactamente $\binom{n-1}{u-1}$ subconjuntos en los cuales cada r_j aparece, con $j = 1, \ldots, n$. Entonces,

$$\begin{aligned}
\frac{1}{u} \sum_{R \in J} \sum_{r \in R} r &= \frac{1}{u} \binom{n-1}{u-1} \sum_{j=1}^{n} r_j \\
&= \frac{1}{u} \frac{(n-1)!}{(u-1)!(n-u)!} \sum_{j=1}^{n} r_j \\
&= \frac{1}{n} \frac{(n)!}{(u)!(n-u)!} \sum_{1 \leq j \leq n} r_j \\
&= \frac{1}{n} \binom{n}{u} \sum_{1 \leq j \leq n} r_j.
\end{aligned}$$

■

Teorema 4.2 ***(Fórmula deformada de la covarianza)***. *La covarianza muestral definida como*

$$\mathrm{Cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y),$$

donde

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n}x_i, \quad \mu_y = \frac{1}{n}\sum_{i=1}^{n}y_i.$$

se puede expresar como

$$\mathrm{Cov}(X,Y) = \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i-x_j)(y_i-y_j).$$

Demostración: Desarrollando la expresión de la covarianza

$$\begin{aligned}
\mathrm{Cov}(X,Y) =& \frac{1}{n}\left[\sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}x_i\mu_y - \sum_{i=1}^{n}y_i\mu_x - \sum_{i=1}^{n}\mu_x\mu_y\right] \\
=& \frac{1}{n}\left[\sum_{i=1}^{n}x_iy_i - \mu_y\sum_{i=1}^{n}x_i - \mu_x\sum_{i=1}^{n}y_i + n\mu_x\mu_y\right] \\
=& \frac{1}{n}\left[\sum_{i=1}^{n}x_iy_i - \frac{(x_1+\ldots+x_n)(y_1+\ldots+y_n)}{n}\right] \\
& - \frac{1}{n}\left[\frac{(y_1+\ldots+y_n)(x_1+\ldots+x_n)}{n}\right] \\
& + \frac{1}{n}\left[\frac{(x_1+\ldots+x_n)(y_1+\ldots+y_n)}{n}\right] \\
=& \frac{1}{n}\left[\sum_{i=1}^{n}x_iy_i - \frac{(x_1+\ldots+x_n)(y_1+\ldots+y_n)}{n}\right] \\
=& \frac{1}{n^2}\left[n\sum_{i=1}^{n}(x_iy_i) - \left(\sum_{i=1}^{n}x_i\right)\left(\sum_{i=1}^{n}y_i\right)\right].
\end{aligned}$$

Por otro lado, observe que

$$
\begin{aligned}
&\frac{1}{n^2}\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i-x_j)(y_i-y_j)\right]\\
=&\frac{1}{n^2}\left[\sum_{i=1}^{n-1}\left[\sum_{j=i+1}^{n}x_iy_i-\sum_{j=i+1}^{n}x_iy_j-\sum_{j=i+1}^{n}y_ix_j+\sum_{j=i+1}^{n}x_jy_j\right]\right]\\
=&\frac{1}{n^2}\left[\sum_{i=1}^{n-1}\left[(n-i)x_iy_i-x_i\sum_{j=i+1}^{n}y_j-y_i\sum_{j=i+1}^{n}x_j+\sum_{j=i+1}^{n}x_jy_j\right]\right]\\
=&\frac{1}{n^2}\left[(n-1)(x_1y_1)-x_1(y_2+\ldots y_n)-y_1(x_2+\ldots+x_n)+x_2y_2+\ldots+x_ny_n\right]\\
&+\frac{1}{n^2}\left[(n-2)(x_2y_2)-x_2(y_3+\ldots y_n)-y_2(x_3+\ldots+x_n)+x_3y_3+\ldots+x_ny_n\right]\\
&\vdots\\
&+\frac{1}{n^2}\left[(n-(n-1))(x_{n-1}y_{n-1})-x_{n-1}(y_n)-y_{n-1}(x_n)+x_ny_n\right].
\end{aligned}
$$

Ignorando el término $\frac{1}{n^2}$. Para cada i se tiene $n-1$ veces la expresión x_iy_i ya que se tiene $n-i$ en la expresión $(n-i)(x_iy_i)$ y aparece $i-1$ veces antes en la expresión $x_1y_1+\ldots+x_ny_n+\cdots+x_{i-1}y_{i-1}+\ldots+x_ny_n$, lo que da un total de $n-1$ veces. Por su parte se tiene que cada x_i se multiplica con cada y_j para $i\neq j$, ya que en la expresión $x_i\sum_{j=i+1}^{n}y_j$, se está multiplicando x_i con y_j para $i<j$ pero en la expresión $y_1(x_2+\ldots+x_n)+\cdots+y_{i-1}(x_i+\ldots+x_n)$ el valor de x_i ya se ha multiplicado con los valores y_j para $i>j$, de lo anterior se tiene que

$$
\begin{aligned}
\frac{1}{n^2}\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i-x_j)(y_i-y_j)\right]&=\frac{1}{n^2}\left[\sum_{k=1}^{n}(n-1)(x_ky_k)-\sum_{k=1}^{n}x_k\sum_{k=1}^{n}y_k+\sum_{k=1}^{n}x_ky_k\right]\\
&=\frac{1}{n^2}\left[n\sum_{k=1}^{n}(x_ky_k)-\sum_{k=1}^{n}x_k\sum_{k=1}^{n}y_k\right].
\end{aligned}
$$

Se concluye que

$$\mathrm{Cov}(X,Y) = \frac{1}{n^2}\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i - x_j)(y_i - y_j)\right].$$

■

Dado que la varianza de una variable aleatoria se puede considerar como la covarianza entre esa variable y ella misma, se tiene el siguiente corolario.

Corolario 4.1 *Si X es una variable aleatoria uniforme discreta sobre el conjunto de n números $\{x_1, \ldots, x_n\}$, entonces*

$$\mathrm{Var}(X) = \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i - x_j)^2$$

Demostración: La varianza de X se puede expresar como:

$$\mathrm{Var}(X) = \mathrm{Cov}(X,X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(x_i - \mu_x),$$

usando el teorema anterior, se tiene que

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(x_i - \mu_x) = \frac{1}{n^2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i - x_j)^2.$$

■

Una vez dadas las definiciones y lemas necesarios se puede probar los teoremas de la siguiente subsección.

Esperanza y varianza del costo en el método de enumeración estocástica

Teorema 4.3 ***(Estimador insesgado)*** *Si $|v|C_{\mathrm{SE}}(T_{\overline{v}})$ es el estimador del método de enumeración estocástica para el bosque $T_{\overline{v}}$, con $\overline{v}$ un hipernodo y $H(S(\overline{v})) = \{\overline{w}_1, \ldots, \overline{w}_d\}$ su conjunto de hiperhijos con cardinalidad d. Entonces*

$$E(C_{\mathrm{SE}}(T_{\overline{v}})) = \frac{\mathrm{Costo}(T_{\overline{v}})}{|\overline{v}|}. \tag{4.2}$$

Demostración: Por la estructura recursiva de la salida del método de enumeración estocástica , se tiene que

$$C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}}) = \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|} C_{\mathrm{SE}}(T_{\overline{\mathbf{W}}}), \tag{4.3}$$

donde $\overline{\mathbf{W}}$ es un hiperhijo de $\overline{\mathbf{v}}$ seleccionado de manera aleatoria con probabilidad uniforme de $H(S(\overline{\mathbf{v}}))$. Para mostrar este resultado se procedera por inducción.

Caso base. Si $h = 0$,

$$\begin{aligned} E(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}})) &= E\left(\frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|} \cdot 0\right) \\ &= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} \\ &= \frac{\sum_{v \in \overline{\mathbf{v}}} c(v)}{|\overline{\mathbf{v}}|} \\ &= \frac{\mathrm{Costo}(T_{\overline{\mathbf{v}}})}{|\overline{\mathbf{v}}|}. \end{aligned}$$

Hipótesis de inducción. Suponga cierto el resultado para alturas $0, 1, \ldots, h-1$.

Paso inductivo. Usando la ecuación (4.3) se obtiene que

$$\begin{aligned} E(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}})) &= E\left(\frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|} \cdot C_{\mathrm{SE}}(T_{\overline{\mathbf{W}}})\right) \\ &= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\frac{1}{d}\sum_{1 \le j \le d} E(C_{\mathrm{SE}}\left(T_{\overline{\mathbf{w}}_j}\right))\right) \\ &= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\frac{1}{d}\sum_{1 \le j \le d} \frac{\mathrm{Costo}(T_{\overline{\mathbf{w}}_j})}{|\overline{\mathbf{w}}|}\right). \end{aligned}$$

Donde la última igualdad es debido a la hipótesis de inducción.

Se tienen los siguientes dos casos:

1. $|S(\overline{\mathbf{v}})| \leq B$. Aquí se tiene, $H(S(\overline{\mathbf{v}})) = \{\overline{\mathbf{w}}_1\}$, $|S(\overline{\mathbf{v}})| = |\overline{\mathbf{w}}_1|$ y $d = 1$, por lo tanto,

$$\begin{aligned}\frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\frac{1}{d}\sum_{1\leq j\leq d}\frac{\text{Costo}(T_{\overline{\mathbf{w}}_j})}{|\overline{\mathbf{w}}|}\right) &= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\frac{\text{Costo}(T_{\overline{\mathbf{w}}_1})}{|\overline{\mathbf{w}}_1|} \\ &= \frac{c(\overline{\mathbf{v}}) + \text{Costo}(T_{\overline{\mathbf{w}}_1})}{|\overline{\mathbf{v}}|} \\ &= \frac{\text{Costo}(T_{\overline{\mathbf{v}}})}{|\overline{\mathbf{v}}|}.\end{aligned}$$

2. $|S(\overline{\mathbf{v}})| > B$. En este caso hay un conjunto de posibles hipernodos que serán escogidos aleatoriamente con probabilidades uniformes de $H(S(\overline{\mathbf{v}}))$. Se tiene que

$$H(S(\overline{\mathbf{v}})) = \{\overline{\mathbf{w}}_1, \ldots \overline{\mathbf{w}}_d\}, \quad d = |H(S(\overline{\mathbf{v}}))| = \binom{|S(\overline{\mathbf{v}})|}{B} > 1.$$

y para $j = 0 \ldots d$, se tiene que $|\overline{\mathbf{w}}_j| = B$. Así que

$$
\begin{aligned}
&\frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\frac{1}{d}\sum_{1\le j\le d}\frac{\text{Costo}(T_{\overline{\mathbf{w}}_j})}{|\overline{\mathbf{w}}|}\right)\\
&= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\binom{|S(\overline{\mathbf{v}})|}{B}^{-1}\sum_{1\le j\le d}\frac{\text{Costo}(T_{\overline{\mathbf{w}}_j})}{B}\right)\\
&= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\binom{|S(\overline{\mathbf{v}})|}{B}^{-1}\sum_{1\le j\le d}\frac{\sum_{w\in\overline{\mathbf{w}}_j}\text{Costo}(T_w)}{B}\right)\\
&= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\binom{|S(\overline{\mathbf{v}})|}{B}^{-1}\frac{\binom{|S(\overline{\mathbf{v}})|}{B}\sum_{w\in S(\overline{\mathbf{v}})}\text{Costo}(T_w)}{|S(\overline{\mathbf{v}})|}\right) \qquad (4.4)\\
&= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\frac{1}{|S(\overline{\mathbf{v}})|}\sum_{w\in S(\overline{\mathbf{v}})}\text{Costo}(T_w)\right)\\
&= \frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\left(\frac{\text{Costo}(T_{S(\overline{\mathbf{v}})})}{|S(\overline{\mathbf{v}})|}\right)\\
&= \frac{c(\overline{\mathbf{v}}) + \text{Costo}(T_{S(\overline{\mathbf{v}})})}{|\overline{\mathbf{v}}|}\\
&= \frac{\text{Costo}(T_{\overline{\mathbf{v}}})}{|\overline{\mathbf{v}}|}.
\end{aligned}
$$

Donde 4.4 se sigue del Lema 4.3, sustituyendo

$$|S(\overline{\mathbf{v}})| = n, \quad \text{Costo}(T_{w_j}) = r_j \text{ para } 1 \le j \le n, \quad u = B, \quad d = \binom{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}.$$

■

Si para el árbol T con raíz v_0 se define $\overline{\mathbf{v}}_0 = \{v_0\}$, entonces el bosque $T_{\overline{\mathbf{v}}_0}$ es precisamente el árbol T y como consecuencia del Teorema 4.3 se tiene el siguiente corolario.

Corolario 4.2 ***(Estimador insesgado del costo)*** *Si T es un árbol con raíz v_0, entonces el método de enumeración estocástica devuelve un estimador insesgado del costo total del árbol, esto es,*

$$E(C_{\text{SE}}(T_{\overline{\mathbf{v}}_0})) = \text{Costo}(T).$$

A continuación se presenta una expresión recursiva para la varianza del estimador del método de enumeración estocástica.

Teorema 4.4 *Si $\overline{v}$ es un hipernodo y $H(S(\overline{v})) = \{\overline{w}_1, \ldots, \overline{w}_d\}$ su conjunto de hiperhijos, entonces*

$$\begin{aligned}\mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{v}})) =& \frac{\left(\frac{|S(\overline{v})|}{|\overline{v}|}\right)^2}{d} \sum_{1\le j\le d} \mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{w}_j})) \\ &+ \frac{\left(\frac{|S(\overline{v})|}{|\overline{v}|}\right)^2}{d} \sum_{1\le i<j\le d} \left(\frac{\mathrm{Costo}(T_{\overline{w}_i})}{|\overline{w}_i|} - \frac{\mathrm{Costo}(T_{\overline{w}_j})}{|\overline{w}_j|}\right)^2.\end{aligned} \tag{4.5}$$

Demostración: Por la ecuación (4.3) y de la ley de la varianza total, se tiene que

$$\begin{aligned}\mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}})) &= \mathrm{Var}\left(\frac{c(\overline{\mathbf{v}})}{|\overline{\mathbf{v}}|} + \frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|} C_{\mathrm{SE}}(T_{\overline{\mathbf{W}}})\right) \\ &= \left(\frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\right)^2 \mathrm{Var}\left(C_{\mathrm{SE}}(T_{\overline{\mathbf{W}}})\right) \\ &= \left(\frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\right)^2 \left(E\left[\mathrm{Var}\left(C_{\mathrm{SE}}\left(T_{\overline{\mathbf{W}}}\right)|\overline{\mathbf{W}}\right)\right] + \mathrm{Var}\left(E\left[C_{\mathrm{SE}}\left(T_{\overline{\mathbf{W}}}\right)|\overline{\mathbf{W}}\right]\right)\right).\end{aligned}$$

Donde $\overline{\mathbf{W}}$ es un hiperhijo de $\overline{\mathbf{v}}$ seleccionado de manera aleatoria con probabilidad uniforme de $H(S(\overline{\mathbf{v}}))$. De lo anterior se tiene que

$$E\left[C_{\mathrm{SE}}\left(T_{\overline{\mathbf{W}}}\right)|\overline{\mathbf{W}}\right] = \frac{1}{d} \sum_{1\le j\le d} \mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{w}}_j})).$$

Por el Corolario 4.2 y el Teorema 4.3 se sigue que

$$\begin{aligned}\mathrm{Var}\left(E\left[C_{\mathrm{SE}}\left(T_{\overline{\mathbf{W}}}\right)|\overline{\mathbf{W}}\right]\right) &= \mathrm{Var}\left(\frac{\mathrm{Costo}(T_{\overline{\mathbf{W}}})}{|\overline{\mathbf{W}}|}\right) \\ &= \frac{1}{d^2} \sum_{1\le i<j\le d} \left(\frac{\mathrm{Costo}(T_{\overline{\mathbf{w}}_i})}{|\overline{\mathbf{w}}_i|} - \frac{\mathrm{Costo}(T_{\overline{\mathbf{w}}_j})}{|\overline{\mathbf{w}}_j|}\right)^2.\end{aligned}$$

Uniendo los dos resultados anteriores se tiene que

$$\begin{aligned}\mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}})) =& \frac{\left(\frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\right)^2}{d} \sum_{i\leq j\leq d} \mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{w}}_j})) \\ &+ \frac{\left(\frac{|S(\overline{\mathbf{v}})|}{|\overline{\mathbf{v}}|}\right)^2}{d} \sum_{i\leq j\leq d} \left(\frac{\mathrm{Costo}(T_{\overline{\mathbf{w}}_i})}{|\overline{\mathbf{w}}_i|} - \frac{\mathrm{Costo}(T_{\overline{\mathbf{w}}_j})}{|\overline{\mathbf{w}}_j|}\right)^2.\end{aligned}$$

■

Corolario 4.3 ***(Estimador de varianza cero)*** *Si $\overline{\boldsymbol{v}}$ es un hiper raíz del bosque $T_{\overline{\boldsymbol{v}}}$ y $S^{(m)}(\overline{\boldsymbol{v}})$ el conjunto de nodos en el nivel m del árbol, con $m = 0,\ldots,h$, entonces el método de enumeración estocástica con presupuesto*

$$B = \max_{0\leq m\leq h} \{|S^{(m)}(\overline{\boldsymbol{v}})|\}.$$

es un estimador de varianza cero.

Demostración: Este resultado es consecuencia del Teorema 4.4, aunado al hecho de que el hiperárbol resultante tiene un único hipernodo $\overline{\mathbf{v}}^{(m)}$ para cada nivel m del árbol, $0 \leq m \leq h$, donde h es la altura del árbol. Además este único hipernodo tiene grado igual uno o cero dependiendo de si es hoja o no.

Sea $\overline{\mathbf{v}}^{(0)}$ la hiperraíz del árbol, los conjuntos de hiperhijos en los niveles $0,\ldots,h$ están dados por ,

$$H\left(S(\overline{\mathbf{v}}^{(0)})\right) = \left\{\overline{\mathbf{v}}^{(1)}\right\},\ldots,H\left(S(\overline{\mathbf{v}}^{(h-1)})\right) = \left\{\overline{\mathbf{v}}^{(h)}\right\}, H\left(S(\overline{\mathbf{v}}^{(h)}\right) = \{\emptyset\}.$$

Del Teorema 4.4 se sigue que

$$\mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}^{(0)}})) = \left(\frac{|S(\overline{\mathbf{v}}^{(0)})|}{|\overline{\mathbf{v}}^{(0)}|}\right) \mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}^{(1)}})) + 0.$$

Iterando la ecuación anterior se obtiene

$$\mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}^{(0)}})) = \prod_{0\leq m\leq h-1} \left(\frac{|S(\overline{\mathbf{v}}^{(m)})|}{|\overline{\mathbf{v}}^{(m)}|}\right) \mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}^{(h)}})).$$

El resultado anterior es cero, debido a que

$$\mathrm{Var}(C_{\mathrm{SE}}(T_{\overline{\mathbf{v}}^{(h)}})) = 0.$$

Lo que concluye la prueba. ■

Mejoras

El método de enumeración estocástica ofrece una mejora cuando se trata de árboles desbalanceados. Sin embargo, esto depende del valor seleccionado para B al ejecutar el algoritmo. En general, cuanto mayor sea el valor de B, el estimador estará más cerca del costo total del árbol en la mayoría de los casos.

Retomando el ejemplo del árbol *Hairbrush*, como se vio anteriormente, el algoritmo de Knuth produce malas estimaciones. En el método de enumeración estocástica, si se toma un valor $B = 2$ se produce un estimador insesgado con varianza cero. Esto se puede comprobar con el Corolario 4.3. En la primera iteración se considera la raíz del árbol, en la segunda iteración se consideran ambos hijos, en general, para todas la iteraciones se consideran dos nodos, como en cada nivel hay dos nodos, se tiene que se consideran todos los nodos del árbol, por lo cual, se obtiene un estimador insesgado de varianza cero.

Lo interesante del problema de calcular el costo del árbol *hairbrush* es que aumentando en uno el valor de B ahora se tiene un estimador insesgado. En general dado un árbol, por el Teorema 4.3 siempre se puede llegar a un estimador insesgado de varianza cero, sin embargo, dependerá del problema si se utiliza el método de enumeración estocástica o algún recorrido. Una de las ventajas del método de enumeración estocástica es su flexibilidad al ajustar el valor de B. A medida que se busca mayor precisión, el valor de B se incrementará. Esto permite adaptarse a las necesidades específicas del problema.

Se ha trabajado tanto con el algoritmo de Knuth como con el método de enumeración estocástica realizando una sola ejecución del algoritmo. Sin embargo, si se realizan múltiples ejecuciones, los algoritmos se vuelven más eficaces al reducir la varianza. Para obtener más detalles, se recomienda consultar [15].

Capítulo 5

Aplicaciones

En este capítulo, se hablará sobre la relación entre diferentes problemas de conteo y el cálculo del costo de un árbol. Igualmente, se explicará cómo se resuelven estos problemas y también se verá cómo se adaptan los algoritmos de enumeración estocástica y Knuth para resolverlos. La información de este capítulo está basada en [15] y [18].

5.1. Enumeración estocástica con oráculos

Al momento de ejecutar el algoritmo de enumeración estocástica, se busca generar trayectorias en el árbol que sean útiles para estimar el costo del árbol. Suponga que v es un vértice que se visita durante la ejecución del método de enumeración estocástica y w es un nodo hijo de v. Si se tiene información que cualquier camino que continúa después de w tendrá costo cero, tiene sentido eliminar a w de la lista de sucesores de v. La decisión de mantener o no a w puede calcularse de forma eficiente mediante un oráculo. Es decir, un oráculo es una forma de calcular si un sucesor de un vértice debe ser considerado o si debe ser ignorado. A continuación se presenta un ejemplo de cómo es utilizar un oráculo con el método de enumeración estocástica.

Ejemplo 5.1 En este ejemplo se pretende mostrar cómo funciona un oráculo. Considere la gráfica dirigida G, con $V = \{a,b,c,d,e,f\}$ y
$E = \{\overrightarrow{ab},\overrightarrow{ac},\overrightarrow{ad},\overrightarrow{bc},\overrightarrow{be},\overrightarrow{dc},\overrightarrow{df},\overrightarrow{ec},\overrightarrow{ef}\}$, se desea encontrar el número total de tra-

yectorias que no se interceptan e inician en el vértice a y termina en el vértice f. En la Figura 5.1 se muestra la representación geométrica de la gráfica G.

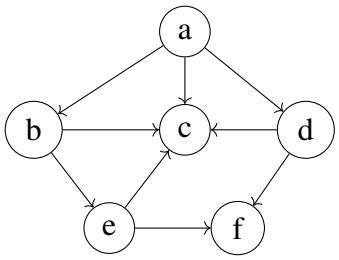
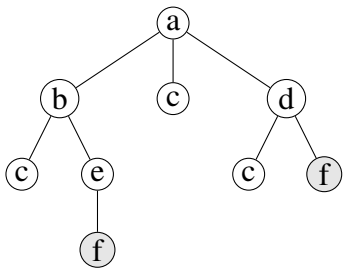
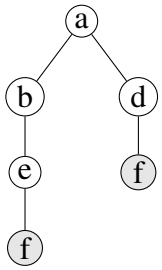


Figura 5.1: Representación geométrica de la gráfica G.

El método de enumeración estocástica empieza por asignar la raíz del árbol al vértice a. Posteriormente para el segundo nivel del árbol hay 3 potenciales sucesores. Sin embargo, las trayectorias que van a través del vértice c nunca llegan al vértice d. Así que el vértice c puede ser eliminado de los sucesores de a. El análisis que determina que a través del vértice c no existe alguna trayectoria satisfactoria, se llama oráculo. Este análisis se sigue para cada paso del algoritmo. En la Figura 5.2 del lado derecho se muestra el árbol original asociado al problema, del lado izquierdo el árbol reducido mediante algún oráculo.



(a) Árbol asociado a la gráfica dirigida G.



(b) Árbol reducido mediante un oráculo.

Figura 5.2: Reducción de un árbol mediante oráculos.

Algoritmo de enumeración estocástica con oráculos

El algoritmo de enumeración estocástica para calcular el costo total del árbol añadiendo oráculos se muestra en la Figura 5.3. Donde $Oracle(H(x))$ representa la función que mediante un oráculo decide qué nodos se mantienen como hiperhijos.

Algoritmo 7: Enumeración estocástica con oráculos.

Entrada Árbol con nodo raíz u.
Salida Estimador C del costo total del árbol y un presupuesto $B \geq 1$.
$D \leftarrow 1$; //Representa el valor $D(\overline{X})$
$X \leftarrow \{u\}$; //Representa los nodos seleccionados en cada iteración
$C \leftarrow c(u)$; //Representa el valor $C(X)$
mientras $|S(X)| > 0$ **hacer**
  $D \leftarrow \frac{|S(X)|}{|X|} D$; //Estima el valor $D(\overline{X}_t)$
  $X \leftarrow$ sucesor uniformemente elegido en *Oracle(H(X))*;
  $C \leftarrow C + \frac{c(X)}{|X|} D$; //Estima el costo del nivel t y lo añade al costo total
devolver C

Figura 5.3: Algoritmo de enumeración estocástica con oráculos.

5.2. Conteo de trayectorias

El problema de conteo de trayectorias, como su nombre lo indica, consiste en determinar el número total de trayectorias posibles que se pueden seguir desde un vértice inicial hasta un vértice final en una gráfica dada.

Debido a que este problema está relacionado con analizar la conectividad entre vértices, puede tener aplicaciones en redes y transporte, redes sociales y recomendaciones, biología y genética, teoría de juegos y economía, entre otros.

Enumeración estocástica en trayectorias

Considere una gráfica G. Se desea encontrar el número total de trayectorias con u como vértice inicial y con v como vértice final. Para contar trayectorias utilizando el método de enumeración estocástica, el primer paso es asignar el vértice inicial u, como la raíz del árbol asociado. Los posible nodos hijos de la raíz son los nodos vecinos de u en la gráfica G. En general, para cada nodo en el árbol sus posibles nodos hijos son sus nodos vecinos en la gráfica G. Se utiliza un oráculo para filtrar los posibles nodos hijos. Se tiene una hoja cuando el nodo del árbol es el vértice final v. Las hojas tienen costo uno y el resto de nodos tienen costo cero. Así al momento de calcular el costo del árbol se estará calculando el número total de trayectorias que tienen a u como vértice inicial y a v como vértice final.

Ejemplo 5.2 Suponga que se tiene la gráfica G_1, con conjunto de vértices $V = \{A,B,C,D,E\}$ y conjunto de aristas $E = \{e_1 = AC, e_2 = AB, e_3 = AE, e_4 = CE, e_5 = AE, e_6 = BD, e_7 = DE\}$. La representación geométrica de la gráfica G se muestra en la Figura 5.4.

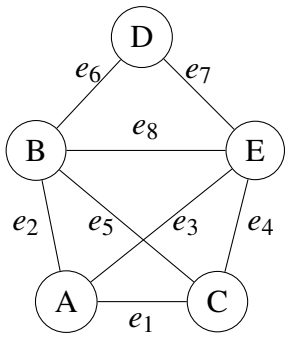


Figura 5.4: Representación geométrica de la gráfica G_1.

Se busca encontrar el número total de trayectorias que tengan como vértice inicial el vértice A y como vértice final el vértice E. Al tener una gráfica con un número pequeño de vértices no es difícil obtener todas las trayectorias de forma analítica. Las trayectorias con A como vértice inicial y E como vértice final son:

$$(e_1,e_4), \quad (e_1,e_5,e_6,e_7), \quad (e_1,e_5,e_8), \quad (e_2,e_8),$$
$$(e_2,e_6,e_7), \quad (e_2,e_5,e_4), \quad (e_3).$$

La notación utilizada para representar las trayectorias se basa en las aristas de la gráfica, con el objetivo de simplificar la notación. El árbol asociado a este problema se muestra en la Figura 5.5. Una vez se cuenta con el árbol asociado se puede estimar el costo a través de cualquier método de conteo de árboles. En este caso el número de vértices y de aristas es bajo, por lo cual, se puede saber con total certeza el número de trayectorias posibles. A continuación se presenta cómo se haría la estimación usando el método de enumeración estocástica con $B = 2$.

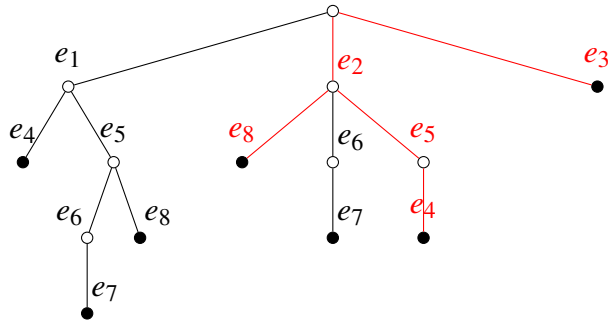


Figura 5.5: Árbol asociado con las trayectorias del vértice u al vértice v, en la gŕafica G_1.

Para realizar la estimación del costo total del árbol a través del método de enumeración estocástica con $B = 2$, se necesita como información la gráfica, el vértice inicial y el vértice final. Considere los valores C y D como en el algoritmo de enumeración estocástica con oráculos visto en la Figura 5.3.

El vértice inicial será la raíz del árbol. En este problema cada nodo en el árbol asociado se asocia a una trayectoria, el nodo raíz hace referencia a una trayectoria que empieza con el vértice u.

Se empieza asignando $D = 1$, $C = 0$. Se revisan las posibles trayectorias que inicien con el vértice u, en este caso se tienen los nodos asociados a las trayectorias: (e_1), (e_2), (e_3). Para cada nodo se revisa que exista una trayectoria del vértice A al vértice E. Esta revisión es lo que se llama oráculo. En este caso se puede utilizar algún recorrido, como BFS o DFS, para verificar que, en efecto, exista alguna trayectoria deseada, que empiece con alguna de las aristas anteriormente mencionadas.

En la primera iteración, se tiene que para cada uno de los nodos hijos de la raíz existe al menos una trayectoria favorable. Se actualiza el valor de $D = (3/1)(1) =$

3. Como se tiene que $B = 2$, se seleccionan aleatoriamente dos nodos, suponga que los nodos seleccionados son los asociados con las trayectorias (e_2) y (e_3). El nodo asociado con la trayectoria (e_2) tiene costo costo cero. En el caso del nodo asociado con la trayectoria (e_3), se tiene que llega al vértice E, lo que implica que se llega a una hoja, por lo cual, el nodo tiene costo igual a 1. Así se hace una nueva asignación al costo, $C = 0 + (1/2)(3) = 3/2$.

En la segunda iteración, los posibles nodos sucesores son los asociados con las trayectorias (e_2, e_6), (e_2, e_8), (e_2, e_5). Se comprueba para cada nodo con el oráculo elegido que existan trayectorias satisfactorias, se obtiene que para cada una de los nodos existen dichas trayectorias. Se actualiza el valor $D = (3/2)(3) = 9/2$. Se prosigue seleccionando aleatoriamente dos de los tres sucesores, suponga que se seleccionan los nodos asociados a las trayectorias (e_2, e_8) y (e_2, e_5). El nodo asociado con la trayectoria (e_2, e_5) tiene costo igual a 0. Por su parte, el nodo asociado a la trayectoria (e_2, e_8) llega al vértice E, por lo cual, tiene costo 1. Se actualiza el valor $C = 3/2 + (1/2)(9/2) = 15/4 = 3.75$.

En la tercera iteración, el nodo asociado con la trayectoria (e_2, e_5) tiene como posibles sucesores los nodos asociados a las trayectorias (e_2, e_5, e_1) y (e_2, e_5, e_4). Con el oráculo elegido se tiene que solo el nodo asociado a la trayectoria (e_2, e_5, e_4) se mantiene. Se hacen las siguientes actualizaciones, el valor $D = (1/2)(3.75) = 1.875$. Una vez se selecciona el nodo asociado a la trayectoria (e_2, e_5, e_4) se llega al vértice E, por lo cual, no hay más sucesores. Se hace la siguientes asignación, $C = 3.75 + (1)(1.875) = 5.625$.

Al no haber más nodos sucesores, se termina la ejecución del algoritmo. La estimación del costo a través del método de enumeración estocástica es 5.625 que difiere de 1.375 del verdadero valor. En la Figura 5.5 de color rojo se muestran las trayectorias usadas para calcular el costo del árbol. •

En el ejemplo que se presentó, se tenía una gráfica con un número reducido de vértices y aristas, lo que facilitó construir el árbol asociado a las trayectorias buscadas, sin embargo, para estimar el costo total del árbol con el método de enumeración estocástica no fue necesario construir el árbol por completo. Esto significa que, en gráficas con un mayor número de vértices y aristas, es posible estimar el número total de trayectorias sin necesidad de construir el árbol asociado por completo.

5.3. Conteo de apareamientos perfectos

El problema de conteo de apareamientos busca determinar cuántos apareamientos perfectos existen en una gráfica bipartita. Debido a que este problema ayuda a determinar todas las asignaciones posibles que cumplen ciertos requisitos o restricciones tiene aplicaciones en teoría de redes y emparejamiento estable, asignación óptima de recursos, biología y genética, entre otros.

Es interesante destacar que el problema de contar el número de apareamientos perfectos es un problema NP-duro, lo que implica que no se ha encontrado un algoritmo eficiente para resolverlo en general.

Matriz de biadyacencia

La matriz de adyacencia de una gráfica bipartita puede ser descompuesta de la siguiente manera

$$A[G] = \begin{bmatrix} 0_{p\times p} & B \\ B^T & 0_{q\times q} \end{bmatrix},$$

para alguna matriz B de tamaño $p \times q$, con $p,q \in \mathbb{R}$. Las matrices de ceros: $0_{p\times p}$ y $0_{q\times q}$. La matriz B representa a toda la gráfica y es llamada matriz de biadyacencia. La matriz B^T representa la matriz transpuesta de B. La definición formal se presenta a continuación.

Definición 5.1 *Sea $G = (V,E)$ una gráfica bipartita con descomposición $U = \{u_1,\ldots,u_p\}$, $W = \{w_1,\ldots,w_q\}$. La matriz de biadyacencia es la matriz B, con dimensión $p \times q$, compuesta de ceros y unos. En esta matriz, la entrada $b_{i,j} = 1$ si y sólo si $(u_i,w_j) \in E$.*

Enumeración estocástica en apareamientos

Suponga que se tiene la gráfica G_2 con conjunto de vértices $V = \{A,B,C,D,S,T,U,V\}$ y conjunto de aristas $E = \{AS,AT,AU,BT,BS,BV,CS,CU,CV,DV,DU,DT\}$. La representación geométrica de la gráfica G_2 se muestra en la Figura 5.6.

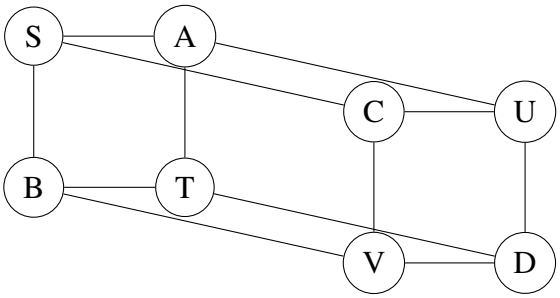


Figura 5.6: Representación geométrica de la grafica G_2.

El problema consiste en encontrar el número total de apareamientos perfectos. Para esto observe que la gráfica G_2 es una gráfica bipartita con partición de los vértices $X = \{A,B,C,D\}$, $Y = \{S,T,U,V\}$. La gráfica tiene como matriz de biadyacencia la siguiente matriz:

$$B[G_2] = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Cada renglón representa a un vértice del conjunto X, por su parte, cada columna representa a un vértice del conjunto Y.

A este problema se le puede asociar un árbol. La raíz no representa algún vértice o arista, solamente es el inicio del emparejamiento. Cada nodo representa un emparejamiento parcial. Cada hijo de la raíz representa el vértice con el que está emparejado el vértice S, los hijos de estos representa el vértice con el que se emparejará el vértice T, así con los emparejamientos de los vértices U y V. Así cada camino (x_1,x_2,x_3,x_4) en el árbol representa los vértices con los que estará emparejado los vértices S, T, U, V.

Por ejemplo, la hoja que tiene asociado el camino (A,B,C,D) tiene costo uno, debido a que $(A-S,B-T,C-U,D-V)$, es un emparejamiento perfecto. En la Figura 5.7 se muestra el árbol asociado a este problema. El costo de los nodos es cero a excepción de las hojas que correspondan a un emparejamiento perfecto.

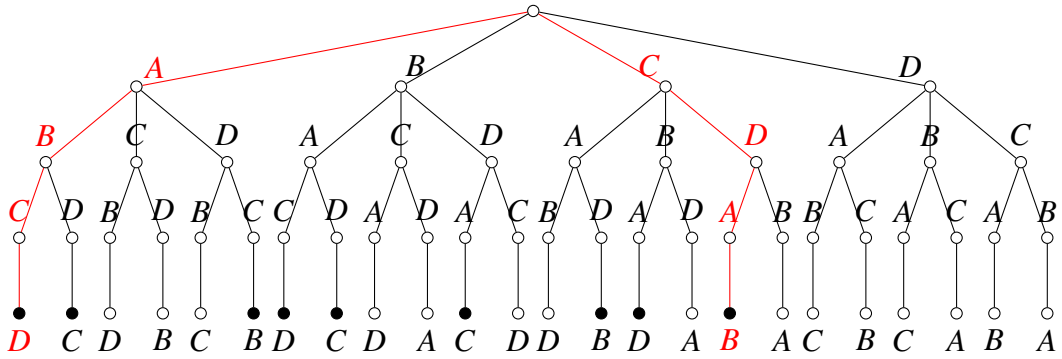


Figura 5.7: Árbol asociado a los emparejamientos perfectos de la grafica G_2.

El oráculo más utilizado en el problema de apareamiento es el método húngaro (*hungarian method*), que verifica en tiempo polinomial si un emparejamiento parcial puede ser extendido a un emparejamiento perfecto, para más detalles puede consultar [12].

Para este ejemplo se usará el método de enumeración estocástica con $B = 2$. Considere los valores C y D como en el algoritmo de enumeración estocástica con oráculos visto en la Figura 5.3. Se empieza con $D = 1, C = 0$.

En la primera iteración, la raíz tiene cuatros posibles hijos, sin embargo, una vez que se aplica el oráculo, quedan solo tres hijos, así que se actualiza el valor $D = (3/1)(1) = 3$. Se seleccionan aleatoriamente dos de los tres nodos, en este caso se eligen los nodos asociados a los emparejamientos parciales (A) y $(C-S)$, cada nodo tiene costo cero. El valor del costo sigue siendo $C = 0$.

En la segunda iteración, se tienen seis posibles hiperhijos, sin embargo, con el oráculo se descarta el nodo asociado al emparejamiento parcial $(A-S, C-T)$. Se actualiza el valor $D = (5/2)(3) = 15/2 = 7.5$. Se seleccionan aleatoriamente dos de los tres nodos, en este caso se elige el nodo asociado al emparejamiento parcial $(A-S, B-T)$ y el nodo asociado al emparejamiento parcial $(C-S, D-T)$. Cada nodo tiene costo cero, por lo cual, el costo se mantiene en cero, $C = 0$.

En la tercera iteración se tienen 4 posibles hiperhijos, sin embargo, el nodo asociado al emparejamiento parcial $(C-S, D-T, B-U)$ es descartado por el oráculo. Se actualiza el valor $D = (3/2)(7.5) = 11.25$. Se seleccionan aleatoriamente dos de los tres vértices, suponga que se eligen los nodos asociados a los emparejamietos parciales $(A-S, B-T, C-U)$ y $(C-S, D-T, A-U)$. Cada nodo tiene costo

cero, así que se mantiene $C = 0$.

En la cuarta iteración hay dos posibles hiperhijos, el oráculo no descarta ninguno, así que, se actualiza el valor $D = (2/2)(11.25) = 11.25$. Como solo se tienen dos hiperhijos, entonces, se seleccionan ambos. Los dos nodos tienen costo 1, por lo cual se actualiza el costo, $C = 0 + (2/2)(11.25) = 11.25$.

No existen sucesores por lo cual la ejecución termina. El costo estimado es 11.25, el cual difiere en 2.25 del costo total del árbol. En la Figura 5.7 de color rojo se muestran las trayectorias usadas para calcular el costo del árbol.

5.4. Conteo SAT

Problema de satisfacción (SAT)

El problema de satisfacción booleana (SAT) es fundamental en la teoría de complejidad computacional y en la lógica matemática. Específicamente, se trata de determinar si una fórmula lógica proposicional puede ser satisfecha al asignar valores de verdad a sus variables, de manera que la fórmula completa sea verdadera. Con base en [9], existen diversas formulaciones para el problema SAT pero la más común tiene en cuenta las dos componentes siguientes:

- Un vector de n variables booleanas $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, llamado asignación de verdad, representa declaraciones que pueden ser `VERDADERAS` o `FALSAS`. La negación lógica `NO` de la variable x_i es denotada $\neg x_i$. Una variable o su negación es llamada literal.
- Un conjunto de m distintas cláusulas $\{C_1, C_2, \ldots, C_m\}$ de la forma $C_i = l_{i_1} \vee l_{i_2} \vee \ldots \vee l_{i_k}$, donde cada l es una literal y $\vee$ denota el operador lógico `O`. Por ejemplo `VERDADERO` $\vee$ `FALSO` $=$ `VERDADERO`. El operador lógico `Y` se denota como $\wedge$.

El problema *SAT* puede ser formulado de la siguiente forma: Encontrar una asignación de $\mathbf{x}$ para la cual todas las cláusulas son verdaderas. El problema SAT anteriormente mencionado se representa como

$$F = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{|C_i|} l_{i_j}. \tag{5.1}$$

Definición 5.2 *Si una fórmula está expresada como en la ecuación (5.1) se dice que está en su forma normal conjuntiva (CNF).*

El problema de decidir si existe una asignación válida, y exhibirla, es llamado el problema SAT de asignación. Un problema SAT de asignación donde cada cláusula contiene exactamente K literales es llamado problema K-SAT. Encontrar el número total de asignaciones válidas es llamado el problema de conteo SAT.

Importancia

Como se mencionó anteriormente, el problema tiene una importancia en el área de la complejidad computacional, esto es debido a que el problema 3-SAT es un problema NP-completo. Un problema NP (*nondeterministic polynomial time*), en resumidas palabras, significa que no se conoce un algoritmo para resolver el problema en el peor de los casos en complejidad temporal polinomial. Sin embargo, decidir si una solución es correcta o incorrecta se puede realizar en tiempo polinomial.

Que sea NP-completo se refiere a la capacidad de ser trasladado en tiempo polinomial a otros problemas NP-completos como lo pueden ser el problema del máximo corte, del agente viajero o de coloración de gráficas. Es decir encontrar una solución efectiva para el problema 3-SAT implica en automático encontrar soluciones efectivas para otros tantos problemas.

De lo anterior el problema SAT está relacionado con resolver problemas tales como planificación y programación de horarios, diseño de circuitos integrados, arquitecturas informáticas, gráficos por computadora, determinación del estado de plegamiento de una proteína, entre otros.

Enumeración estocástica en conteo SAT

Para utilizar el método de enumeración estocástica en el problema de conteo SAT es necesario tener el problema en una forma normal conjuntiva. Una vez enumeradas las variables, la manera de asociar un árbol a este problema es empezar con la raíz que no representa algún vértice o arista, solamente es el inicio de la formula. El nodo raíz tiene dos nodos, el nodo izquierdo representa si la primera variable es verdadera y el nodo derecho si la primera variable es falsa. Después cada nodo del primer nivel vuelve a tener dos hijos que representan la segunda variable e indican si esta es verdadera o falsa, de nueva a cuenta los nodos a la izquierda

indican si la variable es verdadera y los nodos a la derecha indican si la variable es falsa. Así sucesivamente con cada variable, para finalmente tener un árbol con 2 descendientes completo y perfecto, con 2^m hojas donde m representa el número total de variables en la fórmula.

Todos los nodos tienen costo igual a cero a excepción de las hojas que representen una asignación que haga la fórmula verdadera, estas hojas tendrán un costo de uno.

Es importante mencionar que en este problema no existe un oráculo eficiente, es decir, para contar el número total de asignaciones verdaderas dependerá de la estructura de la fórmula, lo ideal es tener una fórmula en su forma normal conjuntiva, en la que cada cláusula no comparte variables con alguna otra cláusula. A continuación, se muestra un ejemplo en el que se tiene una fórmula en su forma normal conjuntiva y las cláusulas no comparten variables. Resolver problemas con fórmulas más extensas que la mostrada en el siguiente ejemplo sigue un proceso muy parecido.

Ejemplo 5.3 Considere la fórmula $(\neg A \vee B) \wedge (S \vee \neg T \vee U)$. Con 21 asignaciones válidas. El primer nivel del árbol representa si la variable A es verdadera o falsa. El segundo nivel representa si la variable B es verdadera o falsa dependiendo de si la variable A fue verdadera o falsa. Así para el resto de variables S, T y U. El árbol asociado a esta formula se muestra en la Figura 5.8.

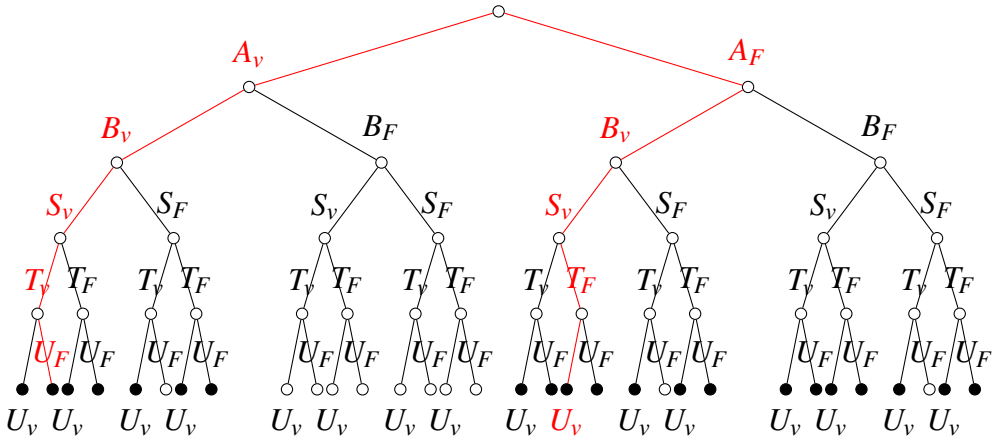


Figura 5.8: Árbol asociado a la formula $F = (\neg A \vee B) \wedge (S \vee \neg T \vee U)$. Los nodos en negro representan las asignaciones válidas.

El método de enumeración estocástica con $B = 2$ para resolver este problema funciona de la siguiente manera. Considere C y D como en el algoritmo de enumeración estocástica con oráculos visto en la Figura 5.3. Se empieza con la raíz que tiene un costo cero, entonces, $D = 1, C = 0$.

En la primera iteración, la raíz tiene dos nodos hijos que representa si la variable A es verdadera o falsa, en este caso como se puede llegar a una asignación satisfactoria no se elimina ningún nodo hijo. Entonces $D = (1)(2) = 2$. Como se tiene $B = 2$ se eligen los dos vértices. El costo se mantiene en cero, $C = 0$.

En la segunda iteración se tienen 4 hiperhijos, sin embargo el camino con A verdadera y B falsa no produce asignaciones satisfactorias por lo cual no se toma en cuenta ese nodo. Así se tiene que $D = (3/2)(2) = 3$, suponga que se eligen los vértices asociados con las asignaciones A-verdadera,B-falsa y A-falsa, B-verdadera. El costo se mantiene en cero, $C = 0$.

En la tercera iteración se vuelven a tener cuatro posibles sucesores, como la variable S pertenece a otra cláusula diferente, entonces, S no tiene restricciones y se mantienen todos los vértices. Por lo cual, $D = (4/2)(3) = 6$. Suponga que se escogen los nodos asociados con las asignaciones A-verdadera,B-verdadera S-verdadera y A-falsa, B-verdadera, S-falsa. El costo se mantiene en cero, $C = 0$.

En la cuarta iteración se vuelven a tener cuatro posibles hiperhijos, en este caso la variable T puede ser verdadera o falsa y existen asignaciones satisfactorias, por lo cual, se mantienen los cuatro hiperhijos y se hace $D = (4/2)(6) = 12$. Suponga que se eligen los nodos asociados con las asignaciones A-verdadera B-verdadera S-verdadera, T-falsa y A-falsa, B-verdadera, S-verdadera, T-falsa. El costo sigue teniendo un valor de cero, $C = 0$.

En la quinta iteración se vuelven a tener cuatro sucesores posibles. Dado que la asignación de S es verdadero, la asignación de U puede ser verdadera o falsa, por lo cual, se mantienen los cuatro hiperhijos. Se hace la asignación $D = (4/2)(12) = 24$. Suponga que se eligen los nodos asociados con los caminos A-verdadera B-verdadera S-verdadera, T-falsa, U-falsa y A-falsa, B-verdadera, S-verdadera, T-Falsa, U-verdadera. Ambos nodos representan asignaciones satisfactorias por lo cual ambas hojas tienen costo de uno. El costo se actualiza, $C = (2/2)(24) = 24$.

No existen más sucesores, así que la ejecución termina. El costo estimado es 24 que difiere en 3 del costo total del árbol. En la Figura 5.8 de color rojo se muestran las trayectorias usadas para calcular el costo del árbol. •

5.5. Confiabilidad en redes

El problema de confiabilidad en redes se refiere a la probabilidad de que una red o sistema de comunicación continúe funcionando correctamente, a pesar de la presencia de fallas o interrupciones en sus componentes. El objetivo es evaluar y mejorar la confiabilidad de una red para garantizar un funcionamiento adecuado y evitar la pérdida de datos, interrupciones del servicio o degradación del rendimiento.

Problema de estimación de la confiabilidad en redes

En esta subsección por simplicidad a la gráfica se le llamará red y a las aristas de la gráfica se les dirá enlaces.

Considere una red tal que contiene n enlaces no fiables. El estado de los enlaces está representado por un vector binario, $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ de variables aleatorias independientes Bernoulli con $P(X_i = 1) = p_i$, $i = 1, \ldots, n$, $p_i \in [0,1]$. Donde $P(X_i = 1)$ significa la probabilidad de que el componente esté funcionando correctamente.

La función de estructura, $H : \{0,1\}^n \to \{0,1\}$, representa el estado del sistema. El objetivo es estimar la probabilidad de fallo del sistema, $\overline{r} = P(H(X) = 0)$, la cual es difícil de estimar ya que todos los componentes son altamente confiables, es decir las probabilidades de dichos componentes de ser fiables son muy cercanas a 1.

En esta ocasión se hará una suposición fuerte, donde todas las aristas tienen probabilidad de fallar q, donde q es un número cercano a cero. Aunque es un caso particular no deja de ser un caso muy especial debido a que muchos de los problemas en confiabilidad de redes hacen este supuesto. En aplicaciones de la industria este es un caso muy estudiado debido a la homogeneidad de los componentes.

Bajo este supuesto se puede usar un método llamado *Spectra* que calcula la probabilidad de fallo de la red, $\overline{r}(q)$.

El enfoque del método *Spectra* considera a $\{e_1, \ldots, e_m\}$ como los enlaces de la red. Los enlaces pueden estar trabajando (*arriba*) o con fallas (*abajo*). Suponga que todos los enlaces inicialmente están arriba y por lo tanto la red está en un estado de funcionamiento. Considere $\pi = (e_{i_1}, \ldots, e_{i_m})$ una permutación de enlaces. Dada la permutación π, se empieza a suprimir enlaces, es decir, se cambia los estados

del enlace, de arriba a abajo, se recorre la permutación de izquierda a derecha y se comprueba el estado de la red después de cada paso. Se encuentra el índice j del primer enlace e_{i_j}, $j = 1,\ldots,m$, para la cual la red cambia de un estado de funcionamiento a un estado de fallo. El índice k es llamado el ancla de π y es denotado por $a(\pi)$.

Ejemplo 5.4 Sea R_1 una red con conjunto de vértices $V = \{s,t,u,v\}$ y conjunto de enlaces $E = \{e_1 = su, e_2 = vt, e_3 = ut, e_4 = sv\}$. La red R_1 está en funcionamiento si el s está conectado con el vértice t, es decir, si existe una trayectoria del vértice s al vértice t. En caso contrario la red está en fallo.

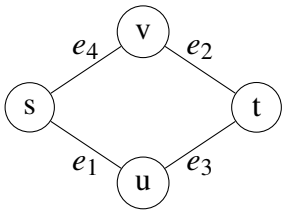


Figura 5.9: Representación geométrica de la red R_1.

Considere la permutación de enlaces $\pi_1 = (e_2, e_4, e_3, e_1)$, en un principio todos los enlaces están arriba, es decir, la red cuenta con todos los enlaces. Suprimir un enlace, se puede entender como quitar un enlace de la red. Entonces, hasta que se suprime el tercer enlace, de izquierda a derecha, la red cambia de un estado de funcionamiento a un estado de fallo. Por lo tanto, el ancla de π_1 es 3, es decir, $a(\pi_1) = 3$. •

Se asigna una distribución uniforme sobre el conjunto de todas las permutaciones de enlaces, esto es, $P(\Pi = \pi) = \frac{1}{m!}$ y se define el conjunto que tienen una ancla igual a k como:

$$A(k) = \{\pi : a(\pi) = k\}.$$

Definición 5.3 ***(Destrucción spectra)*** *Sea* Π *una permutación aleatoria sobre el conjunto de enlaces, se define*

$$f_k = P(\Pi \in A(k)) = \frac{|A(k)|}{m!}, \quad k = 0,\ldots,m.$$

Entonces,

$$S_p = \{f_0, f_1, \ldots, f_m\}$$

es llamado destrucción spectra o simplemente D-spectra de la red.

Definición 5.4 ***(D-spectra acumulativo)*** *El D-spectra acumulativo está definido por*

$$C_{s_p} = \{F(0), F(1), \ldots, F(m)\},$$

donde

$$F(k) = \sum_{i=0}^{k} f_i$$

.

Una noción similar se ocupa para definir *construcción spectra* o *C-spectra*, el cual es obtenido al considerar una red donde todos los enlaces están abajo. Aquí la red se encuentra en un estado de fallo. De igual forma que en D-*spectra*, sea $\pi = (e_{i_1}, \ldots, e_{i_m})$, una permutación de las aristas de la red. Con esta permutación, se empiezan a cambiar el estado de los enlaces de abajo hacia arriaba recorriendo la permutación de izquierda a derecha. El ancla de construcción de π, $(a'(\pi))$ se define como el primer índice del enlace, para la cual, la red entra en un estado de funcionamiento. De manera análoga se define el conjunto que tienen una ancla de construcción igual a k como:

$$A'(k) = \{\pi : a'(\pi) = k\}.$$

Definición 5.5 ***(Construcción spectra)*** *Sea* Π *una permutación aleatoria sobre el conjunto de enlaces, se define*

$$f'_k = P(\Pi \in A'(k)) = \frac{|A'(k)|}{m!}, \quad k = 0, \ldots, m.$$

El conjunto

$$S_p = \{f'_0, f'_1, \ldots, f'_m\},$$

es llamado construcción spectra o simplemente C-spectra de la red.

Definición 5.6 ***(C-spectra acumulativo)*** *El C-spectra acumulativo está definido por*

$$C'_{s'_p} = \{F'(0), F'(1), \ldots, F'(m)\},$$

donde

$$F'(k) = \sum_{i=0}^{k} f'_i.$$

Proposición 5.1 ***(Construcción y destrucción spectra equivalencia)*** *Para la construcción y destrucción spectra se tiene que*

$$f_k = f'_{m-k}.$$

Demostración: Sea $\pi = (e_{i_1}, e_{i_2}, \ldots, e_{i_m})$ una permutación de enlaces, se define π^r como $\pi^r = (e_{i_m}, \ldots, e_{i_1})$. Suponiendo que $\pi \in A(k)$, esto es, $a(\pi) = k$, con $k = \{0, 1, \ldots m\}$.
Note que para la permutación π se tiene que:

- Si las aristas $e_{i_1}, \ldots, e_{i_k}$ están abajo y las aristas $e_{i_{k+1}}, \ldots, e_{i_m}$ están arriba, entonces la red está en un estado de fallo.
- Si las aristas $e_{i_1}, \ldots, e_{i_k}$ están arriba y las aristas $e_{i_{k+1}}, \ldots, e_{i_m}$ están abajo, entonces la red está en un estado de funcionamiento.

A partir de estas observaciones, se sigue que $a'(\pi^r) = m - k$, esto es $\pi^r \in A'(m - k)$. Por lo tanto, si $\pi \in A(k)$ implica que $\pi^r \in A'(m-k)$, de lo cual, se puede concluir que $|A(k)| = |A'(m-k)|$. Combinando este resultado con las Definiciones 5.3 y 5.5 la prueba es completa, ya que

$$f_k = \frac{A(k)}{m!} = \frac{A'(m-k)}{m!} = f'_{m-k}.$$

■

La característica interesante de D-*spectra* y C-*spectra* es que una vez se dispone de estas se puede calcular directamente la probabilidad de fallo de la red, $\bar{r}(q)$. Para ver esto se presenta el siguiente análisis.

Definición 5.7 *Un conjunto de fallos se define como un conjunto ordenado de aristas tal que su fallo hace a la red entrar en un estado de fallo. Denotamos por* $\mathcal{N}(k)$ *al número de conjuntos de fallos de tamaño k.*

Proposición 5.2 *El número de conjuntos de fallos de tamaño k es,*

$$\mathcal{N}(k) = \binom{m}{k} F(k). \tag{5.2}$$

Demostración: Esto se cumple debido a la siguiente explicación combinatoria. El D-*spectra* acumulativo, $F(k)$, es la fracción de todos los conjuntos de fallos de tamaño k entre todos los subconjuntos de tamaño k tomados de m componentes.

■

Observe lo siguiente

- La red está en estado de fallo si y solo si está en uno de los conjuntos de fallo.
- Para una q fija, cada conjunto de fallo de tamaño k tiene probabilidad $q^k(1-q)^{m-k}$

Combinando lo anterior con la ecuación (5.2) se obtiene que

$$\begin{aligned}\bar{r}(q) &= \sum_{k=0}^{m} \binom{m}{k} F(k) q^k (1-q)^{m-k} \\ &= \sum_{k=0}^{m} \mathcal{N}(k) q^k (1-q)^{m-k} \end{aligned} \tag{5.3}$$

La ecuación (5.3) hace ver que solo hace falta calcular D-*spectra* para calcular la probabilidad de fallo de la red para cualquier valor de q.

Algoritmo PMC

Como consecuencia inmediata de la Proposición 5.1, uno puede escoger libremente si trabajar con C-*spectra* o D-*spectra*. Desafortunadamente tanto C-*spectra* como D-*spectra* generalmente no están disponibles analíticamente. Por lo anterior, uno de los enfoques para resolver este problema es a través de procesos MonteCarlo (PMC). En la Figura 5.10 se muestra el algoritmo PMC para calcular D-*spectra*. El algoritmo para calcular C-*spectra* es análogo.

Algoritmo 8: Algoritmo PMC

Entrada Gráfica $G(V,E)$ con $|E| = m$, N número de iteraciones.
Salida Estimadores $\widehat{F(k)}$ con $k = \{0,1,\dots m\}$.
$\widehat{f_0} \leftarrow 0, \widehat{f_1} \leftarrow 0, \dots, \widehat{f_m} \leftarrow 0;$
para $1,2,\cdots,N$ **hacer**
    $\Pi \leftarrow (e_{i_1},\dots,e_{i_m})$ (generar una permutación de aristas aleatoria)
    $k \leftarrow a(\Pi)$ (encontar el ancla)
    $\widehat{f_k} \leftarrow \widehat{f_k} + 1$
para $k = 0,1,\dots,m$ **hacer**
    `//Estimar D-spectra`
    $\widehat{f_k} = \frac{\widehat{f_k}}{N}$.
    `//Estimar D-spectra`
    acumulativo $\widehat{F(k)} \leftarrow \sum_{i=1}^{k} \widehat{f_i}$
devolver $\widehat{F(0)}, \widehat{F(1)}, \cdots, \widehat{F(m)}$

Figura 5.10: Algoritmo PMC para calcular D-*spectra*.

Con base en [3] y [18], para redes altamente fiables, es decir, redes con una q muy cercana a cero, el algoritmo PMC falla para producir resultados fiables, debido a que se requieren cálculos que involucran estimación de probabilidades de eventos raros. Para mejorar la estimación se puede utilizar el método de enumeración estocástica.

Enumeración estocástica para confiabilidad en redes

La manera en que se utilizará el método de enumeración estocástica será para calcular D-*spectra*. Considere el conjunto de todas las permutaciones de aristas, con esas permutaciones es posible construir un árbol al que se le dice árbol de permutaciones, donde para cada trayectoria de la raíz a una hoja corresponde a una permutación específica π. Por su parte, habrá nodos de color negro que corresponderán al ancla de destrucción asociada a cada permutación, a estos nodos se les llamará nodos de anclaje.

Ejemplo 5.5 Suponga que se tiene la red R con conjunto de vértices $V = \{s,t,u,v\}$ y conjunto de enlaces $E = \{e_1 = us, e_2 = vt, e_3 = uv, e_4 = st\}$. La red se muestra

en la Figura 5.11. Considere que la red está en un estado de funcionamiento si los nodos s y t están conectados y en estado de fallo en otro caso.

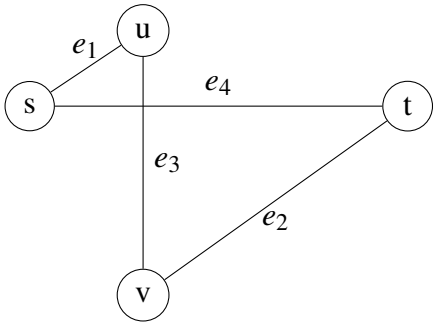


Figura 5.11: Representación geométrica de la red R.

El árbol de permutación asociado al problema se muestra en la Figura 5.12. Donde las anclas de destrucción de cada permutación de enlaces son los nodos en negro. Vea que las permutaciones (e_1, e_4, e_3, e_2) y (e_1, e_4, e_2, e_3) tienen como ancla de destrucción al enlace e_4.

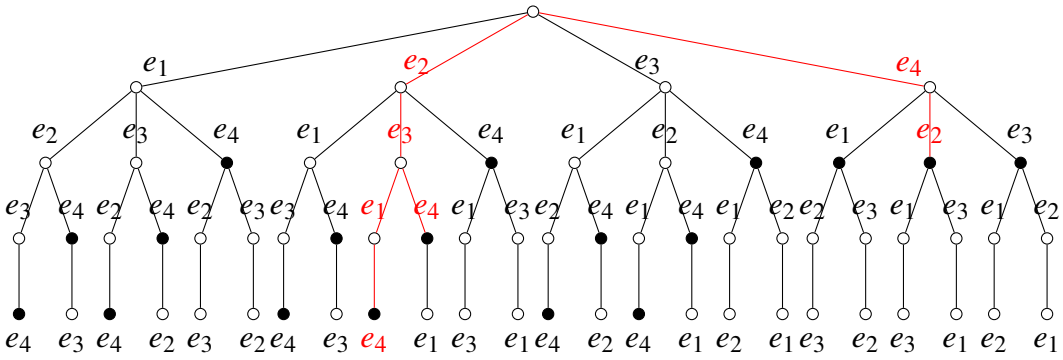


Figura 5.12: Árbol de permutación asociado a la red R.

A partir del árbol de permutación se puede calcular D-*spectra*. Note que se tienen 6 nodos de anclaje en el nivel 2 del árbol y cada uno de estos nodos, induce dos trayectorias que terminan en las hojas del árbol, así que

$$f_2 = \frac{6 \cdot 2}{4!} = \frac{1}{2}$$

En el nivel 3 del árbol se tienen 6 nodos de anclaje, cada uno de estos induce solo una trayectoria, así que

$$f_3 = \frac{6 \cdot 1}{4!} = \frac{1}{4}$$

Igualmente, en el nivel 4 se tienen 6 nodos de anclaje, cada uno de estos induce solo una trayectoria, así que

$$f_4 = \frac{6 \cdot 1}{4!} = \frac{1}{4}$$

En los niveles 0 y 1 del árbol no hay nodos de anclaje así que $f_0 = f_1 = 0$ y se tiene que

$$F(0) = 0, F(1) = 0, F(2) = \frac{1}{2}, F(3) = \frac{3}{4}, F(4) = 1.$$

Teniendo esta información se puede calcular la no fiabilidad de la red para cualquier valor de q a través de la ecuación (5.3). •

El ejemplo anterior proporciona una idea de cómo utilizar el árbol de permutación para obtener la probabilidad de fallo de la red. Esta idea se generaliza a través del siguiente teorema.

Teorema 5.1 *Si T es un árbol de permutación asociado a una red R con m enlaces, entonces*

$$f_k = \frac{\text{Total de nodos de anclaje en el nivel } k \text{ del árbol.}}{\text{Total de nodos en el nivel } k.}.$$

Demostración: Sea $L(k)$ el conjunto de nodos de anclaje en el nivel k del árbol. Considere la trayectoria de la raíz del árbol a algún vértice $v \in L(k)$. Un nodo de anclaje no puede aparecer en está trayectoria en el nivel j, con $j < k$ o $j > k$, ya que en ese caso $v \notin L(k)$, así que,

$$|A(k)| = |L(k)| \cdot (\text{El total de trayectorias inducidas de } v \in L(k)).$$

Teniendo en cuenta que cada nodo en el nivel k induce

$$(m-k)(m-(k+1)) \cdots (m-(m-1)) = (m-k)!$$

trayectorias hacia las hojas, se concluye que

$$|A(k)| = |L(k)|(m-k)!.$$

También note que el número total de nodos en el nivel k es $(m)(m-1)\cdots(m-k+1) = \frac{m!}{(m-k)!}$. Por la Definición 5.3 se tiene que

$$\begin{aligned} f_k &= \frac{|A(k)|}{m!} \\ &= \frac{|L(k)|(m-k)!}{m!} \\ &= \frac{|L(k)|}{\frac{m!}{(m-k)!}} \\ &= \frac{\text{Total de nodos de anclaje en el nivel } k \text{ del árbol.}}{\text{Total de nodos en el nivel } k.} \end{aligned}$$

■

El teorema anterior brinda una manera de calcular la probabilidad de fallo usando el método de enumeración estocástica. Los pasos a seguir son los siguientes:

1. En el árbol de permutación, se asigna costo de uno a los nodos de anclaje y costo de cero al resto de nodos. Se utiliza el método de enumeración estocástica para estimar los costos de cada nivel del árbol.
2. Estimar el valor f_k para cada nivel k del árbol. Esto se hace estimando el costo del nivel k y dividiendolo sobre $m!/(m-k)!$
3. Una vez se tienen estimados los valores f_k, calcular F_k.
4. Una vez se cuenta con F_k para cada nivel k del árbol, a través de la ecuación (5.3), se puede calcular la probabilidad de fallo de la red.

A continuación se muestra un ejemplo de cómo estimar $D-$*spectra* con el método de enumeración estocástica siguiendo los pasos anteriores.

Ejemplo 5.6 Considere la red R de la Figura 5.11, se utilizará el método de enumeración estocástica con $B = 2$ para estimar los costos de cada nivel del árbol de permutación. Consideré los valores de C y D como en el algoritmo de enumeración estocástica presentado en la Figura 5.3. Se inicia con raíz que tiene costo cero, entonces, $D = 1$, $C = 0$.

En la primera iteración, la raíz tiene cuatro sucesores posibles, se hace la asignación $D = 4$. Suponga que se elige los nodos asociados a las trayectorias (e_2) y

(e_4). Ambos nodos tienen costo cero, ya que para ninguna trayectoria son nodos de anclaje. El costo se mantiene en cero, $C = 0$. Así que el valor estimado de f_1 es 0.

Para la segunda iteración, se tienen 6 hiperhijos, se hace la asignación $D = (6/2)(4) = 12$. Suponga que se eligen el nodo asociado a la trayectoria (e_2, e_3) cuyo costo es cero y el nodo (e_4, e_2) cuyo costo es uno, debido a que es un nodo de anclaje. Se tiene que el costo estimado del segundo nivel es $C = (1/2)(12) = 6$.

En la tercera iteración se tienen dos hiperhijos. El valor $D = (2/2)(12) = 12$. Al ser solo dos nodos se eligen ambos, el nodo asociado a la trayectoria (e_2, e_3, e_1) cuyo costo es cero, y el nodo asociado a la trayectoria (e_2, e_3, e_4) cuyo costo es uno por ser un nodo de anclaje. El costo estimado del tercer nivel es $C = (1/2)(12) = 6$.

Para la cuarta iteración se cuenta con un solo sucesor. Se hace la asignación $D = (1/2)(12) = 6$. Se elige al único sucesor, el cual tiene asociada la trayectoria (e_2, e_3, e_1, e_4). El costo estimado del cuarto nivel es $C = (1/1)(6) = 6$.

En la Figura 5.12 de color rojo se muestran las trayectorias usadas para calcular el costo del árbol.

Una vez se tienen los costos por nivel se pueden estimar los valores de f_1, f_2, f_3, f_4, denotaremos por $\widehat{f_i}$ al valor estimado de f_i y por $\widehat{F(i)}$ al valor estimado de $F(i)$.

Se tienen los siguientes resultados:

$$
\begin{aligned}
\widehat{f_0} &= 0, \\
\widehat{f_1} &= 0, \\
\widehat{f_2} &= \frac{6}{\frac{4!}{(4-2)!}} = \frac{6}{12} = \frac{1}{2}, \\
\widehat{f_3} &= \frac{6}{\frac{4!}{(4-3)!}} = \frac{6}{24} = \frac{1}{4}, \\
\widehat{f_4} &= \frac{6}{\frac{4!}{(4-4)!}} = \frac{6}{24} = \frac{1}{4}.
\end{aligned}
$$

Las estimaciones de *D-spectra* acumulativo son

$$\widehat{F(0)} = 0, \quad \widehat{F(1)} = 0, \quad \widehat{F(2)} = \frac{1}{2}, \quad \widehat{F(3)} = \frac{3}{4}, \quad \widehat{F(4)} = 1.$$

En este caso, las estimaciones son exactas, sin embargo, es importante señalar que esto no siempre está garantizado.

•

Conclusiones

Este trabajo ha proporcionado una visión general y accesible sobre el problema de conteo de árboles y cómo enfrentarlo de manera eficiente con el método de enumeración estocástica. Se han presentado los fundamentos matemáticos y las herramientas probabilísticas necesarias para analizar y comprender cómo se aplica el problema de conteo de árboles. Los capítulos nos han llevado desde la teoría de gráficas hasta la presentación de estructuras de datos y algoritmos clave, permitiendo desarrollar una comprensión sólida de las diferentes técnicas de conteo de árboles.

Se hizo un énfasis especial en el método de enumeración estocástica, que es una generalización del algoritmo de Knuth. Este método nos permite obtener un estimador de varianza cero, lo que significa que podemos confiar en que obtendremos un costo preciso con un presupuesto adecuado. Un ejemplo de su utilidad es contar el número exacto de trayectorias entre dos vértices de una gráfica. Sin embargo, es importante destacar que este método no es la forma más óptima de realizar este cálculo. Hasta ahora, no se ha encontrado un algoritmo óptimo conocido para resolver este problema.

Como trabajo a futuro, se planea implementar los algoritmos en un lenguaje de programación eficiente en términos de uso de recursos y velocidad de ejecución, como lo puede ser, el lenguaje de programación “C”. También se investigará las estructuras de datos adecuadas para implementar estos algoritmos, teniendo en cuenta los posibles oráculos que se pueden utilizar.

Un tópico interesante de explorar es la relación entre las redes bayesianas (ver [11]) y el conteo de árboles. Se buscará comprender cómo estas dos áreas se conectan y si existe alguna aplicación práctica de esta relación.

Bibliografía

[1] Bang-Jesen J., Gutin G., *Digraphs: theory, algorithms and applications.* Springer-Verlag, 2nd ed. (2002).

[2] Biggs N., *Algebraic graph theory.* Cambridge Mathematical Library, 2nd ed. (1993).

[3] Botev Z. I., L'Ecuyer P., Rubino G., Simard R., Tuffin B., *Static network reliability estimation via generalized splitting.* INFORMS J. Comput., vol. 25, no. 1, pp. 56–71, (2013).

[4] Canek P., *Estructura de datos con Java moderno*. Las Prensas de Ciencias, (2018).

[5] Chang S. K., *Data structures and algorithms*. World Scientific, Vol. 13, (2003).

[6] Chartrand G., Lesniak L., Zhang P., *Graphs & digraphs*. CRC Press, 6th ed. (2016).

[7] Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., *Introduction to algorithms*. The MIT Press, 3rd ed. (2009).

[8] González-M. D. A., *Introducción a la teoría de gráficas*. Departamento de Matemáticas Aplicadas y Sistemas. Universidad Autónoma Metropolitana - Cuajimalpa. (2017).

[9] Gu J., Purdom P. W., Franco J., Wah B. W., *Algorithms for the satisfiability (SAT) problem: a survey. In satisfiability problem: theory and applications*. DIMACS Series in Discrete Mathematics, Vol. 35, (1996).

[10] James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. *Resampling methods. An introduction to statistical learning: with applications in python*. Cham: Springer International Publishing, (2023).

[11] Koller D, Friedman N., *Probabilistic graphical models: principles and techniques*.
MIT press, (2009).

[12] Kuhn H. W., *The Hungarian method for the assignment problem*.
Naval Research Logistics Quarterly, DOI 2:83–97, (1955).

[13] Rincón L., *Introducción a la probabilidad*.
Las Prensas de Ciencias, 1era. ed. (2014).

[14] Ross S., *A first course in probability*.
Pearson, 8th ed. (2010).

[15] Rubinstein R. Y., Kroese D. P., *Simulation and the Monte Carlo method*.
Wiley, 3rd ed., (2017).

[16] Skiena S. S., *The algorithm design manual*.
Springer, 3rd ed. (2020).

[17] Vaisman R., Kroese D. P., *Stochastic enumeration method for counting trees*.
Springer, DOI 10.1007/s11009-015-9457-4, (2015).

[18] Vaisman R., Kroese D. P., Gertsbakh I. B., *Improved sampling plans for combinatorial invariants of coherent systems.*
IEEE Transactions on Reliability, DOI 10.1109/TR.2015.2446471, (2014).

[19] Zhang, Y., Wu, H. and Cheng, L., *Some new deformation formulas about variance and covariance*.
IEEE, (2012).