Análisis de un algoritmo más eficiente para determinar la similitud entre genomas con un enfoque en la teoría de gráficas

José Guadalupe de Jesús Marín Parra[1], Erik Rangel Limón[1]

1. *Facultad de Ciencias, Universidad Nacional Autónoma de México, Ciudad Universitaria, Coyoacán 04510, CDMX, México*

1. Introducción

La matriz de distancias de distintos pares de genoma han sido clave para el análisis filogenético de las especies, por lo general la matriz de distancias de n especies es una matriz $n \times n$ tal que toda entrada a_{ij} determina un valor que representa qué tan lejana es la especie i de la especie j; una especie muy parecida a otra tendrá un valor menor en su entrada (si son la misma, entonces será 0), y mientras más distintas sean, este valor crecerá.

Hay muchas formas de crear ésta matriz de distancias, una de las más comunes es mediante los algoritmos de alineamiento de secuencias, sin embargo éstos algoritmos tienen una complejidad considerablemente alta, tan sólo la propuesta de *Needleman-Wunsch* toma tiempo $O(nm)$ siendo n y m el tamaño de dos secuencias respectivamente, por lo que si queremos analizar dos secuencias de una misma especie, ésto tomaría tiempo $O(n^2)$, lo cual haría muy ineficiente crear una matriz de distancias que contempla todas las comparaciones posibles entre dos especies.

Es posible crear una matriz de distancias prescindiendo del algoritmo de alineamiento, usando un algoritmo enfocado en la teoría de gráficas propuesto en el artículo *A new graph-theoretic approach to determine the similarity of genome sequences based on nucletide triplets*[1], en donde la comparación de dos secuencias toma tiempo $O(n)$ siendo n la mayor de las secuencias según su longitud.

Este método permite que la matriz de distancias pueda ser creada con diversas secuencias en un tiempo considerablemente menor que usando cualquier algoritmo de alineamiento.

La idea de usar gráficas de manera general se utiliza para contar los distintos tripletes de nucleótidos que hay en una secuencia, lo cual puede ser útil para muchos otros usos.

1.1. Objetivos

- Repasaremos algunas definiciones que utilizaremos sobre la teoría de gráficas.
- Describiremos el funcionamiento del algoritmo.
- Compararemos la complejidad de la creación de la matriz de distancias utilizando el algoritmo de alineamiento con la creación de la misma utilizando el algoritmo enfocado en gráficas.
- Implementaremos el algoritmo.
- Probaremos el algoritmo para crear el arbol filogenético de varias especies por medio del algoritmo UPGMA.

2. Métodos

El algoritmo enfocado a la teoría de gráficas es un método muy conveniente para analizar secuencias, en éste usamos lo que se llama una gráfica bipartita completa $K_{16,4}$ para modelar todos los posibles tripletes de nucléotidos.

La base de éste algoritmo es crear una gráfica bipartita para cada secuencia que queramos analizar, y a cada una de las aristas les asignaremos un peso según el número de veces que aparezca el tri-nucleótido modelado por la arista.

2.1. Sobre teoría de gráficas

Una gráfica es un par ordenado (V, A) donde V es el conjunto de vértices y A es un conjunto de aristas. El conjunto de aristas puede modelar una relación arbitraria simétrica y no reflexiva entre dos vértices de V (recordemos que la simetría en una relación $\mathcal{R}$ quiere decir que $\forall_{a,b} a\mathcal{R}b \Rightarrow b\mathcal{R}a$, y reflexividad es que $\forall_a a\mathcal{R}a$); por lo tanto para una relación arbitrariaria $\mathcal{R}$ entre los vértices de V, podemos definir al conjunto de aristas como sigue:

$$A = \{uv \mid u, v \in V, u\mathcal{R}v\}$$

(a) Gráfica G (b) Gráfica H

Figura 1: Ejemplos de gráficas cualesquiera

Sabiendo ésto, recordaremos algunas definiciones sobre teoría de gráficas que utilizaremos en el algoritmo:

- Dada una gráfica G, denotaremos como $V(G)$ a su conjunto de vértices, y denotaremos como $A(G)$ a su conjunto de aristas.
- Sean $u, v \in V(G)$ diremos que u y v son *adyacentes* si y sólo si existe la arista $uv \in A(G)$.

(a) Los vértices u y v son *adyacentes* (b) Los vértices w y y **no** son *adyacentes*

Figura 2: Ejemplos de adyacencias

- Sea A un subconjunto de $V(G)$, diremos que A es un *conjunto independiente de vértices* si y sólo si para todo par de vértices $u, v \in A$, u y v **no** son adyacentes.

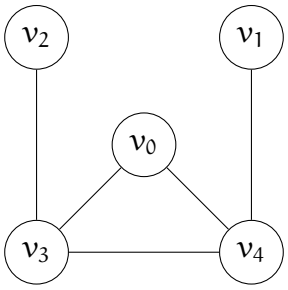


Figura 3: $\{v_0, v_1, v_2\}$ es un *conjunto independiente*, pues no existen adyacencias entre ellos

- Diremos que una gráfica G es *bipartita* si y sólo si existe una partición de $V(G)$ en dos conjuntos V_1, V_2 tales que V_1 y V_2 son *conjuntos independientes de vértices*.

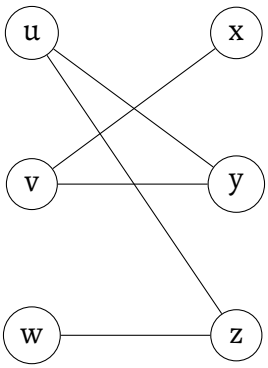
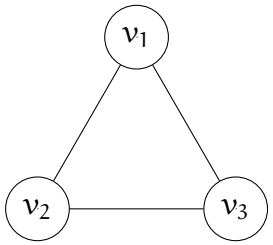
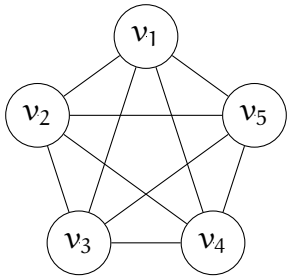


Figura 4: Ejemplo de gráfica *bipartita*. $V_1 = \{u, v, w\}$ y $V_2 = \{x, y, z\}$

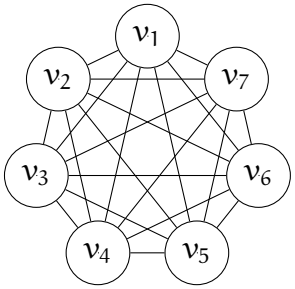
- Una gráfica es *completa* si y sólo si para todo par de vértices $u, v \in V(G)$, u y v son adyacentes. Éstas serán denotaras por K_n, con n el número de vértices.



(a) Gráfica K_3



(b) Gráfica K_5



(c) Gráfica K_7

Figura 5: Ejemplos de gráficas K_n

- Una gráfica será *bipartita completa* si y sólo si la gráfica es bipartita por dos conjuntos V_1, V_2 y además para todo vértice $u \in V_1$, u es adyacente a todo vértice $v \in V_2$. Esta gráfica se denotará como $K_{i,j}$ siendo i la cardinalidad de V_1 y j la cardinalidad de V_2.

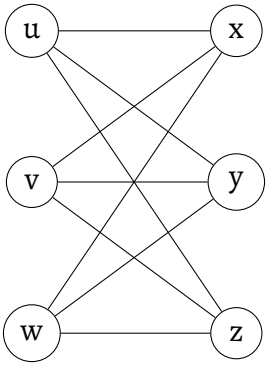


Figura 6: Ejemplo de gráfica *bipartita completa* $K_{3,3}$

- Para denotar el peso que va a tener una arista en una gráfica G, se tiene que definir una función $w : A(G) \to \mathbb{R}$.

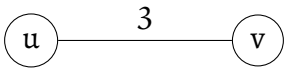
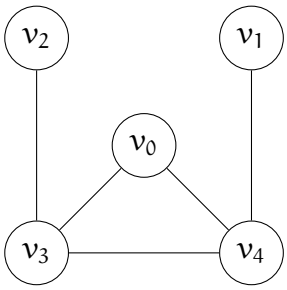


Figura 7: Ejemplo de peso de una arista $w(uv) = 3$

- La matriz de adyacencias de una gráfica G, digamos M, será una matriz bidimensional $n \times n$ siendo n el número de vértices de G, si los vértices son $V(G) = \{v_1, \ldots, v_n\}$, la entrada M_{ij} de la matriz sera 1 si y sólo si $v_i v_j \in A(G)$, y será 0 en cualquier otro caso.



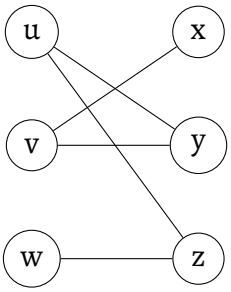
(a) Gráfica G

$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

(b) Matriz de adyacencias de G

Figura 8: Ejemplo de matriz de adyacencias

- La matriz de adyacencias de una gráfica bipartita por conjuntos independientes V_1 y V_2 se puede reducir a una matriz de tamaño $|V_1| \times |V_2|$, dado que sabemos que entre los vértices de V_1 nunca habrá una adyacencia, y tampoco entre los vértices de $|V_2|$. De esta manera, la matriz de adyacencia reducida de una gráfica *bipartita completa* será una matriz que tiene 1 en todas sus entradas.



(a) Gráfica *bipartita*

$$\begin{array}{c} \\ u \\ v \\ w \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

(b) Matriz de adyacencias reducida

Figura 9: Ejemplo de gráfica *bipartita* y su respectiva matriz de adyacencias reducida.

2.2. Conteo de tri-nucleótidos por medio de gráficas

Lo que hacemos en éste algoritmo es utilizar una gráfica $K_{16,4}$ que representan a todos los tripletes de nucléotidos, en donde el conjunto independiente de 16 vértices son todas las posibles formas de tener una cadena con dos caracteres de "A", "C", "T" o "G"; mientras que el conjunto de 4 vértices son cada uno de los caracteres mencionados anteriormente.

De forma visual, podemos ver a la gráfica mencionada como sigue:

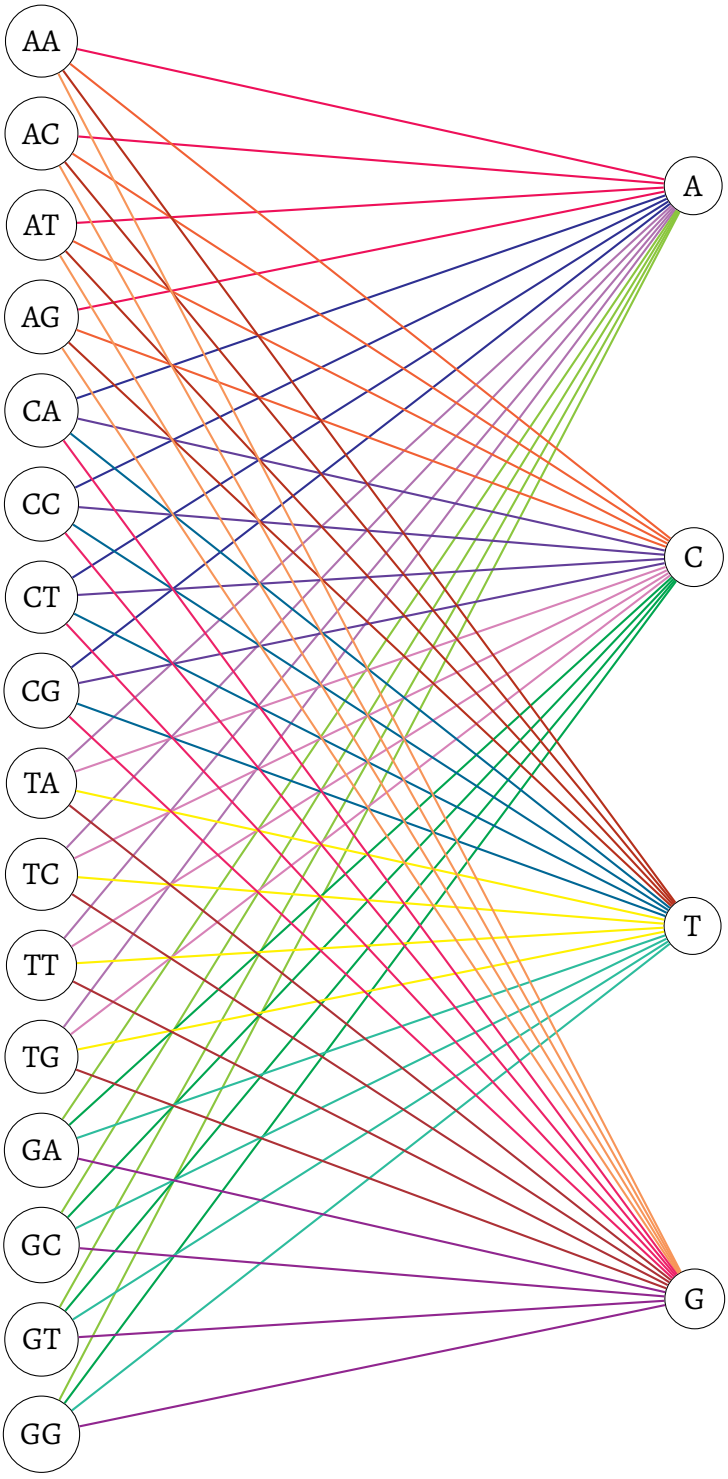


Figura 10: Gráfica bipartita de posibles tripletes de nucleótidos (Los colores sólo son para diferenciar mejor a las aristas)

De esta manera, para cada secuencia que queramos analizar, generaremos una matriz de tamaño 16×4, que representaría la matriz de adyacencias reducida de la gráfica bipartita de posibles tripletes de nucleótidos; sin embargo, como todas las entradas serían iguales, llenaremos dicha matriz con el peso de la arista que representa cada entrada.

Como dijimos anteriormente, el peso de la arista representará las veces que aparece dicho tri-nucleótido en la secuencia, y además no consideraremos la posición en la que se encuentra. Por ejemplo, veamos cuál sería el peso de las aristas en la subsecuencia hipotética "ACCTGATTGGAC".

A	C	C	T	G	A	T	T	G	G	A	C
A	C	C	·	·	·	·	·	·	·	·	·
·	C	C	T	·	·	·	·	·	·	·	·
·	·	C	T	G	·	·	·	·	·	·	·
·	·	·	T	G	A	·	·	·	·	·	·
·	·	·	·	G	A	T	·	·	·	·	·
·	·	·	·	·	A	T	T	·	·	·	·
·	·	·	·	·	·	T	T	G	·	·	·
·	·	·	·	·	·	·	T	G	G	·	·
·	·	·	·	·	·	·	·	G	G	A	·
·	·	·	·	·	·	·	·	·	G	A	C

Tabla 1: Tripletes de nucleótidos contenidos en la subsecuencia.

De esta manera, en la arista que representa cada triplete de nucleótido que aparece en la tabla anterior se anotará su correspondiente peso, y como cada triplete aparece sólo una vez, entonces cada uno de ellos tendrá peso uno, mientras que el resto será 0.

2.3. Creación de la matriz de distancias

Como resultado del proceso anterior, si analizamos k secuencias distintas, obtendríamos una matriz tridimensional R de tamaño $k \times 16 \times 4$. Para crear la respectiva matriz de distancias, crearemos una matriz bidimensional D de tamaño $k \times k$, en donde toda entrada D_{xy} con $x, y \in \{0, \ldots, k-1\}$ tendrá el siguiente valor:

$$D_{xy} = \frac{\sum_{i=0}^{15} \sum_{j=0}^{3} |R_{xij} - R_{yij}|}{64}$$

Notemos entonces que $|R_{xij} - R_{yij}|$ nos estaría dando la diferencia entre las apariciones del triplete de nucleótido ij de las secuencias x y y, de manera que ésta operación es el promedio aritmético de todas estas diferencias. Y en efecto, cumplirá que mientras mayor sea el número de la entrada D_{xy}, más diferentes serán las secuencias x y y una de otra.

2.4. ¿Por qué este algoritmo calcula efectivamente la diferencia entre dos secuencias del DNA?

Éste no es el primer algoritmo con el que se ha pretendido calcular la diferencias entre dos secuencias prescindiendo del algoritmo de alineamiento, anteriormente ha habido varios intentos con el uso de métodos numéricos, de los cuales destacamos aquellos que presentan un acercamiento probabilístico, pues de ahí proviene la idea central de nuestro algoritmo.

En estos métodos, se suelen usar lo que se conoce como la frecuencia de palabras o k-meros, en donde se calcula la frecuencia que cierta subcadena de tamaño k, donde k es cualquier entero positivo. En éstos métodos se ha mostrado que para ciertos valores de ésta k, ésta puede mostrar una buena efectividad al calcular la distancia entre ciertas secuencias, pero no con todas las secuencias se tiene un resultado bueno. Por lo que el encontrar ésta k óptima se vuelve necesario.

En el artículo *Optimal choice of k-mer in composition vector method for genome sequence comparison*[2] se muestra que el tomar ésta k como 3 ha mostrado resultados satisfactorios en la comparación completa del genoma en la mayoría de los casos.

Tomando como base entonces las subcadenas de 3 nucleótidos y calculando su respectivo vector de frecuencia a partir de un modelo de gráfica bipartita, es que tenemos una forma de determinar la diferencia entre dos secuencias.

2.5. Ajuste para distintas subcadenas de tamaño k

Es posible realizar el algoritmo variando el tamaño de esta k si se requiere, en este caso sólo tomaremos $k = 3$. De manera general, los conjuntos independientes se pueden tomar como las posibles formas de tener subcadenas de $\lceil \frac{k}{2} \rceil$ y $\lfloor \frac{k}{2} \rfloor$ elementos respectivamente; sin embargo entre mayor sea ésta k mayor es el espacio que se requiere para almacenar la matriz de la gráfica bipartita.

2.6. Cambios en la implementación

En la implementación original del algoritmo, para cada secuencia a analizar, itera sobre todos los posibles tripletes de nucleótidos, guardando en su respectiva entrada el número de veces que aparece en la cadena; sin embargo, hacerlo de esta forma requiere de recorrer la misma cadena 64 veces, y considerando que una sola cadena puede tener una longitud considerablemente grande, esta tarea puede tomar demasiado tiempo.

En nuestra implementación la matriz tridimensional tendrá 0 en todas sus entradas, y recorreremos cada caracter tomando a sus dos caracteres siguientes, y sumaremos 1 a su respectiva entrada en la matriz, recorriendo así una sola vez la cadena completa.

Para la construcción de la matriz de distancias seguiremos el mismo proceso que describe el artículo sin mayores cambios.

2.7. Complejidad en tiempo y comparativa

Supongamos que queremos analizar k secuencias, de las cuales la mayor de ellas tiene longitud n. Generar la matriz de la gráfica bipartita de una secuencia, con la modificación que hicimos, debería tomar a lo más n pasos, pues se recorre una sola vez toda la secuencia. Si realizamos ésto con las k secuencias, ésto se puede realizar en tiempo $O(kn)$.

Por otra parte la matriz de distancias se crea llenando cada entrada de una matriz $k \times k$ con un promedio aritmético que toma 64 pasos, por lo que en total le toma $64k^2 \in O(k^2)$.

Ésto nos da una complejidad mejor comparado con cualquier algoritmo de alineamiento, supongamos que de igual manera llenamos una matriz de distancias $k \times k$ siendo k el número de secuencias a analizar, y para cada entrada en la matriz, si estamos en la entrada ij, entonces realizaremos el algoritmo de alineamiento *Needleman-Wunsch*, y por tanto creamos una matriz con la longitud de la secuencia i por la longitud de la secuencia j, y como n es la mayor de las secuencias, ésta matriz puede tomar un tamaño a lo más $n \times n$, donde para cada entrada en la matriz se determina un valor; tan solo hasta éste momento, nos habremos tomado n^2 pasos, y sin importar cuanto tiempo nos tome determinar la distancia entre ambas secuencias, la creación de la matriz nos habrá tomado al menos $O(k^2n^2)$.

2.8. Creación del árbol filogenético con UPGMA (Unweighted Pair Group Method using arithmetic Averages)

UPGMA, por sus siglas en inglés, es un método de agrupamiento de pares no ponderados usando promedios aritméticos, y éste algoritmo consiste en lo siguiente:

Ya teniendo la matriz de distancias, para cada secuencia analizada le corresponde el nodo de un arbol binario. Este algoritmo obtenemos la pareja (i, j) cuya distancia es menor en la matriz (con $i \neq j$), con la que formamos un nuevo árbol que tiene como hijas a las secuencias que representan i y j (izquierda y derecha respectivamente), la raíz de éste árbol sería la distancia que tenían antes el par (i, j) entre dos.

Si $\mathcal{A}$ es un árbol binario, denotaremos como $|\mathcal{A}|$ al número de hojas del árbol.

De la matriz eliminamos las filas y columnas correspondientes a los elementos i y j, supongamos que éstos eran los clústeres $\mathcal{A}$ y $\mathcal{B}$, y en su lugar agregamos la fila y columna correspondiente para el elemento $\mathcal{A} \cup \mathcal{B}$; sin embargo hay que actualizar las entradas incidentes con $\mathcal{A} \cup \mathcal{B}$, para ello utilizaremos la siguiente fórmula para añadir la distancia de una entrada incidente (digamos $\mathcal{X}$) con la unión especificada.

$$d_{\mathcal{A}\cup\mathcal{B},\mathcal{X}} = \frac{|\mathcal{A}|d_{\mathcal{A},\mathcal{X}} + |\mathcal{B}|d_{\mathcal{B},\mathcal{X}}}{|\mathcal{A}| + |\mathcal{B}|}$$

Y repetimos éste proceso hasta tener un sólo árbol posible.

Esta se trata de una implementación trivial, que en cada iteración disminuye en uno el número de árboles que representa la matriz de distancias y termina cuando sólo queda una, y en cada iteración se busca a la pareja menor en la matriz, lo cual toma tiempo a lo más $O(n^2)$, el resto de pasos se hace con una iteración de menor tamaño, por lo que al final, éste algoritmo tiene complejidad $O(n^3)$ en tiempo.

3. Resultados

Para recrear las pruebas es necesario tener instalado lo siguiente:

- `Python3` para el script en general (*En nuestro caso fueron realizadas con la versión* `3.10.8`)
- Paquetería `binarytree` para el uso de árboles binarios como estructura de datos.
- `Graphviz` para la graficación del árbol filogenético en archivo pdf.

Realizamos dos pruebas: El árbol filogenético de 41 mamíferos a partir de su DNA mitocondrial y el árbol filogenético de 59 genomas completos pertenecientes a bacterias de 15 familias distintas.

La primera prueba es rápida pues los archivos consultados tienen un tamaño de 16,000 pares de bases en promedio; y todos los archivos para recrear éste árbol filogenético se encuentran en el siguiente repositorio.

La segunda prueba es más intensiva pues en ésta los archivos consultados tienen un tamaño de 3,000,000 pares de bases en promedio; para recrear éste árbol filogenético es necesario primero descomprimir el archivo en `src/input/bacteria.zip` del repositorio.

Ambos árboles se generan con la siguiente orden:

```
$ python graph_comparison.py
```

3.1. Mamíferos

Resultado completo de las pruebas en mamíferos
Ubicado en `src/output/mamiferos.pdf` dentro del repositorio.

Veamos la siguiente tabla con cada uno de los mamíferos analizados dado el animal y la longitud de su genoma.

Mamíferos y su respectiva longitud de genoma.

Mamífero	Tamaño del genoma (pares de bases)
Human	16569
Pigmy chimpanzee	16563
Common chimpanzee	16554
Gorilla	16364
Gibbon	16472
Baboon	16521
Vervet monkey	16389
Bornean orangutan	16389
Sumatran orangutan	16499
Cat	17009
Dog	16727
Pig	16680
Sheep	16616
Goat	16640
Cow	16338
Buffalo	16355
Wolf	16774
Tiger	16990
Leopard	16964
Indian Rhinoceros	16829
White Rhinoceros	16832

Mamíferos y su respectiva longitud de genoma (continuación).

Black Bear	16868
Brown Bear	17020
Polar Bear	17017
Giant Panda	16805
Rabbit	16245
Hedgehog	17447
MacacaThibet	16586
Squirrel	16507
Dormouse	16602
Blue whale	16402
Bowhead Whale	16390
Chiru	16498
Common warthog	16719
Donkey	16670
Fin Whale	16398
Gray Whale	16412
Horse	16660
Indus River Dolphin	16324
North Pacific Right Whale	16386
Taiwan serow	16524

Lo que nos arroja un árbol filogenético en el cual podemos apreciar diferencias y similitudes en cada mamífero que analizamos.

Veamos algunos ejemplos de esto en nuestras especies.

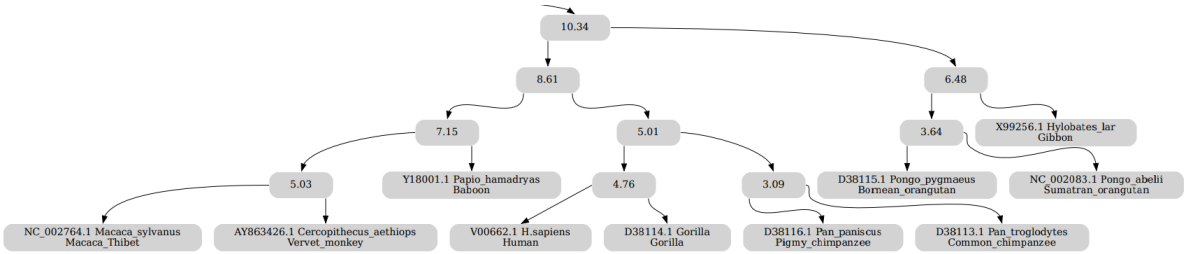


Figura 11: Marca la similitud entre *Macaco del Tibet*, *Mono Verde*, *Babuino*, *Humano*, *Gorila*, *Chimpancé Pigmeo*, *Chimpancé Común*, *Orangután Borneo*, *Orangután de Sumatra* y el *Gibón*

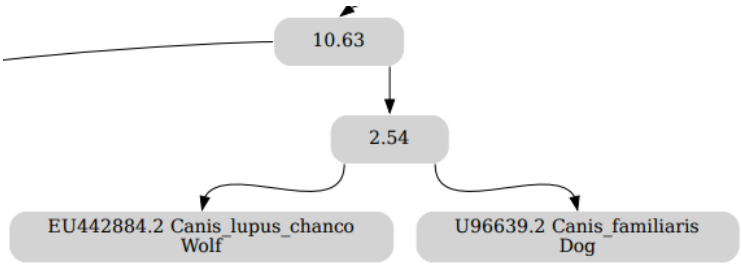


Figura 12: Marca la similitud entre *Perros* y *Lobos*

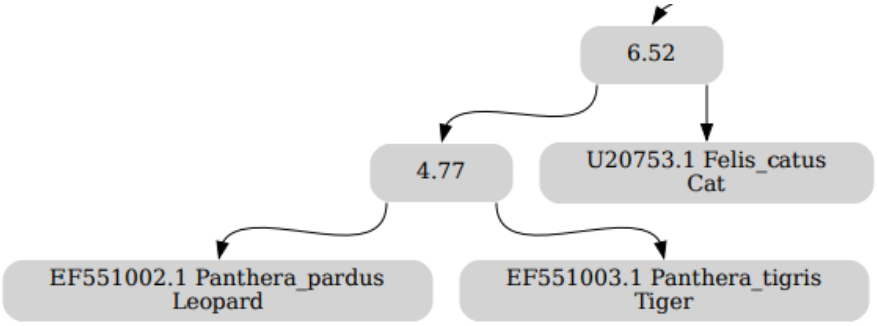


Figura 13: Marca la similitud entre *Leopardos*, *Tigres* y *Gatos*

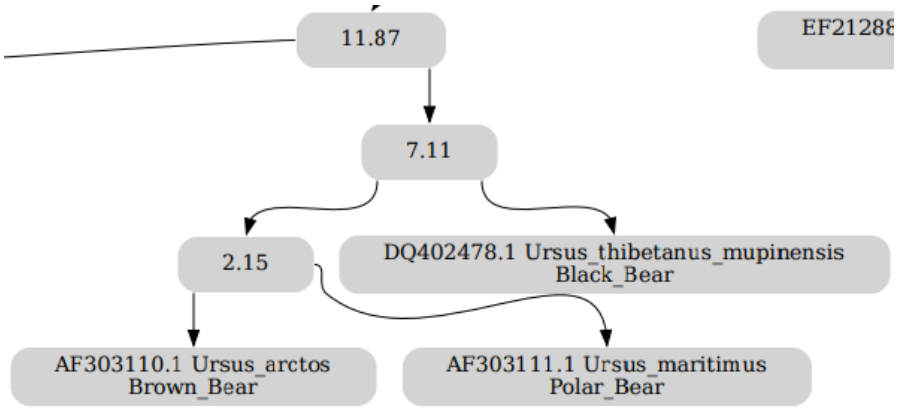


Figura 14: Marca la similitud entre *Osos Cafés*, *Osos Negros* y *Osos Polares*

Dejándonos con todas las demás especies con diferencias claras entre ellas dadas por la distancia a la que se encuentran, es decir, existen muy pocas similitudes . Por ejemplo, podemos decir que un *Humano* y un *Oso Polar* no son similares, pues la distancia entre ellos es más larga.

3.2. Bacterias

Resultado completo de las pruebas en bacterias
Ubicado en `src/output/bacterias.pdf` dentro del repositorio.

La siguiente es una tabla con cada una de las bacterias analizadas y la respectiva familia a la que pertenecen.

Y en el árbol filogenético podemos diferenciar a éstas familias, así como también ver el parecido que tienen unas con otras.

Veamos el ejemplo con las tres primeras familias en la tabla:

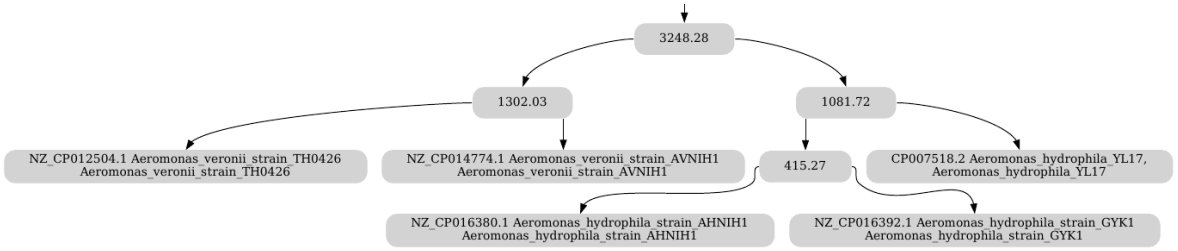


Figura 15: Familia *Aeromonadaceae* identificada en un único clúster

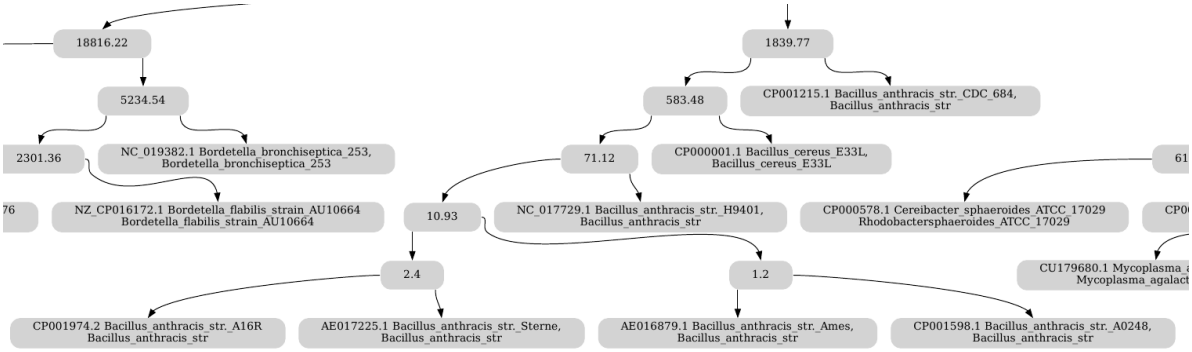


Figura 16: Familia *Bacillaceae* identificada en un único clúster

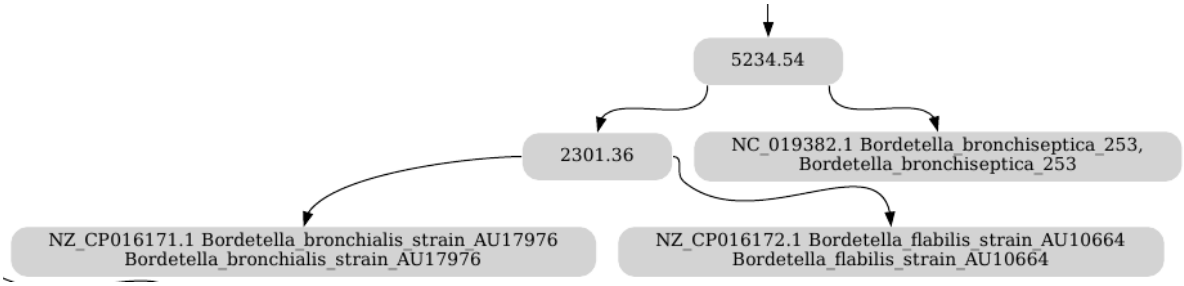


Figura 17: Familia *Alcaligenaceae* identificada en un único clúster

Tabla 2: Bacterias y sus respectivas familias.

Familia	Especie
Aeromonadaceae	Aeromonas hydrophila strain AHNIH1
	Aeromonas hydrophila strain GYK1
	Aeromonas hydrophila YL17
	Aeromonas veronii strain AVNIH1
	Aeromonas veronii strain TH0426
Bacillaceae	Bacillus anthracis str. A0248
	Bacillus anthracis str. A16R
	Bacillus anthracis str. Ames
	Bacillus anthracis str. CDC 684
	Bacillus anthracis str. H9401
	Bacillus anthracis str. Sterne
	Bacillus cereus E33L
Alcaligenaceae	Bordetella bronchialis strain AU17976
	Bordetella bronchiseptica 253
	Bordetella flabilis strain AU10664
Borreliaceae	Borrelia duttonii Ly
	Borrelia hermsii DAH
	Borrelia recurrentis A1
	Borrelia turicatae 91E135
Caulobacteraceae	Phenylobacteriumzucineum HLK1
	Caulobacter crescentus CB15
	Caulobacter crescentus NA1000
Clostridiaceae	Clostridium perfringens ATCC 13124
	Clostridium perfringens SM101
	Clostridium perfringens str. 13 DNA
Desulfovibrionaceae	Desulfovibrio vulgaris DP4
	Desulfovibrio vulgaris Hildenborough
	Desulfovibrio vulgaris RCH1
Erwiniaceae	Erwinia pyrifoliae DSM 12163
	Erwinia sp. Ejp617
	Erwinia tasmaniensis strain ET1-99
Lactobacillaceae	Lactobacillus acidophilus NCFM
	Lactobacillus helveticus DPC 4571
	Lactobacillus johnsonii NCC 533
Mycoplasmataceae	Mycoplasma agalactiae PG2
	Mycoplasma conjunctivae HRC-581T
	Mycoplasma fermentans JER
Burkholderiaceae	Ralstoniaeutropha H16
	Ralstoniaeutropha JMP134
Rhodobacteraceae	Rhodobactersphaeroides 2.4.1
	Rhodobactersphaeroides ATCC 17029
	Rhodobactersphaeroides KD131

4. Conclusión

Generar la matriz de distancias es un paso muy importante para el análisis filogenético de las especies, y los métodos que la calculan por medio de algoritmos de alineamiento pueden resultar útiles para obtener información

Tabla 3: Bacterias y sus respectivas familias (continuación)

Staphylococcaceae	Staphylococcus carnosus subsp. carnosus TM300
	Staphylococcus epidermidis ATCC 12228
	Staphylococcus epidermidis RP62A
	Staphylococcus haemolyticus JCSC1435 DNA
	Staphylococcus lugdunensis HKU09-01
Yersiniaceae	Yersinia pestis Antiqua
	Yersinia pestis CO92
	Yersinia pestis D106004
	Yersinia pestis KIM10+
	Yersinia pestis Z176003
Enterobacteriaceae	Escherichia coli ABU 83972
	Escherichia coli APEC O1
	Escherichia coli ATCC 8739
	Escherichia coli BL21
	Shigella flexneri 2002017
	Shigella flexneri 2a str. 301
	Shigella sonnei Ss046

precisa entre pocas secuencias que se requieran analizar; sin embargo, si se requiere procesar una gran cantidad de información, el uso de éste método se vuelve inviable debido principalmente a su complejidad computacional.

Por ésta razón se vuelve necesario investigar métodos que prescindan del algoritmo de alineamiento, y el acercamiento a este problema por medio de gráficas bipartitas resulta en una solución que además de mejorar el tiempo de analizar cada secuencia a lineal, también es precisa para el cálculo de distancias como vimos en los resultados. Por lo que el uso de éste método puede ser de gran utilidad para procesar una gran cantidad de datos.

5. Datos

La lista completa de los datos utilizados en las pruebas se encuentran en el siguiente link.

Se encuentra en la carpeta `src/input/datos` *del repositorio*

6. Bibliografía

1. Das, S., Das, A., Bhattacharya, D. & Tibarewala, D. (2020). A new graph-theoretic approach to determine the similarity of genome sequences based on nucleotide triplets. *Genomics*, 112(6), 4701-4714. https://www.sciencedirect.com/science/article/pii/S088875431930638X

2. Das, S., Deb, T., Dey, N., Ashour, A. S., Bhattacharya, D. & Tibarewala, D. (2018). Optimal choice of k-mer in composition vector method for genome sequence comparison. *Genomics*, 110(5), 263-273. https://www.sciencedirect.com/science/article/pii/S0888754317301453

3. Wikipedia contributors. (2022, 16 octubre). *UPGMA*. Wikipedia. https://en.wikipedia.org/wiki/UPGMA