ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE CIENCIAS

IMPLEMENTACIÓN Y EVALUACIÓN DE HEURÍSTICAS PARA CONSTRUIR ÁRBOLES FILOGENÉTICOS BASADOS EN MATRICES DE DISTANCIAS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MATEMÁTICO

BLANCA EVELYN QUISHPE GOYES
GUSTAVO PATRICIO RECALDO VÁSQUEZ

DIRECTOR: DR. LUIS MIGUEL TORRES

QUITO, JULIO 2006

DECLARACIÓN

Nosostros, BLANCA EVELYN QUISHPE GOYES y GUSTAVO PATRICIO RECALDE VÁSQUEZ, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____ _____

Evelyn Quishpe Goyes Gustavo Recalde Vásquez

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por BLANCA EVELYN QUISHPE GOYES y GUSTAVO PATRICIO RECALDE VÁSQUEZ, bajo mi supervisión.

_____

Dr. Luis Miguel Torres
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Dejamos constancia de nuestro sincero agradecimiento:

Al Dr. Luis Miguel Torres por su apoyo e incentivo permanentes a lo largo de la realización de esta tesis y sobre todo su gran paciencia y dedicación.

A Alexis Abad por su apoyo incondicional.

DEDICATORIA

A Jehová, mi Dios, quien me ha dado todo

Evelyn

DEDICATORIA

Dedico este trabajo a mis padres Rosita Vásquez y Gustavo Recalde, quienes me apoyaron siempre e hicieron todo lo posible para que culmine mis estudios.

Gustavo

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ALGORITMOS

RESUMEN

Una filogenia o árbol filogenético es una representación gráfica que ilustra las relaciones de parentesco entre las especies de un conjunto. Una filogenia puede ser construida por distintos métodos que parten de secuencias de ADN de cada especie.
En los métodos basados en matrices de distancias, las secuencias de ADN son empleadas para calcular distancias evolutivas entre las especies, las mismas que luego se emplean como un criterio para decidir que tan cercanas o lejanas deben estar dos especies dentro de árbol filogenético.
En este estudio describimos e implementamos computacionalmente una metaheurística basada en el esquema de Recocido Simulado para la construcción de filogenias. Nuestro algoritmo emplea como entrada la información de secuencias de ADN para cada especie o alternativamente una matriz de distancias entre las especies. Mediante un método de agrupamiento se genera una primera filogenia. Empleamos luego un movimiento local para generar nuevas filogenias a partir de la filogenia inicial y realizar una exploración controlada del espacio de búsqueda. Las distintas filogenias son evaluadas comparativamente mediante una medida de ajuste obtenida por medio del método de mínimos cuadrados.
Reportamos, finalmente, los resultados de pruebas computacionales efectuados sobre problemas reales obtenidos del Internet.

Palabras clave: Árboles filogenéticos, matrices de distancias, recocido simulado, mínimos cuadrados, Branch-and-Bound, optimizaciòn combinatoria, métodos de agrupamiento.

Capítulo 1

INTRODUCCIÓN

1.1. ANTECEDENTES

La gran diversidad de organismos vivos ha motivado al hombre desde la antiguedad a clasificar a este extenso conjunto de formas de vida, desarrollando varios sistemas de clasificación. Uno de los primeros interesados en dar una organización al mundo viviente fue Aristóteles (384-322 A.C) quien estudió y describió más de 500 especies animales; estableció la primera clasificación de acuerdo a las caracteristicas físicas de los organismos que no fue superada hasta el siglo XVIII por Carl Linné (Hull [1973]). Carl Linné (1707-1778) estableció una clasificación de las especies basándose en el concepto de *especie* como un grupo de individuos semejantes, con antepasados comunes (Hull [1973]). Más adelante Charles Darwin (1809-1882), expuso sus ideas sobre la evolución de las especies por medio de la selección natural en su libro *" El origen de las especies "* (Darwin [1859]). Darwin sostenía que una especie podía dar lugar a varias, utilizó los árboles ramificados para representar esto, así se explicaba muchos aspectos morfológicos como caracteres originados en un antepasado común que por diversificación de las especies han producido los caracteres homólogos(Darwin [1859]).

La Sistemática, ciencia que se encarga del estudio de la diversidad los organismos junto a la Taxonomía que clasifica a los organismos en grupos, toman la percepción de Darwin, y la forma de clasificación se enfoca a las relaciones de parentesco entre las especies. Una forma de representar los resultados de esta clasificación donde se muestran relaciones de parentesco se conoce como *Filogenia*.

1.2. SISTEMÁTICA FILOGENÉTICA

La Sistemática Filogenética, nacida con Hennig [1966], es planteada como la búsqueda del orden natural. Conceptualmente Ax [1987] describe a la Sistemática Filogenética como la tarea de descubrir las relaciones genéticas de parentesco entre organismos y trasladar el orden que percibimos en la naturaleza a un sistema equivalente humano.

1.2.1. GRUPOS

De acuerdo a distintos rasgos y diferencias biológicas se pueden distinguir los siguientes grupos:

Grupo monofilético. Monofilum o clado

Un grupo monofilético es un grupo de especies descendientes de una especie simple (especie troncal) que incluye todos los descendientes de la misma (Hennig [1966]). Un grupo monofilético debe contener, por definición, al menos dos especies. Está formado por un grupo de especies filogenéticamente *cerrado*, es decir, son un grupo de especies que mantienen entre ellas la relación de parentesco más cercana posible, sin que ningún otro grupo externo tenga una relación igual o más cercana.

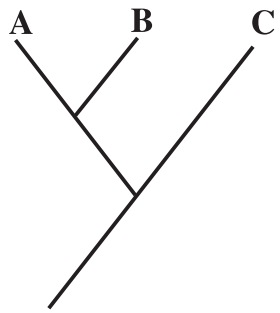


Figura 1.1: (AB) forma un grupo monofilético,((AB)C) también pero (BC) y (AC) no.

Taxón

Un taxón es un grupo de organismos a los que se da un nombre. Según el concepto de clado hay dos tipos de taxones:

- *Taxón natural:* Es un grupo de organismos que existe en la naturaleza. En el ámbito de la Sistemática Filogenética se entiende a una especie particular o a cualquier grupo de organismos monofilético.
- *Taxón artificial:* Son aquellos que no existen en la naturaleza. Son grupos de organismos no monofiléticos.

Especie

Técnicamente una especie corresponde al taxón menor que se puede formar, es la unidad sobre la que actúa la evolución y, por tanto, también es la unidad del estudio filogenético, el único grupo real que se puede encontrar en la naturaleza.

Ancestros

Un ancestro es un organismo primitivo que ha dado lugar a un conjunto de especies. Los ancestros se encuentran en la base del grupo.

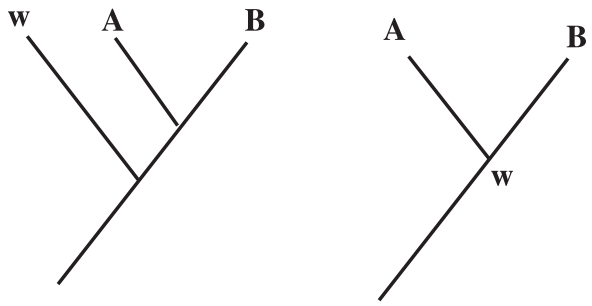


Figura 1.2: En la primera figura w da lugar a las especies A y B, el ancestro w como un terminal. La segunda figura muestra al ancestro w como un nodo hipotético (nodo interno).

Caracteres

Un caracter se define como un rasgo o atributo observable en un ser vivo. Los caracteres que se utilizan habitualmente son los morfológicos, los nucleótidos de ADN y los aminoácidos de las secuencias proteicas, siendo los nucleótidos de ADN los caracteres que se emplearán en esta tesis.

1.2.2. REPRESENTACIÓN DE FILOGENIAS

Una filogenia se representa habitualmente por medio de diagramas que indican la secuencia de la evolución de los organismos. Dependiendo del modo en que han sido construidos, o la información que proporcionen serán de un tipo u otro. A continuación definiremos algunas de ellos.

Dendrogramas

Los dendrogramas son diagramas con forma de árbol. Se utilizan para representar las relaciones entre organismos. Dependiendo de sus características y uso reciben distintos nombres. Dendrograma, por tanto es un nombre genérico que se aplica indistintamente a cualquier tipo de árbol.

Terminología utilizada en los dendrogramas

- *Nodo:* El punto donde cada rama se bifurca [1] o termina
- *Nodo interno(o internodo):* Si tiene ramas descendientes
- *Nodo terminal:* Si el linaje termina en él. Normalmente corresponde a los taxones observados (especies)
- *Rama:* Un nodo y sus internodos
- *Raíz:* El nodo más reciente a todos los nodos e internodos

La Figura 1.3 muestra un dendograma con sus respectivas componentes.

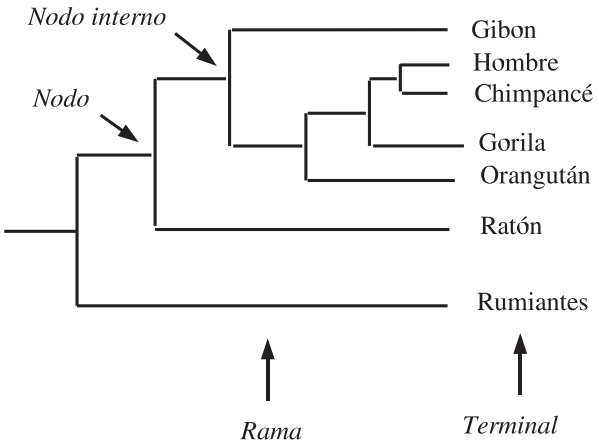


Figura 1.3: Dendrograma

Fenogramas

Los fenogramas son dendrogramas que se fundamentan en una evaluación del parecido global de los organismos. La longitud de sus ramas es proporcional al grado de semejanza fenotípica que es el valor que indica que tan aparentadas

[1] Una bifurcación es un nodo en un árbol que conecta exactamente tres ramas.

estan dos especies de acuerdo a sus caracteres hereditarios, mientras más parentesco haya entre un par de especies, la longitud de las ramas correspondientes a cada especie se incrementará. No da ninguna información sobre la evolución de los organismos proporcionando únicamente un diagrama de relación fenotípica. La Figura 1.4 muestra un fenograma.

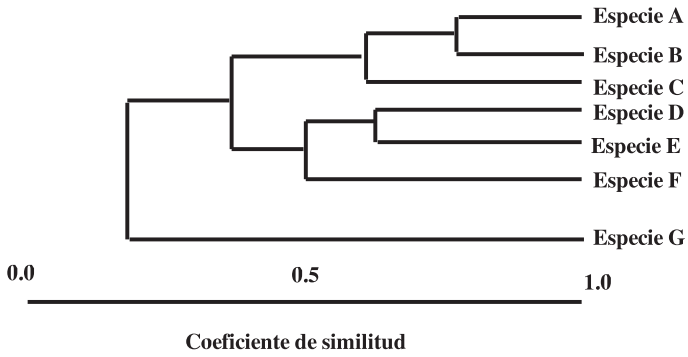


Figura 1.4: Fenograma. El eje que contiene el coeficiente de similitud que indica la semejanza fenotípica entre los organismos.

Filogramas (dendrogramas tipológicos)

Los dendrogramas tipológicos son propios del evolucionismo y muestra las ramificaciones sufridas en el proceso de la evolución, las ramas son proporcionales al grado de divergencia filogenética que es el valor que indica cuan genéticamente separadas están dos especies (eje horizontal), contienen especies ancestrales en los nodos y en el interior de las ramas. Difieren de los fenogramas por el significado del eje vertical que representa el tiempo de evolución de un ancestro a la especie actual.

Árboles filogenéticos

Actualmente los árboles filogenéticos son las representaciones más utilizadas en la filogenia y los cuales serán el eje fundamental para nuestro estudio. Estos son gráficos que representan las relaciones filogenéticas entre los diferentes taxones tal como son entendidas por un investigador particular. Es una hipótesis sobre las relaciones filogenéticas de un taxón. En este sentido cualquier árbol que indique relaciones filogenéticas puede ser considerado un árbol filogenético, pero su uso se reserva a la Sistemática Filogenética.

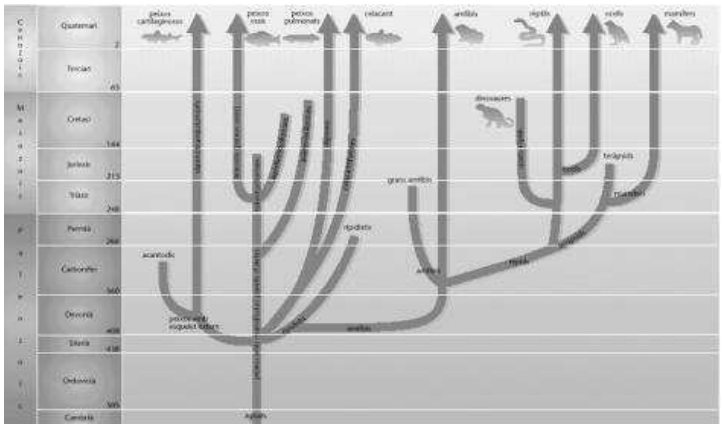


Figura 1.5: Filograma. El eje vertical representa el tiempo, el horizontal la divergencia filogenética, en los nodos y las ramas se encuentran las especies ancestrales.

Diagrama de relaciones filogenéticas. Árbol evolutivo

A un árbol evolutivo también se lo llama árbol filogenético o cladograma temporalizado. Se caracteriza por colocar en el eje vertical el tiempo y/o aceptar ancestros en los nodos no considerarlos como terminales sino como internodos. El eje horizontal y el ángulo de las bifurcaciones no tienen ningún significado especial. Los diagramas de relaciones filogenéticas son también llamados árboles evolutivos (Steipe [1995]). Ver Figura 1.6.

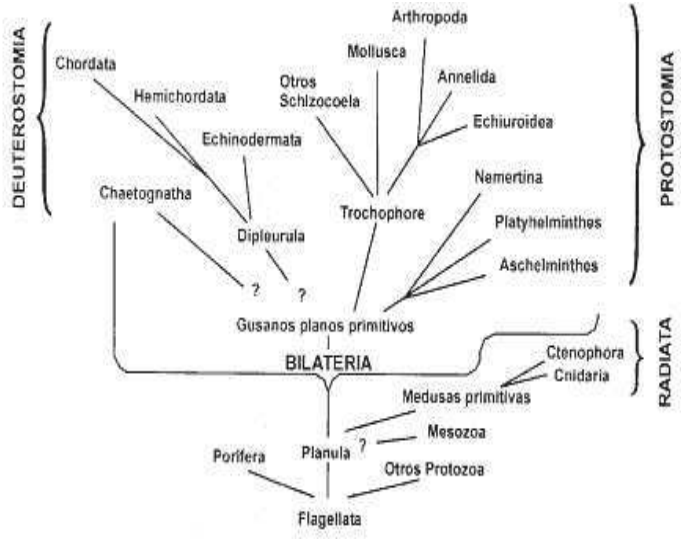


Figura 1.6: Diagrama de relaciones filogenéticas. En el eje vertical está el tiempo y en los internodos los ancestros.

1.3. ETAPAS DEL ANÁLISIS FILOGENÉTICO

El análisis filogenético comienza con la adquisición de datos biomoleculares, como secuencias de ADN. Después, se construye un árbol que representa la evolución hipotética de las secuencias, o de los organismos estudiados.

Generalmente, las hojas del árbol representan a los organismos sobre los que disponemos de información, mientras que los nodos interiores representan las especies de las que supuestamente evolucionaron.

El proceso evolutivo se modeliza mediante un tipo especial de árboles con pesos en sus ramas, que son los árboles filogenéticos.

Las etapas de un análisis filogenético son:

- Construir el alineamiento múltiple de las secuencias en estudio
- Cálculo de distancias genéticas entre especies
- Construir el árbol filogenético
- Evaluar el árbol filogenético construido

1.3.1. ALINEAMIENTO DE SECUENCIAS DE ADN

El ADN es una molécula formada por dos filamentos. Cada hilo consiste en una cadena compuesta por `A` (adenina), `T` (timina), `C` (citosina), y `G` (guanina) que son bases nitrogenadas, por lo que una secuencia de ADN no es más que una sucesión de bases nitrogenadas o nucleótidos. Se muestra un ejemplo en la Figura 1.7.

```
S = ATGACCAACATCCGAAAATCCCACCCGCTAATCAAAATTATCAATCACTCATTTATCG
ACCTACCAACCCCATCAAACATCTCATCCTGATGAAACTTTGGCTCCCTTTTAGGAATATGC
```

Figura 1.7: Secuencia (Tapirus terrestris cytochrome b gene,mitochondrial) AF056030, longitud = 60 nucleótidos. Tomado del banco de datos (GenBank)

El alineamiento de secuencias se puede definir como la comparación entre dos o más secuencias para observar si existen zonas o caracteres (nucleótidos) comunes en el grupo de secuencias analizadas, permitiendo establecer una función similar, una estructura similar o un origen evolutivo entre un grupo de especies.

La idea del alineamiento de secuencias genéticas se basa en la inserción de espacios en las secuencias tal que los nucléotidos iguales calcen o se alineen con espacios entre pares de secuencias.

Veamos un ejemplo:

Sea $s_1 =$ GACGGATT y $s_2 =$ GATCGGTT

Con un alineamiento se tendrá:

GA-CGGATT
GATCGG-TT

Las longitudes de las secuencias de ADN pueden llegar a contener miles de nucleótidos por tanto el alineamiento entre un grupo de especies se vuelve un problema considerable dentro del análisis filogenético.

El desarrollo e implementación de algoritmos para el alineamiento de secuencias constituye en sí un campo importante de investigación dentro de la bioinformática en la actualidad. Su estudio detallado no está dentro de los objetivos de esta tesis; el lector interesado puede encontrar información adicional al respecto en Felsenstein [2004a] y en la web en Gómez [2005] y Abascal [2006].

En adelante asumiremos que las secuencias de ADN ya han sido previamente alineadas.

1.3.2. DISTANCIAS GENÉTICAS Y MODELOS DE EVOLUCIÓN

Una vez que las secuencias están alineadas el siguiente paso necesario para la construcción de árboles filogenéticos es el cálculo de las distancias genéticas. La distancia genética entre dos secuencias es un valor de disimilitud de las mismas.

Estas distancias son almacenadas en matrices simétricas.

Debido a que algunas de estas diferencias resultan de múltiples eventos de mutación (cambio hereditario en el material genético, que aparece bruscamente), este valor es normalmente un cálculo subestimado de la verdadera distancia evolutiva, es por ello que se han aplicado varios modelos de evolución.

El modelo de evolución es el que provee el criterio para cuantificar las diferencias entre las secuencias. Su uso es fundamental para la construcción del árbol filogenético.

El criterio más simple es considerar que cada cambio en las posiciones de las secuencias obedece a un único evento mutacional, es decir, que la tasa de cambio de cualquier base hacia otra cualquiera es la misma.

De acuerdo a la base tenemos dos tipos de cambios:

Transición: A $\leftrightarrow$ G C $\leftrightarrow$ T
Transversión: A $\leftrightarrow$ C A $\leftrightarrow$ T C $\leftrightarrow$ G G $\leftrightarrow$ T

Tomando el criterio más simple, la matriz de distancias genéticas tiene como elementos el número total de diferencias entre cada par de secuencias (sin tomar en cuenta que se tenga una transición o transversión). La Figura 1.8 muestra un ejemplo de como se construye dicha matriz.

	1	2	3	4	5	6	7	8	9	10	11	12
Especie 1	A	A	**G**	**G**	T	**T**	**G**	C	G	C	**G**	**T**
Especie 2	A	A	**C**	**T**	T	**C**	**A**	C	G	C	**A**	**A**
Especie 3	A	C	C	T	T	G	A	C	C	C	A	A
Especie 4	A	C	C	T	T	C	A	C	C	C	A	A

	Especie 1	Especie 2	Especie 3	Especie 4
Especie 1	0	**6**	8	8
Especie 2	6	0	3	2
Especie 3	8	3	0	1
Especie 4	8	2	1	0

Figura 1.8: Matriz de Datos y Matriz de distancias evolutivas $\mathbb{D}$.

El número de diferencias de nucleótidos entre la Especie 1 y la Especie 2 es 6, este valor corresponde al elemento $d_{12} \in \mathbb{D}$ (en negrilla en la Figura 1.8). De igual forma se obtienen los demás elementos $d_{ij} \in \mathbb{D}$ con $i \neq j$, los elementos $d_{ii} = 0$ ya que representan a una sola especie.

Cuando se considera pesos en las transiciones y transversiones, se han propuesto varios modelos descritos a continuación.

Modelo de Jukes-Cantor

Este modelo fue propuesto por Jukes & Cantor [1969] posee un parámetro α que representa una tasa de sustitución o cambio entre todos los pares de bases. Tomando el criterio anterior, este modelo asigna una misma probabilidad de ocu-

rrencia tanto a las transversiones como a las transiciones. La Figura 1.9 muestra el esquema general de este modelo.

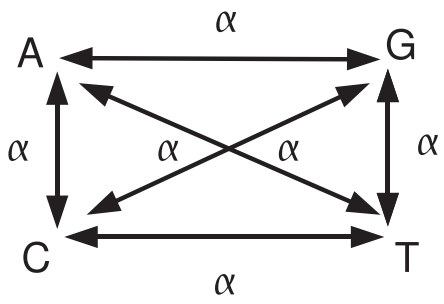


Figura 1.9: Modelo de Jukes y Cantor.

Para el cálculo de las distancias evolutivas, se emplean probabilidades de transición [2].

Éstas probabilidades fueron modeladas sobre una distribución de Poisson en función de la longitud de ramas a través del tiempo t y la tasa de cambio α, por tanto la probabilidad de transición de un nucleótido x a otro y se calcula como:

$$P(x|y,\alpha,t) = \frac{1}{4}(1 - e^{-\frac{4}{3}\alpha t}) \tag{1.1}$$

Supongamos que $x =$ C y $y =$ A , entonces hay tres nucleótidos más T, G, A que podrían cambiar a una A, así la probabilidad que este nucleótido $x =$ C sea diferente a los otros es :

$$P_s = \frac{3}{4}(1 - e^{-\frac{4}{3}\alpha t}) \tag{1.2}$$

El producto αt, de la tasa de cambio y el tiempo es un buen estimador de la distancia evolutiva y fácilmente se lo obtiene de la ecuación 1.2 como sigue:

[2]Las probabilidades de transición no se refiere a los tipos de cambios de una base a otra que se llamó transiciones, sino a la definición propiamente dicha: la probabilidad de cambio de un estado a otro, en este caso de un nucleótido a otro.

$$\begin{aligned} P_S &= \frac{3}{4}(1 - e^{-\frac{4}{3}\widehat{\alpha t}}) \\ \frac{4}{3}P_S &= 1 - e^{-\frac{4}{3}\widehat{\alpha t}} \\ 1 - \frac{4}{3}P_S &= e^{-\frac{4}{3}\widehat{\alpha t}} \\ \ln(1 - \frac{4}{3}P_S) &= -\frac{4}{3}\widehat{\alpha t} \\ -\frac{3}{4}\ln(1 - \frac{4}{3}P_S) &= \widehat{\alpha t} \end{aligned}$$

Así las distancias evolutivas d_{ij} para este modelo se calculan como:

$$d_{ij} = -\frac{3}{4}\ln(1 - \frac{4}{3}P_{s_i s_j}) \tag{1.3}$$

donde $P_{s_i s_j}$ representa el número de diferencias entre las especies s_i y s_j.

Modelo de Kimura (Dos parámetros)

Este modelo lo propuso Kimura [1980] y tiene la presencia de dos parámetros : α que representa las transiciones y β a las transversiones, es decir, se asigna una cierta probabilidad para las transiciones y otra para las transversiones. Ver Figura 1.10.

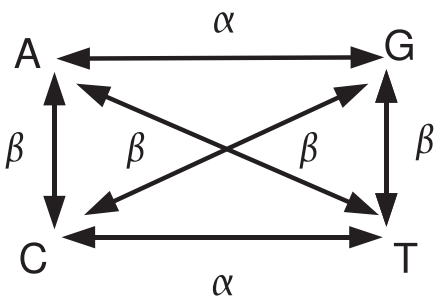


Figura 1.10: Modelo de Kimura. Dos parámetros

Un procedimiento similar al modelo anterior se sigue para obtener las distancias evolutivas.

Las distancias evolutivas se calculan con :

$$\widehat{t} = -\frac{1}{4}ln[(1-2Q)(1-2P-Q)^2 \qquad \widehat{R} = \frac{-ln(1-2P-Q)}{-ln(1-2Q} - \frac{1}{2} \tag{1.4}$$

donde Q es el número de diferencias con transversion y P es el número de diferencias con transicion. Para detalles de la construcción de ecuaciones anteriores ver Felsenstein [2004b].

Existen otros modelos con más parámetros como el modelo de Tamura & Nei [1993] de tres parámetros, el modelo general de tiempo reversible (Lanave et al. [1984]) de 6 parámetros y varios más que se adaptan propiamente a características biológicas.

1.3.3. CONSTRUCCIÓN DE ÁRBOLES FILOGENÉTICOS

Con un alineamiento de un grupo de especies y una matriz de distancias genéticas empleando uno de los modelos de evolución, tenemos la información necesaria para dar inicio a la construcción de árboles filogenéticos mediante métodos de reconstrucción filogenética descritos más adelante.

El propósito de esta tesis se enmarca en el tratamiento de datos moleculares (secuencias de ADN) por tanto de aquí en adelante hablaremos de construcción de árboles filogenéticos con secuencias de ADN.

Hemos venido hablando de árboles filogenéticos en términos biológicos pero matemáticamente un árbol filogenético se define como un árbol $H = (V, T)$ dentro de la teoría de grafos, donde V es el conjunto de nodos externos (especies) e internos (ancestros) y T es el conjunto de aristas (ramas) de H.

Por las características del árbol filogenético podemos dar una definición matemática que se ajusta de una mejor manera:

Dado un conjunto $S = \{s_1, s_2, ..., s_n\}$ de especies, un árbol filogenético $H = (V, T)$ es un árbol con n hojas y m nodos internos de grado tres. Como H es un árbol, entonces tiene $n + m - 1$ aristas , además por técnicas de enumeración de árboles filogenéticos (Ver detalles más adelante) tenemos que H tiene $2n - 3$ aristas, por tanto el número de nodos internos es: $n - 3$. Asi H es un árbol con $|V| = 2n - 2$ y $|T| = 2n - 3$.

Recordemos que un *árbol* es un grafo conexo y acícilo y una *hoja* es un nodo de grado uno, es decir, que dicho nodo es incidente a una sola arista.

Raíz de un árbol filogenético

La raíz de un árbol filogenético H es el nodo que da origen a todos los nodos terminales. Un árbol puede o no tener raíz. Los árboles sin raíz especifican relaciones filogenéticas pero no caminos evolutivos.

Un árbol sin raíz puede tener varios árboles con raíz, por lo que a veces resulta conveniente trabajar con árboles sin raíz por su menor número y luego enraizarlos.

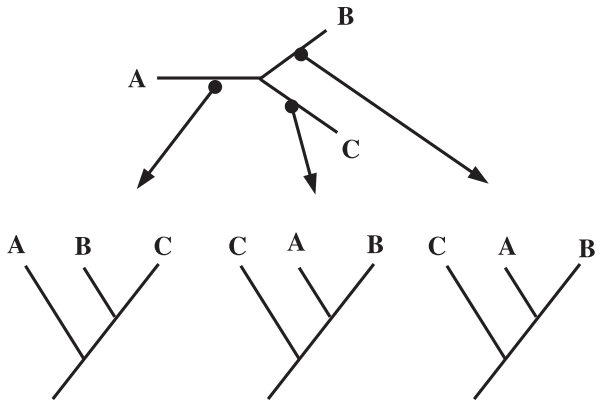


Figura 1.11: Un árbol no dirigido puede dar lugar a 3 árboles filogenéticos con raíz

Topología de un árbol filogenético

En terminos biológicos una topología de un árbol filogenético es un patrón de ramificación del árbol, es decir, el padrón de distribución de las ramas en el árbol.

Matemáticamente, sea $H = (V, T)$ un árbol filogenético, una topología $H' = (V', T')$ es un subgrafo generador de H, ya que $V = V'$

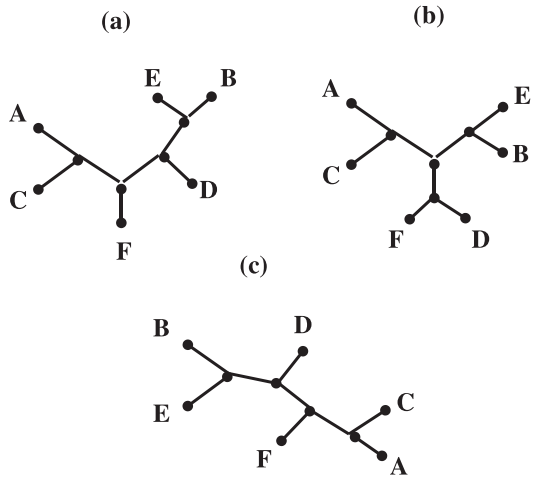


Figura 1.12: Los árboles en a) y c) tienen la misma topología.

Dos árboles van a ser distintos si su topología es distinta o siendo esta la misma, la asignación de las diferentes especies a las distintas hojas no es igual.

Enumeración de árboles

Considerar todo el espacio de árboles en un análisis filogenético implica enumerar cada elemento de este espacio, es decir enumerar todos los posibles árboles que genera un grupo de n especies.

El proceso de enumeración argumenta la adición de especies. Partiendo de un par de especies, se añade una nueva, hay únicamente un lugar donde hacerlo, así se forma un primer árbol con tres especies, sobre éste, nuevamente se añade una especie, ahora hay tres posibles lugares donde añadirla dando origen a tres árboles, sobre cada uno de éstos se repite la adicion de una nueva especie, originando más árboles. Este proceso prosigue hasta alcanzar añandir las n especies. La Figura 1.13 muestra el proceso de construcción de todos los posibles árboles.

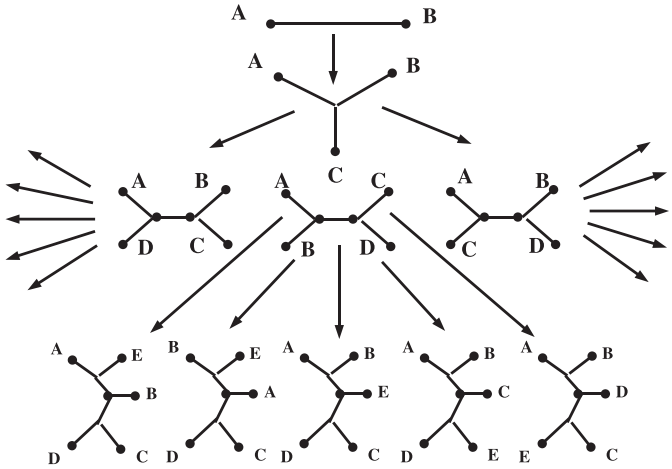


Figura 1.13: Enumeración de árboles. El proceso añade una nueva especie en todos los posibles lugares. Para $n = 4$ se muestra la consecuencia de añadir una quinta especie a uno de los posibles árboles, los otros únicamente muestran saetas que indican el número de árboles que surgirán con la especie añadida.

La cantidad de árboles sin raíz de un determinado número de especies n se calcula como:

$$U_n = \prod_{i=3}^{n}(2i-5) \tag{1.5}$$

Demostración: Para $n = 1, 2$ por definición el producto de cero elementos es $U_1 = U_2 = 1$.

Para $n \geq 3$ la demostración es por inducción:

Sea $n = 3$. Entonces hay exactamente $U_3 = \prod_{i=3}^{3}(2i-5) = 1$ árboles sin raíz con 3 hojas.

Sea $n > 3$. Hay exactamente n hojas, $n-3$ nodos internos y $2n-3$ aristas.

Un nuevo nodo es insertado en cualquiera de las $2n-3$ ramas. Entonces por hipotesis de inducción, tenemos:

$$\begin{aligned} U_{n+1} &= U_n * (2n-3) \\ &= \prod_{i=3}^{n} (2i-5)(2(n+1)-5) \\ &= \prod_{i=3}^{n+1} (2i-5) \end{aligned}$$

Por cada árbol sin raiz deben de existir un número determinado de árboles con raiz, todos los posibles árboles que resultan añadiendo a cada rama de dicho árbol una raiz, por tanto, en cada árbol sin raiz habrá $2n-3$ árboles con raiz.

Partiendo de la ecuación 1.5 y como existen $2n-3$ árboles con raiz por cada árbol sin raiz, entonces:

$$R_n = (2n-3) * U_n \tag{1.6}$$

donde R_n es la cantidad de árboles con raíz.

Para la construcción de la ecuación 1.6 ver Schröder [1870].

Aplicando las ecuaciones para árboles con raíz y sin raíz a determinados n, tenemos la Tabla 1.1.

Se puede observar que tanto el número de árboles con raíz y sin raíz crece exponencialmente con el número de especies n. Esto es una dificultad para la reconstrucción de filogenias e intentar una enumeración exhaustiva no sería conveniente.

Varios métodos se han enfrentado a esta dificultad usando distintas técnicas y explorando solo una parte significativa del espacio de árboles.

Métodos de reconstrucción filogenética

Se presenta un resumen corto de los distintos métodos de reconstrucción filogenética, su idea, sus ventajas y desventajas.

Estos métodos pueden clasificarse en Métodos de distancias (Distance methods), Métodos de parsimonia (Parsimony methods) , Métodos de similitud (Likelyhood methods) y Métodos bayesianos (Inferencia bayesiana).

Número de especies	Árboles con raíz	Árboles sin raíz
2	1	1
3	3	1
4	15	3
5	105	15
6	945	105
7	10,395	945
8	135,135	10,395
9	2,027,025	135,135
10	34,459,425	2,027,025
11	654,729,075	34,459,425
12	13,749,310,575	654,729,075
13	316,234,143,225	13,749,310,575
20	8.2×10^{21}	2.22×10^{20}

Cuadro 1.1: Número de árboles con y sin raíz.

Métodos de Distancias (Análisis de cluster)

Estos métodos parten de una matriz de distancias evolutivas, construyen árboles uniendo pares de nodos con distancias menores, según dicha matriz. A cada paso de este proceso, dos nodos terminales se reemplazan por un nodo interno, hasta que quedan solamente dos nodos.

Los modelos que utilizan este tipo de supuesto son : *WPGMA (weighted pair group method using arithmetic averages)* , y *UPGMA (unweighted pair group method using arithmetic averages)*. También en este grupo se encuentra el método Neighbor Joining, que actualmente es aceptado de muy buena manera por la comunidad de biólogos por su facilidad en reconocer una topología aceptable del árbol y su velocidad en hacerlo. La idea de este método es ir agrupando especies vecinas tal que la longitud entre ellas sea mínima.

Adheriéndose a este grupo como un complemento está el método de los mínimos cuadrados MMC el cual constituye el objeto de nuestro estudio. Mediante minimización se obtiene un sistema de ecuaciones que permite encontrar longitudes de ramas de un árbol y una medida de ajuste que permite ver que tan buena fue la aproximación con respecto a la matriz de distancias evolutivas dada.

Este grupo de métodos será detallado en el Capítulo 2 de esta tesis.

Máxima parsimonia

Entre los primeros métodos para la construcción de filogenias se encuentran los métodos de parsimonia. La parsimonia suministra un criterio sencillo para la elección del mejor árbol, la topología elegida es aquella que minimiza el número total de cambios evolutivos (entre secuencias) que han ocurrido. Para determinar el número total de cambios evolutivos en un árbol existen varios algoritmos que permiten contarlos. Entre ellos el algoritmo de Fitch [1971] y el algoritmo de Sankoff. Para una detallada explicación de estos algoritmos ver Felsenstein [2004c].

Máxima Verosimilitud

Este método fue introducido por Cavalli-Sforza & Edwards [1967]. La verosimilitud se la considera como la probabilidad de observar los datos alineados X dado un modelo θ , $P(X|\theta)$, donde los parámetros de θ incluyen un árbol, longitudes de ramas y un modelo de evolución. El principio de la verosimilitud busca parámetros que maximicen la probabilidad P. La función de verosimilitud se define como $L(\theta) = P(X|\theta)$.

La máxima verosimilitud es el valor de θ que maximiza $L(\theta)$. El procedimiento básico es calcular la probabilidad de cada columna del alineamiento, por tanto la probabilidad del alineamiento entero es simplemente el producto de las probabilidades de cada columna. La verosimilitud del alineamiento esta dada por:

$$L(\theta) = \prod_{i=1}^{n} P(X_i|\theta) \tag{1.7}$$

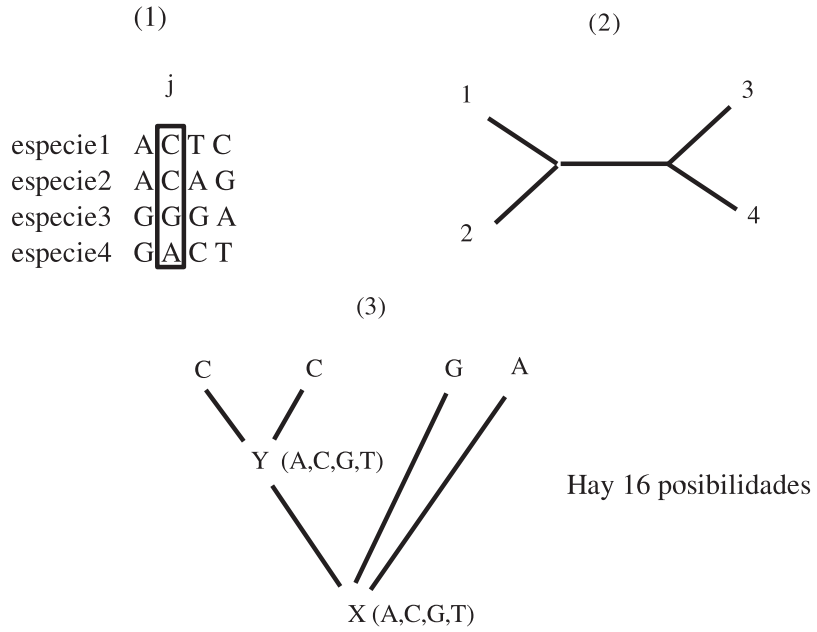
En la Figura 1.14 se muestra el proceso de la máxima verosimilitud.

Para detalles de este método revisar Felsenstein [2004d] y Gascuel [2004a].

Inferencia bayesiana

Los métodos bayesianos son similares a los de máxima verosimilitud, trabaja con probabilidades.

Del método de máxima verosimilitud tenemos la probabilidad $P(X|\theta)$, la idea de la inferencia bayesiana es buscar la probabilidad del modelo θ con respecto a los datos alineados $P(\theta|X)$. Como θ es dado, tenemos una distribución a priori de θ, $P(\theta)$, esto se combina con la verosimilitud o la probabilidad de los datos alineados dado θ, $P(X|\theta)$ y a través del Teorema de Bayes podemos buscar la probabilidad a posteriori $P(\theta|X)$:



(4) Verosimilitud del sitio j

$$L(j) = \text{Prob}\left| \begin{array}{c} C \quad C \quad G \quad A \\ A \\ \quad\quad A \end{array} \right| + \text{Prob (15 otras probabilidades)}$$

(5) Verosimilitud de todo el árbol

$$L = L(1)\,L(2)\ldots L(N) = \prod_{j=1}^{N} L(j)$$

Figura 1.14: Máxima verosimilitud con secuencias de ADN

$$P(\theta|X) = \frac{P(\theta)P(X|\theta)}{P(X)} \tag{1.8}$$

La probabilidad marginal de los datos alineados $P(X)$, es una constante normalizada y se obtiene integrando con respecto a θ el numerador de 1.8, entonces se tendría:

$$P(\theta|X) = \frac{P(\theta)P(X|\theta)}{\int P(\theta)P(X|\theta)d\theta} \tag{1.9}$$

El cálculo del denominador de la ecuación 1.9 implica un trabajo analíticamente duro [3], por tanto se dificulta el cálculo directo de $P(\theta|X)$. Este inconveniente ha sido superado con el uso de los métodos de Cadenas de Markov Monte Carlo MCMC, que nos proprociona muestras con una distribución a posteriori y ya no se necesita calcular el denominador de la ecuación 1.9. La idea de los métodos MCMC es buscar aleatoriamente en un espacio de árboles de tal forma que se alcance la distribución deseada (en este caso, la distribución a posteriori). El algoritmo de Metropolis-Hastings (Metropolis et al. [1953]) se encarga de este proceso. Para una descripción más profunda ver Felsenstein [2004e] y Gascuel [2004b]. La Tabla 1.2 resume ventajas y desventajas de los métodos mencionados anteriormente.

Los métodos de matrices de distancias son muy rápidos pero no permiten explorar todo el espacio de árboles o al menos los árboles más significativos. Los métodos de máxima verosimilitud y los métodos bayesianos producen resultados similares, su diferencia es en el tiempo de cálculo. Su uso dependerá de los recursos disponibles y de la necesidad de obtener los resultados.

[3] La expresión $\int P(\theta)P(X|\theta)d\theta$, implica la suma sobre todos las posibles topologías

Métodos	Ventajas	Desventajas
Matrices de distancias	Rápidos	Secuencias son transformadas y se pierde información
Máxima parsimonia	Es robusto si las ramas son cortas	Mala representación cuando las ramas tienen una considerable longitud
Máxima verosimilitud	La verosimilitud captura toda la información de los datos bajo el modelo dado	Computacionalmente lento
Inferencia bayesiana	Las distribuciones a a priori pueden ser especificadas	Dificultad para determinar un criterio de parada en las Cadenas de Markov Monte Carlo

Cuadro 1.2: Ventajas y desventajas de los métodos de reconstrucción filogenética.

Capítulo 2

MÉTODOS BASADOS EN MATRICES DE DISTANCIAS

La construcción de árboles filogenéticos ha venido representando una tarea fundamental dentro de varias ramas de la biología. El surgimiento de la biología molecular en los últimos años ha proporcionado gran cantidad de información genética acerca de los organismos, permitiéndo una reconstrucción filogenética más precisa, pero a la vez computacionalmente más complejo.

Formalmente, el problema de reconstrucción de árboles filogenéticos puede describirse como sigue:
Dados un conjunto $S = \{s_1, s_2 s_3, ..., s_n\}$ de especies y para cada especie $s_i \in S$ una sucesión finita $g_i < c_{i1}, c_{i2}, ..., c_{ir} >$ de r caracteres genéticos, encontrar un árbol $H = (V, T)$ y un vector $l \in \mathbb{R}^T$ de longitudes para las aristas que satisfagan:

- $V := S \bigcup W$ donde W es el conjunto de nodos internos (ancestros)
- $d(s_i) = 1, \qquad \forall s_i \in S$
- $d(w) = 3, \qquad \forall w \in W$
- $\forall s_i, s_j \in S$, la longitud e_{ij} del único camino que une s_i y s_j en T será mayor cuanto más distintas sean las secuencias g_i y g_j entre sí

La última condición puede interpretarse de distintas maneras, de acuerdo al paradigma empleado para la reconstrucción filogenética.

Varios modelos y herramientas matemáticas aplicados a este problema antes de la explosión de la biología molecular se ha venido actualizando para tratar con este tipo de información, tal es el caso de los métodos basados en matrices de distancias que si bien funcionan con caracteres morfológicos también lo hacen con información molecular (secuencias de ADN, proteínas u otro tipo de información). Estos métodos fueron introducidos por Cavalli-Sforza & Edwards [1967] y Fitch & Margoliash [1967].

La idea general de estos métodos se basa en el cálculo de una medida de la distancia, llamada distancia evolutiva, entre cada par de especies a partir de secuencias alineadas de ADN (información molecular).

La distancia evolutiva es una métrica , es decir una función $d : S \times S \longrightarrow \mathbb{R}$ que satisface las siguientes propiedades:

i) No negatividad: $d_{ij} \geq 0 \quad para \quad i \neq j, \quad i, j \in S$

ii) Unicidad: $d_{ij} = 0 \Leftrightarrow i = j, \quad \forall i, j \in S$

iii) Simetría: $d_{ij} = d_{ji} \quad \forall i, j \in S$

iv) Desigualdad triangular: $\forall i, j, k \in S : d_{ik} \leq d_{ij} + d_{jk}$

Empleando el criterio de distancia evolutiva, podemos formular de manera más precisa el problema de reconstrucción filogenética, en la versión en la que será tratado en esta tesis.

Problema de la reconstrucción filogenética basada en distancias evolutivas

Dados un conjunto de especies $S = \{s_1, s_2, ..., s_n\}$ y una matriz simétrica $\mathbb{D} = (d_{ij}) \in \mathbb{R}^{n \times n}$ de distancias evolutivas entre éstas, encontrar un árbol $H = (V, T)$ y un vector $l \in \mathbb{R}^T$ que satisfagan:

- $V := S \bigcup W$ donde W es el conjunto de nodos internos (ancestros)
- $d(s_i) = 1, \qquad \forall s_i \in S$
- $d(w) = 3, \qquad \forall w \in W$
- $\forall i, j : e_{ij}$ se aproxima lo más posible a d_{ij}

Donde e_{ij} denota, como ya se señaló antes, la longitud del único camino en H con extremos en s_i y s_j.

2.1. EVALUACIÓN DEL AJUSTE DE UNA TOPOLOGÍA

Introductoriamente en este capítulo se dijo que la idea básica de los métodos basados en matrices de distancias consiste en obtener un árbol filogenético y un vector l de longitudes para las aristas, de tal forma que las distancias entre los pares de especies sobre este árbol reflejen de la manera más fiel posible sus distancias evolutivas.

Si d_{ij} representa la distancia evolutiva entre un par de especies $s_i, s_j \in S$ y e_{ij} es la distancia calculada para las mismas especies dentro de un árbol filogenético $H = (V, T)$ con un vector de longitudes $l \in \mathbb{R}^T$, entonces un criterio para medir el ajuste de H y l a las distancias evolutivas observadas es:

$$Q = \sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} w_{ij}(d_{ij} - e_{ij})^2 \tag{2.1}$$

donde w_{ij} son factores de ponderación no negativos.

Notar que $Q \geq 0$ y que este valor crece conforme a las distancias obtenidas a partir de la filogenia difieren más de las distancias observadas. Por otra parte, si $Q = 0$ entonces H y l predicen con exactitud las distancias evolutivas.

Existen varios criterios para elegir los valores w_{ij} :

- Criterio de Cavalli-Sforza & Edwards [1967]: $w_{ij} = 1$;
- Criterio de Fitch & Margoliash [1967]: $w_{ij} = \dfrac{1}{d_{ij}^2}$;
- Criterio de Beyer et al. [1974]: $w_{ij} = \dfrac{1}{d_{ij}}$.

La ecuación 2.1 puede usarse en el sentido inverso: dado un árbol filogenético $H = (V, T)$, encontrar un vector de longitudes $l^* \in \mathbb{R}^T$ que minimice Q. Este planteamiento da origen al Método de los Mínimos Cuadrados (MMC).

Método de los Mínimos Cuadrados (MMC)

El MMC puede combinarse con heurísticas de búsqueda local que generan iterativamente varios árboles filogenéticos. Se calcula para cada uno de ellos el mejor valor de ajuste Q y se seleccionan aquellos árboles donde Q es pequeño (Ver más detalles en la Sección 2.3).

El MMC determina las longitudes de las aristas que minimizan Q. Notar que la ecuación 2.1 es una expresión cuadrática sobre estas longitudes, pues los valores de e_{ij} dependen de l .

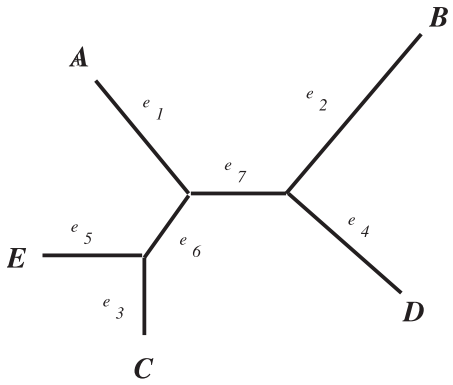


Figura 2.1: Longitudes de aristas sobre un árbol como variables.

Específicamente,

$$d_{ij} = \sum_{e \in T} x_{ij}^e l_e \tag{2.2}$$

donde x_{ij}^e es un parámetro determinado por:

$$x_{ij}^e = \begin{cases} 1 & \text{Si la arista } e \text{ pertenece al camino de la especie } i \text{ hacia la especie } j \text{ en } H \\ 0, & \text{caso contrario} \end{cases} \tag{2.3}$$

Reemplazando (2.2) en (2.1) tenemos:

$$Q = \sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} w_{ij}(d_{ij} - \sum_{e \in T} x_{ij}^e l_e)^2 \tag{2.4}$$

Podemos ahora minimizar la expresión 2.4 al derivar respecto a cada variable l_e e igualar a cero:

$$\frac{dQ}{dl_e} = -2 \sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} w_{ij} x_{ij}^e (d_{ij} - \sum_{e \in T} x_{ij}^e l_e) = 0 \tag{2.5}$$

De esta forma obtenemos un sistema de ecuaciones lineales para las longitudes de las aristas. Por ejemplo para el caso de la Figura (2.1) y asumiendo un peso $w_{ij} = 1$, las ecuaciones del sistema formado por (2.5) son:

$$\begin{aligned}
d_{AB} + d_{AC} + d_{AD} + d_{AE} &= 4e_1 + e_2 + e_3 + e_4 + e_5 + 2e_6 + 2e_7 \\
d_{AB} + d_{BC} + d_{BD} + d_{BE} &= e_1 + 4e_2 + e_3 + e_4 + e_5 + 2e_6 + 3e_7 \\
d_{AC} + d_{BC} + d_{CD} + d_{CE} &= e_1 + e_2 + 4e_3 + e_4 + e_5 + 3e_6 + 2e_7 \\
d_{AD} + d_{BD} + d_{CD} + d_{DE} &= e_1 + e_2 + e_3 + 4e_4 + e_5 + 2e_6 + 3e_7 \\
d_{AE} + d_{BE} + d_{CE} + d_{DE} &= e_1 + e_2 + e_3 + e_4 + 4e_5 + 3e_6 + 2e_7 \\
d_{AC} + d_{AE} + d_{BC} + d_{BE} & \\
+ d_{CD} + d_{DE} &= 2e_1 + 2e_2 + 3e_3 + 2e_4 + 3e_5 + 6e_6 + 4e_7 \\
d_{AB} + d_{AD} + d_{BC} + d_{CD} & \\
+ d_{BE} + d_{DE} &= 2e_1 + 3e_2 + 2e_3 + 3e_4 + 2e_5 + 4e_6 + 6e_7
\end{aligned} \tag{2.6}$$

El sistema anterior caracteriza el conjunto de candidatos para minimizar Q. Para formularlo matricialmente, sea $\widetilde{d} \in \mathbb{R}^p$, con $p = \binom{n}{2} = \frac{n(n-1)}{2}$, el vector que contiene las distancias evolutivas entre cada par de especies. Por ejemplo, para el árbol de la Figura 2.1:

$$\widetilde{d} = \begin{bmatrix} d_{AB} \\ d_{AC} \\ d_{AD} \\ d_{AE} \\ d_{BC} \\ d_{BD} \\ d_{BE} \\ d_{CD} \\ d_{CE} \\ d_{DE} \end{bmatrix} \tag{2.7}$$

Los coeficientes x_{ij}^{e} pueden ser almacenados en una matriz $\mathbb{X}$ de orden $p \times q$, denominada matriz topológica, donde $q := |T|$ es el número de aristas de H, es decir, cada fila corresponde a una distancia d_{ij} del vector $\widetilde{d}$ y contiene un elemento igual a 1 si la arista e_{ij} esta en el camino de s_i hacia s_j sobre H. Nuevamente, para el árbol de la Figura 2.1:

$$\mathbb{X} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.8}$$

Con esto el sistema (2.6) puede formularse como:

$$\mathbb{X}^{\mathrm{T}}\widetilde{\mathrm{d}} = (\mathbb{X}^{\mathrm{T}}\mathbb{X})\mathit{l} \tag{2.9}$$

Para el caso en que los w_{ij}, toman valores arbitrarios, éstos se almacenan en

una matriz diagonal $\mathbb{W}$ de dimensión $(p \times p)$:

$$\mathbb{W} = \begin{bmatrix} w_{00} & 0 & 0 & \dots & 0 & 0 \\ 0 & w_{11} & 0 & \dots & 0 & 0 \\ 0 & 0 & w_{22} & \dots & 0 & 0 \\ 0 & 0 & 0 & w_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & w_{pp} \end{bmatrix} \tag{2.10}$$

En este caso, 2.9 toma la forma:

$$\mathbb{X}^{\mathrm{T}}\mathbb{W}\widetilde{\mathrm{d}} = (\mathbb{X}^{\mathrm{T}}\mathbb{W}\mathbb{X})\boldsymbol{I} \tag{2.11}$$

Estos sistemas (2.9) y (2.11) pueden ser resueltos con métodos directos como la eliminación Gaussiana o iterativos como Jacobi. (Ver Burden & Faires [1985]. Capítulos 6 y 8)

2.2. ALGORITMOS ENUMERATIVOS EXACTOS

Los algoritmos enumerativos exactos dentro del análisis filogenético son aquellos que garantizan la obtención de un árbol con el valor mínimo de Q. Son métodos enumerativos por lo que su aplicación se ve restringida a un cierto números de especies, pues como se apreció en la Tabla 1.1 el número de árboles generados es astrómicamente grande para conjuntos de 20 especies o más.

Los algoritmos exactos pueden ejecutarse por medio de una búsqueda exhaustiva o empleando la técnica de Branch and Bound.

Búsqueda Exhaustiva

La búsqueda exhaustiva consiste en formar un conjunto de todas las soluciones de un problema y elegir la mejor. En un análisis filogenético la búsqueda exhaustiva es limitada para cantidades pequeñas de especies, generalmente cuando $n \leq 12$ (Foulds & Graham [1982] y Day et al. [1986]).

Los árboles se generan por inserción sucesiva de las especies en todas las posiciones posibles.

Branch and bound B&B

El algoritmo Branch and Bound B&B es un método comúnmente usado para la solución de programas enteros. Consiste en la “ enumeración controlada ” de

las soluciones factibles. A diferencia de los métodos de enumeración exhaustiva, en B&B se emplea algún criterio de acotamiento para restringir la exploración del espacio de búsqueda mientras se garantiza la optimalidad de la solución final.

En la reconstrucción filogenética, esta técnica fue introducida por primera vez por Hendy & Penny [1982] y Swofford & Olsen [1990]. Ellos proponen un algoritmo exacto que puede utilizarse cuando el número de especies es menor o igual a 30.

El algoritmo parte del único árbol posible formado por las tres primeras especies. Una nueva especie es añadida sobre este primer árbol, existen tres posibles lugares donde puede ser añadida, esto da origen a tres árboles, a partir de este conjunto pequeño de tres árboles, se determina algún criterio de acotamiento (en nuestro caso, la medida de ajuste Q dada por el MMC) para escoger uno de ellos, el escogido es aquel que posee menor valor sobre el criterio elegido (nuevamente tomando nuestro caso, escogeríamos el árbol con menor valor de Q), sobre el árbol elegido se repite la adición de una nueva especie, generando un conjunto más numeroso de árboles, seguidamente se obtiene el valor de discriminación para cada uno de ellos y se elige el de menor valor. Este proceso se repite hasta alcanzar las n especies.

Esta técnica va tomando únicamente árboles significativos para el conjunto de las n especies, evitando una enumeración exhaustiva.

La Figura 2.2 muestra el procedimiento.

2.3. HEURÍSTICAS

Cuando la dimensión de ciertos problemas crece, el número de operaciones requeridas por los algoritmos exactos puede llegar a aumentar tan explosivamente como para hacer imposible su aplicación. Se plantea entonces como una alternativa el uso de heurísticas. Una *heurística* para un problema de optimización es un algoritmo eficiente que encuentra una solución factible “ suficientemente buena ”, aunque no necesariamente óptima.

Las heurísticas se emplean con frecuencia para resolver ciertos problemas particularmente “ duros ” desde el punto de vista computacional, conocidos como problemas “ NP-difíciles ”. La Teoría de Complejidad provee argumentos , uno de ellos el Teorema de Cook (Cook et al. [1998]. Capítulo 9, página 718) para suponer que esos problemas no admiten un algoritmo eficiente para su solución exacta. Como se ha demostrado que la reconstrucción de filogenias en base a matrices de distancias pertenece a esta clase de problemas (Day [1983] y Foulds & Graham [1982]), las heurísticas juegan un papel importante en la solución de instancias

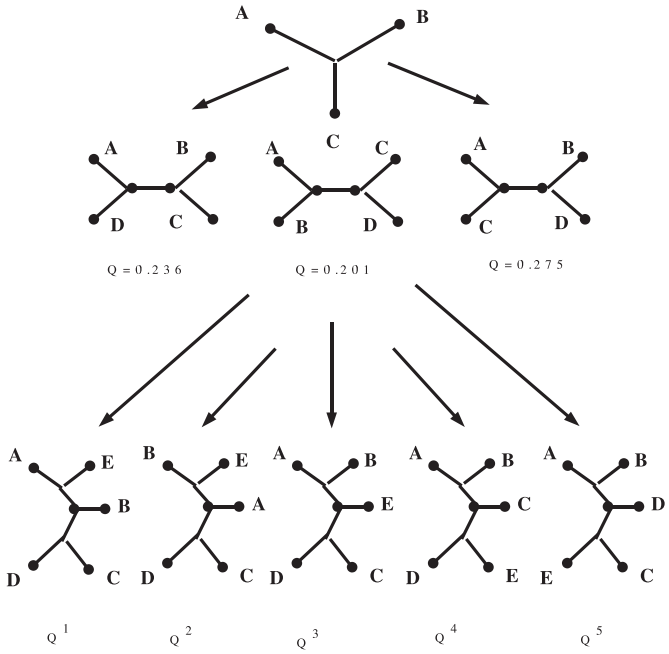


Figura 2.2: Método B&B. El árbol central del grupo de tres árboles es aquel que posee menor medida de ajuste, por tanto sobre este se añade una nueva especie, generando un nuevo grupo de 5 árboles, y se elige el árbol de menor valor de Q .

prácticas.

2.3.1. HEURÍSTICAS DE AGRUPAMIENTO DIRECTO

Dentro de este grupo encontramos los métodos más sencillos de agrupamiento (clustering). Se trata de algoritmos glotones que construyen un árbol filogenético al agrupar una tras otra las especies (o subgrupos de especies) que minimizan un cierto criterio de distancia. Entre los algoritmos más comunes se encuentran: WPGMA, UPGMA, Neighbor Joining.

Método WPGMA (Weighted Pair Group Method Using Arithmetic Average)

Este fue uno de los primeros métodos de agrupamiento y el más sencillo para la construcción de árboles con raíz. El WPGMA entrega como resultado un árbol en el cual las distancias entre la raíz y cada especies son las mismas (lo que se conoce como distancias ultramétricas).

Este algoritmo recibe un conjunto $S = \{1, 2, ..., n\}$ de especies y una matriz $\mathbb{D} \in \mathbb{R}^{n \times n}$ que contiene las distancias evolutivas entre todos los pares de especies.

El Algoritmo 2.1 presenta una descripción esquemática.

ENTRADA:

- Un conjunto $S = \{1, 2, ..., n\}$ de especies
- Una matriz simétrica no negativa $\mathbb{D} = (d_{ij}) \in \mathbb{R}^{n \times n}$ de distancias

SALIDA:

Un árbol (con raíz) ultramétrico $H := (V, T)$ y un vector $l \in \mathbb{R}^T$ de longitudes de aristas.

Inicialización:

- $V := S$,
- $T := \emptyset$
- $Q := S$

Fase principal:

mientras |Q| $\neq \emptyset$ **hacer**

- Determinar $i, j \in Q$ con $d_{ij} :=$ min { $d_{kl} : k, l \in Q$ }
- Agregar un nuevo elemento z a V
- Añadir nuevas aristas entre i, j y z :
  - $T := T \bigcup \{iz, jz\}$

  Para $k \in Q \setminus \{z\}$

$$d_{zk} := \frac{d_{ik} + d_{jk}}{2} \tag{2.12}$$

- $Q := Q \setminus \{i, j\} \bigcup \{z\}$

fin mientras

Algoritmo 2.1: WPGMA

Básicamente, este método consiste en ir seleccionando pares de especies con menor distancia evolutiva e ir formando grupos. Cada vez que dos especies son agrupadas, son reemplazadas por un nuevo nodo (ancestro hipotético), el mismo que entra a formar parte del conjunto Q de nodos por procesar.

La distancia del ancestro hipotético hacia los demás nodos en Q es el promedio simple de las distancias de los nodos agrupados. El proceso se repite mientras $Q \neq \emptyset$.

Método UPGMA (Unweighted Pair Group Method Using Arithmetic Average)

Este método fue introducido por Sokal & Michener [1958] como parte de los métodos de agrupamiento. Similar al método WPGMA , su diferencia radica en que las distancias son calculadas con un promedio aritmético que depende de el número de especies de cada grupo. Si no se quiere trabajar con un simple promedio , el UPGMA es una buena opción.

El Algoritmo 2.2 presenta una descripción esquemática del UPGMA.

ENTRADA:

- Un conjunto $S = \{1, 2, ..., n\}$ de especies
- Una matriz simétrica no negativa $\mathbb{D} = (d_{ij}) \in \mathbb{R}^{n \times n}$ de distancias

SALIDA:

Un árbol (con raíz) ultramétrico $H := (V, T)$ y un vector $l \in \mathbb{R}^T$ de longitudes de aristas.

Inicialización:

- $n_i := 1, \qquad \forall i \in S$
- $V := S,$
- $T := \emptyset$
- $Q := S$

Fase principal:

mientras |Q| $\neq \emptyset$ **hacer**

- Determinar $i, j \in Q$ con $d_{ij} :=$ min { $d_{kl} : k, l \in Q$ }
- Agregar un nuevo elemento z a V
- Añadir nuevas aristas entre i, j y z :
  - $T := T \bigcup \{iz, jz\}$
  - $n_z := n_i + n_j$

  Para $k \in Q \setminus \{z\}$

$$d_{zk} := \frac{n_i}{n_z} d_{ik} + \frac{n_j}{n_z} d_{jk} \tag{2.13}$$

- $Q := Q \setminus \{i, j\} \bigcup \{z\}$

fin mientras

Algoritmo 2.2: UPGMA

Este método requiere de n^3 operaciones para inferir una filogenia con n especies. Cada búsqueda del elemento más pequeño dentro de la matriz de distan-

cias demanda una cantidad proporcional a n^2 operaciones. Sin embargo, el proceso puede hacerse más eficiente creando una lista de tamaño n que almacene para cada columna, el índice de la fila que contiene el elemento más pequeño (o viceversa), así la búsqueda del mínimo requiere una cantidad proporcional a n operaciones en lugar de n^2. El trabajo extra requerido para manterner actualizada esta lista luego de cada agrupamiento es proporcional a n. En total esta variante del algoritmo requiere n^2 operaciones.

La metodología de este método es similar al método WPGMA, difieren en la forma de calcular las distancias.

Método del vecino más cercano (Neighbor joining- NJ)

Este algoritmo fue desarrollado por Saitou & Nei [1987] y es muy popular dentro de la comunidad de biólogos. Se lo ha considerado como el método de agrupamiento de menor costo, que produce aproximaciones satisfactoriamente buenas. Puede trabajar con un centenar de especies, empleando la potencia de cálculo de un ordenador común de escritorio . El algortimo 2.3 presenta la descripción del esquema del Neighbor Joining.

El algoritmo NJ se estructura de manera similar al WPGMA y el UPGMA, pero su diferencia radica en la forma de tomar el valor mínimo de la matriz de distancias evolutivas. Un factor adicional forma parte del NJ, el vector r de divergencias que almacena las diferencias de una especie con respecto a las otras, éstas nos permiten calcular una nueva distancia $\widehat{d_{ij}}$ que muestra una relación entre la distancia evolutiva d_{ij} de un par de especies con respecto a los otras. Con éstas, se construye la matriz $\mathbb{D}'$ y de esta matriz se toma el valor mínimo. Las longitudes de las aristas se calculan como indican las ecuaciones 2.16 y 2.17.

El número de operaciones que requiere este algoritmo es una cantidad proporcional a n^3 debido a la necesidad de actualizar r_i.

2.3.2. METAHEURÍSTICAS O BÚSQUEDA LOCAL

Los métodos de búsqueda local son métodos iterativos que empiezan con una solución factible y la van mejorando progresivamente. El procedimiento consiste en explorar una vecindad previamente definida para cada punto del espacio de soluciones y elegir una nueva solución dentro de tal vecindad.

El esquema general de una búsqueda local para el problema de reconstrucción filogenética es:

1. Generar un árbol inicial

ENTRADA:

- Un conjunto $S = \{1, 2, ..., n\}$ de especies
- Una matriz no negativa $\mathbb{D} \in \mathbb{R}^{n \times n}$ de distancias

SALIDA:

Un árbol (con raíz) aditivo $H := (V, T)$ y un vector $l \in \mathbb{R}^T$ de longitudes de aristas.

Inicialización:

- $V := S$,
- $T := \emptyset$
- $Q := S$

Fase principal:

mientras |Q| $\neq \emptyset$ **hacer**

- Calcular:

$$r_i = \sum_{j \neq i} \frac{d_{ij}}{|Q| - 2} \qquad \forall i \in Q \tag{2.14}$$

- Calcular

$$\widehat{d}_{ij} = d_{ij} - r_i - r_j \qquad \forall i, j \in Q \tag{2.15}$$

- Determinar $i, j \in Q$ con $\widehat{d}_{ij} :=$ min $\{ \widehat{d}_{kl} : k, l \in Q \}$
- Agregar un elemento nuevo z a V
- Añadir nuevas aristas entre i, j y z.
  - $T := T \bigcup \{iz, jz\}$
- Calcular

$$l_{i_z} = \frac{1}{2}(d_{ij} + r_i - r_j) \qquad l_{j_z} = \frac{1}{2}(d_{ij} + r_j - r_i) \tag{2.16}$$

- Para $k \in Q \setminus \{z\}$

$$d_{z,k} = \frac{1}{2}(d_{ik} + d_{jk} - d_{ij}) \tag{2.17}$$

$$Q := Q \setminus \{i, j\} \bigcup \{z\}$$

fin mientras

Algoritmo 2.3: Neighbor Joining

2. Mientras no se satisfaga un criterio de parada:
   Sustituir el árbol actual por un “ vecino ”.

Las variaciones entre estos métodos son los criterios de vecindad y los mecanismos para controlar la búsqueda en el paso 2.

Movimientos locales

Los movimientos locales son criterios que permiten la generación de árboles. La idea de “ movimiento ” constituye la base de las heurísticas de búsqueda local. La forma en que estos movimientos se realizan dan lugar a las siguientes heurísticas.

Nearest-neighbor interchanges (NNI)

Nearest Neighbor Interchange es un criterio de búsqueda de árboles mediante movimientos sobre un árbol.

Por la definición de árbol filogenético, cada arista interna tiene cuatro subárboles conectados a ésta (un subárbol puede ser un solo nodo). El NNI elimina una rama interna e_{ij}, desconectando a los cuatro subárboles adheridos a ésta y busca alternativas distintas de volver a conectarlos a e_{ij}. Hay únicamente dos alternativas de reconección y se muestra en la Figura 2.3.

Este proceso se repite para cada arista interna del árbol.

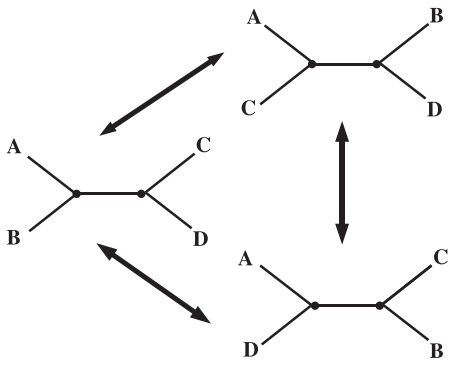


Figura 2.3: Nearest-neighbor interchanges (NNI)

Como se vio en el Capitulo 1, si H es un árbol de n especies, entonces habrá $n-3$ aristas internas, así se examinará $2(n-3)$ vecinos sobre H.

Subtree pruning and regrafting (SPR)

Esta heurística remueve una arista e ya sea interna o externa de un árbol H y lo separa en dos subárboles H^1 y H^2, la arista removida que queda atada a uno de los subárboles es insertada en cualquier lugar posible del otro árbol. La Figura 2.4 muestra el proceso.

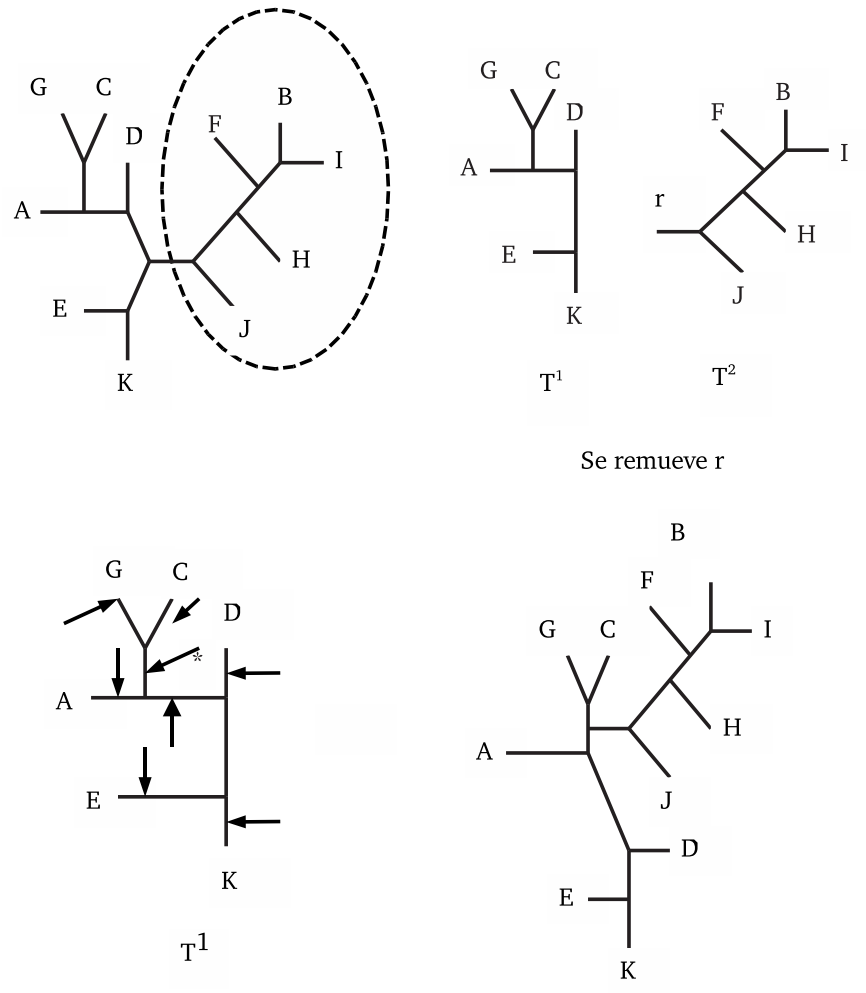


Figura 2.4: Subtree pruning and regrafting (SPR)

En general , si H^1 tiene n_1 y H^2 tiene n_2 especies, se tendrán $2n_1 - 3$ posibles lugares para volver a insertar e. Uno de estos posibles lugares dará el árbol original H. Es decir, H tendrá $(2n_1 - 3 - 1) + (2n_2 - 3 - 1) = 2n - 8$ árboles vecinos por cada arista interior, y $2n - 6$ vecinos por cada arista externa, el total de vecinos que examina SPR será $n(2n - 6) + (n - 3)(2n - 8) = 4(n - 3)(n - 2)$. Sin embargo algunos de estos vecinos pueden coincidir entre sí.

Tree bisection and reconnection (TBR)

La heuristica TBR es similar a (SPR) la diferencia radica en que ahora no solamente puede enlazarse la rama removida sino cualquiera de las ramas que pertenezcan al subárbol, como lo demuestra la Figura 2.5.

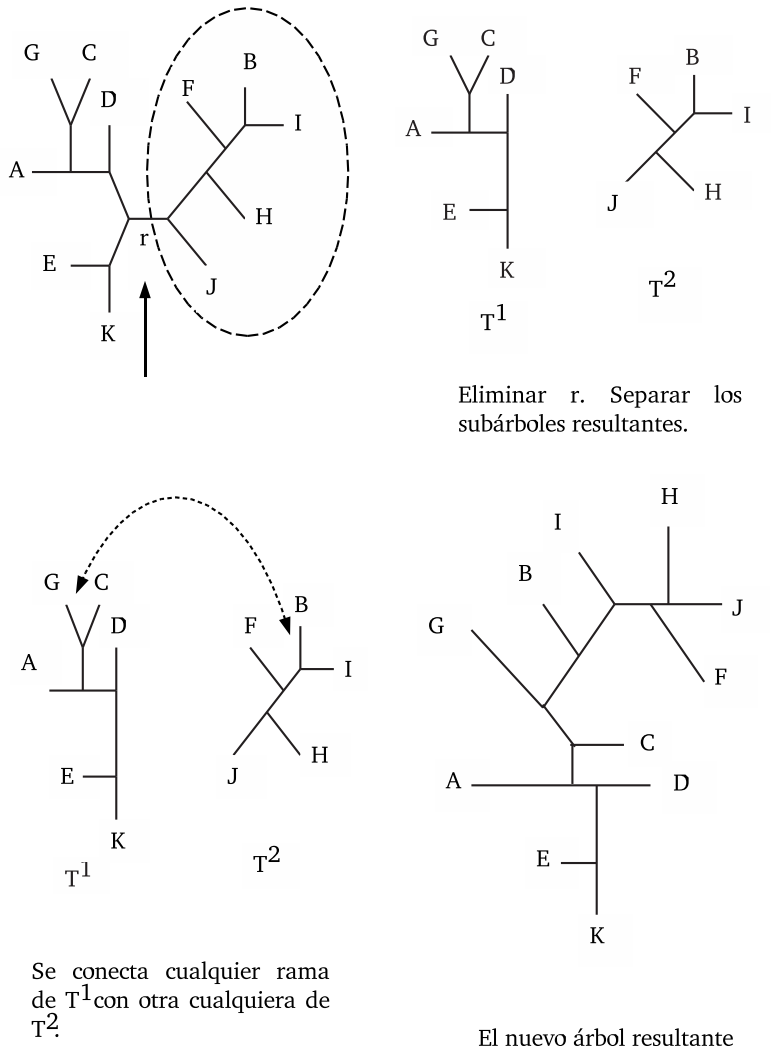


Figura 2.5: Tree bisection and reconnection (TBR)

Si H^1 tiene n_1 especies y H^2 , n_2 especies , se tendrá $(2n_1-3)(2n_2-3)$ posibles formas de reconectar H^1 y H^2, claro entre ellas la que nos devolverá H. El número de vecinos explorados por TBR se especifica en Allen & Steel [2001].

Estos tres criterios de vecindad presentados son un tipo de métodos exhaustivos. Cuando el número de especies es pequeño para un $n < 11$ éstos realizan

una búsqueda exhaustiva exitosa pero cuando este número es significativo se emplean otros métodos con estrategias de simulación como los algoritmos genéticos (Ver detalles en Matsuda [1996]), Tree-fusing (Golobof [1999]) entre otros.

Recocido Simulado (Simulated Annealing)

El Recocido Simulado (RS) es una de las primeras metaheurísticas más empleadas dentro de la Optimización Combinatoria. Fue introducido por Kirkpatrick et al. [1983]. Esta metaheurística es un método de búsqueda local que permite movimientos que empeoran la función objetivo (movimientos " ascendentes ") para evitar quedar atrapado anticipadamente en un óptimo local.

Su nombre se deriva de un algoritmo diseñado para simular el enfriamiento de un metal , proceso denominado " recocido ".

El Recocido Simulado controla la admisión de movimientos ascendentes mediante una función de probabilidad que se vuelve más restrictiva conforme avanza la búsqueda.

El uso de este *control* se basa en el trabajo de Metropolis et al. [1953] en el campo de la termodinámica estadística. Metrópolis modeló el proceso de recocido, simulando cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable (congelado). Las leyes de la termodinámica dicen que a una temperatura t la probabilidad de un incremento energético de magnitud δE puede aproximar por:

$$P(\delta E) = \exp(-\delta E/kt) \tag{2.18}$$

siendo k una constante física denominada de Boltzman. El algoritmo de Metrópolis genera una perturbación aleatoria en un sistema y calcula los cambios de energía resultantes : si hay una caída energética, el cambio se acepta automáticamente caso contrario, si se produce un incremento energético, el cambio es aceptado con una probabilidad dada por (2.18). Observar que cuando la temperatura es alta el exponente es cercano a 1 y se permiten casi todos los desplazamientos, mientras que a medida que baja se van restringiendo los desplazamientos degradantes. Esto permite que la solución escape de mínimos locales. El algoritmo produce de esta forma una cadena de Markov que se aproxima al equilibrio termodinámico. El proceso se repite durante un número predefinido de iteraciones en series decrecientes de temperatura, hasta que el sistema esté frío.

Publicaciones de Kirkpatrick et al. [1983] y Cerny [1985] mostraron cómo este proceso podría ser aplicado a problemas de optimización, asociando conceptos

Simulación Termodinámica	Optimización Combinatoria
Estados del Sistema	Soluciones factibles
Energía	Costo
Cambios de estado	Solución de control
Temperatura	Parámetro de control
Estado congelado	Solución heurística

Cuadro 2.1: Simulación termodinámica en relación con la optimización combinatoria

clave del proceso original de simulación con elementos de optimización combinatoria. Ver Tabla 2.1 (Dowsland & Díaz [2001]).

Para el caso de la construcción de árboles filogenéticos el Recocido Simulado puede formularse como se muestra en el algoritmo 2.4.

Búsqueda Tabú (Tabu Search)

La Búsqueda Tabú es un procedimiento metaheurístico cuya filosofía se da con respecto al funcionamiento del cerebro humano. Se afirma que una persona posee una memoria a corto, mediano y largo plazo, en la cual se basa el comportamiento a lo largo de su vida.

La búsqueda local del método tabú esta controlada por ciertas prohibiciones que vienen dadas por los “ recuerdos ” del procedimiento, así como por ciertos criterios para evaluar el beneficio asociado a aceptar un movimiento. El método guarda en memoria las últimas soluciones visitadas (memoria a corto plazo) e impide el retorno a las mismas durante un número dado de iteraciones. Las soluciones prohibidas constituyen una lista tabú que se actualiza con cada iteración y que permite que el algoritmo salga de óptimos locales sin dejar de efectuar buenos movimientos.

A esta memoria a corto plazo se agregan en algunos casos las memorias a mediano y largo plazos, las cuales permiten emprender una intensificación y diversificación de la búsqueda.

La búsqueda tabú cuenta con un criterio, denominado de aspiración, que permite que una solución tabú deje de constar en lista.

Este criterio generalmente va ligado a un fuerte decrecimiento de la función objetivo, aunque también se pueden presentar otras alternativas de aspiración dependientes incluso de la iteración en que se encuentre el proceso. Es claro que un procedimiento de este tipo requiere de gran cantidad de memoria para almacenar

Sea Q el costo de la solución (medida de ajuste) del árbol H y sea $N(H)$ su vecindad.

Inicialización:

- Escoger un árbol inicial H_0 , una temperatura inicial $t_0 = 0$, una función de reducción de temperatura α, un número de iteraciones K , una longitud de piso L y un criterio de parada N.
- $t = t_0$

Fase principal:

repetir

repetir

Seleccionar aleatoriamente un árbol $H = N(H_0)$;

Sea $\delta = Q(H) - Q(H_0)$;

si $\delta < 0$ **entonces**

$H = H_0$;

si no

Calcular

$$P(\delta) = \exp(-\delta/t) \tag{2.19}$$

Generar aleatoriamente $x \in U[0, 1]$;

si $P < x < 1$ **entonces**

aceptar H

si no

rechazar H

fin si

fin si

hasta que iteraciones % L= 0

$t = \alpha t$;

hasta que N = CIERTO

Algoritmo 2.4: Recocido Simulado

soluciones pasadas, razón por la cual se vuelve necesario buscar herramientas que permitan optimizar el espacio utilizado. Una estrategia usual consiste en el almacenamiento de atributos de movimientos. El Algoritmo 2.5 ilustra una representación esquemática del método de búsqueda tabú para el problema de la reconstrucción de filogenias.

Notación

X : conjunto de árboles admisibles

Q : medida de ajuste

A : función de aspiración

T : lista tabú

Q^+ : cota inferior de la medida de ajuste

max : número máximo de iteraciones para mejorar un árbol.

Inicialización:

- Escoger un árbol inicial $H \in X$ y fijar $iter := 0$; $T := \emptyset$;
- Se inicializa la función de aspiración A

Fase principal:

mientras $Q(H) > Q^+$,$iter - cont < max$ **hacer**

$iter := iter + 1$

Generar una vecindad $N(H)$

Escojer el mejor árbol $H' \in N(H)$ tal que $H' \notin T$ *ó* $Q(H') \leqslant A(Q(H))$

Actualizar A y T

$H := H'$

si $Q(H) < Q^*$ **entonces**

$H^* := H$; $cont := iter$

fin si

fin mientras

Algoritmo 2.5: Búsqueda Tabú

Capítulo 3

UNA METAHEURÍSTICA PARA LA CONSTRUCCIÓN DE ÁRBOLES FILOGENÉTICOS

3.1. DESCRIPCIÓN GENERAL DEL ALGORITMO

Como se vió en el Capítulo 1, el número de posibles árboles crece explosivamente con el número de especies, por lo que la exploración exhaustiva de todo el espacio de búsqueda correspondiente a un grupo de n especies se vuelve intratable computacionalmente aún para los valores de n relativamente bajos como 13 o 14. Los diversos métodos y heurísticas usados en la práctica únicamente exploran un espacio de árboles representativo y dentro de éste buscan el mejor árbol. El criterio de selección del mejor árbol varía entre las distintas técnicas empleadas.

Esta tesis formula un algoritmo de búsqueda bajo el esquema del Recocido Simulado (RS) descrito en el capítulo anterior, cuyo criterio de elección del mejor árbol se basa en la medida de ajuste Q dada por la ecuación 2.1.

El esquema general de nuestra metaheurística comprende los siguientes pasos (resumidos en el esquema de la Figura 3.1):

1. Recepción de datos

   - Datos almacenados como secuencias de ADN
   - Datos almacenados como distancias evolutivas

   Los datos presentados como secuencias de ADN, deben estar previamente alineados.

2. Cálculo de la matriz de distancias evolutivas si los datos fueron ingresados como secuencias de ADN, de acuerdo al modelo de evolución de Jukes & Cantor [1969] descritos en la Sección 1.3.2 en la página 9. Si se presentan datos como matrices de distancias entra directamente.

3. Construcción de la filogenia inicial H_0 con el método Neighbor Joining (vecino más cercano) presentado en la Subsección 2.3.1, página 31.

4. Cálculo de la medida de ajuste Q_0 y las longitudes de las ramas de la filogenia inicial por medio del método de los Mínimos Cuadrados MMC.(Ver Sección 2.1, pag.23)

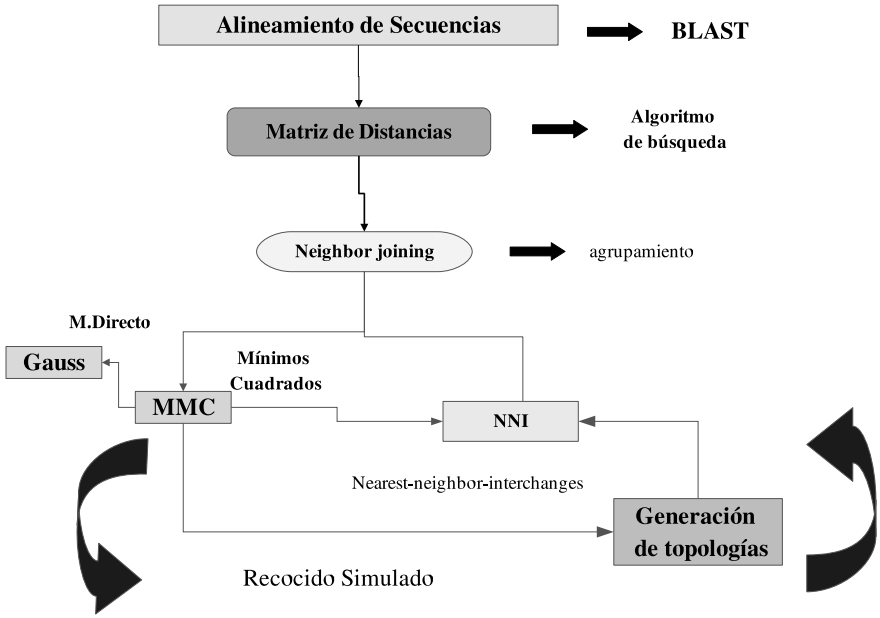


Figura 3.1: Esquema general de la metaheurística

5. Búsqueda del mejor árbol con el Recocido Simulado

   a) Inicializar los parámetros del RS (temperatura inicial $t_0 = t$, longitud de piso L y control de la temperatura α) con valores calibrados para nuestro propósito; $t_0 = 50, L = 150$ y $\alpha = 0{,}85$ Ver detalles de la calibración en la Sección 4.3.

   b) Generar una nueva filogenia H' a partir de H_0, mediante movimientos sobre las ramas dados por la heurística NNI descrita en la sección 2.3.1 página 33.

   c) Calcular la medida de ajuste Q' para H' con el MMC con sus respectivas longitudes de las aristas.

   d) Comparar Q_0 con Q' . Si $Q' < Q_0$ se acepta H' , caso contrario

   - Calcular $\delta = Q' - Q_0$ y generar un valor $x \rightsquigarrow U[0, 1]$
   - Calcular $P = e^{\frac{\delta}{t_0}}$
     Si $P < x < 1$ aceptar H' con probabilidad P.
     Si no se rechaza H'

   Este proceso iterativo se hace hasta alcanzar un primer criterio de parada que es la modulidad 0 de L , esto significa que en cada L iteraciones se inicia una nueva búsqueda, y el parámetro t disminuye $t := \alpha * t$ pero para parar este primer procedimiento, un segundo criterio de parada es considerado,

si no hay una mejora en la medida de ajuste después de 4 bloques de L, termina el algoritmo.

Se describe en las siguientes secciones uno por uno todos los pasos del esquema anterior sobre la base de un ejemplo práctico. Detalles de la implementación computacional concernientes a cada uno de los métodos que intervienen en la metaheurística serán tratados en el siguiente capítulo.

3.2. ALMACENAMIENTO DE DATOS

Los datos de entrada están almacenados en un archivo de texto ya sea como secuencias de ADN o como matrices de distancias. El archivo que contiene los datos de secuencias de ADN empieza con el número de especies y la longitud de las secuencias ya sea en una o 2 líneas del archivo, cada línea siguiente contiene el nombre de la especie y en los 10 primeros caracteres seguidos por secuencia de ADN respectiva. La Figura 3.2 muestra el archivo de entrada para la instancia `tapir5.txt` [1] , la misma que contiene información de secuencias de ADN para un grupo de cinco especies, de las cuales cuatro son especies de tapires y la quinta es el caballo.

```
5 37
caballus..GGATTCCAAGACGCAACATCCCCTATTATAGAAGAAC
bairdii...GGATTTCAAGATGCAACATCTCCCATTATAGAAGAAC
terrestrisGGATTTCAAGATGCAACATCTCCTATTATAGAAGAAC
pinchag...GGATTTCAAGATGCAACATCTCCTATTATAGAAGAAC
indicus...GGATTTCAAGATGCAACATCTCCTATTATAGAAGAGC
```

Figura 3.2: Secuencias de 5 especies con una longitd de 37. Instancia `tapir5.txt`

Este tipo de formato se ha basado en parte en el formato NEXUS, que es un estándar actual para los programas de reconstrucción de filogenias (Ver Apéndice B).

El archivo de matriz de distancias inicia con las dimensiones de la matriz y contiene a continuación los elementos de la misma, fila por fila. Al igual que para el archivo de secuencias, los 10 primeros caracteres de cada línea están reservados

[1] Para fines ilustrativos se presenta aquí únicamente subsecuencias de 37 nucleótidos de las secuencias originales de 637 nucleótidos

para almacenar el nombre de una especie. Luego vienen las distancias hacia cada una de las otras especies, separadas entre sí por espacios en blanco. (Ver Figura 3.3)

```
5 5
caballus..0.000000  0.189816  0.210049  0.192835  0.202766
bairdii...0.189816  0.000000  0.098899  0.087952  0.095324
terrestris0.210049  0.098899  0.000000  0.025743  0.110315
pinchag...0.192835  0.087952  0.025743  0.000000  0.102677
indicus...0.202766  0.095324  0.110315  0.102677  0.000000
```

Figura 3.3: Matriz de distancias evolutivas para la instancia `tapir5.txt`

Tal como se señalo en la Sección 2.1, la matriz de distancias evolutivas $\mathbb{D}$,es una matriz simétrica donde cada elemento d_{ij} indica la variabilidad genética entre las especies i e j de acuerdo al modelo de evolución. En nuestro ejemplo, la variabilidad genética d_{01} entre el caballo (0) y el tapir bairdii (1) es 0.189816 de acuerdo al modelo de Jukes & Cantor [1969].

Los datos ingresados como secuencias se procesan y se almacenan en una matriz de distancias evolutivas, de acuerdo al modelo de evolución de Jukes & Cantor [1969], el cual consiste en asignar igual probabilidad a todas las transiciones y transversiones de se dan entre pares de secuencias.
Esta matriz de distancias constituye el punto de partida para nuestro algoritmo de reconstrucción filogenética.

3.3. CONSTRUCCIÓN DE LA FILOGENIA INICIAL

El esquema del Recocido Simulado (RS), necesita de una primera filogenia para empezar la búsqueda. Para generarla, se emplea el algoritmo de agrupamiento Neighbor Joining (NJ) descrito en la Subsección 2.3.1.

Las distancias evolutivas son almacenadas en una matriz $\mathbb{D}$. Se parte de una estructura en estrella que representa al árbol filogenético no resuelto y se van agrupando sucesivamente los diferentes nodos, el par de nodos elegido es aquel que posee la menor distancia sobre $\mathbb{D}$.

Mostramos a continuación la aplicación de este algoritmo de agrupamiento sobre la instancia `tapir5.txt`.

1. Se ubican las especies en el árbol en forma de estrella. (Por simplicidad, nos referimos en adelante a las especies con los números 0,...,4, como se indica en la Figura 3.4.

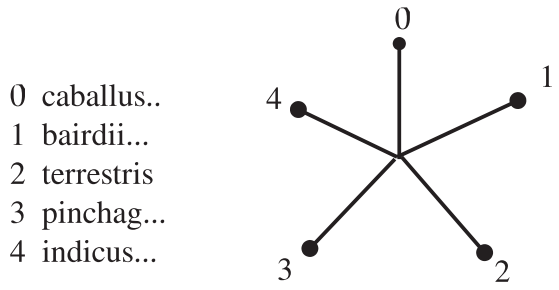


Figura 3.4: Asignación de un número para cada especie y la estructura inicial de árbol en forma de estrella

2. Se construye la matriz $\mathbb{D}'$ a partir de la matriz de distancias evolutivas $\mathbb{D}$ y las divergencias **r** para cada nodo.

   La divergencia para cada nodo es la suma de las diferencias de nucléotidos que tiene con respecto a las demás especies, calculándose como:

   $$r_i = \sum_{j \neq i} \frac{d_{ij}}{|Q| - 2} \qquad \forall i, j \in Q$$

   donde Q es la cardinalidad del grupo actual de especies.

   Así, los elementos de $\mathbb{D}'$ se obtienen como:

   $$d'_{ij} = d_{ij} - r_i - r_j \qquad i, j = 0, ..., 4 \ .$$

   A continuación se elige el menor valor.

	0	1	2	3	4
0	0	-0.226449	-0.198622	-0.203448	-0.22669
1	-0.226449	0	-0.201319	-0.200596	-0.226845
2	-0.198622	-0.201319	0	-0.25238	-0.202888
3	-0.203448	-0.200596	-0.25238	0	-0.198786
4	-0.22669	-0.226845	-0.202888	-0.198786	0

Figura 3.5: Matriz $\mathbb{D}'$ donde el rectángulo contiene el menor valor.

La Figura 3.5 indica que el elemento d'_{23} contiene el menor valor, por tanto el primer par de nodos a agrupar es $\{2,3\}$. Se crea un nuevo nodo 5 que representa dicha agrupación. Ahora este nuevo nodo forma parte del árbol en forma de estrella y los nodos 2 y 3 ya no se tomarán en cuenta para los siguientes agrupamientos. El nuevo árbol se muestra la figura 3.6.

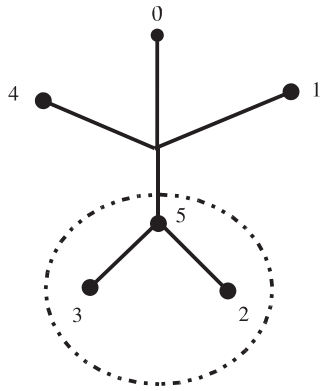


Figura 3.6: Árbol con el primer grupo formado

3. La matriz de distancias evolutivas $\mathbb{D}$ se actualiza con las nuevas distancias originadas por el grupo conformado, calculando las distancias de 2 y 3 al nuevo nodo 5 como:

$$d_{2,5} = \frac{1}{2}(d_{23} + r_2 - r_3) \qquad d_{3,5} = \frac{1}{2}(d_{23} + r_3 - r_2) \quad (3.1)$$

y las distancias entre el nuevo nodo 5 y cada una de los nodos restantes como:

$$d_{(23),k} = \frac{1}{2}(d_{2k} + d_{3k} - d_{23}) \quad (3.2)$$

para $k = 0, 1, 4$ Se eliminan las columnas y filas correspondientes a las especies 2 y 3 y se añaden una nueva fila y una nueva columna para el grupo (23).

4. Otra vez se obtienen las divergencias a partir de $\mathbb{D}$, se construye $\mathbb{D}'$ y se selecciona el elemento de menor valor. En este caso se trata de d_{14} (Ver Figura 3.7), por lo que los nodos 1 y 4 son agrupados introduciendo un nuevo nodo 6, tal como se indica en la Figura 3.8.

	0	1	5	4
0	0	-0.276109	-0.276406	-0.276290
1	-0.27610	0	-0.276290	-0.276406
5	-0.27640	-0.276290	0	-0.276109
4	-0.27629	-0.276406	-0.276109	0

Figura 3.7: Matriz $\mathbb{D}'$ actualizada donde el rectangulo contiene el menor valor.

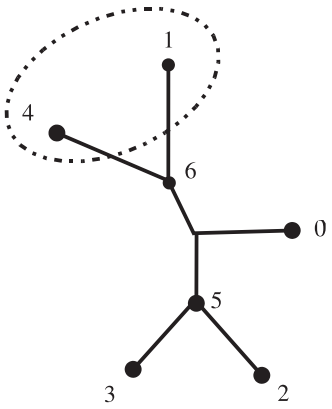


Figura 3.8: Árbol con el segundo grupo formado

5. El proceso anterior se repite, generando primero la matriz $\mathbb{D}'$ entre los nodos 0, 5, y 6, representada en la Figura 3.9 y luego el árbol correspondiente (Ver Figura 3.10).

	0	5	6
0	0	-0.198622	-0.364963
5	-0.364963	-0.25238	-0.364963
6	-0.364963	-0.202888	0

Figura 3.9: Matriz $\mathbb{D}'$ donde el rectángulo contiene el menor valor.

6. Una última iteración da como resultado el árbol de la Figura 3.11.

3.4. DETERMINACIÓN DEL VALOR DE AJUSTE

Para poder seleccionar uno de los posibles árboles durante el proceso de búsqueda local, es necesario disponer de algún cirterio para evaluar qué tanto se ajusta una determinada filogenia a los datos de entrada provistos por la matriz de distancias evolutivas. En nuestro caso, tomamos como medida de ajuste el valor Q obtenido por el método de Mínimos Cuadrados descritos en la Sección 2.1 , página 37.

El cálculo de Q se da por la ecuación matricial 2.9. Como se describió en la Sección 2.1, este sistema se obtiene al tratar de determinar las longitudes de

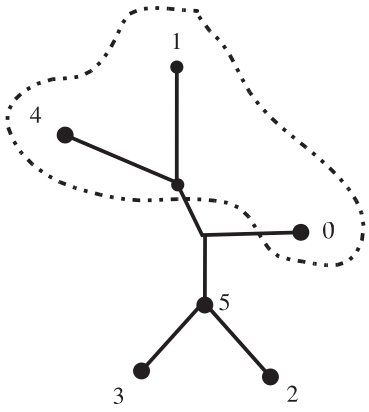


Figura 3.10: Árbol con el tercer grupo formado

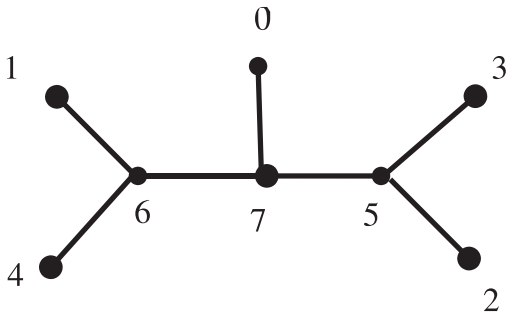


Figura 3.11: Árbol filogenético completo

las aristas que minimizan la suma de los cuadrados de las diferencias entre las distancias calculadas sobre el árbol y las distancias evolutivas observadas.

El sistema de ecuaciones lineales dado por el MMC tiene como parámetros la matriz topológica $\mathbb{X}$ y el vector $\widehat{d}$ de todas las distancias entre pares de nodos.

Las variables están dadas por el vector l de las longitudes de las aristas, por tanto el sistema es : $\mathbb{X}^T\widehat{d} = (\mathbb{X}^T\mathbb{X})l$.

En el caso de la instancia `tapir5.txt` $\mathbb{X}$ y $\widehat{d}$ tomam los valores presentados en la Figura 3.12.

$$\mathbb{X} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \widehat{d} = \begin{bmatrix} 0{,}202864 \\ 0{,}186832 \\ 0{,}196812 \\ 0{,}0972286 \\ 0{,}0867451 \\ 0{,}0937178 \\ 0{,}0251464 \\ 0{,}107861 \\ 0{,}100756 \end{bmatrix}$$

Figura 3.12: Matriz topológica $\mathbb{X}$ y vector $\widehat{d}$ para la instancia `tapir5.txt`

Resolviendo el sistema de ecuaciones por el método directo de Gauss (Burden & Faires [1985]) se obtiene el vector de longitudes de ramas dado en la figura 3.13.

$$l = \begin{bmatrix} 0{,}143766 \\ 0{,}0407583 \\ 0{,}0181767 \\ 0{,}00696969 \\ 0{,}0529595 \\ 0{,}0385088 \\ 0{,}00020674 \end{bmatrix}$$

Figura 3.13: Longitudes de ramas del árbol

Una vez obtenido l, se calcula Q con 2.9 y se tiene $Q := 0{,}0528007$

El algoritmo retorna siempre el árbol con el menor valor de Q de entre todos los explorados durante la búsqueda local.

3.5. MOVIMIENTOS LOCALES

El algoritmo del Recocido Simulado a más de requerir una filogenia inicial, necesita un espacio de árboles para explorar y sobre éste encontrar un árbol cuya medida de ajuste sea mínima.

Para generar este espacio, se empleó la heurística NNI. Sobre la filogenia inicial obtenido por el NJ, se realizaron los dos tipos de movimientos que propone dicha heurística, descritos a continuacion.

Un primer paso, fue escoger una arista interna $e_{ij} := e_0$ y desconectar todas las aristas adyacentes a e_{ij}.

Supongamos que las aristas desconectadas son: $e_{ki} := e_1, \quad e_{li} := e_2, \quad e_{jm} := e_3, \quad e_{jn} := e_4$, el primer movimiento, las reconecta nuevamente intercambiando las aristas e_2 y e_3.

El segundo movimiento, al igual que el primero intercambia dos aristas, en este caso se intercambian las aristas e_2 y e_4.

La Figura 3.14 muestra los respectivos movimientos aplicados a la instancia en estudio (`tapir5.txt`).

Una vez realizados los dos movimientos, se obtienen los dos primeros posibles árboles para nuestro espacio. Este proceso se realiza para cada una de las aristas internas del árbol inicial, y a su vez sobre cada uno de los árboles que se van generando.

Metaheurística RS en acción

La filogenia inicial H_0 , su medida de ajuste Q_0, y la heurística de generación de árboles NNI constituyen los elementos principales para poner en marcha el algoritmo del Recocido Simulado.

Mediante el ejemplo de la instancia `tapir5.txt` se ilustra este algoritmo.

Iniciamos generando un árbol H a partir del H_0 con el NNI, evaluamos su medida de ajuste Q y la comparamos con Q_0. Ver Figura 3.15.

Sea $\delta = Q - Q_0$, entonces si $\delta < 0$, se acepta H como un árbol mejor al inicial, si no se calcula $P(\delta) = \exp(-\delta/t)$. Se genera un número aleatorio x entre 0 y 1 , de tal forma que si $P < x < 1$, se acepta H con una probabilidad P, si no cumple una de estas condiciones, se genera un nuevo árbol con el otro movimiento del NNI sobre el árbol inicial.

En nuestro ejemplo $\delta = 0{,}0033915 > 0$ por tanto, no se acepta el segundo árbol. Calculamos $P(\delta) = 0{,}99932193$ y generamos un número aleatorio $x = 0{,}0023$, como $P < x < 1$ no cumple, se vuelve a rechazar H.

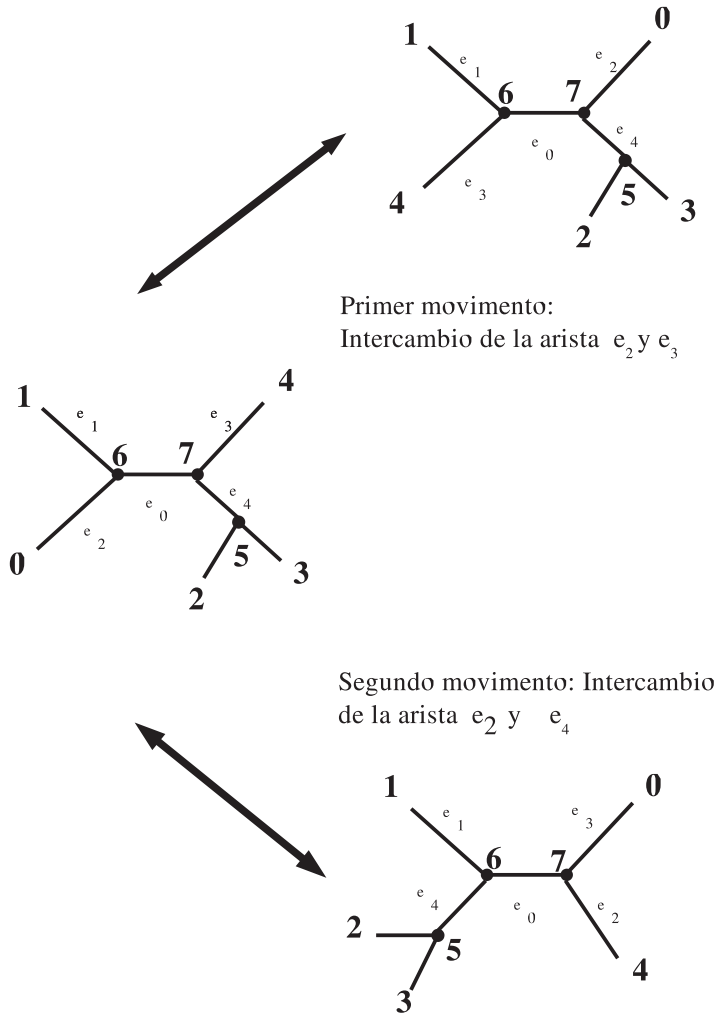


Figura 3.14: Cambios del NNI sobre el árbol inicial.

Ahora generamos otro árbol con el segundo movimiento de NNI. Ver Figura 3.16.

Ahora tenemos un $\delta = 0{,}0041044 > 0$, nuevamente no aceptamos este segundo árbol y calculamos $P(\delta) = 0{,}999762$ y $x = 0{,}999999$ como $P < x < 1$, se cumple, se acepta este árbol con una probabilidad de $0{,}999762$.

Si una de las condiciones dadas se cumple, el árbol H elegido toma el lugar del árbol inicial y se repite el proceso anterior. Este procedimiento se lo hace hasta alcanzar el primer criterio de parada, en nuestro caso, las L iteraciones luego de varias disminuciones del parámetro $t := \alpha \times t$. El algoritmo termina cuando se han alcanzado los cuatro bloques de L.

En nuestro ejemplo luego de varias iteraciones hechas por el RS, se obtuvo un árbol significativo con una medida de ajuste mínima, mostrado en la Figura 3.17.

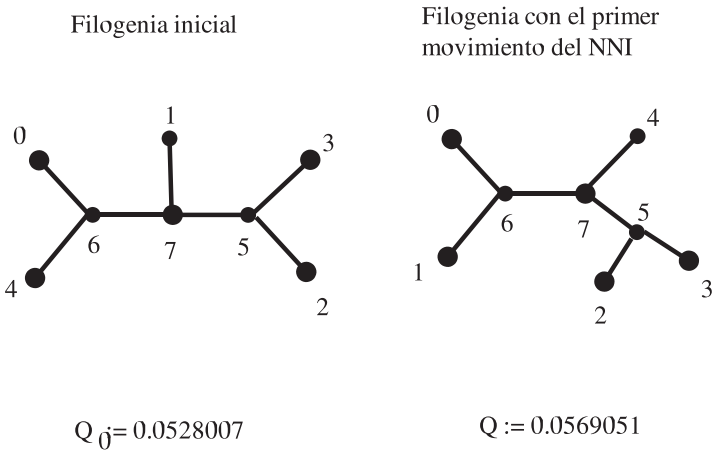


Figura 3.15: Primer movimiento del NNI en la instancia `tapir5.txt`

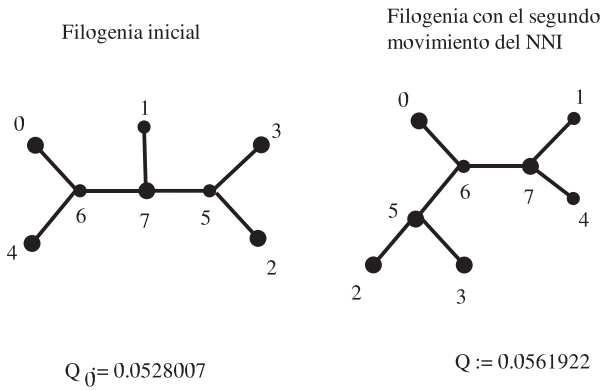


Figura 3.16: Segundo movimiento del NNI `tapir5.txt`

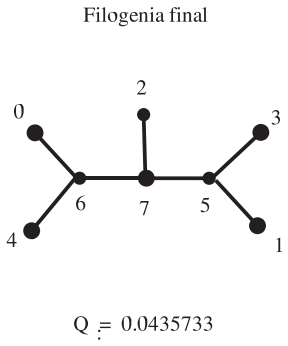


Figura 3.17: Árbol obtenido por el RS `tapir5.txt`

Capítulo 4

RESULTADOS COMPUTACIONALES

En este capítulo se presentan aspectos concernientes a la implementación computacional del modelo. Se describen, en primer lugar, las distintas estructuras de datos empleadas en el programa. Luego se discuten en base a un ejemplo, las funciones que implementan los algoritmos de Neighbor Joining, Recocido Simulado (RS) y Branch-and-Bound (B & B). Finalmente, se detallan resultados de pruebas reales experimentales sobre conjuntos de datos reales obtenidos a través del internet.

Para fines posteriores hemos nombrado al programa de construcción de filogenias con secuencias de ADN como **FILOMAT**.

4.1. DETALLES DE LA IMPLEMENTACIÓN

Para la implementación de cada uno de los algoritmos que emplea el programa se usó el lenguaje de programación C++ (Kdevelop 2.1) sobre el sistema operativo Linux SUSE 9.0 bajo un compilador GNUCC en plataforma Linux de una PC PENTIUM de 2.4 Mhz.

4.1.1. ESTRUCTURA DE DATOS

Cuando la información es presentada como secuencias de ADN, se requiere el número de especies y la longitud de las secuencias. Algunas veces se requiere que la información se presente como matrices de distancias con distintos modelos de evolución, razón por la cual éstas matrices se obtienen con la ayuda de otros programas como el PHYLIP 3.5 (Ver detalles en Felsentein [1993]).

En esta tesis se calcula la matriz de distancias evolutivas con el primer modelo de evolución de Jukes & Cantor [1969]. La función que se encarga de este cálculo es `matriz_distancias()` ; que cuenta el número de diferencias en las secuencias entre cada par de especies y multiplica por su respectiva probabilidad de ocurrencia.

Para procesar la información disponible presentada como secuencias de ADN, se ha empleado la clase `secuencia`, además los datos también se pueden presentar como una matriz de distancias evolutivas, para esto hemos empleado la clase `matriz`.

El árbol filogenético compuesto de nodos internos y externos y longitud de ramas propició la creación de la clase `arbol` fundamental dentro del programa.

No se describen aquí los parámetros de las funciones implementadas que integran clases y en general el programa. Referirse para ello al Apéndice A que contiene fragmentos del código comentado de FILOMAT.

Clase `secuencia`

Esta clase es un arreglo de caracteres (arreglo $n \times l$) y tiene el siguiente aspecto:

```
FILOMAT_secuencia
Inicio
Inicializar;
    asignar;
    insertar_base;
```

La función `asignar` asigna memoria a los objetos de la clase: número de especies n y longitud de las secuencias l.
La función `insertar_base` introduce un caracter (nucleótido A,C,G,T) de la secuencia en una matriz $n \times l$.

Clase `matriz`

También siendo un arreglo $m \times n$, la clase `matriz` presenta la siguente configuración:

```
FILOMAT_matriz
Inicio
    Inicializar;
    asignar;
```

La función `asignar` asigna memoria a los objetos de la clase : número de filas (m) y número de columnas (n).

Clase `arbol`

La clase `arbol` [1], es un vector de nodos de tamaño N , se complementa con dos clases adicionales: la clase `arco` y la clase `nodo`.

[1] Se describe únicamente la estructura general de la clase, las funciones auxiliares de esta clase que utilizan los algoritmos se detallan en la Sección 4.2

La clase `arco` almacena una tripleta donde el primer elemento representa el nodo inicial, el segundo, el nodo final, y el tercero,el índice de la rama, por ejemplo: (1,4,3) indica que la rama 3, tiene como nodo inicial 1 y nodo final 4.

La clase `nodo` es una lista de arcos incidentes al nodo respectivo. Con estas clases complementarias , la clase `arbol` se estructura como:

```
FILOMAT_arbol
Inicio
    Inicializar;
    asignar;
    agregar_arco;
    copiar();
```

La función `asignar` asigna memoria a los objetos de la clase : número de nodos externos $(ne1)$ y número de nodos internos $(ni1)$, donde la dimensión del vector es: $N = ne1 + ni1$.

La función `agregar_arco` introduce un arco al vector de nodos.
La función `copiar()` reproduce una copia de un árbol.

Las tres clases anteriores poseen sus respectivos operadores de lectura y escritura, además de su constructor y destructor para la asignación y liberación de memoria.

4.2. ALGORITMOS

En esta sección se describen en forma detallada la configuración e implementación de los tres algoritmos fundamentales en la metaheurística: Neighbor Joininig, Recocido Simulado(RS) y Branch-and-Bound (B&B).

4.2.1. NEIGHBOR JOINING

Este algoritmo permite la construcción de una primera filogenia, a la cual se la toma como una solución inicial para iniciar el proceso iterativo del Recocido Simulado. La configuración del algoritmo fue la siguiente manera:

```
FILOMAT_Neighbor Joining
topologia();  //Nombre de la función del algoritmo
Inicio
Inicializar;
        divergencias_r();
```

```
        Daproximada();
        pos_min();
        agregar_arco;
actualizacion
mientras haya especies por unir
        actualizarA();
        divergencias_r();
        actualizarmatrizD();
        actualizarpos_min();
        agregar_arco;
termina
```

La función `divergencias_r()` calcula las divergencias (diferencias de las distancias evolutivas de una especie i con respecto a las otras) de cada especie y las almacena en un vector de dimensión n , donde n es el número de especies.

$$r_i = \sum_{j \neq i}^{m} \frac{d_{ij}}{m-2} \qquad i = 1, ..., m$$

m es el número de especies actuales en el grupo.

La función `Daproximada()` calcula las distancias evolutivas aproximadas para cada par del grupo de especies con sus respectivas divergencias y las almacena en la matriz $\mathbb{D}'$.

$$d'_{ij} = d_{ij} - r_i - r_j,$$

La función `pos_min();` busca el mínimo valor de $\mathbb{D}'$, d'_{ij} además proporciona los índices i y j que representan el par de especies con menor distancia, por ejemplo, sea d'_{23} el menor valor de D', entonces la especie 2 con la especie 3 pasan a formar un nuevo nodo.

Auxiliarmente se creó un vector de nodos V' donde se va marcando cada par de especies seleccionadas para evitar nuevamente tomarlas , además se introduce en el vector V' el nuevo nodo formado.

La función `agregar_arco` a partir de la función `pos_min()` introduce el arco (i, j, k) al vector de nodos que representa el árbol, donde i e j son los índices seleccionados por `pos_min()`. Una primera iteración representa la descripción anterior de la funciones, seguidamente se procede a la actualización.

La función `actualizarA()` modifica la matriz de distancias evolutivas inicial $\mathbb{D}$, inhabilita los elementos ya tomados en cuenta (pares (i, j)) , asigna un lugar k para el nuevo nodo resultante de la unión de las especies (i, j) y actualiza los valores de D respectivamente. La dimensión de $\mathbb{D}$ no se altera.

De manera similar trabajan las otras funciones restantes, inhabilitan, asignan y actualizan la información sin modificar sus dimensiones.

Finalmente se obtiene el árbol, vector de nodos , se ha calculado además las longitudes de las ramas respectivas con la función `MMC()`.

La función `MMC()` es el algortimo del Método de los Mínimos Cuadrados, el cual genera un sistema de ecuaciones con las distancias entre pares de nodos y lo resuelve con el método directo de Gauss, implementado en la función `Gauss();`. La soluciones obtenidas se utilizaron para determinar la medida de ajuste Q mediante la función `puntajeQ()`, para detalles de la implementación de estas funciones ver el Apéndice A).

Se ha tomado la instancia `tapir5.txt` para ilustrar la implementación de clases. La Figura 4.1 muestra los datos del archivo `tapir5.txt` como secuencias de ADN y matriz de distancias evolutivas.

```
5 637
caballus..GGATTCCAAGACGCAACATCCCCTATTATAGAAGAAC ...
bairdii...GGATTTCAAGATGCAACATCTCCCATTATAGAAGAAC ...
terrestrisGGATTTCAAGATGCAACATCTCCTATTATAGAAGAAC ...
pinchag...GGATTTCAAGATGCAACATCTCCTATTATAGAAGAAC ...
indicus...GGATTTCAAGATGCAACATCTCCTATTATAGAAGAGC ...
```

		0	1	2	3	4
0	caballus..	0	0.184851	0.202864	0.186832	0.196812
1	bairdii...	0.184851	0	0.0972286	0.0867451	0.0937178
2	terrestris	0.202864	0.0972286	0	0.0251464	0.107861
3	pinchag...	0.186832	0.0867451	0.0251464	0	0.100756
4	indicus...	0.196812	0.0937178	0.107861	0.100756	0

Figura 4.1: Secuencias de ADN y matriz de distancias evolutivas para un grupo de cinco especies

La Figura 4.2 muestra el árbol inicial construido con el algoritmo Neighbor Joining, se puede apreciar también la estructura de árbol que emplea la clase de este mismo nombre. Las longitudes de ramas también se ilustran en dicha figura conjuntamente con la medida de ajuste Qo obtenida con la función `puntajeQ()`.

```
Árbol inicial (Neighbor Joining)

5 especies (nodos externos),        Longitudes de ramas
3 ancestros(nodos internos),          -----------------
7 ramas
0: (0 7 0)                              0 0.14878
1: (1 6 1)                              1 0.0410133
2: (2 5 2)                              2 0.018838
3: (3 5 3)                              3 0.006905
4: (4 7 4)                              4 0.0539857
5: (5 2 2)(5 3 3)(5 6 5)                5 0.0395408
6: (6 1 1)(6 5 5)(6 7 6)                6 0.00017375
7: (7 0 0)(7 4 4)(7 6 6)

                                      Q : 0.0570562
```

Figura 4.2: Estructura del primer árbol obtenido con el Neighbor Joining ,longitudes de ramas y la medida de ajuste Q

Para facilidad de interpretación de datos se implementó la función `grafico();` que representa un árbol, como muestra la Figura 4.3.

Este gráfico es representativo hasta un grupo de 10 especies, para un número mayor a este, únicamente se muestra el árbol en su estructura de grafo.

4.2.2. RECOCIDO SIMULADO

El Recocido Simulado (RS) fue configurado como sigue:

```
FILOMAT_Recocido Simulado
recocido(); //Nombre de la función del algoritmo
Inicio
Inicializar;   Q0=Q (medida de ajuste);
               Qini=Q0; Gmin=G (arbol G);
hacer
     NNI();
     MMC();
     hacer
        si(Q<Q0)
            aceptar G;
```

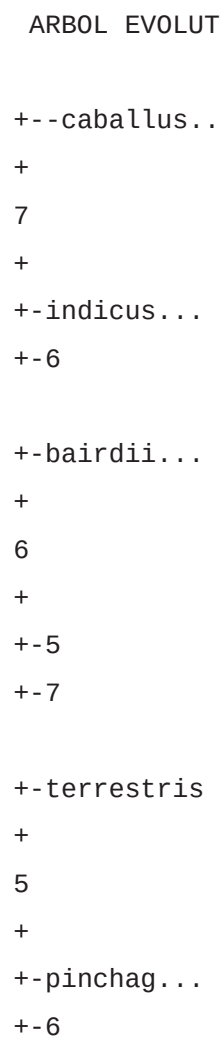


Figura 4.3: Representación gráfica del árbol construido por el Neighbor Joining

```
        si no
            p=(exp-(Q-Q0)/t)
            x=drand48();
            si(p<x<1)
            acepto G;
            si no
            no acepto G;
            si acepto G
                Q0=Q;
            si no acepto G;
                NNI();
MMC();
            si (Q0<Gmin)
                Qmin=Q0;
                Gmin=G;
    hasta iteracion %L=0;
    t=alfa*t;
hasta criterio de parada=N
```

Donde `t` es la temperatura , `L` longitud de piso (iteraciones requeridas) y `N` criterio de parada.

Únicamente dos funciones intervinieron en el Recocido Simulado:

La función `NNI()` que realiza movimientos en las ramas a partir del árbol inicial para generar nuevas topologías. Borra una arista interior sobre el árbol conjuntamente con sus aristas adheridas a ésta, dando lugar a cuatro subárboles desconectados por la eliminación de dicha arista. Esto permite realizar dos conecciones posibles entre estos cuatro subárboles. Se detalla la implementación en el Apéndice A.

La función `MMC()` fue mencionada en el algoritmo anterior.

Se ilustra el resultado de la aplicación RS al archivo `tapir5.txt` en la Figura 4.4.

4.2.3. BRANCH-AND-BOUND (B&B)

La técnica Branch-and-Bound (B&B) constituye un respaldo trascedente para el Recocido Simulado, nos permite sustentar la solución final dada por el RS, en grupos de hasta 12 especies. En esta implementación se usó la estructura `stack S` (pila) para almacenar el conjunto de árboles.

El algoritmo (B&B) se ha configurado como sigue:

```
Árbol más cercano
5 especies (nodos externos),          Longitudes de ramas
3 ancestros(nodos internos),          -----------------
7 ramas
0: (0 6 0)                              0 0.143775
1: (1 5 1)                              1 0.0538832
2: (2 7 2)                              2 0.0375054
3: (3 5 3)                              3 0.0328619
4: (4 6 4)                              4 0.0530371
5: (5 3 3)(5 1 1)(5 7 5)                5 -0.0196904
6: (6 0 0)(6 4 4)(6 7 6)                6 0.0194512
7: (7 2 2)(7 6 6)(7 5 5)


                                      Q : 0.0435733
```

Figura 4.4: Árbol obtenido con el Recocido Simulado y longitudes de ramas

```
FILOMAT_Branch-and-Bound
BAB();//Nombre de la función del algoritmo
Inicio
Inicializar; Qmin=G.Q;
    asignar;
        repetir;
        agregar_arco;  // creación del primer árbol G1
        hasta 3;         con 3 especies
    S.push(G1);     // introduce el primer elemento G1 en la pila
G2.copiar(G1);
MMC(G1);
mientras S no vacia
    hacer
        G1=S.top();  // Asigno a G1 el último elemento de la pila
        S.pop();     // elimina el último elemento de la pila
        MMC(G1);
        si (G1.Q<Qmin)
            insertar_e ;             //inserto una nueva especie
            si aun hay especies        creo un nuevo arbol G2
                MMC(G2);
                S.push(G2);
            si no
```

```
                MMC(G2);
                si(G2.Q<Qmin)
                    Gmin=G2;
                    Qmin=G2.Q;
parar
```

Cabe mencionar que se ha utilizado las librerías STL para trabajar con las colas (`stack`), listas (`list`) y vectores (`vector`) empleando directamente sus funciones.

La función `insertar_`e crea un árbol G1 a partir un árbol G, introduciendo una nueva especie en G1.

Se muestra el resultado en la Figura 4.5.

```
Árbol óptimo
5 especies (nodos externos),          Longitudes de ramas
3 ancestros(nodos internos),          -----------------
7 ramas
0: (0 7 0)                              0 0.143775
1: (1 6 1)                              1 0.0538832
2: (2 5 2)                              2 0.0375054
3: (3 6 3)                              3 0.0328619
4: (4 7 4)                              4 0.0.530371
5: (5 7 6)(5 6 5)(5 2 2)                5 -0.0196904
6: (6 3 3)(6 5 5)(6 1 1 )                6 0.0194512
7: (7 4 4)(7 5 6)(7 0 0)

                                       Q : 0.0435733
```

Figura 4.5: Árbol obtenido con Branch-and-Bound

Su representación gráfica se observa en la Figura 4.6.

Las respectivas comparaciones entre los árboles obtenidos por el RS y por B&B se detallan en la Sección 4.4.

4.3. CALIBRACIÓN DE PARÁMETROS

Los parámetros del Recocido Simualdo fueron calibrados con 5 grupos de especies de diferente tamaño:

- Un grupo pequeño de 5 especies (instancia: `tapir5.txt`)

```
 ARBOL EVOLUTIVO
+-indicus...
+
7
+
+5
+--caballus..

+-pinchag...
+
6
+
+5
+-bairdii...

+-7
+
5
+
+-6
+-terrestris
```

Figura 4.6: Representación gráfica del árbol construido por Branch-and-Bound

- Dos grupos medianos: de 9 especies (instancia:`bacterias9.txt`) y uno de 14 especies (instancia:`tortugas14.txt`)
- Dos grupos grandes: uno de 17 especies (instancia:`humano17.txt`) y uno de 54 especies (instancia: `adh54.txt`).

Estos grupos contienen datos reales que se han tomados del Internet. El nombre del archivo se deriva del conjunto de especies del cual se dispone información y el número indica cuántas especies lo conforman. Por ejemplo el archivo `tortugas14.txt` indica que es un conjunto de 14 especies de tortugas.

Se seleccionaron 3 conjuntos de 6 valores cada uno, para cada uno de los tres parámetros del RS:

- factor de disminución de temperatura α ={0.60,0.70,0.80,0.85,0.90,0.99}
- temperatura inicial t_o ={10,50,100,150,200,400}

- longitud de piso L={10,50,150,300,400,500}

El criterio de elección de los parámetros se enfocó en la elección de las tripletas que cumplían la condición de que la medida de ajuste $Qmin$ sea menor a la medida de ajuste inicial Q_0, luego de esta selección, se tomaron en cuenta los tiempos de corrida menores. La Tabla 4.3 muestra los resultados de calibración paramétrica del cual podemos resumir que los mejores valores para cada parámetro son:

$$\alpha = 0{,}85 \qquad T_0 = 50 \qquad L = 150$$

Instancias	α	to	L	Qo	Qmin	Mej. %	Tiempo
tapir5.txt	0.6	10	10	0.0528007	0.043573	17.4759	0.08
	0.6	50	10	0.0528007	0.043573	17.4759	0.08
	0.6	150	10	0.0528007	0.043573	17.4759	0.08
	0.85	10	150	0.0528007	0.043573	17.4759	0.08
	0.9	10	400	0.0528007	0.043573	17.4759	0.08
bacterias9.txt	0.6	10	400	5.27231	2.17313	58.7822	0.71
	0.7	400	10	5.27231	2.11858	59.8169	0.71
	0.85	10	200	5.27231	2.11858	59.8169	0.71
	0.85	50	50	5.27231	2.17313	58.7822	0.71
	0.85	400	150	5.27231	2.17313	58.7822	0.69
tortugas14.txt	0.7	10	10	0.519209	0.378677	27.0666	4.14
	0.8	150	150	0.519209	0.370502	28.6409	4.22
	0.85	400	10	0.519209	0.370503	28.6409	4.26
	0.9	50	10	0.519209	0.363311	30.026	5.01
	0.99	10	400	0.519209	0.0.363372	30.0144	4.18
humano17.txts	0.6	500	50	11.5522	8.27609	28.3593	8.48
	0.7	10	50	11.5522	8.27609	28.3593	8.82
	0.7	10	200	11.5522	8.27609	28.3593	8.84
	0.85	50	10	11.5522	8.27609	28.3593	9.6
	0.9	10	200	11.5522	8.29811	28.1687	8.25
adh54.txt	0.6	10	10	27.5571	19.0996	30.4095	1713
	0.6	10	50	27.5571	19.1039	30.6752	1632
	0.7	50	50	27.5571	16.4999	36.421	1386
	0.8	10	10	27.5571	18.3393	34.4861	1853
	0.85	10	150	27.5571	17.7358	35.6432	1483

Cuadro 4.1: Calibración de parámetros

Resultados similares se obtuvieron al realizar la calibración de parámetros con información almacenada como matrices de distancias evolutivas.

4.4. ANÁLISIS DE RESULTADOS COMPUTACIONALES

Los datos que se emplearon en este estudio han sido tomados en su mayoría del Internet. Los formatos de entrada y salida de la información son muy similares

a los utilizados por otros programas de reconstrucción filogenética. Tanto los datos de secuencias como los formatos de distancias están almacenados en archivos de texto.

Se han empleado 5 instancias con datos reales: `tapir5.txt`, `chimpan7m.tx`, `dog8.txt`, `bacterias9.txt`, `chicken11.txt`, las otras instancias ha sido construidas con simulación aleatoria para realizar las respectivas comparaciones entre el Recocido Simulado y Branch-and-Bound (B&B). Se ha podido reportar la brecha de optimalidad entre las 7 primeras instancias, ya que a partir de la octava instancia, la carga computacional requerida por Branch-and-Bound es demasiado elevada. El Cuadro 4.2 muestra la brecha de optimalidad de estas estancias, además se adjunto instancias hasta de 60 especies entre reales [2] y simuladas para registrar el tiempo de cálculo de la metaheurística.

No.	Instancia	Qo	Qmin RS	Qmin B&B	T-RS	T-B&B	Brecha de optimalidad
1	tapir5.txt	0.0528007	0.0475997	0.0435733	0.08	0.01	0.092405
2	6.txt	0.31722	0.2008	0.2004	0.14	0.01	0.001999
3	chimpan7m.txt	20.9527	16.6478	16.5356	0.26	0.66	0.006785
4	dog8.txt	209594	197443	189533	0.45	11.34	0.041778
5	bacterias9.txt	5.27231	2.58005	2.17655	0.65	66.36	0.18538
6	10.txt	0.594	0.3030	0.2963	1.31	1853	0.0226
7	chicken11m.txt	0.768494	0.398027	0.346689	1.37	1086	0.125706
8	12.txt	1.420	1.087	-	1.82	-	-
9	13.txt	1.6539	1.2364	-	2.48	-	-
10	tortugas14.txt	0.51209	0.378658	-	3.7	-	-
11	15.txt	2.093	1.480	-	4.34	-	-
12	evol16.txt	2.13166	1.56629	-	5.21	-	
13	humano17.txt	2.4358	1.8572	-	7.34	-	
14	18.txt	3.306	2.429	-	9.51	-	-
15	19.txt	3.7320	2.9291	-	11.39	-	-
16	20.txt	4.4147	3.4748	-	16.43	-	-
17	30.txt	12.202	6.1518	-	96.07	-	-
18	40.txt	17.1421	8.2828	-	427	-	-
19	adh54.txt	27.5571	15.5307	-	1853	-	-
20	60.txt	114.378	925716	-	5020	-	-

Cuadro 4.2: Resultados comparativos entre RS vs. B&B

[2] Las instancias reales son aquellas que muestran el nombre del conjunto de especies, `humano17.txt`, las demás son simuladas `53.txt`

La brecha de optimalidad está entre el 0.1 % al 12 % con una media del 5.62 %.

El tiempo de cálculo es un factor importante en la metaheurística. Se aplicaron distintos tipos de regresión [3] a los tiempos registrados en el Cuadro 4.2 hasta 60 especies. Por motivos de espacio únicamente se muestran aquí las 20 primeras instancias. El Apéndice C, muestra la tabla completa depara las 60 especies. Los tiempos de cálculo (en función del tamaño del problema) se ajustan a una regresión potencial. La Figura 4.5 muestra la curva de regresión obtenida.

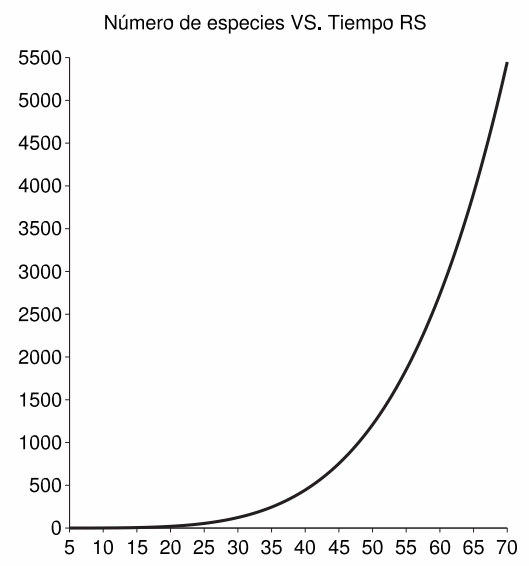


Figura 4.7: Crecimiento potencial del numero de especies con respecto al tiempo RS. La ecuación de regresión es $y = 0{,}00003x^{4{,}4763}$ con ajuste del 0.9922

Se tiene que el tiempo de cálculo estimado para 70 especies es de aproximadamente 90.82 minutos (5449.21 segundos).

El Recocido Simulado permite trabajar con instancias de hasta 70 especies en una cantidad de tiempo aceptable y se espera su buen desemvolvimiento para instancias mayores a 70.

El conjunto de los tiempos de cálculo obtenidos por Branch-and-Bound es limitado, por lo que hacer un análisis similar al del Recocido Simulado requerirá de más observaciones.

[3] Con la ayuda del Excel, se operaron los distintos métodos de regresión

Capítulo 5

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

1. Los árboles que entregan los algoritmos de agrupamiento que están alejados en promedio un 20 % del óptimo constituyen así una buena opción para considerarlos como una solución inicial para los distintos métodos iterativos y de consenso en la reconstrucción de árboles filogenéticos.

2. El empleo de heurísticas para la generación de topologías permiten construir un aceptable conjunto de árboles donde se pueda realizar una buena búsqueda del mejor árbol inclusive para decenas de especies.

3. El método de los mínimos cuadrados MMC combinado con heurísticas de búsqueda local permite una exploración significativa de árboles, al tomar únicamente los árboles que tengan medidas de ajuste pequeñas. El esquema del Recocido Simulado explora el espacio de árboles, mediante movimientos " ascendentes " que empeoran la medida de ajuste, evita que la búsqueda termine en un óptimo local.

4. El algoritmo B&B tiene una aplicabilidad en instancias de hasta 11 especies con un tiempo de cálculo de 1086 segundos sobre los equipos empleados en nuestras pruebas: PC Pentium de 2.4 Mhz.

5. La soluciones obtenidas con la metaheurística del Recocido Simulado se encuentran en promedio un 5.62 % alejadas del óptimo obtenido por Branch and Bound.

5.2. RECOMENDACIONES

1. El Capítulo 1 contiene conceptos y definiciones básicas sobre Sistemática Filogenética, una lectura previa de este capítulo es recomendable para apreciar de una mejor manera el estudio y análisis de las filogenias.

2. Si se requiere profundizar de los temas de carácter biológico, se recomienda una revisión del libro de Joseph Felsenstein " Inferring Phylogenies ",(Felsenstein [2004f]) donde se puede encontrar uan variedad de algoritmos de reconstrucción filogenética y otros temas relacionados con información molecular.

3. Instancias con $n > 70$ pueden ser tratadas con un trabajo computacional en paralelo.

Apéndice A

CÓDIGO DEL PROGRAMA FILOMAT

Se presenta la distribución de ficheros y cabeceras empleados en la metaheurística y el contenido de cada uno de ellos. Cada clase y función tiene su respectiva explicación de su labor dentro del programa. La organización se archivos se muestra a continuación:

```
Ficheros C/C++ :

    Fichero principal:      main.cpp
    Ficheros secundarios:   arbol.cpp
                            funciones.cpp

Cabeceras C/C++ :           funciones.h
                            const.h
```

El fichero `main.cpp` contiene librerias y funciones principales de la metaheurística.

El fichero `arbol.cpp` presenta funciones básicas para la construcción del árbol, además incluye la clase `secuencia` y la clase `matriz`.

El fichero `funciones.cpp` incluye todas las funciones que intervienen en el algoritmo Neighbor Joining, la función `Gauss();` que resuelve el sistema de ecuaciones dado por `MMC();`, funciones auxiliares (lectura, escritura, suma y producto) para operar con matrices. Este fichero contiene la función `matriz_distancias();` que calcula la matriz de distancias evolutivas.

La cabecera `funciones.h` contiene todas las declaraciones de clases y funciones.

La cabecera `const.h` tiene los parámetros del Recocido Simulado, to ,α , L definidos como parámetros globales, además de una variable boolena para tratar los mensajes de salida del programa.

A continuación y en el orden presentado, se muestra el código de FILOMAT.

Fichero principal `main.cpp`

```
/***************************************************************
               Metaheuristica RS:main.cpp-  description
                           -------------------
  begin            : mar ene 3 2006
  copyright        : (C) 2006 by Evelyn Quishpe & Gustavo Recalde
  email            : evelyn_quishpe@yahoo.es , tavitorec@yahoo.com

***************************************************************/

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <list>
#include <vector>
#include <stack>
#include <fstream.h>
#include "funciones.h"
#include "const.h"
#include <time.h>
#include <ctime>

using namespace std;

/*************************************************************************/
void bfs(arbol &G, int *P, int i);
/*************************************************************************
bfs.- Crea un vector P de predecesores que indica el "camino" (nodos
que son visitados) desde un nodo i hacia cualquier otro nodo en el
arbol
 *************************************************************************/
void bfs(arbol &G, int *P, int i)
{
   ...
```

```
}

/*****************************************************************/
void caminos(arbol &G,int *P, double *MTA,double *d);
/*****************************************************************
caminos.-Genera una matriz "Topologica" (o de caminos) MTA[l,k] donde l
         representa el camino de i a j para i=0,...,ne y j=i+1,...,ne y
 MTA[l,k] = 1 si se pasa del nodo i al j por la arista k
         Parametros: Arbol G, Vector de predecesores P, vector de
 distancias d entre cada par de nodos y la Matriz donde se
         representa los caminos entre cada par de nodos por unos y ceros
(MTA)
 *****************************************************************/
void caminos(arbol &G,int *P, double *MTA,double *d)
{
    ...
}

void grafico(secuencia &ADN, arbol&G, matriz &M,double *rama,
 int datos ,ostream& file);
/*****************************************************************
La función gráfico, representa la filogenia de n especies con sus
respectivas longitudes de ramas r.
*****************************************************************/
void grafico(secuencia &ADN,arbol &G,  matriz &M,double *rama,
int datos, ostream& file)

{
...
}

void escribir_MDA(secuencia& ADN, const double *Matriz,
                  int m_filas, int n_columnas, ostream& file);
/*****************************************************************
escribir_MDA.- Esta función permite mostrar la Matriz de distancias
encabezada con la respectiva especie en cada fila.
*****************************************************************/
void escribir_MDA(secuencia& ADN, const double *Matriz, int m_filas,
```

```
  int n_columnas, ostream& file)
{
    ...

}

void escribir_MDA1(matriz& M, int m_filas,ostream& file);
/*********************************************************************
escribir_MDA1.-Similar a la funcion escribir_MDA pero esta unicamente
escribe el numero de identificacion de cada especie.
*********************************************************************/
void escribir_MDA1(matriz& M, int m_filas,ostream& file)
{
    ...
}

/*********************************************************************/
void MMC(secuencia &ADN,arbol &G,matriz& MA,ostream& file,int &datos);
/*********************************************************************
MMC.- Metodo Minimos Cuadrados.
    * Genera el sistema de ecuaciones de las distancias entre nodos
a partir de la matriz topologica y las distancias entre los
nodos externos, y se resuelve por el método de Gauss(Reduce la
matriz Ad a una triangular superior y las variables se obtienen
    con la funcion "variables").
    * Calcula el puntaje Q de la filogenia con la funcion "puntajeQ"
*********************************************************************/
void MMC(secuencia &ADN,arbol &G,matriz& MA,ostream& file,int &datos)
{
      ...
}

/*********************************************************************/
void topologia(secuencia& ADN,arbol &G,double *B,int &datos);
/*********************************************************************
  topologia.- Funcion que genera la topología de la filogenia usando el
              método del vecino más cercano: Neighbor Joining .
  Algoritmo clustering
```

```
 ****************************************************************/
void topologia(secuencia& ADN,arbol &G,double *B,int &datos)
{
      ...
}


/****************************************************************/
void recocido(secuencia &ADN,arbol& G,matriz &MA,int N,int &datos);
/****************************************************************
recocido.- Esta función representa la metaheurística del recocido
   simulado RS
           Es una cadena de algoritmos que toma una filogenia
   inicial con una medida Qini, y mediante probabilidad
   y propiedades propias del RS busca una nueva filogenia
   con menor medida de ajuste Qmin.
****************************************************************/
void recocido(secuencia &ADN,arbol& G,matriz &MA, int N,int &datos)
{

   ...
}
void BAB(secuencia &ADN,arbol &G,matriz& MA, int &datos);
/************************************************************
BAB.- Es un algoritmo exacto basado en la técnica branch and
  bound.
  Busca implicitamente todas las soluciones pero elimina
  selectivamente la gran mayoria(soluciones vagas)incluso
      antes de buscarlas.
************************************************************/
void BAB(secuencia &ADN,arbol &G,matriz& MA, int &datos)
{
    ...
}


/****************************************************************/
void busqueda(secuencia& ADN, arbol& G, matriz &MA,double *M,
```

```
  int &datos);
/*********************************************************************
busqueda.- Esta funcion permite elegir el metodo que se requiera.
           Contiene las funciones recocido() y BAB().
*********************************************************************/
void busqueda(secuencia& ADN, arbol& G, matriz &MA,double *M,
 int &datos)
{
    ...
}

int main (int argc, char * argv[])
{
    arbol G, Gmin;
    secuencia ADN;
    matriz M;
    int datos;
    do
    {
      cout<<endl<<"       Arboles Filogeneticos"<<endl<<endl;
      cout<<" 1. Datos de archivo con matriz de distancias"<<endl;
      cout<<" 2. Datos de archivo con Secuencias de ADN"<<endl;
      cout<<" 3.- SALIR  "<<endl<<endl<<"Digite la opción deseada:";
      cin>>datos;
      if (datos==1)
      {

        ifstream f;
        string s;
        cout << "Nombre del archivo: ";
        cin >> s;
        f.open(s.c_str());
        f>>M;
        ADN.asignar(M.f,0);
        busqueda(ADN,G,M,M.M,datos);
      }
      if (datos==2)
      {
```

```
        ifstream f;
        string s;
        cout << "Nombre del archivo: ";
        cin >> s;
        f.open(s.c_str());
        f>>ADN;
        G.asignar(ADN.n,ADN.n-2);
        busqueda(ADN,G,M,G.MD,datos);
      }
    }
    while(datos!=3);
    return EXIT_SUCCESS;
}
```

Fichero secundario `funciones.cpp`

```
/***************************************************************
                      funciones.cpp  -  description
                           -------------------
 begin            : mar ene 3 2006
 copyright        : (C) 2006 by Evelyn Quishpe & Gustavo Recalde
 email            : evelyn_quishpe@yahoo.es , tavitorec@yahoo.com
 ***************************************************************/

#include <iostream.h>
#include <stdlib.h>
#include <fstream.h>

using namespace std;

/***************************************************************
  sub.- Funcion para la subìndices matriciales, devuelve la
pocisiòn de la lista en la que los valores de la
matriz estan almacenados
                                        0 1 2
                                    0 ( a b c )
        A=(a b c d e f g h i) ->   A= 1 ( d e f )
```

```
                    5               2 ( g h i )

        A[sub(1,3,2)=1*3+2= 5 ->  A[1,2]= f
  Parámetros: Indices de la matriz i,j y numero de columnas n
*****************************************************************/
int sub(int i,int n,int j)
{
    return (i*n+j);
}


/*****************************************************************
 escribir .- Muestra en pantalla una matriz en formato de filas y
 columnas
 Parametros: como parametros al numero de filas y columnas de la
             matriz y a la matriz donde se van almacenar los datos.
*****************************************************************/
void escribir(const double *Matriz, int m_filas, int n_columnas,
ostream& file)
{
   ...
}

/*****************************************************************
leerM.- Almacena una matriz ingresada desde el teclado
        Parametros: el numero de filas y columnas de la matriz y
a la matriz donde se van almacenar los datos.
 *****************************************************************/
void leerM(double *Matriz, int m_filas, int n_columnas)
{
  for (int i=0;i<m_filas;i++)
    for (int j=0;j<n_columnas;j++)
    {
      cout<< "A["<<i<<j<<"]= ";
      cin>>Matriz[sub(i,n_columnas,j)];
    }
}
```

```
/*****************************************************************
producto.- Realiza el producto entre dos matrices.
           Parametros: Las 2 matrices con sus respectivo numero de
   filas y columnas y la Matriz donde se almacenara el
           resultado (C_mq).
 *****************************************************************/
void producto(double *A_mn,double *B_nq, double *C_mq, int m,
int n,int q)
{
  ...
}

/*****************************************************************
transp.- Transpone una matriz
         Parametros: Matriz (X) y su numero de filas y columnas, y
 la matriz donde se va almacenar el resultado (XT)
 *****************************************************************/
void transp(double *X,double *XT, int filas,int columnas)
{
    for (int i=0;i<filas;i++)
      for (int j=0;j<columnas;j++)
        XT[sub(i,columnas,j)]=X[sub(j,filas,i)];
}

/*****************************************************************
unir.- Une un vector a una matriz y forma otra matriz.
       Parametros: Matriz (A) con su numero de filas y columnas,
   vector al que se va unir y a la matriz donde se va almacenar
       el resultado (Ad)
 *****************************************************************/
void unir(double *A,double *d,double *Ad,int m_filas,int n_columnas)
{
    for (int i=0;i<m_filas;i++)
      for (int j=0;j<n_columnas-1;j++)
      {
        Ad[sub(i,n_columnas,j)]=A[sub(i,n_columnas-1,j)];
        Ad[sub(i,n_columnas,n_columnas-1)]= d[i];
      }
```

```
}

/*********************************************************************
  Gauss.- Reduce una matriz a una matriz triangular superior
 *********************************************************************
void Gauss(double *A, int m, int n)
{
   double aux;
   for(int k=0;k<m;k++)
      for(int i=k+1;i<m;i++)
      {
        aux=A[sub(i,n,k)];
        for(int j=k;j<n;j++)
          A[sub(i,n,j)]= A[sub(i,n,j)]-(A[sub(k,n,j)]*aux/
  A[sub(k,n,k)]);
      }
}

/*********************************************************************
  variables.- Resuelve una matriz traingular superior
 *********************************************************************/
void variables (double *M1,double *x, int m, int n)
{
    ...
}

/*********************************************************************
  leer.- Almacena una matriz de caracteres por teclado
 *********************************************************************/
void leer(char *S, int m, int n)
{
    for (int i=0;i<m;i++)
      for (int j=0;j<n;j++)
      {
        cout<< "S["<<i<<j<<"]= ";
        cin>>S[sub(i,n,j)];
      }
}
```

```
/*****************************************************************
escriturachar.- Muestra en pantalla una matriz de caracteres
*****************************************************************/
void escriturachar(char *S, int m, int n)
{
    ...
}

/*****************************************************************
matriz_distancias.-Genera la matriz de distancias contando las
   diferencias entre los caracteres de cada par de
                  filas e la matriz de caracteres
         Parametros: Matriz de caracteres de las secuencias de
 ADN (S), numero de especies (n), longitud de las secuencias
 (l) y la matriz donde se van almacenar las distancias (A)
 *****************************************************************/

void matriz_distancias(int n,int l, char *S,double *MD)
{
    ...
}


/*****************************************************************
  Daproximada.- Calcula una matriz aproximada de distancias de donde
se obtendran el par de nodos que se van a unir.
  Parametros: Matriz de distancias entre cada especie (MD) con el
    numero de columnas(n), vector de divergencias(r) y
        la matriz donde se almacenaran los datos MDA
 *****************************************************************/
void Daproximada(double *MD, double *r, int n, double *MAD)
{
    ...
}

/*****************************************************************
pos_min.- Identifica la posición del valor de la distancia mínima en
```

```
  una matriz de distancias si es que tal pocisión no esta
  marcada.
          Parametros: Matriz de distancias (MD), el vector de
  marcas(M), numero de columnas (n) y los parametros de
    resultado para la pocision
          de fila(a) y columna(b)
 ****************************************************************/
void pos_min(double *MD,int *M,int n,int &a, int &b)
{
    ...
}


/****************************************************************
actualizarA.- Esta función sobreescribe valores sobre una copia de
  la matriz de distancias MD, para aplicar sobre esta
      actualización el algoritmo Neighbor Joininig
 ****************************************************************/
void actualizarA(double *A, int *M, int &a,int &b,int n)
{
    ...
}


/****************************************************************
divergencias_r.- Calcula las divergencias de cada especie
    Parametros: Matriz de distancias MD, numero de columnas(n),
el vector de marcas(M), un parametro para el decremento de
n (ll) y el vector de resultado (r)
 ****************************************************************/
void divergencias_r(double *A, int n, int *M, double *r,int ll)
{
    ...
}

/****************************************************************
actualizarmatrizD.- Esta función sobrescribe valores sobre
la matriz  de distancias aproximadas y trabaja
```

```
  sobre esta.
 ********************************************************************/
void actualizarmatrizD(double *A, double *r, int *M, int n, double *D)
{
   ...
}

/*********************************************************************
actualizarpos_min.- Recalcula la pocisión del minimo de los valores
en la nueva matriz de distancias
             Parametros: Matriz de distancias(MD), dimension(n),
 vecctor de marcas(M) y los parametros para las nuevas
 posiciones.
 ********************************************************************/
void actualizarpos_min(double *D,int *M,int n,int &a, int &b)
{
    ...
}
```

Cabecera `funciones.h`

```
/*********************************************************************
                      funciones.h  -  description
                           -------------------
  begin          : mar ene 3 2006
  copyright      : (C) 2006 by Evelyn Quishpe & Gustavo Recalde
  email          : evelyn_quishpe@yahoo.es , tavitorec@yahoo.com
 ********************************************************************/


#ifndef FUNCIONES_H
#define FUNCIONES_H

#include <list>
#include <vector>
#include <fstream.h>

using namespace std;
```

```
class arco
{
    public:
    int inicio,final,ind;
    arco (int x, int y, int z) {inicio = x; final = y; ind= z;}
};

class nodo
{
    public:
    list<arco> L;
};

class arbol
{
    public:
    int ne,ni,m;            //# de nodos externos, internos y arcos
    vector<nodo> V;         //vector de arcos
    double *MD;
    double Q;               //Matriz de distancias
    arbol();
    arbol(const arbol& G1) {
       MD=NULL;
       copiar(G1);
    }
    ~arbol();
    void asignar(int ne1,int ni1);
    void liberar();
    void agregar_arco(int i, int j, int ind);
    void insertar_distancia(double *D, int i,int j, double d, int ne);
    int  ind_arco(int i, int j);
    int  nes();
    void Varcos(int *Va);
    void insertar_e(arbol &G1, int e, int r);
    void nodos_arco(int ind, int &a, int &b);
    void NNI(int &c, int &ind);
    void optimo(int c, int &p, int &u);
```

```
    void puntajeQ(double *MDA,double *MTA,double *x,int c,ostream& file);
    void copiar(const arbol& G1);//const;
    arbol& operator=(const arbol &G1);
    friend istream& operator >>(istream& is, arbol& G);
    friend ostream& operator <<(ostream& os, const arbol& G);
    int n() {return ne + ni;}
};

class secuencia
{
    public:
    int  n,l;
    char *S;
    secuencia() {S=NULL;}
    ~secuencia();
    void insertar_base(char *S, int i,int j, char s, int l);
    void asignar(int n1,int l1);
    friend istream& operator >>(istream& is, secuencia& ADN);
    friend ostream& operator <<(ostream& os, secuencia& ADN);
};


class matriz
{
    public:
    int  f,c;
    double *M;
    matriz();
    ~matriz();
    string *t;
    void asignar(int f1,int c1);
    friend istream& operator >>(istream& is, matriz& A);
    friend ostream& operator <<(ostream& os, matriz& A);

};

int  sub(int i,int n,int j);
void escribir(const double *Matriz, int m_filas, int n_columnas, ostream& file);
```

```
void leerM(double *Matriz, int m_filas, int n_columnas);
void producto(double *A_mn,double *B_nq, double *C_mq, int m, int n, int q);
void transp(double *X,double *XT, int filas,int columnas);
void unir(double *A,double *d,double *Ad,int m_filas,int n_columnas);
void Gauss(double *A,int m, int n);
void variables(double *M1,double *x, int m, int n);
void jacobi(double *A, double *xj,double *xj1, int m,int n, int &iteracion);
void leer(char *S, int m, int n);
void escriturachar(char *S, int m, int n);
void matriz_distancias(int n,int l, char *S,double *A);
void Daproximada(double *MD, double *r, int n, double *MAD);
void pos_min(double *MD,int *M,int n,int &a, int &b);
void actualizarA(double *A, int *M,int &a,int &b,int n);
void divergencias_r(double *A, int n,int *M, double *r,int ll);
void actualizarmatrizD(double *A, double *r,int *M, int n, double *D);
void actualizarpos_min(double *D,int *M,int n,int &a, int &b);

#endif
```

Cabecera arbol.cpp

```
/***************************************************************
                  arbol.cpp  -  description
                             -------------------
    begin           : mar ene 3 2006
    copyright       : (C) 2006 by Evelyn Quishpe & Gustavo Recalde
    email           : evelyn_quishpe@yahoo.es , tavitorec@yahoo.com
 ***************************************************************/

#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <list>
#include <vector>
#include <fstream.h>
#include "funciones.h"

using namespace std;
```

```
/*****************************************************************
  Constructor
*****************************************************************/
arbol::arbol()
{
    ne=ni=m=0;
    MD=NULL;

}

/*****************************************************************
  Destructor
*****************************************************************/
arbol::~arbol()
{
    liberar();
}

/*****************************************************************
  Libera memoria de la clase arbol
*****************************************************************/
void arbol::liberar()
{

  ne=ni=m=0;
    for(unsigned int i=0; i<V.size(); i++)
            V[i].L.clear();
  delete[] MD;

}

/*****************************************************************
  Asignar: Asigna memoria a la clase.
          Parametros: Numero de nodos externos (ne1) e internos (ni1)
*****************************************************************/
void arbol::asignar(int ne1,int ni1)
{
   ...
```

```
}
/*********************************************************************
agregar_arco.- Agrega un arco al vector de nodos.
        Parametros: Nodos inicial (ini), final (fin) y el indice del
arco (ind)
*********************************************************************/
void arbol::agregar_arco(int ini,int fin, int ind)
{
    arco a(ini,fin,ind);
    V[ini].L.push_back(a);
    m++;
}


/*********************************************************************
Ind_arco.- Identifica el índice o numero de un arco.
        Parametros: Los numeros de los nodos extremos del arco.
*********************************************************************/
int arbol::ind_arco(int i, int j)
{
    ...
}


/*********************************************************************
buscar_arco.- Identifica los nodos a, b de los extremos del arco ind.
         Busca en las listas de adyacencia de los nodos internos.
         Parametros.- Indice o identificador del arco o rama (ind).
 Parametros de para los extremos a(inicial) y b(final)
*********************************************************************/
void arbol::nodos_arco(int ind, int &a, int &b)
{
  ...
}

/*********************************************************************
Realiza una copia del arbol G1 de nombre G2
*********************************************************************/
```

```
void arbol::copiar(const arbol& G1) //const
{
    ...
}

arbol& arbol::operator = (const arbol &G1)//para una asignación
  multiple
{
  copiar(G1);
  return (*this);
}

/*********************************************************************
  Insertar_e.- Se crea un arbol G1 a partir de el arbol G insertando
   una especie o nodo externo en una de las ramas del
   arbol
  Parametros: Arbol que se va a crear (G), especie(e) y rama del
         arbol(r) en la que se va insertar.
*********************************************************************/
void arbol::insertar_e(arbol &G1, int e, int r)
{
    ...
}

/*********************************************************************
nes.- Cuenta y devuelve el numero de nodos externos o especies que
  estan en un arbol que no esta "completo")
*********************************************************************/
int arbol::nes()
{
    ...
}

/*********************************************************************
  Varcos.- Crea un vector con las especies presentes en el arbol
   incompleto
*********************************************************************/
void arbol::Varcos(int *Va)
```

```
{
    ...
}

/*********************************************************************
Insertar_distancia.- Inserta la distancia entre el nodo i e j a la
 matriz de distancias del arbol.
          Parametros: Indices de la matriz (i,ne,j), la distancia
 a ser insertada y la matriz donde se va almacenar
     las distancias (Matriz de distancias MD)
*********************************************************************/
void arbol::insertar_distancia(double *MD, int i,int j, double d, int ne)
{
    MD[sub(i,ne,j)]=d;
    MD[sub(j,ne,i)]=d;
}

/*********************************************************************
  NNI Intercambios del vecino más cercano.- Se borran una rama interior
  sobre el árbol y las dos ramas conectadas a el en cada extremo
  (es decir 5 ramas)
  Los 4 subárboles desconectados pueden ser intercambiados de tres
  formas mas posibles una de ellas la original. Para esto se realiza
  el intercambio en los nodos de la lista de adyacencia de las ramas
  afectadas. Intercambios:
              S    T   c=1   S    U          S    T   c=2   S    T
               \__/    --->   \__/            \__/    --->   \__/
               /  \           /  \            /  \           /  \
              U    V         T    V          U    V         V    U

  Parámetros: Tipo de cambio (c) y el índice de la rama afectada (ind)
*********************************************************************/
void arbol::NNI(int &c, int &ind)
{
  ...
}
```

```
/*********************************************************************
puntajeQ.- Calcula una especie de puntaje global que abarca las
   distancias de todas la ramas del arbol.
           Parametros: Las matrices de distancias aproximadas (MDA)
           y topologica(MTA) y el vector de soluciones del sistema
   de ecuaciones (x)
*********************************************************************/
void arbol::puntajeQ(double *MDA,double *MTA,double *x, int co,
    ostream& file)
{
    ...
}

/*********************************************************************
  Operadores de entrada y salida de datos.
*********************************************************************
/
istream& operator >>(istream& is, arbol& G)
{
   ...
}

ostream& operator <<(ostream& os, const arbol& G)
{
    ...
}


                                //Clase Secuencia
/*************************
         destructor
*************************/

secuencia::~secuencia()
{
  if(S!=NULL)
    delete []S;
}
```

```
/*****************************************************************
insertar_base.- Inserta un caracter (base (s)) de la secuencia para ir
       formando la matriz de los datos de las secuencias
       Parametros: caracter(s), Matriz de Datos de las secuencias(S),
   con sus indices de fila(i) y columna(j) y la longitud de
   secuencia(l)
*****************************************************************/
void secuencia::insertar_base(char *S, int i,int j, char s, int l)
{
  S[sub(i,l,j)]=s;
}


/*****************************************************************
Asigna memoria a los objetos de la clase secuencia.
  Parametros: Numero de secuencias (n1) y longitud de la sescuencias (
  (l1)
*****************************************************************/
/
void secuencia::asignar(int n1,int l1)
{
    ...
}


/*****************************************************************
  Operadores de entrada y salida de datos.
*****************************************************************/
istream& operator >>(istream& is,secuencia& ADN)
{
    ...
}

ostream& operator <<(ostream& os, secuencia & ADN)
{
    os<<endl<<" Numero de especies: "<<ADN.n<<endl;
    os<<" Longitud de las secuencias: "<<ADN.l-10<<endl<<endl;
```

```
    escriturachar(ADN.S,ADN.n,ADN.l);
    return os;
}

                          //Clase Matriz

/*********************************************************************
  Asigna memoria a los objetos de la clase matriz.
  Parametros: Numero de filas (m1) y columnas (n1)
*********************************************************************/
matriz::matriz() //constructor no tiene tipo de retorno
{

f=c=0;
M=NULL;
t=NULL;
}

matriz::~matriz() //constructor no tiene tipo de retorno)
{
delete[] M;
delete[] t;
}
void matriz::asignar(int f1,int c1)
{
...
}

/*********************************************************************
  Operadores de entrada y salida de datos.
*********************************************************************/
istream& operator >>(istream& is, matriz& A)
{
   ...
}

ostream& operator <<(ostream& os, matriz& A)
{
```

```
...
}
```

Cabecera const.h

```
/***************************************************************
                      const.h -  description
                        -------------------
    begin           : mar ene 3 2006
    copyright       : (C) 2006 by Evelyn Quishpe & Gustavo Recalde
    email           : evelyn_quishpe@yahoo.es , tavitorec@yahoo.com
 ***************************************************************/


#ifndef CONST_H
#define CONST_H

const double t0=50;
const double alfa0= 0.85;
const int L = 150;
bool mensajes = true;
bool parada;

#endif
```

Apéndice B

FORMATO NEXUS

Este tipo de formato es el más usado por los programas de análisis filogénetico.

```
#NEXUS
begin data;
dimensions ntax=12 nchar=898;
format datatype=dna interleave=no gap=-;
matrix
Tarsius_syrichta    AAGTTTCATTGGAGCCACCACTCTTATAATTGCCCATGGCCTCACCTCCTC
CCTATTATTTTGCCTAGCAAATACAAACTACGAACGAGTCCACAGTCGAACAATAGCACTAGCCCGTGGCC
TTCAAACCCTATTACCTCTTGCAGCAACATGATGACTCCTCGCCAGCTTAACCAACCTGGCCCTTCCCCCA
ACAATTAATTTAATCGGTGAACTGTCCGTAATAATAGCAGCATTTTCATGGTCACACCTAACTATTATCTT
AGTAGGCCTTAACACCCTTATCACCGCCCTATATTCCCTATATATACTAATCATAACTCAACGAGGAAAAT
ACACATATCATATCAACAATATCATGCCCCCTTTCACCCGAGAAAATACATTAATAATCATACACCTATTT
CCCTTAATCCTACTATCTACCAACCCCAAAGTAATTATAGGAACCATGTACTGTAAATATAGTTTAAACAA
AACATTAGATTGTGAGTCTAATAATAGAAGCCCAAAGATTTCTTATTTACCAAGAAAGTA-TGCAAGAACT
GCTAACTCATGCCTCCATATATAACAATGTGGCTTTCTT-ACTTTTAAAGGATAGAAGTAATCCATCGGTC
TTAGGAACCGAAAA-ATTGGTGCAACTCCAAATAAAAGTAATAAATTTATTTTCATCCTCCATTTTACTAT
CACTTACACTCTTAATTACCCCATTTATTATTACAACAACTAAAAAATATGAAACACATGCATACCCTTAC
TACGTAAAAAACTCTATCGCCTGCGCATTTATAACAAGCCTAGTCCCAATGCTCATATTTCTATACACAAA
TCAAGAAATAATCATTTCCAACTGACATTGAATAACGATTCATACTATCAAATTATGCCTAAGCTT

Lemur_catta    AAGCTTCATAGGAGCAACCATTCTAATAATCGCACATGGCCTTACATCATCCATAT
TATTCTGTCTAGCCAACTCTAACTACGAACGAATCCATAGCCGTACAATACTACTAGCACGAGGGATCCAA
ACCATTCTCCCTCTTATAGCCACCTGATGACTACTCGCCAGCCTAACTAACCTAGCCCTACCCACCTCTAT
CAATTTAATTGGCGAACTATTCGTCACTATAGCATCCTTCTCATGATCAAACATTACAATTATCTTAATAG
GCTTAAATATGCTCATCACCGCTCTCTATTCCCTCTATATATTAACTACTACACAACGAGGAAAACTCACA
TATCATTCGCACAACCTAAACCCATCCTTTACACGAGAAAACACCCTTATATCCATACACATACTCCCCCT
TCTCCTATTTACCTTAAACCCCAAAATTATTCTAGGACCCACGTACTGTAAATATAGTTTAAA-AAAACAC
TAGATTGTGAATCCAGAAATAGAAGCTCAAAC-CTTCTTATTTACCGAGAAAGTAATGTATGAACTGCTAA
CTCTGCACTCCGTATATAAAAATACGGCTATCTCAACTTTTAAAGGATAGAAGTAATCCATTGGCCTTAGG
AGCCAAAAA-ATTGGTGCAACTCCAAATAAAAGTAATAAATCTATTATCCTCTTTCACCCTTGTCACACTG
ATTATCCTAACTTTACCTATCATTATAAACGTTACAAACATATACAAAAACTACCCCTATGCACCATACGT
AAAATCTTCTATTGCATGTGCCTTCATCACTAGCCTCATCCCAACTATATTATTTATCTCCTCAGGACAAG
AAACAATCATTTCCAACTGACATTGAATAACAATCCAAACCCTAAAACTATCTATTAGCTT
```

Homo_sapiens AAGCTTCACCGGCGCAGTCATTCTCATAATCGCCCACGGGCTTACATCCTC
ATTACTATTCTGCCTAGCAAACTCAAACTACGAACGCACTCACAGTCGCATCATAATCCTCTCTCAAGGAC
TTCAAACTCTACTCCCACTAATAGCTTTTTGATGACTTCTAGCAAGCCTCGCTAACCTCGCCTTACCCCCC
ACTATTAACCTACTGGGAGAACTCTCTGTGCTAGTAACCACGTTCTCCTGATCAAATATCACTCTCCTACT
TACAGGACTCAACATACTAGTCACAGCCCTATACTCCCTCTACATATTTACCACAACACAATGGGGCTCAC
TCACCCACCACATTAACAACATAAAACCCTCATTCACACGAGAAAACACCCTCATGTTCATACACCTATCC
CCCATTCTCCTCCTATCCCTCAACCCCGACATCATTACCGGGTTTTCCTCTTGTAAATATAGTTTAACCAA
AACATCAGATTGTGAATCTGACAACAGAGGCTTA-CGACCCCTTATTTACCGAGAAAGCT-CACAAGAACT
GCTAACTCATGCCCCCATGTCTAACAACATGGCTTTCTCAACTTTTAAAGGATAACAGCTATCCATTGGTC
TTAGGCCCCAAAAATTTTGGTGCAACTCCAAATAAAAGTAATAACCATGCACACTACTATAACCACCCTAA
CCCTGACTTCCCTAATTCCCCCCATCCTTACCACCCTCGTTAACCCTAACAAAAAAAACTCATACCCCCAT
TATGTAAAATCCATTGTCGCATCCACCTTTATTATCAGTCTCTTCCCCACAACAATATTCATGTGCCTAGA
CCAAGAAGTTATTATCTCGAACTGACACTGAGCCACAACCCAAACAACCCAGCTCTCCCTAAGCTT

Pan AAGCTTCACCGGCGCAATTATCCTCATAATCGCCCACGGACTTACATCCTCATTATTAT
TCTGCCTAGCAAACTCAAATTATGAACGCACCCACAGTCGCATCATAATTCTCTCCCAAGGACTTCAAACT
CTACTCCCACTAATAGCCTTTTGATGACTCCTAGCAAGCCTCGCTAACCTCGCCCTACCCCCTACCATTAA
TCTCCTAGGGGAACTCTCCGTGCTAGTAACCTCATTCTCCTGATCAAATACCACTCTCCTACTCACAGGAT
TCAACATACTAATCACAGCCCTGTACTCCCTCTACATGTTTACCACAACACAATGAGGCTCACTCACCCAC
CACATTAATAACATAAAGCCCTCATTCACACGAGAAAATACTCTCATATTTTTACACCTATCCCCCATCCT
CCTTCTATCCCTCAATCCTGATATCATCACTGGATTCACCTCCTGTAAATATAGTTTAACCAAAACATCAG
ATTGTGAATCTGACAACAGAGGCTCA-CGACCCCTTATTTACCGAGAAAGCT-TATAAGAACTGCTAATTC
ATATCCCCATGCCTGACAACATGGCTTTCTCAACTTTTAAAGGATAACAGCCATCCGTTGGTCTTAGGCCC
CAAAAATTTTGGTGCAACTCCAAATAAAAGTAATAACCATGTATACTACCATAACCACCTTAACCCTAACT
CCCTTAATTCTCCCCATCCTCACCACCCTCATTAACCCTAACAAAAAAAACTCATATCCCCATTATGTGAA
ATCCATTATCGCGTCCACCTTTATCATTAGCCTTTTCCCCACAACAATATTCATATGCCTAGACCAAGAAG
CTATTATCTCAAACTGGCACTGAGCAACAACCCAAACAACCCAGCTCTCCCTAAGCTT

Gorilla AAGCTTCACCGGCGCAGTTGTTCTTATAATTGCCCACGGACTTACATCATCATTA
TTATTCTGCCTAGCAAACTCAAACTACGAACGAACCCACAGCCGCATCATAATTCTCTCTCAAGGACTCCA
AACCCTACTCCCACTAATAGCCCTTTGATGACTTCTGGCAAGCCTCGCCAACCTCGCCTTACCCCCCACCA
TTAACCTACTAGGAGAGCTCTCCGTACTAGTAACCACATTCTCCTGATCAAACACCACCCTTTTACTTACA
GGATCTAACATACTAATTACAGCCCTGTACTCCCTTTATATATTTACCACAACACAATGAGGCCCACTCAC
ACACCACATCACCAACATAAAACCCTCATTTACACGAGAAAACATCCTCATATTCATGCACCTATCCCCCA
TCCTCCTCCTATCCCTCAACCCCGATATTATCACCGGGTTCACCTCCTGTAAATATAGTTTAACCAAAACA
TCAGATTGTGAATCTGATAACAGAGGCTCA-CAACCCCTTATTTACCGAGAAAGCT-CGTAAGAGCTGCTA
ACTCATACCCCCGTGCTTGACAACATGGCTTTCTCAACTTTTAAAGGATAACAGCTATCCATTGGTCTTAG

GACCCAAAAATTTTGGTGCAACTCCAAATAAAAGTAATAACTATGTACGCTACCATAACCACCTTAGCCCT
AACTTCCTTAATTCCCCCTATCCTTACCACCTTCATCAATCCTAACAAAAAAAGCTCATACCCCCATTACG
TAAAATCTATCGTCGCATCCACCTTTATCATCAGCCTCTTCCCCACAACAATATTTCTATGCCTAGACCAA
GAAGCTATTATCTCAAGCTGACACTGAGCAACAACCCAAACAATTCAACTCTCCCTAAGCTT

Pongo AAGCTTCACCGGCGCAACCACCCTCATGATTGCCCATGGACTCACATCCTCCCTA
CTGTTCTGCCTAGCAAACTCAAACTACGAACGAACCCACAGCCGCATCATAATCCTCTCTCAAGGCCTTCA
AACTCTACTCCCCCTAATAGCCCTCTGATGACTTCTAGCAAGCCTCACTAACCTTGCCCTACCACCCACCA
TCAACCTTCTAGGAGAACTCTCCGTACTAATAGCCATATTCTCTTGATCTAACATCACCATCCTACTAACA
GGACTCAACATACTAATCACAACCCTATACTCTCTCTATATATTCACCACAACACAACGAGGTACACCCAC
ACACCACATCAACAACATAAAACCTTCTTTCACACGCGAAAATACCCTCATGCTCATACACCTATCCCCCA
TCCTCCTCTTATCCCTCAACCCCAGCATCATCGCTGGGTTCGCCTACTGTAAATATAGTTTAACCAAAACA
TTAGATTGTGAATCTAATAATAGGGCCCCA-CAACCCCTTATTTACCGAGAAAGCT-CACAAGAACTGCTA
ACTCTCACT-CCATGTGTGACAACATGGCTTTCTCAGCTTTTAAAGGATAACAGCTATCCCTTGGTCTTAG
GATCCAAAAATTTTGGTGCAACTCCAAATAAAAGTAACAGCCATGTTTACCACCATAACTGCCCTCACCTT
AACTTCCCTAATCCCCCCCATTACCGCTACCCTCATTAACCCCAACAAAAAAAACCCATACCCCCACTATG
TAAAAACGGCCATCGCATCCGCCTTTACTATCAGCCTTATCCCAACAACAATATTTATCTGCCTAGGACAA
GAAACCATCGTCACAAACTGATGCTGAACAACCACCCAGACACTACAACTCTCACTAAGCTT

Hylobates AAGCTTTACAGGTGCAACCGTCCTCATAATCGCCCACGGACTAACCTCTTCCCTG
CTATTCTGCCTTGCAAACTCAAACTACGAACGAACTCACAGCCGCATCATAATCCTATCTCGAGGGCTCCA
AGCCTTACTCCCACTGATAGCCTTCTGATGACTCGCAGCAAGCCTCGCTAACCTCGCCCTACCCCCCACTA
TTAACCTCCTAGGTGAACTCTTCGTACTAATGGCCTCCTTCTCCTGGGCAAACACTACTATTACACTCACC
GGGCTCAACGTACTAATCACGGCCCTATACTCCCTTTACATATTTATCATAACACAACGAGGCACACTTAC
ACACCACATTAAAAACATAAAACCCTCACTCACACGAGAAAACATATTAATACTTATGCACCTCTTCCCCC
TCCTCCTCCTAACCCTCAACCCTAACATCATTACTGGCTTTACTCCCTGTAAACATAGTTTAATCAAAACA
TTAGATTGTGAATCTAACAATAGAGGCTCG-AAACCTCTTGCTTACCGAGAAAGCC-CACAAGAACTGCTA
ACTCACTATCCCATGTATGACAACATGGCTTTCTCAACTTTTAAAGGATAACAGCTATCCATTGGTCTTAG
GACCCAAAAATTTTGGTGCAACTCCAAATAAAAGTAATAGCAATGTACACCACCATAGCCATTCTAACGCT
AACCTCCCTAATTCCCCCCATTACAGCCACCCTTATTAACCCCAATAAAAAGAACTTATACCCGCACTACG
TAAAAATGACCATTGCCTCTACCTTTATAATCAGCCTATTTCCCACAATAATATTCATGTGCACAGACCAA
GAAACCATTATTTCAAACTGACACTGAACTGCAACCCAAACGCTAGAACTCTCCCTAAGCTT

Macaca_fuscata AAGCTTTTCCGGCGCAACCATCCTTATGATCGCTCACGGACTCACCTCTTC
CATATATTTCTGCCTAGCCAATTCAAACTATGAACGCACTCACAACCGTACCATACTACTGTCCCGAGGAC
TTCAAATCCTACTTCCACTAACAGCCTTTTGATGATTAACAGCAAGCCTTACTAACCTTGCCCTACCCCCC
ACTATCAATCTACTAGGTGAACTCTTTGTAATCGCAACCTCATTCTCCTGATCCCATATCACCATTATGCT
AACAGGACTTAACATATTAATTACGGCCCTCTACTCTCTCCACATATTCACTACAACACAACGAGGAACAC

TCACACATCACATAATCAACATAAAGCCCCCCTTCACACGAGAAAACACATTAATATTCATACACCTCGCT
CCAATTATCCTTCTATCCCTCAACCCCAACATCATCCTGGGGTTTACCTCCTGTAGATATAGTTTAACTAA
AACACTAGATTGTGAATCTAACCATAGAGACTCA-CCACCTCTTATTTACCGAGAAAACT-CGCAAGGACT
GCTAACCCATGTACCCGTACCTAAAATTACGGTTTTCTCAACTTTTAAAGGATAACAGCTATCCATTGACC
TTAGGAGTCAAAAACATTGGTGCAACTCCAAATAAAAGTAATAATCATGCACACCCCCATCATTATAACAA
CCCTTATCTCCCTAACTCTCCCAATTTTTGCCACCCTCATCAACCCTTACAAAAAACGTCCATACCCAGAT
TACGTAAAAACAACCGTAATATATGCTTTCATCATCAGCCTCCCCTCAACAACTTTATTCATCTTCTCAAA
CCAAGAAACAACCATTTGGAGCTGACATTGAATAATGACCCAAACACTAGACCTAACGCTAAGCTT

M_mulatta AAGCTTTTCTGGCGCAACCATCCTCATGATTGCTCACGGACTCACCTCTTCCATA
TATTTCTGCCTAGCCAATTCAAACTATGAACGCACTCACAACCGTACCATACTACTGTCCCGGGGACTTCA
AATCCTACTTCCACTAACAGCTTTCTGATGATTAACAGCAAGCCTTACTAACCTTGCCCTACCCCCCACTA
TCAACCTACTAGGTGAACTCTTTGTAATCGCGACCTCATTCTCCTGGTCCCATATCACCATTATATTAACA
GGATTTAACATACTAATTACGGCCCTCTACTCCCTCCACATATTCACCACAACACAACGAGGAGCACTCAC
ACATCACATAATCAACATAAAACCCCCCTTCACACGAGAAAACATATTAATATTCATACACCTCGCTCCAA
TCATCCTCCTATCTCTCAACCCCAACATCATCCTGGGGTTTACTTCCTGTAGATATAGTTTAACTAAAACA
TTAGATTGTGAATCTAACCATAGAGACTTA-CCACCTCTTATTTACCGAGAAAACT-CGCGAGGACTGCTA
ACCCATGTATCCGTACCTAAAATTACGGTTTTCTCAACTTTTAAAGGATAACAGCTATCCATTGACCTTAG
GAGTCAAAAATATTGGTGCAACTCCAAATAAAAGTAATAATCATGCACACCCCTATCATAATAACAACCCT
TATCTCCCTAACTCTCCCAATTTTTGCCACCCTCATCAACCCTTACAAAAAACGTCCATACCCAGATTACG
TAAAAACAACCGTAATATATGCTTTCATCATCAGCCTCCCCTCAACAACTTTATTCATCTTCTCAAACCAA
GAAACAACCATTTGAAGCTGACATTGAATAATAACCCAAACACTAGACCTAACACTAAGCTT

M_fascicularis AAGCTTCTCCGGCGCAACCACCCTTATAATCGCCCACGGGCTCACCTCTTC
CATGTATTTCTGCTTGGCCAATTCAAACTATGAGCGCACTCATAACCGTACCATACTACTATCCCGAGGAC
TTCAAATTCTACTTCCATTGACAGCCTTCTGATGACTCACAGCAAGCCTTACTAACCTTGCCCTACCCCCC
ACTATTAATCTACTAGGCGAACTCTTTGTAATCACAACTTCATTTTCCTGATCCCATATCACCATTGTGTT
AACGGGCCTTAATATACTAATCACAGCCCTCTACTCTCTCCACATGTTCATTACAGTACAACGAGGAACAC
TCACACACCACATAATCAATATAAAACCCCCCTTCACACGAGAAAACATATTAATATTCATACACCTCGCT
CCAATTATCCTTCTATCTCTCAACCCCAACATCATCCTGGGGTTTACCTCCTGTAAATATAGTTTAACTAA
AACATTAGATTGTGAATCTAACTATAGAGGCCTA-CCACTTCTTATTTACCGAGAAAACT-CGCAAGGACT
GCTAATCCATGCCTCCGTACTTAAAACTACGGTTTCCTCAACTTTTAAAGGATAACAGCTATCCATTGACC
TTAGGAGTCAAAAACATTGGTGCAACTCCAAATAAAAGTAATAATCATGCACACCCCCATCATAATAACAA
CCCTCATCTCCCTGACCCTTCCAATTTTTGCCACCCTCACCAACCCCTATAAAAAACGTTCATACCCAGAC
TACGTAAAAACAACCGTAATATATGCTTTTATTACCAGTCTCCCCTCAACAACCCTATTCATCCTCTCAAA
CCAAGAAACAACCATTTGGAGTTGACATTGAATAACAACCCAAACATTAGACCTAACACTAAGCTT

M_sylvanus AAGCTTCTCCGGTGCAACTATCCTTATAGTTGCCCATGGACTCACCTCTTCCATA
TACTTCTGCTTGGCCAACTCAAACTACGAACGCACCCACAGCCGCATCATACTACTATCCCGAGGACTCCA
AATCCTACTCCCACTAACAGCCTTCTGATGATTCACAGCAAGCCTTACTAATCTTGCTCTACCCTCCACTA
TTAATCTACTGGGCGAACTCTTCGTAATCGCAACCTCATTTTCCTGATCCCACATCACCATCATACTAACA
GGACTGAACATACTAATTACAGCCCTCTACTCTCTTCACATATTCACCACAACACAACGAGGAGCGCTCAC
ACACCACATAATTAACATAAAACCACCTTTCACACGAGAAAACATATTAATACTCATACACCTCGCTCCAA
TTATTCTTCTATCTCTTAACCCCAACATCATTCTAGGATTTACTTCCTGTAAATATAGTTTAATTAAAACA
TTAGACTGTGAATCTAACTATAGAAGCTTA-CCACTTCTTATTTACCGAGAAAACT-TGCAAGGACCGCTA
ATCCACACCTCCGTACTTAAAACTACGGTTTTCTCAACTTTTAAAGGATAACAGCTATCCATTGGCCTTAG
GAGTCAAAAATATTGGTGCAACTCCAAATAAAAGTAATAATCATGTATACCCCCATCATAATAACAACTCT
CATCTCCCTAACTCTTCCAATTTTCGCTACCCTTATCAACCCCAACAAAAAACACCTATATCCAAACTACG
TAAAAACAGCCGTAATATATGCTTTCATTACCAGCCTCTCTTCAACAACTTTATATATATTCTTAAACCAA
GAAACAATCATCTGAAGCTGGCACTGAATAATAACCCAAACACTAAGCCTAACATTAAGCTT

Saimiri_sciureus AAGCTTCACCGGCGCAATGATCCTAATAATCGCTCACGGGTTTACTTCGTC
TATGCTATTCTGCCTAGCAAACTCAAATTACGAACGAATTCACAGCCGAACAATAACATTTACTCGAGGGC
TCCAAACACTATTCCCGCTTATAGGCCTCTGATGACTCCTAGCAAATCTCGCTAACCTCGCCCTACCCACA
GCTATTAATCTAGTAGGAGAATTACTCACAATCGTATCTTCCTTCTCTTGATCCAACTTTACTATTATATT
CACAGGACTTAATATACTAATTACAGCACTCTACTCACTTCATATGTATGCCTCTACACAGCGAGGTCCAC
TTACATACAGCACCAGCAATATAAAACCAATATTTACACGAGAAAATACGCTAATATTTATACATATAACA
CCAATCCTCCTCCTTACCTTGAGCCCCAAGGTAATTATAGGACCCTCACCTTGTAATTATAGTTTAGCTAA
AACATTAGATTGTGAATCTAATAATAGAAGAATA-TAACTTCTTAATTACCGAGAAAGTG-CGCAAGAACT
GCTAATTCATGCTCCCAAGACTAACAACTTGGCTTCCTCAACTTTTAAAGGATAGTAGTTATCCATTGGTC
TTAGGAGCCAAAAACATTGGTGCAACTCCAAATAAAAGTAATA---ATACACTTCTCCATCACTCTAATAA
CACTAATTAGCCTACTAGCGCCAATCCTAGCTACCCTCATTAACCCTAACAAAAGCACACTATACCCGTAC
TACGTAAAACTAGCCATCATCTACGCCCTCATTACCAGTACCTTATCTATAATATTCTTTATCCTTACAGG
CCAAGAATCAATAATTTCAAACTGACACTGAATAACTATCCAAACCATCAAACTATCCCTAAGCTT
;
end;

Apéndice C

TABLA DE RESULTADOS

No.	Instancia	Qo	Qmin RS	Qmin B&B	T-RS	T-B&B	Brecha de optimalidad
1	tapir5.txt	0.0528007	0.0475997	0.0435733	0.08	0.01	0.092405
2	6.txt	0.31722	0.2008	0.2004	0.14	0.01	0.001999
3	chimpan7m.txt	20.9527	16.6478	16.5356	0.26	0.66	0.006785
4	dog8.txt	209594	197443	189533	0.45	11.34	0.041778
5	bacterias9.txt	5.27231	2.58005	2.17655	0.65	66.36	0.18538
6	10.txt	0.594	0.3030	0.2963	1.31	1853	0.0226
7	chicken11m.txt	0.768494	0.398027	0.346689	1.37	1086	0.125706
8	12.txt	1.420	1.087	-	1.82	-	-
9	13.txt	1.6539	1.2364	-	2.48	-	-
10	tortugas14.txt	0.51209	0.378658	-	3.7	-	-
11	15.txt	2.093	1.480	-	4.34	-	-
12	evol16.txt	2.13166	1.56629	-	5.21	-	
13	humano17.txt	2.4358	1.8572	-	7.34	-	
14	18.txt	3.306	2.429	-	9.51	-	-
15	19.txt	3.7320	2.9291	-	11.39	-	-
16	20.txt	4.4147	3.4748	-	16.43	-	-
17	23.txt	5.977	3.901	-	26.6	-	-
18	25.txt	7.8147	5.0600	-	41.01	-	-
19	27.txt	7.7219	4.4598	-	53.46	-	-
20	30.txt	12.20	6.1518	-	96.07	-	-
21	33.txt	8.7514	6.1205	-	171.57	-	-
21	35.txt	14.310	6.4537	-	254.52	-	-
22	37.txt	12.282	8.68108	-	283.65	-	-
23	40.txt	17.1421	8.2828	-	427	-	-
24	43.txt	15.643	10.5587	-	635	-	-
25	45.txt	20.0453	10.058	-	924	-	-
26	47.txt	20.064	13.9243	-	1278	-	-
27	50.txt	25.0384	13.046	-	1327	-	-

No.	Instancia	Qo	Qmin RS	Qmin B&B	T-RS	T-B&B	Brecha de optimalidad
28	53.txt	22.9656	17.2552	-	1468	-	-
29	adh54.txt	27.5571	15.5307	-	1853	-	-
30	56.txt	104.363	92.2541	-	2341	-	-
31	58.txt.txt	104.313	89.94	-	4077	-	-
32	60.txt	114.378	92.5712	-	5020	-	-

Cuadro C.1: Resultados comparativos entre RS vs. B&B y tiempos de cálculo para instancias de hasta 60 especies para el Recocido Simulado

BIBLIOGRAFÍA

Abascal (2006). Alineamiento de secuencias. Universidad Autónoma de Madrid.

Allen & Steel (2001). Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics 1(5), 1–15*.

Ax (1987). The phylogenetic System. *Chichester 1, 57–58*. John Wiley and Sons.

Beyer, Stein, Smith & Ulam (1974). A molecular sequence metric and evolutionary trees. *Mathematical Biosciences (19), 9–25*.

Burden & Faires (1985). *Análisis Numérico* (First edition ed.). ISBN 968-7270-09-8. Wadsworth. Inc. Belmont California 94002. E.E.U.A.: Grupo Editorial Iberoamérica.

Cavalli-Sforza & Edwards (1967). Phylogenetic analysis: Models and estimation procedures. *American Journal of Human Genetics (19), 233–257*. Evolution 21: 550-570.

Cerny (1985). A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applic (45), 41–55*.

Cook, Cunningan, W. & Schrijver (1998). *Combinatorial Optimization* (First edition ed.). London: John Murray, Albemarle Street: Wiley Interscience. On-line Facsimile Edition:Electronic Scholarly Publishing.

Darwin (1859). *Of the Origin of the Species by means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* (First edition ed.). London: John Murray, Albemarle Street: M.A. Fellow, Royal,Geological, and Linnean Societies. On-line Facsimile Edition:Electronic Scholarly Publishing.

Day (1983). Computationally difficult parsimony problems in Phylogenetic systematics. *Journal of Theoretical Biology (103), 429–438*.

Day, Johnson & Sanko (1986). The computational complexity of inferring rooted phylogenies by parsimony. *Math. Biosci (81), 33–42*.

Dowsland & Díaz (2001). Diseño de Heurísticas y Fundamentos del Recocido Simulado. *Revista Iberoamericana de Inteligencia Artificial (20), 34–52*.

Felsenstein (2004a). Inferring Phylogenies, chapter 29, pp. 496–520. ISBN 0-87893-177-5. Sinauer Assocaites.

—— (2004b). Inferring Phylogenies, chapter 13, pp. 196–198. ISBN 0-87893-177-5. Sinauer Assocaites.

—— (2004c). Inferring Phylogenies, chapter 1, pp. 1–10. ISBN 0-87893-177-5. Sinauer Assocaites.

—— (2004d). Inferring Phylogenies, chapter 16, pp. 248–288. ISBN 0-87893-177-5. Sinauer Assocaites.

—— (2004e). Inferring Phylogenies, chapter 18, pp. 288–306. ISBN 0-87893-177-5. Sinauer Assocaites.

—— (2004f). *Inferring Phylogenies* (First Edition ed.). ISBN 0-87893-177-5. Sinauer Assocaites,Inc.,PO Box 407,23 Plumtree Road, Sunderland, MA, 01375, U.S.A: Sinauer Associates. Internet: publisher@sinauer.com;http://www.sinauer.com.

Felsentein (1993). PHYLIP 3.5 (Software para construcción de filogenias). 1980 programado en Pascal. DisponIble en http://evolution.gs.whasinton.edu/phylip.html.

Fitch (1971). Toward defining the course of evolution : Minimun change for a especified tree topology. *Systematic Zoology (20), 406–416*.

Fitch & Margoliash (1967). Construction of phylogenetic trees. *Science (155), 279–284*.

Foulds & Graham (1982). The Steiner problem in phylogeny is NP-complete. *Adv. Appl. Math. (3), 43–49*.

Gascuel (2004a). Mathematics of Evolution and Phylogeny, chapter 2, pp. 33–62. ISBN 0 19 856610 7. OXFORD University Press.

—— (2004b). Mathematics of Evolution and Phylogeny, chapter 3, pp. 63–91. ISBN 0 19 856610 7. OXFORD University Press.

Golobof (1999). Analyzing large data sets in reasonable times: Solutions for composite optima. *Cladistics (15), 415–428*.

Gómez (2005). Alineamiento de secuencias. Centro de Biología Molecular de Severo Ochoa.

Hendy & Penny (1982). Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences (60), 133–142*.

Hennig (1966). Phylogenetic Systematics. *University of Illinois 1*. Press, Urbana.

Hull (1973). Darwin and His Critics. *Harvard Un. Press, 67*.

Jukes & Cantor (1969). Evolution of protein molecules. *Mammalian Protein Metabolism III, 31–231*.

Kimura (1980). A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution 16, 111–120*.

Kirkpatrick, Gallat & Vecchi (1983). Optimization by simulated annealing. *Science (220), 671–680*.

Lanave, Preparata, Saccone & Serio (1984). A new method for calculating evolutionary substitution rates. *Journal of Molecular Evolution (20), 86–93*.

Matsuda (1996). Protein Phylogenetic inference using maximun likelihood with a genetic algorithm. *Pacific Symposium on Biocomputing 1(1), 512–523*.

Metropolis, Rosenbluth & Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics (21), 1087–1092*.

Saitou & Nei (1987). The neighbor-joining meted: A new meted for reconstructing Phylogenetic trees. *Molecular Biology and Evolution (4), 406–425*.

Schröder (1870). Vier combinatorische Probleme. *Zeitschrift für Mathematik und Physik (15), 361–376*.

Sokal & Michener (1958). A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin 38, 1409–1438*.

Steipe (1995). Phylogenetic Analysis. *1 1(1), 1*. También disponible desde Internet en: http://www.lmb.uni-muenchen.de/groups/bioinformatics/ch10/chap-10.html.

Swofford & Olsen (1990). Phylogeny reconstruction, chapter 11, pp. 411–501. Massachusetts: Sinaeur Associates. Sunderland.

Tamura & Nei (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution (10), 512–526*.