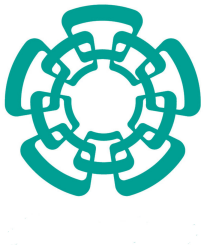


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Laboratorio de Tecnologías de la Información

Metaheurísticas para la resolución del problema de máxima parsimonia

Tesis que presenta:

Karla Esmeralda Vázquez Ortiz

Para obtener el grado de:

Maestro en Ciencias en Computación

Director de la Tesis:
Dr. Eduardo Arturo Rodríguez Tello

Cd. Victoria, Tamaulipas, México. Diciembre, 2011

Nota aclaratoria:

Las reglas ortográficas que se siguieron en la escritura de esta tesis son las publicadas en el año 2010 por la Real Academia Española y la Asociación de Academias de la Lengua Española. De acuerdo a estas reglas para las palabras *este*, *esta*, *estos*, *estas*, *ese*, *esa*, *esos*, *esas*, *aquel*, *aquella*, *aquellos*, *aquellas* no se toma en consideración la posible ambigüedad entre determinantes demostrativos y pronombres demostrativos. Antiguamente se acentuaban cuando tenían función de pronombre. Las palabras *esto, eso, aquello,* que solo pueden ser pronombres, nunca se han acentuado. La palabra *'solo'* puede funcionar como adjetivo o como adverbio. Antiguamente se acentuaba cuando tenía función de adverbio, equivalente a solamente (Real Academia Española, 2010).

La tesis presentada por Karla Esmeralda Vázquez Ortiz fue aprobada por:

Dr. José Gabriel Ramírez Torres

Dr. Gregorio Toscano Pulido

Dr. Eduardo Arturo Rodríguez Tello, Director

Cd. Victoria, Tamaulipas, México., 14 de Diciembre de 2011

A mis hijos, mi motivación en la vida

Agradecimientos

- Esta investigación fue parcialmente financiada mediante los proyectos: CONACyT 99276, Algoritmos para la Canonización de Covering Arrays; 51623 Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas.
- Gracias a mis hijos porque me dieron una razón para continuar con mi superación profesional y sin reproches me permitieron dedicar mucho del tiempo que les correspondía a la realización de este trabajo de investigación
- Un gran agradecimiento a mi director el Dr. Eduardo Rodríguez Tello por su apoyo moral e intelectual y por que supo ser un excelente guía, sin él no hubiera sido posible concluir con éxito este trabajo
- Quiero agradecer al Dr. Jean-Michel Richer de la Universidad de Angers en Francia, su apoyo también forma parte de mi trabajo
- A mis compañeros Luis Carlos, Edna, Yazmin, Erika, Gerardo, Guillermo, Jorge, Marco, Pedro y Ezra por mencionar solo algunos les agradezco los gratos momentos que compartieron conmigo
- Agradezco también al Dr. José Gabriel Ramírez Torres y al Dr. Gregorio Toscano Pulido, por dedicar tiempo a la revisión de este trabajo, sus observaciones fueron de gran utilidad
- Este trabajo se llevo a cabo con el apoyo económico de CONACyT y el apoyo técnico y administrativo del laboratorio de tecnologías de información (Cinvestav - Tamaulipas).

Sin el apoyo de todas estas personas no habría podido llevar a cabo este trabajo. A todos ellos muchas gracias

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Publicaciones

Karla E. Vázquez-Ortiz, Eduardo Rodríguez-Tello. *Metaheuristics for the Maximum Parsimony Problem*, The sixth IASTED International Conference on Computacional Intelligence and Bioinformatics (CIB 2011), Pittsburgh, USA, November 2011, páginas 105-113.

Resumen

Metaheurísticas para la resolución del problema de máxima parsimonia

por

Karla Esmeralda Vázquez Ortiz

Maestro en Ciencias del Laboratorio de Tecnologías de la Información
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2011
Dr. Eduardo Arturo Rodríguez Tello, Director

El problema de Máxima Parsimonia (MP) consiste en encontrar una topología de árbol, entre todas las posibles, que permita representar las relaciones evolutivas para un grupo de especies dado, de forma tal que esta contenga el menor número de cambios evolutivos (mutaciones) posibles. El problema de MP pertenece a la clase de problemas NP-completos y es altamente combinatorio, por lo que los algoritmos exactos existentes para resolverlo solo funcionan para instancias pequeñas del problema (con menos de 10 especies). Una alternativa viable a esta problemática consiste en el uso de algoritmos aproximados para resolverlo.

En este trabajo de investigación se analizaron diversos métodos aproximados para resolver el problema de MP y se seleccionaron dos de ellos para su implementación: la búsqueda local iterativa (*ILS, iterated local search*) y el recocido simulado (*SA, Simulated Annealing*), los cuales llamamos ILS-MP y SA-MP, respectivamente. Para cada componente esencial de estos algoritmos metaheurísticos, se consideraron diferentes opciones entre las que se encuentran dos métodos de inicialización, dos funciones de vecindad básicas y diecisiete vecindarios combinados. En el caso de ILS-MP se analizó el impacto de la función de vecindad en el desempeño del algoritmo de búsqueda local embebido. Por otra parte para el algoritmo SA-MP se estudiaron dos diferentes métodos para determinar el valor de la temperatura inicial, un esquema de enfriamiento de la temperatura de tipo estático así como uno dinámico y dos criterios de paro para el algoritmo.

A partir de una extensa comparación experimental, utilizando 18 instancias estándar de la literatura, se logró identificar la mejor combinación de componentes esenciales para cada una de las metaheurísticas propuestas. ILS-MP y SA-MP fueron comparados empleando sus mejores componentes y se demostró que SA-MP permite obtener mejores resultados que los proporcionados por ILS-MP. Posteriormente, se efectuó la sintonización de los parámetros de entrada de SA-MP mediante una técnica inspirada en pruebas de interacción combinatoria con la finalidad de potenciar su desempeño. La versión sintonizada de SA-MP se comparó empíricamente contra tres métodos representativos del estado del arte, permitiendo observar que SA-MP fue capaz de mejorar en el $28\,\%$ de las instancias las mejores soluciones previamente conocidas y en el $72\,\%$ restante igualar estos resultados. Finalmente, se ejemplificó la utilidad práctica del algoritmo SA-MP, en el área de epidemiología, para la construcción de árboles filogenéticos usados en el análisis de un problema real compuesto por 148 cepas del virus de influenza.

Abstract

Metaheuristics for solving the maximum parsimony problem

by

Karla Esmeralda Vázquez Ortiz
Master of Science from the Information Technology Laboratory
Research Center for Advanced Study from the National Polytechnic Institute, 2011
Dr. Eduardo Arturo Rodríguez Tello, Advisor

The Maximum Parsimony problem (MP) consists in finding a tree topology, among all the possibles ones, which permits to represent the evolutionary relationships for a given set of species in such a way that it contains the minimum number of evolutionary changes (mutations). The MP problem belongs to the class NP-complete and it is highly combinatorial, for this reason the existing exact algorithms for solving it only can manage small instances of the problem (with less than 10 species). The most suitable alternative is thus the use of approximate algorithms.

In this research work, different approximate methods for solving the MP problem were analyzed and two of them were selected for their implementation: Iterated Local Search (ILS) and Simulated Annealing (SA), which are called ILS-MP and SA-MP, respectively. For each essential component of these metaheuristic algorithms different options were considered, among them two initialization methods, two basic neighborhood functions and seventeen composed neighborhoods. For the case of ILS-MP, the impact of the neighborhood function on the performance of the embedded local search algorithm was analyzed. For the SA-MP algorithm, two different methods to determine the value of the initial temperature, a dynamic as well as a static cooling scheme and two stopping criteria were studied.

Through an extensive experimental comparison, employing 18 standard benchmark instances taken from the literature, the best combination of essential components for each implemented metaheuristic was identified. Using their best components ILS-MP and SA-MP were compared, showing that SA-MP allows to obtain better results than those reached by ILS-MP. Later, the input

parameters used by SA-MP were tuned, for improving its performance, by using a technique inspired by combinatorial interaction testing. The fine-tunned version of SA-MP was empirically compared against three representative state-of-the-art methods showing that SA-MP was able to improve the previous best-known solutions for $28\,\%$ of the benchmark instances, and to equal those results for the remaining of the instances.

Finally, the practical utility of SA-MP for constructing phylogenetic trees was shown using a real epidemiology example composed by 148 strains of the influenza virus. to assess the practical utility of SA-MP in the epidemiology area, for which we have used as input a compound of 148 strains of the influenza virus.

1

Introducción

1.1 Antecedentes

La Bioinformática es un área de investigación interdisciplinaria donde intervienen biología, estadística, matemáticas aplicadas, química, bioquímica, inteligencia artificial y ciencias de la computación. La Bioinformática estudia el desarrollo de métodos computacionales y técnicas estadísticas para resolver problemas prácticos y teóricos que se derivan del almacenamiento, extracción, manipulación y distribución de información relativa a macromoléculas biológicas como el ADN (ácido desoxirribonucleico), el ARN (ácido ribonucleico) y las proteínas. Dichas macromoléculas, se encuentran representadas por secuencias de caracteres.

En lo sucesivo se utilizará el término macromoléculas biológicas para hacer referencia a secuencias de ADN, ARN y proteínas.

Dentro de la bioinformática se pueden distinguir dos vertientes. La primera lleva a cabo el desarrollo de herramientas computacionales y bases de datos (BD). La segunda comprende la

aplicación de estas herramientas y BD en la generación de conocimiento biológico para comprender mejor los sistemas vivos. Cabe mencionar que estos dos campos son complementarios entre sí.

Algunas de las principales áreas de investigación en bioinformática son: alineamiento de secuencias, predicción de genes, predicción de estructuras de proteínas, predicción de la expresión génica, interacciones proteína-proteína, modelado de la evolución y reconstrucción de árboles filogenéticos, por mencionar solo algunas. En este trabajo de tesis se abordará justamente esta última área de investigación.

Haeckel en 1866 fue el primero en sugerir una secuencia evolutiva para explicar el origen de la primera célula viva (Dose, 1981), lo que dio origen al término filogenia. Este concepto se refiere a una clasificación en forma de árbol que refleja la historia evolutiva de una especie o grupo de organismos.

Por su parte la reconstrucción filogenética se basa en la búsqueda de caracteres compartidos entre los miembros del grupo de organismos estudiado con la finalidad de analizar esas relaciones evolutivas.

Existen dos enfoques principales utilizados para la reconstrucción filogenética. El primero, conocido como *morfología comparativa*, se basa en el análisis de características anatómicas, fisiológicas y de conducta compartidas entre los organismos en estudio. El segundo, llamado *bioquímica comparativa*, estudia los caracteres comunes (aminoácidos[1] o nucleótidos[2]) dentro de las macromoléculas pertenecientes a un grupo de individuos analizados. Esta tesis empleará el enfoque de bioquímica comparativa.

En el área de filogenética existen diferentes métodos para la reconstrucción de árboles filogenéticos, siendo el de *Máxima Parsimonia* (MP) uno de los principales y con el que se trabajará en esta investigación.

La MP es un método de reconstrucción filogenética que está basado en la suposición de que la hipótesis más probable es aquella que requiere el menor número de explicaciones. Esta es utilizada para clasificar los seres vivos con el fin de recuperar su historia filogenética. Según el cladismo (rama de la biología que define las relaciones evolutivas entre los organismos basándose en similitudes derivadas)

[1] Las secuencias de proteínas están formadas por aminoácidos.
[2] Las secuencias de ADN y ARN están formadas por nucleótidos.

las especies se deben clasificar de acuerdo a grupos monofiléticos (formados por el ancestro común y todos sus descendientes) (Hennig, 1966).

El método de MP busca maximizar la similitud evolutiva, esto involucra la identificación de las topologías de árbol con menor longitud, es decir, aquellas que requieren el menor número de cambios evolutivos (transformaciones en estados de caracteres) para explicar las diferencias observadas entre las secuencias correspondientes a las macromoléculas biológicas de cada una de las especies (Kluge y Farris, 1969).

1.2 Motivación

Existen dos razones principales por las que es interesante el estudio de la reconstrucción de árboles filogenéticos. La primera concerniente a ciencias biológicas donde la reconstrucción de árboles filogenéticos es utilizada para la generación de nuevas vacunas, antibacteriales, herbicidas y en el estudio de la dinámica de comunidades microbianas (Pace, 1997). Por otra parte, en la industria farmacéutica ayuda al desarrollo inteligente de nuevos fármacos. Cabe mencionar que en biología molecular la máxima parsimonia también se utiliza en la predicción de estructuras de proteínas, determinación de homología de secuencias y la clasificación de proteínas entre otras (Murakami y Jones, 2006).

La segunda razón obedece a que el problema de MP es NP-completo, puesto que equivale al problema de optimización combinatoria conocido como el problema del árbol de Steiner en un hipercubo, y el cual está comprobado que es NP-completo (Garey y Johnson, 1977).

Adicionalmente, el tamaño del espacio de búsqueda, *i.e.* el número total de posibles topologías de árbol con raíz, para este problema está dado por la siguiente expresión (Xiong, 2006):

$$\frac{(2n-3)!}{2^{n-2}(n-2)!} \tag{1.1}$$

donde n es el número de especies estudiadas.

Es fácil observar que esta expresión crece de forma factorial en función de n lo que genera espacios

de búsqueda de grandes dimensiones.

Por las razones expuestas anteriormente, en ciencias de la computación representa un reto importante el desarrollar algoritmos capaces de entregar soluciones de buena calidad para problemas altamente complejos como este, en tiempos de cómputo razonables.

1.3 Objetivos de la Tesis

1.3.1 Objetivo general

El objetivo general de este trabajo de investigación consiste en desarrollar un algoritmo metaheurístico capaz de resolver el problema de MP de manera altamente competitiva con respecto a las técnicas propuestas en el estado del arte.

1.3.2 Objetivos específicos

Con el fin de dar cumplimiento al objetivo general de esta tesis es necesario alcanzar los siguientes objetivos específicos:

- Obtener conocimiento acerca de los diferentes métodos de solución propuestos en la literatura
- Identificar en la literatura diversas metaheurísticas como el Recocido Simulado (Kirkpatrick, Gelatt, y Vecchi, 1983; Cerny, 1985), la Búsqueda Tabú (Glover y Laguna, 1999), los Algoritmos Meméticos (Moscato, 1999; Ulder, Aarts, Bandelt, van Laarhoven, y Pesch, 1991) y otras, con la finalidad de seleccionar aquella que permita resolver de manera más eficiente el problema de MP y además contribuir en el estado del arte con una comparativa de las mismas
- Implementar la metaheurística seleccionada para resolver el problema de MP
- Identificar los valores de los parámetros de control que incrementan el desempeño del algoritmo propuesto tanto en calidad de solución como en tiempo de cómputo consumido

- Validar la metaheurística de solución implementada mediante un estudio comparativo con respecto a las mejores técnicas reportadas en la literatura, utilizando para ello métricas estándar
- Publicar los resultados que se obtengan de este trabajo de investigación en un congreso internacional

1.4 Organización de la tesis

Esta tesis esta formada por 4 capítulos más que se encuentran organizados de la siguiente manera:

- El Capítulo 2 muestra una breve historia sobre el origen del problema en estudio, posteriormente se muestra el planteamiento formal del problema así como su complejidad computacional, finalmente se describen las diferentes técnicas que se han utilizado para resolverlo
- En el Capítulo 3 se presentan las metaheurísticas: búsqueda local iterativa (ILS) y el algoritmo de recocido simulado (SA) como métodos de solución al problema de MP. Se identifican los componentes necesarios para su implementación y se describe el objetivo de cada uno de ellos
- En el Capítulo 4 se analizan de manera experimental diferentes opciones de componentes para los algoritmos ILS y SA con el fin de identificar aquellos que les permitan mejorar su desempeño. Posteriormente se comparan experimentalmente entre sí ambos algoritmos implementados en este trabajo para identificar aquel que mejores soluciones entrega para el problema de MP. El algoritmo SA resultó ser la mejor implementación de este trabajo, la cual es posteriormente comparada con tres de los mejores algoritmos reportados en el estado del arte. Para finalizar este capítulo, se muestra una aplicación práctica del problema de MP donde se emplean árboles filogenéticos (construidos con nuestro algoritmo SA) para resolver un caso práctico en el que se determina el origen de una epidemia.
- En el capítulo 5 se presentan las conclusiones obtenidas durante este trabajo de investigación y se abordan algunas posibilidades de trabajo futuro

2

Estado del arte

En este capítulo se presenta una breve reseña histórica sobre los acontecimientos que dieron origen al problema de máxima parsimonia, posteriormente se muestran la definición formal del problema, su complejidad computacional y diferentes métodos exactos y aproximados que se han empleado para resolverlo.

2.1 Historia

Podemos decir que el análisis filogenético de las especies tuvo su origen a partir del siglo XVIII, cuando surgen las teorías de la evolución y tiene como base fundamental las leyes de la herencia. Es por esta razón que iniciaremos con una breve reseña histórica de hechos importantes.

Darwin (1859) publicó el libro "El origen de las especies", en el que proponía el mecanismo que había dado origen a toda la biodiversidad. De esta gran obra nacen las teorías evolutivas. A grandes rasgos podemos decir que los seres vivos se reproducen, y que solo sobreviven los más aptos, es decir, aquellos que poseen capacidades de adaptación al entorno. Así, generación tras generación, la naturaleza va eligiendo a los más aptos y con el tiempo se van creando nuevas especies. Por lo

que se llega a la conclusión de que toda especie viva se ha generado a partir de otra anterior, dado que todos los seres vivos compartimos el mismo código genético, podemos deducir que todos los habitantes del planeta descendemos de un antepasado común (Darwin, 1859).

Según Van Der Pas (1959), Gregor Mendel realizó trabajos en un pequeño jardín, que consistían en cruzar plantas que deberían diferir en al menos dos caracteres. Tras un período de observaciones planteo las leyes de la herencia y en el año de 1866 publicó sus resultados con el título "Ensayos sobre los híbridos vegetales". Sin embargo, su trabajo no fue reconocido y fue hasta el año 1900 que la comunidad científica se comenzó a interesar en los mecanismos de transmisión genética.

Dose (1981) expone en su trabajo de investigación que Haeckel (1866) apoyado en la idea de Darwin, defendió de forma radical la teoría de la recapitulación (actualmente descartada), que dice, que los procesos que sufren los seres vivos desde la fecundación hasta su plenitud y madurez siguen a los cambios y evolución de las especies. Anticipó el hecho de que la clave de los factores hereditarios residen en el núcleo de la célula. Fue el primero en establecer una hipótesis filogenética de la diversidad biológica ajustada a la teoría de la evolución.

Hennig (1966) propuso el cladismo, que es un método utilizado para reconstruir la genealogía de los organismos, y se basa en tres principios:

1. Los taxones (organismos que después de un proceso de clasificación resultan emparentados) se unen en grupos basándose en los rasgos compartidos

2. Todos los grupos descienden de un antepasado común (es decir son monofiléticos)

3. El árbol que tiene la mayor probabilidad de ser correcto es aquel que demuestre ser el más sencillo y que contenga el menor número de cambios. Esto se define como el árbol más parsimonioso

El problema de máxima parsimonia surge de la necesidad de clasificar las diferentes especies, con el fin de obtener su historia evolutiva. Los árboles filogenéticos representan la posibilidad de realizar una representación de la historia evolutiva de un grupo de especies bajo el criterio de parsimonia. Es

decir, bajo este criterio el árbol más parsimonioso es aquel que implica el menor número de cambios evolutivos.

Edwards y Sforza (1963), fueron los primeros en mencionar el criterio de máxima parsimonia en filogenia. Introdujeron métodos computacionales para calcular árboles evolutivos a partir de información genética. Su contribución más importante fue la introducción de métodos estocásticos de estimación, aplicados a modelos estocásticos de evolución.

Después de haber analizado los acontecimientos históricos que sirvieron como base a este problema de investigación, se procede a revisar los métodos utilizados para resolver el problema de MP.

2.2 Definición del problema

Sea un conjunto $\mathscr{S} = \{S_1, S_2, \ldots, S_n\}$ de n secuencias de longitud k (taxones operacionales previamente alineados) sobre un alfabeto $\mathscr{A}$ previamente definido. Un *árbol filogenético con raíz* $T = (V, E)$ que representa relaciones ancestrales está compuesto por un conjunto de nodos $V = \{v_1, \ldots, v_r\}$ y un conjunto de aristas $E \subseteq V \times V = \{\{u, v\} | u, v \in V\}$. El conjunto de nodos $|V| = (2n - 1)$ se divide en dos subconjuntos: I, contiene $n - 1$ *nodos internos* (ancestros hipotéticos), cada uno con dos descendientes; y L compuesto de n *hojas*, i.e., nodos sin descendientes.

La secuencia de parsimonia P_w de cada nodo interno $I_w \in I$ del cual sus descendientes son $S_u = \{x_1, \cdots, x_k\}$ y $S_v = \{y_1, \cdots, y_k\}$ se calcula con la siguiente expresión:

$$\forall i, 1 \leq i \leq k, z_i = \begin{cases} x_i \cup y_i, & \text{si } x_i \cap y_i = \emptyset \\ x_i \cap y_i, & \text{sino} \end{cases} \tag{2.1}$$

El costo de parsimonia de la secuencia (mutaciones) P_w está definido como:

$$\phi(P_w) = \sum_{i=1}^{k} C_i \qquad \text{donde} \qquad C_i = \begin{cases} 1, & \text{si } x_i \cap y_i = \emptyset \\ 0, & \text{sino} \end{cases} \tag{2.2}$$

y el costo de parsimonia para el árbol T se obtiene con la siguiente fórmula:

$$\phi(T) = \sum_{\forall w \in I} \phi(P_w) \tag{2.3}$$

Por lo tanto el problema de MP consiste en encontrar una topología de árbol T^* para la cual $\phi(T^*)$ sea mínimo, i.e.,

$$\phi(T^*) = \text{mín}\{\phi(T) : T \in \mathscr{T}\} \tag{2.4}$$

donde $\mathscr{T}$ es el conjunto de todas las posibles topologías de árbol (espacio de búsqueda).

2.3 Complejidad computacional

En esta sección se presentan diferentes aspectos relacionados con la complejidad computacional del problema de MP. Adicionalmente, se muestra el análisis de la cardinalidad del conjunto de todas las soluciones potenciales (espacio de búsqueda) al problema.

El problema de MP es NP-completo (Gusfield, 1997). Esto es fácilmente comprobable puesto que es equivalente al problema del árbol de Steiner en un hipercubo, el cual es un problema conocido de optimización combinatoria que pertenece a la clase NP-completo (Garey y Johnson, 1977).

Definición 1 (Hipercubo) *Un hipercubo d es un grafo no dirigido con nodos 2^d biyectivamente etiquetados, es decir, cada etiqueta pertenece a un solo elemento en el grafo y que todos los elementos en el grafo pertenecen al menos a un elemento del grupo de etiquetas, donde las etiquetas están representadas por los números enteros entre 0 y $2^d - 1$. Dos nodos de un hipercubo son adyacentes si y solo si, los equivalentes del binario de sus etiquetas se diferencian por un solo bit.*

Para facilitar la comprensión de la reducción del problema de MP al problema del árbol de Steiner en un hipercubo se toma como base el trabajo realizado por Goëffon (2006), donde se pude ver a un árbol filogenético como un conjunto de puntos conectados en una secuencia particular dentro del hipercubo. Consideraremos dos secuencias binarias S_1 y S_2, con $\mathscr{A} = \{0, 1\}$ y 3 puntos de forma simultánea, donde cada secuencia está compuesta por uno o dos caracteres (ver Tabla 2.1).

Ejemplo 1	Ejemplo 2	Ejemplo 3
S_1 0	S_1 00	S_1 00
S_2 1	S_2 01	S_2 11

Tabla 2.1: Tres ejemplos con dos secuencias binarias S_1 y S_2.

Ahora solo es cuestión de imaginar un hipercubo de dimensión m, cuyos nodos se representan por cada una de las 2^m secuencias binarias distintas y que de acuerdo a la definición se consideran adyacentes solamente si su distancia de Hamming (la diferencia entre una palabra de código válida y otra) es igual a uno.

Nótese que con dos taxones, solo puede construirse un árbol filogenético. En la Figura 2.1 se muestra la forma de representar un árbol en un hipercubo, donde los círculos más grandes corresponden a las secuencias del problema y están etiquetados con cada una de las $2^m - n$ secuencias posibles. En este contexto el problema de MP consiste en encontrar el árbol más corto dentro del hipercubo y que a su vez contenga todos los nodos correspondientes a los datos específicos. Cada arista del árbol simboliza un cambio de estado, y la puntuación de parsimonia del árbol es igual a su número de aristas (longitud).

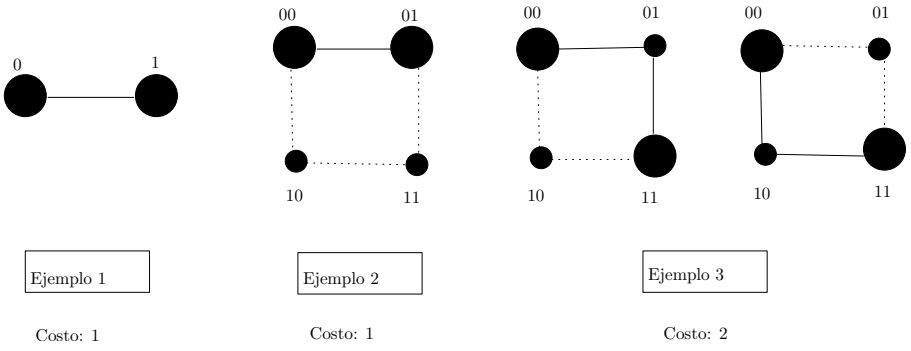


Figura 2.1: Ejemplo de representación del problema de MP en un hipercubo.

El árbol que se obtiene no necesariamente será binario, pero es relativamente sencillo convertirlo en uno. Primero se insertan nodos de grado 3 cuando ciertos nodos del hipercubo tienen un grado superior. Después, se extraen los nodos particulares del hipercubo, para transformarlos en hojas, si su grado es igual a 2. Finalmente, para completar el árbol binario obtenido, es suficiente con remplazar todos los caminos $(x_i, \ldots, x_{i+l})$ donde solamente x_i y x_{i+l} tienen un grado diferente de 2, por una

arista simple (x_i, x_{i+l}) etiquetada con la longitud l del camino. Es así como se obtiene una solución sin raíz al problema, pero es posible insertar una raíz en cualquier parte del árbol.

En la Figura 2.1 el *Ejemplo 1* representa un hipercubo de dimensión uno, el cual solo tiene una solución al problema. En el *Ejemplo 2*, además de la topología trivial de un árbol con dos hojas, se representa el árbol más corto con lo que se optimiza el resultado de la parsimonia. Del mismo modo en el *Ejemplo 3*, se tienen dos posibles explicaciones: un ancestro hipotético etiquetado con 01 y otro con 10.

Definición 2 (Árbol de Steiner.) *Sea $G = (V, E)$ un grafo no dirigido. V es el conjunto de nodos y $E \subseteq V^2$ aristas. En cada arista $(v_i, v_j) \in E$ se asocia un peso no negativo $\omega_{ij} \in \mathbb{N}^*$. Sea $X \subseteq V$ un subconjunto de nodos. Un árbol de Steiner $ST_G(X) = (V', E')$ es un árbol tal que $X \subseteq V' \subseteq V$ y $E' \subseteq E$. Los nodos de $V' \setminus X$ son puntos de Steiner. El peso $ST_G(X)$, que se denota $\Omega(ST_G(X))$, se define por:*

$$\Omega(ST_G(X)) = \sum_{\substack{i,j \\ (v_i,v_j)\in E'}} \omega_{ij} \tag{2.5}$$

El problema de Steiner nombrado así por su creador, el matemático suizo Jakob Steiner (1796-1863), trata de encontrar el árbol Steiner de longitud mínima dado G y X. El problema de Steiner se genera cuando las aristas de G no tienen un valor asociado. En este caso, $\omega_{ij} = 1$ para todo i, j tal que $(v_i, v_j) \in E'$.

El problema de MP al mapearse a secuencias binarias equivale al problema de Steiner en un hipercubo no ponderado. Un hipercubo de dimensión m contiene un registro para cada una de las 2^m cadenas de m bits. Dos secuencias separadas por un cambio de estado, están representadas por dos vértices adyacentes en el hipercubo. Es fácil observar que como el árbol de Steiner es el más corto, este equivale al árbol más parsimonioso, tolerando la posibilidad de un árbol no binario. El número de aristas del árbol de Steiner es igual a la puntuación mínima de parsimonia. Sankoff y Rousseau (1975) fueron los primeros en establecer un enlace entre el problema de MP y el problema de Steiner.

Regresando al ejemplo propuesto por Goëffon (2006), en el que se utilizan cinco secuencias de longitud cuatro definidas a continuación:

Secuencia 1	0000
Secuencia 2	0011
Secuencia 3	0101
Secuencia 4	0110
Secuencia 5	1001

El problema consiste en encontrar el árbol más parsimonioso. Para resolver el problema de Steiner, se considera un hipercubo $H = (V, E)$ de dimensión 4 y un conjunto de cinco nodos $X = \{0000, 0011, 0101, 0110, 1001\}$. En la Figura 2.2.a, se representa a H en las cinco esquinas denotadas por los círculos obscuros de mayor tamaño, así como un árbol de Steiner de longitud 6. Este árbol se muestra más claramente en la Figura 2.2.b, donde cada arista representa un cambio de estado. Los nodos iniciales se muestran en negrita, mientras que los otros nodos representan los puntos de Steiner

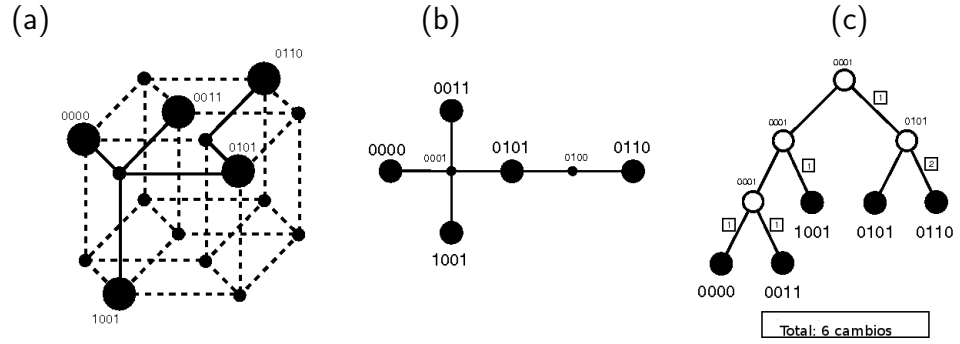


Figura 2.2: Problema de Steiner en un hipercubo de dimensión 4.

En la práctica, no es necesario ir más allá, porque el árbol de la Figura 2.2.b, mediante la eliminación de los puntos innecesarios de Steiner (nivel 2), muestra toda la información requerida. Sin embargo, para lograr un mayor rigor y cumplir con las Definiciones 1 y 2, el árbol se representa con mayor claridad en la Figura 2.2.c. En él se puede notar una puntuación de 6.

De acuerdo con Foulds y Graham (1982), el problema de Steiner en un hipercubo es NP-completo, y la equivalencia al problema de MP también, por lo tanto, es poco probable que existan

algoritmos exactos que corran en tiempo polinomial para resolver este difícil problema de optimización combinatoria.

2.4 Análisis del espacio de búsqueda

El tamaño del espacio de búsqueda para el problema de MP está dado por el número total de posibles topologías de árbol para el conjunto de n especies analizadas. La siguientes expresión permite calcular de manera exacta dicho tamaño:

$$N_R = (2n-3)!/2^{n-2}(n-2)! \tag{2.6}$$

donde N_R es el número total de posibles topologías de árbol con raíz.

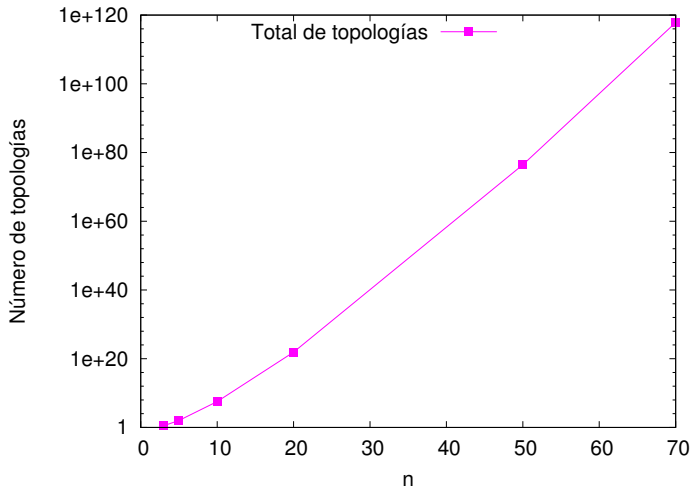


Figura 2.3: Número total de topologías de árbol con raíz en función del número n de especies analizadas.

Por ejemplo, si aplicamos las fórmulas anteriores para un grupo de $n = 6$ especies se observa que existen 945 posibles topologías de árbol. Para este ejemplo el tamaño del espacio de búsqueda es relativamente pequeño. Sin embargo, para instancias con $n = 125$ especies, manejadas comúnmente en la literatura, el número total de posibles topologías de árboles con raíz es $N_R = 1.62129384\text{E}{+}243$.

En la Figura 2.3 se muestra gráficamente la forma en que crece el número de las posibles topologías de árboles filogenéticos con raíz en función del número n de especies analizadas.

A partir de esta gráfica se puede observar claramente que el tamaño del espacio de búsqueda para este problema se incrementa de forma factorial en función del valor de n.

2.5 Métodos de resolución del problema de MP

Desde los años 1960s se ha desarrollado mucho trabajo para la reconstrucción de árboles filogenéticos. Podemos mencionar dos enfoques utilizados con este fin:

- *Basados en distancias.* Este tipo de métodos requieren una alineación previa de las secuencias en estudio. Se construye una matriz completa de correspondencias que describen las distancias existentes entre cada par de secuencias.
    - Basados en agrupamiento
        - UPGMA (unweighted Pair Group Method Using Aritmetic Average). Sokal y Michener (1958) propusieron el método de agrupamiento de pares no ponderado con media aritmética. En este método al inicio se identifican los dos taxones que son más similares entre sí, los agrupa como uno solo, posteriormente realiza la misma operación con los taxones restantes. Este proceso se repite hasta que solo quedan dos taxones. Finalmente se calcula la distancia existente entre los taxones que han sido agrupados. En la construcción de árboles filogenéticos se utiliza solo cuando la tasa de evolución es aproximadamente constante a través de las diferentes líneas evolutivas.
    - Basados en optimalidad
        - Fitch y Margoliash (1967), proponen seleccionar el mejor árbol basándose en la mínima desviación entre las distancias calculadas en la totalidad de las ramas del árbol y las distancias del conjunto de datos originales.

- Evolución mínima. Emplea programación lineal para la construcción de árboles. Este método busca un árbol con el menor número de mutaciones. Rzhetsky y Nei (1993) aplicaron este método para la construcción de árboles filogenéticos.

- *Basados en caracteres*

  - Máxima Verosimilitud. Es un método exhaustivo que busca todas las posibles topologías y considera cada posición en un alineamiento (no solo los sitios informativos). Foster y Hickey (1999) aplicaron este método a la reconstrucción filogenética.
  - Máxima Parsimonia. Busca el árbol con el menor número de sustituciones para explicar de mejor manera las relaciones evolutivas entre los taxones estudiados.

Debido a que Máxima Parsimonia es el método más utilizado por ser más eficiente, consideramos apropiado aplicarlo en este trabajo de investigación.

Existen diferentes métodos para resolver el problema de MP, algunos proporcionan una solución exacta pero su tiempo de calculo limita el tamaño de la instancia, es decir, solo se puede emplear en instancias pequeñas. Los métodos aproximados dan una solución en un tiempo razonable mas no garantizan que sea la solución óptima del problema. En las siguientes secciones se presentan los diferentes métodos empleados para resolver este problema.

2.5.1 Métodos exactos

Los métodos exactos exploran el espacio de búsqueda proporcionando la mejor solución posible. Sin embargo, representan un método inviable para problemas en los que el espacio de búsqueda crece de forma exponencial o similar, en función del tamaño de las instancias, que esta dado por el n número de taxones estudiados.

2.5.1.1. Búsqueda exhaustiva

Este método enumera todos los posibles árboles para posteriormente calcular su puntuación de parsimonia. Debido a que el número de árboles se incrementa de forma factorial como se vio en la

Sección 1.5, este método resulta ineficiente. Su aplicabilidad está limitada a instancias con menos de diez taxones (Swofford, Olsen, Waddell, y Hillis, 1996), de las que se obtienen todas las posibles topologías de árbol entre las que se encuentra el más parsimonioso. En la práctica nadie utiliza la búsqueda exhaustiva ya que en las instancias reales el número de taxones es superior a diez.

2.5.1.2. Algoritmo de ramificación y poda

Dentro de los algoritmos que resuelven con precisión los problemas combinatorios está el algoritmo de ramificación y poda (RP) (Land y Doig, 1960), que se adapta a muchos problemas de optimización. Hendy y Penny (1982) realizaron la primera aplicación al problema de MP donde construyen inicialmente todos los posibles árboles agregando un taxón a la vez y verifican el costo del árbol parcial. En cuanto se encuentra una puntuación de parsimonia que supera una cota C, se abandona la construcción de árboles de forma parcial. La cota C se calcula inicialmente para un árbol generado de manera aleatoria o heurística. La cota C, se ajusta durante la búsqueda.

El tiempo de cómputo del algoritmo de RP depende de la elección de la cota C (Swofford, Olsen, Waddell, y Hillis, 1996), el hecho de que MP sea un problema altamente combinatorio, hace que sea inviable el uso de un método exacto para su resolución cuando el número de taxones es mayor a quince.

2.5.2 Métodos aproximados

Los métodos aproximados o heurísticos para construcción de árboles generan de forma iterativa una solución aproximada al problema. Algunos enfoques resultan ser eficientes para resolver ciertos problemas pero en otros casos resultan más costosos debido a que consumen más tiempo de computo. Este tipo de métodos se aplican cuando el obtener una solución mediante métodos exactos se torna imposible. Aunque, si bien proporcionan un árbol filogenético en tiempo razonable, no garantizan que sea el más parsimonioso.

A continuación analizaremos algunos métodos aproximados representativos del estado del arte.

2.5.2.1. Algoritmos voraces

Este tipo de algoritmos se utilizan frecuentemente en reconstrucción filogenética con el nombre de adición por pasos. Funcionan insertando taxones en lugares del árbol que reduzcan al mínimo el aumento de puntuación de parsimonia. Solo algunas variantes de este algoritmo son no deterministas.

Andreatta y Ribeiro (2002) propusieron una comparación entre tres algoritmos voraces de diferente complejidad y eficiencia:

- *1stRotuGbr* cada iteración selecciona un taxón al azar y se inserta en el lugar que reduce al mínimo la puntuación de parsimonia;
- *Gstep_wR* selecciona un taxón y una rama de tal manera que al adicionar este taxón en esa posición, el aumento de puntaje ocasionado por esta adición sea mínimo;
- *GRstep* es una variante de *Gstep_wR* donde la rama y el taxón seleccionados no aumentan la puntuación de parsimonia en más del 10 % de la generada por el mejor.

De sus experimentos se concluye que, *Gstep_wR* es ligeramente más eficaz que *1stRotuGbr* pero su complejidad ($O(n^3)$ en contra de $O(n^2)$) aumenta dramáticamente el tiempo de cálculo. *GRstep* solo es útil cuando se combina con un método de búsqueda local.

Los algoritmos voraces determinan su efectividad en tiempo de cálculo cuando las instancias a tratar son de gran tamaño, por lo que es poco probable acercarse a los resultados obtenidos por otros algoritmos heurísticos.

2.5.2.2. Búsqueda local estocástica

El problema de MP tiene espacios de búsqueda muy grandes por lo que se ha comprobado empíricamente que los métodos heurísticos de búsqueda local estocástica (o métodos de vecindario) son apropiadas para resolver este problema (Ganapathy, Ramachandran, y Warnow, 2004). Este tipo de métodos inician con una solución S_0, para posteriormente realizar búsquedas iterativas al rededor de ella, en caso de encontrar una mejor solución, sustituye S_0 con la nueva solución. Este proceso se

repite hasta cumplir con la condición de paro del algoritmo, que puede ser el número de soluciones vecinas exploradas.

*2.5.2.3. Descenso puro con intercambio de ramas (*branch-swapping*)*

El algoritmo de descenso puro genera un árbol inicial, posteriormente busca un árbol cercano (dentro de su vecindario), que debe tener una puntuación inferior, este proceso se repite hasta que ya no se encuentra un árbol con puntaje inferior en el vecindario. Este árbol representa una solución óptima local, más no necesariamente representa la solución óptima del problema. Weber et al. (2006) realizaron una comparación de diferentes métodos para la reconstrucción de árboles filogenéticos incluyendo un algoritmo de descenso puro.

Los algoritmos de descenso, son la base de todos los métodos que resuelven de forma eficiente el problema de MP, aún y cuando existen técnicas que proporcionan mejor calidad en las soluciones entregadas.

Durante este trabajo analizaremos, dos vecindarios de árboles: NNI y SPR. Los cuales revisaremos a continuación.

Intercambio del vecino más cercano (NNI)

NNI es una técnica propuesta por Waterman y Smith (1978), que consiste en intercambiar dos subárboles separados por un nodo interior. El tamaño del vecindario es $O(n)$. Cada árbol calculado tiene $2n-6$ vecinos NNI, $n-3$ nodos internos y dos posibles movimientos por arista (Robinson, 1971). Esta función trata de seleccionar el árbol que demuestre tener mejor desempeño que el resto de sus vecinos. Una representación gráfica de este concepto puede ser observada en la Figura 2.4.

Poda y reconexión de un subárbol (SPR)

El método SPR (Swofford y Olsen, 1990), elimina un nodo interno existente junto con sus hijos y lo reinserta en otro lugar creando un nuevo nodo interno. Para cada árbol existen $2(n-3)(2n-7)$ reordenamientos posibles de SPR (Allen y Steel, 2001), por lo que el tamaño del vecindario es $O(n^2)$. La aplicación de este vecindario se puede observar en la Figura 2.5.

(a) Subárboles a intercambiar (b) Árbol resultante

Figura 2.4: Ejemplo de vecindario NNI.

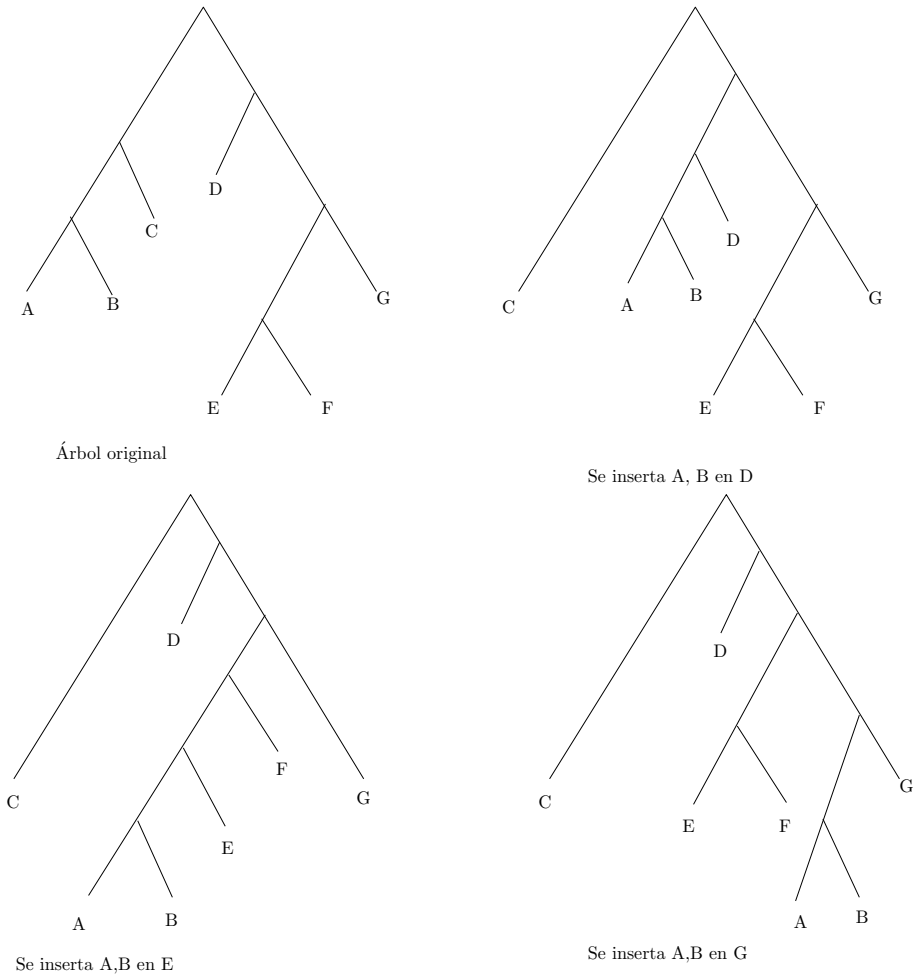


Figura 2.5: Ejemplo de vecindario SPR.

2.5.2.4. Descenso de vecindad variable

Un algoritmo de descenso con un vecindario muy pequeño podría devolver de forma rápida un óptimo local de mala calidad, mientras que en uno muy grande se corre el riesgo, ya sea de repetir vecinos analizados previamente, o bien, de dejar zonas del espacio de búsqueda sin explorar.

Existen varias formas de obtener un óptimo local, sin embargo, la mayoría de las técnicas alteran la configuración actual, lo cual genera altos costos. La mejor manera de crear una nueva configuración es cambiar el vecindario (Mladenović y Hansen, 1997; Hansen y Mladenović, 1999).

El algoritmo de descenso de vecindad variable (Variable Neighborhood Descent) propuesto por Ribeiro y Vianna (2005) (ver Algoritmo 2.1) sigue este patrón para resolver el problema de MP, tiene un buen comportamiento pero sus tiempos de cálculo son relativamente altos.

Algoritmo 2.1 Ejemplo de descenso de vecindad variable.

Entrada: Una matriz de caracteres A, un conjunto de vecindarios $\mathscr{N}^1, \mathscr{N}^2, \ldots, \mathscr{N}^{l_{\text{máx}}}$
Salida: La mejor solución encontrada s_0

```
Construir un árbol inicial s_0
l ← 1
mientras l ⩽ l_máx hacer
  Elegir el mínimo s' ∈ N^l(s_0)
  si f(s') < f(s_0) entonces
    s_0 ← s'
    l ← 1
  sino
    l ← l + 1
  fin si
fin mientras
```

Goëffon et al. (2005) proponen un mecanismo en el que utilizan el concepto de exploración y explotación. Inician la búsqueda explorando en espacios grandes y conforme avanza y se tienen buenas soluciones, restringe su búsqueda a espacios más pequeños aplicando el concepto de explotación.

2.5.2.5. Recocido simulado

Es una metaheurística que se basa en el trabajo de Metropolis et al. (1953) en el campo de la termodinámica estadística, en el que modelan el proceso de recocido, mediante la simulación de los

cambios energéticos en un sistema de partículas conforme decrece la temperatura hasta que converge a un estado estable (congelado). De forma independiente Kirkpatrick et al. (1983) y Cerny (1985) mostraron como este proceso podría ser aplicado a problemas de optimización, asociando conceptos clave del proceso de simulación, con elementos de optimización combinatoria.

El Algoritmo 2.2 representa el método básico de recocido simulado para problemas de minimización. El parámetro t_z representa la temperatura, por lo que una solución con un incremento

Algoritmo 2.2 Algoritmo básico de recocido simulado.

Entrada: Función de costo $f(s)$, solución inicial s_0, temperatura inicial $t_0 > 0$, función de reducción de la temperatura ω, un criterio de parada CP, temperatura final t_f
Salida: Mejor solución visitada

```
t_z ← t_0
mientras t_z < t_f hacer
  mientras CP hacer
    Seleccionar aleatoriamente una solución s en el vecindario de s_0 denotado como N(s_0)
    Δ ← f(s) − f(s_0)
    si Δ < 0 entonces
      s_0 ← s
    sino
      generar aleatorio r ∈ (0, 1)
      si r < exp(−Δ/t_z) entonces
        s_0 ← s
      fin si
    fin si
  fin mientras
  t_z ← ω(t_z)
fin mientras
```

Δ en la función de costo será aceptada con probabilidad $e^{(-\Delta/t_z)}$. Por lo tanto, si se permite a t_z alcanzar valores suficientemente pequeños, ya no habrá más movimientos a soluciones peores y habrá convergencia a un óptimo local.

En reconstrucción filogenética esta metaheurística fue aplicada por Barker (2003) en su trabajo denominado LVB y posteriormente fue analizada por Goëffon et al. (2005) quienes propusieron un algoritmo de recocido simulado con mejores resultados incluso que DNAPARS, que es un software libre para resolver el problema de MP. Sin embargo, existe una instancia de prueba en la literatura llamada *zilla* para la que ninguno de los programas entregó buenos resultados.

2.5.2.6. Procedimientos de búsqueda voraz aleatorizada y adaptativa (GRASP)

Este procedimiento se divide en dos fases, durante la primera fase, se aplica el procedimiento propio de un algoritmo voraz, en que se agrega paso a paso un taxón en la posición en la que se reduce el incremento de la puntuación de parsimonia, el método híbrido de agregación paso a paso e intercambio de rama sugiere que una vez que se ha insertado el nuevo taxón en el árbol se realice un intercambio de ramas, con el fin de realizar una corrección en el árbol que mejore el costo de parsimonia.

Durante la segunda fase, se aplica un procedimiento de búsqueda local, que devuelve la mejor solución encontrada (ver Algoritmo 2.3). Ribeiro y Vianna (2005) aplicaron el algoritmo GRASP para resolver el problema de MP, realizando pruebas con instancias generadas de manera aleatoria y otras tomadas de la literatura demostrando que la nueva heurística mostraba mejor desempeño en lo que se refiere a calidad de solución, con respecto a los algoritmos registrados en la literatura hasta ese momento.

Algoritmo 2.3 Algoritmo GRASP.

Entrada: $MaxIteraciones$
Salida: Mejor solución visitada s_0
  $s \leftarrow$ Construcción de solucion mediante método voraz
  $s_0 \leftarrow s$
  para $i \leftarrow 1$ **to** $MaxIteraciones$ **hacer**
    $s \leftarrow$ buscarSolucion $\in \mathscr{N}(s_0)$
    $s_0 \leftarrow$ MejorSolucion(s_0,s)
  fin para

2.5.2.7. Algoritmos meméticos

El concepto de algoritmos genéticos (AG) fue introducido por Holland (1975) y popularizado por Goldberg (1989) y se inspira en el proceso de selección natural.

Según Hoos y Stützle (2005) en la mayoría de los casos un algoritmo genético no es suficientemente eficaz, ya que los operadores de cruza y mutación no aumentan la eficiencia de la búsqueda.

Por esta razón los algoritmos genéticos se han hibridado con métodos de búsqueda local, en los que el operador de búsqueda local sustituye o sucede al de mutación. Esto permite realizar búsquedas en distintas áreas, guiadas a su vez por mecanismos genéticos de selección y cruza. Este tipo de métodos hibridados es lo que se conoce como algoritmos meméticos (Moscato, 1999) o algoritmos de búsqueda local genética (Ulder, Aarts, Bandelt, van Laarhoven, y Pesch, 1991) (ver Algoritmo 2.4).

Algoritmo 2.4 Algoritmo Memético.

Entrada: Espacio de búsqueda ER, función de evaluación f, número de individuos en la población z, número máximo de iteraciones $maxIter$, probabilidad de cruza P_c, probabilidad de mutación P_m

Generar (P, z) /* $P \leftarrow \{x_1, x_2, ..., x_z\},\ P \subseteq ER$ */
Busqueda en Vencindario(P)
Evaluar(P, f)
$X \leftarrow$ *Mejor*(P) /* devuelve el mejor individuo */
$nbIter \leftarrow 1$
mientras $nbIter < maxIter$ **hacer**
  $P' \leftarrow$ *Cruza*(P, P_c)
  $P'' \leftarrow$ *Mutación*$(P',\ P_m)$
  Busqueda en Vencindario(P'')
  Evaluar(P'', f)
  $s \leftarrow$ *Mejor*(P'')/* devuelve el mejor individuo en P'' /*
  $P \leftarrow$ *Seleccionar*$(P'' \cup P)$
  si $f(s) < f(s_0)$ **entonces**
    $s_0 \leftarrow s$
  fin si
  $nbIter \leftarrow nbIter + 1$
fin mientras

Los algoritmos meméticos han tenido poca aplicación en reconstrucción filogenética. Hasta el momento solo se registran los trabajos realizados por Matsuda (1996) y Lewis (1998). La aplicación más reciente de estos algoritmos en la resolución del problema de MP la realizaron Richer et al. (2009), con su algoritmo Hydra, en el que se empleó un grupo de 12 instancias del sitio TreeBase (http://www.treebase.org), cuyo tamaño varía entre 116 y 299 taxones y su puntaje de parsimonia no es conocido, esto con el fin de realizar las pruebas. Los resultados de dichas pruebas fueron comparados contra TNT (software más popular y que registra los mejores resultados) obteniendo

una mejoría en 6 de las instancias de prueba, Hydra solo empeoró en una instancia y mostró igual resultado en 5 de las instancias.

Debido a que Hydra es el algoritmo que ha mostrado los mejores resultados hasta el momento, se decidió emplear los resultados obtenidos de su implementación para realizar las comparaciones con los resultados que se obtengan durante este trabajo de investigación.

2.5.3 Resumen del capítulo

Los árboles filogenéticos tienen su origen en el siglo XVIII con las teorías de la Evolución de Darwin. Años mas tarde, Haeckel apoyado en estas teorías, establece la primer hipótesis filogenética de la diversidad biológica ajustada a la teoría de la evolución. Hennig propuso el cladismo, que es un método utilizado para reconstruir la genealogía de los organismos. Posteriormente, surge la necesidad de una clasificación taxonómica de las diferentes especies existentes en el planeta con el fin de obtener su historia evolutiva. Los árboles filogenéticos representan una forma adecuada de realizar esta clasificación bajo el criterio de máxima parsimonia. El problema es que el espacio de búsqueda crece en razón del número de especies que se estudian y es computacionalmente imposible obtener el árbol filogenético más parsimonioso mediante un método exacto para instancias con más de 10 taxones.

Derivado de esto se han implementado métodos de búsqueda heurística que no garantizan proporcionar la solución exacta como: algoritmos voraces, búsqueda local estocástica, descenso puro con intercambio de ramas (*branch-swapping*), recocido simulado, algoritmos meméticos, entre otros.

En este trabajo de tesis se busca por lo tanto desarrollar un algoritmo metaheurístico que proporcione de manera eficiente una solución al problema de MP.

En el siguiente capítulo se muestran dos algoritmos metaheurísticos que se implementaron durante este trabajo de investigación.

3

Metaheurísticas para resolver el problema de Máxima Parsimonia

En este capítulo se hace una breve introducción a las metaheurísticas: recocido simulado (*Simulated Annealing*, SA) y búsqueda local iterativa (*Iterated Local Search*, ILS), mostrando a detalle los componentes necesarios para su implementación. Para cada uno de estos componentes se describe su objetivo y se analizan diferentes variantes especialmente diseñadas para la resolución del problema de MP.

3.1 Introducción

Existen diferentes métodos para resolver el problema de Máxima Parsimonia. Sin embargo, es difícil y en ocasiones inviable obtener una solución exacta para instancias con mas de diez taxones, debido a que su espacio de búsqueda crece de forma factorial, como se mostró en la Sección 2.4. Después de hacer un estudio minucioso de diferentes metaheurísticas se decidió implementar 2 de ellas: búsqueda local iterativa (ILS) y recocido simulado (SA). Esto debido a que notamos que

estas metaheurísticas tenían las cualidades necesarias, para proporcionar buenos resultados para el problema MP.

A continuación describimos a detalle ambas metaheurísticas así como sus componentes esenciales. Es importante hacer notar que para algunos de estos componentes se desarrollaron diferentes propuestas, las cuales se analizarán a lo largo de este capítulo.

3.2 Búsqueda Local Iterativa

La metaheurística ILS construye iterativamente una secuencia de soluciones generadas por la heurística de búsqueda local embebida. Tiene cinco componentes esenciales: método de inicialización, búsqueda local, perturbación, criterio de aceptación de búsqueda local y condición de paro.

ILS explora un conjunto de soluciones, partiendo de una solución s_0 creada mediante un método de inicialización, hacia otra solución cercana mediante una exploración heurística. Una vez que se tiene s_0, ILS realiza una búsqueda local mediante el algoritmo de descenso obteniendo un óptimo local s. Posteriormente de forma iterativa genera un estado intermedio s^* mediante un mecanismo de perturbación. A la nueva solución se le aplica búsqueda local con el fin de obtener un nuevo óptimo local $s^{*'}$. Finalmente se aplica un criterio de aceptación a s y $s^{*'}$. Este proceso termina cuando se cumple con una condición de paro.

En el Algoritmo 3.5 se puede observar el pseudocódigo de ILS que se propone para dar solución al problema de MP, el cual es llamado ILS-MP en lo sucesivo.

Algoritmo 3.5 ILS-MP

Entrada: Función de vecindad N, función de evaluación f, condición de paro CP
Salida: La mejor solución encontrada s

```
s_0 ← Generar solución inicial mediante algoritmo voraz
s ← Búsqueda local mediante algoritmo de descenso en (s_0)
mientras CP hacer
    s* ← Perturbación(s)
    s*' ← Búsqueda local mediante algoritmo de descenso en (s*)
    s ← Aplicar criterio de aceptación(s, s*')
fin mientras
```

En este trabajo de tesis se probaron diversas opciones para los componentes clave que integran esta metaheurística a fin de lograr un mejor comportamiento del algoritmo.

3.2.1 Método de inicialización

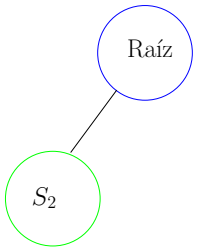
La obtención de una solución inicial, representa el primer paso en todas las técnicas que buscan obtener soluciones de buena calidad a problemas de optimización, en este caso el de Máxima Parsimonia. En una metaheurística la solución inicial es de gran importancia ya que permite iniciar la búsqueda explorando en el espacio de una solución de buena calidad, lo que nos puede guiar hacia mejores resultados. Durante este trabajo se analizaron dos métodos para la generación de una solución inicial. En la siguiente sección mostramos el método aleatorio.

3.2.1.1. Método aleatorio de inicialización

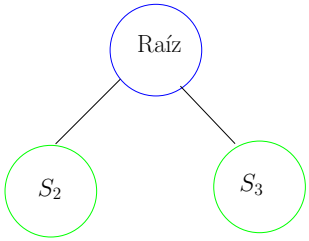
Este método construye un árbol filogenético asignando a cada uno de sus componentes una posición aleatoria. En su primer fase genera una permutación sobre los taxones de la instancia. Esta permutación indica el orden en que cada elemento será colocado en el árbol. En la segunda fase se colocan los elementos de la permutación en posiciones aleatorias del árbol. El siguiente ejemplo muestra de forma clara este procedimiento.

Primera fase: Supongamos que tenemos una instancia de cuatro taxones $\{S_0, S_1, S_2, S_3, \}$, el primer paso es generar una permutación aleatoria sobre ellos obteniendo por ejemplo: $\{S_2, S_3, S_0, S_1, \}$. Posteriormente se crea el nodo raíz del árbol, al que se liga el primer taxón de la permutación en una posición aleatoria (rama izquierda o derecha), como se muestra en la Figura 3.1(a). El segundo taxón se coloca en la rama contraria a la seleccionada para el primer taxón lo que se puede notar en la Figura 3.1(b).

Segunda fase: En esta fase, se agregan los taxones restantes. Estos al agregarse al árbol generan la creación de un nodo interno. Las nuevas hojas se posicionan en el árbol aleatoriamente, es decir, parten de la raíz para elegir una ruta en búsqueda de su posición (izquierda o derecha). En la Figura 3.2(a) las líneas violetas muestran las dos posibles rutas. En esta misma figura podemos notar que



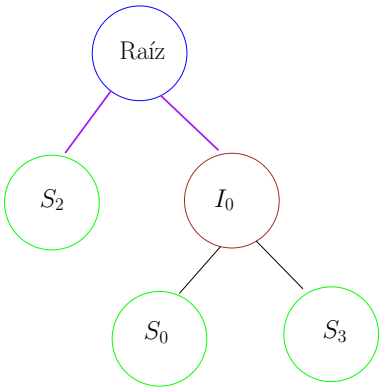
(a)
Inserción del taxón S_2 a la izquierda



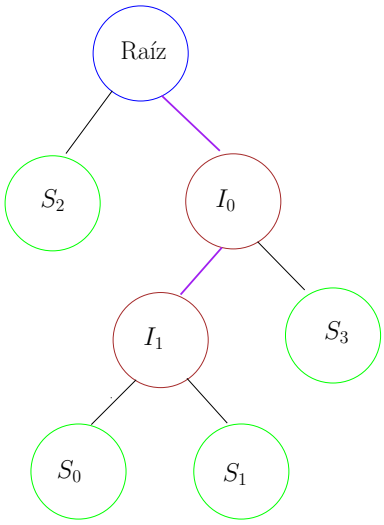
(b) Inserción del taxón S_3 a la derecha

Figura 3.1: Agregando los primeros elementos de la permutación.

hacia el lado izquierdo hay una hoja y al derecho un nodo interno. Si se eligiera la rama izquierda ya se habría encontrado la posición de inserción puesto que se trata de una hoja. Al elegir la rama derecha hay que continuar la búsqueda repitiendo el criterio de elección de ruta aplicado en la raíz. Este procedimiento se repite hasta encontrar una hoja donde se creará el nodo interno para insertar el taxón S_1. La Figura 3.2(b) muestra la inserción del taxón S_1. La ruta que siguió en busca de su posición en el árbol se representa en lineas violetas. Este procedimiento se resume en el Algoritmo 3.6.



(a)
Localización rutas (lineas moradas) y sitios de inserción (círculos verdes)



(b) Creación del nodo I_1 e inserción del taxón S_1

Figura 3.2: Inserción del taxón S_1 y generación del nodo I_1.

Algoritmo 3.6 Método de inicialización aleatoria.

Entrada: Archivo en formato phylip
Salida: Una solución inicial

```
Generar una permutación P con los n taxones de la instancia
i ← 0
mientras i = n hacer
  si i = 0 entonces
    Generar nodo I_0 para raíz
    I_0 → izq ← P[i]
    I_0 → der ← P[i + 1]
    i ← i + 2
  sino
    Generar un número aleatorio r ∈ (0, 1)
    si r = 0 entonces
      Buscar una hoja S_0 del lado izquierdo de la raíz siguiendo direcciones aleatorias
    sino
      Buscar una hoja S_0 del lado derecho de la raíz siguiendo direcciones aleatorias
      Generar I_0
      I_0 → izq ← S_0
      I_0 → der ← P[i]
      i ← i + 1
    fin si
  fin si
fin mientras
```

3.2.1.2. Método voraz de inicialización

Los métodos de construcción voraz intentan construir estratégicamente una solución. Analizan a cada paso las diferentes posiciones en las que los elementos del árbol pueden ser colocados. Su propósito es minimizar el incremento en el costo del árbol. Durante este trabajo de investigación se implementó el método voraz descrito en la Sección 3.2.1.2. Este método de inicialización fue estudiando por Andreatta y Ribeiro (2002) quienes definieron que era uno de los mejores métodos voraces, debido a que proporciona calidad en sus resultados y tiempos de ejecución adecuados.

El Algoritmo 3.7 se implementó en este trabajo de tesis. Este aplica un método en el que al inicio se genera una permutación aleatoria con todos los taxones de la instancia. Posteriormente se crea el nodo raíz al que se ligan los dos primeros taxones de la permutación. El resto de los taxones se agregan al árbol analizando las diferentes posiciones en las que pueden ser colocados. Trata de

identificar la mejor posición para el nuevo elemento, con la finalidad de construir el árbol filogenético con el mejor costo.

Algoritmo 3.7 Método de inicialización voraz.

Entrada: Archivo en formato phylip
Salida: Una solución inicial

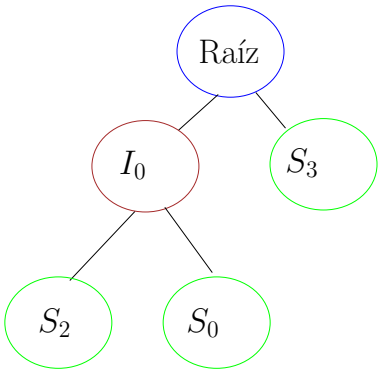
```
Generar una permutación P con los n taxones de la instancia
i ← 0
mientras i = n hacer
  si i = 0 entonces
    Generar nodo I_0 para raíz
    I_0 → izq ← P[i]
    I_0 → der ← P[i + 1]
    i ← i + 2
  sino
    Buscar todas las hojas S del árbol T
    MejorCosto ← ∞
    mientras no se hayan explorado todas las hojas S del árbol T hacer
      S_0 ← Hoja actual
      T_0 ← Crear I_0 en T
      I_0 → izq ← S_0
      I_0 → der ← P[i]
      Evaluar el árbol nuevo T_0 mediante algoritmo Fitch
      si T_0 → costo < MejorCosto entonces
        T ← T_0
        MejorCosto ← T_0 → costo
      sino
        Deshacer movimiento
      fin si
    fin mientras
    i ← i + 1
  fin si
fin mientras
```

El siguiente ejemplo muestra a detalle el funcionamiento de este método.

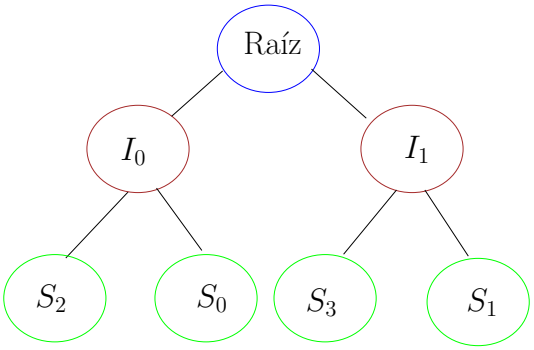
Primera fase: Igual a la primera fase del método aleatorio de inicialización

Segunda fase: Esta fase se aplica a partir del tercer elemento de la permutación, donde al agregar un nuevo elemento al árbol se analizan todas las posibles posiciones en las que puede insertarse. La Figura 3.3(a) muestra en tono verde los círculos correspondientes a las hojas, considerados como los

sitios de inserción. Supongamos que el costo del árbol representado en la Figura 3.3(a) es de 20, y este se incrementa de acuerdo a la posición en la que se inserta S_1 de la siguiente manera: en S_2 se incrementa a 27, en S_0 se incrementa a 25 y en S_3 a 24. Debido a que se trata de minimizar el incremento en el costo, se puede notar que la mejor posición de inserción es S_3 por lo que se elije esta posición para insertar S_1. En la Figura 3.3(b) se puede observar el árbol resultante de este procedimiento.



(a) Árbol de costo 20, sitios de inserción marcados en verde



(b) Creación de nodo I_1 e inserción de S_1. Costo del árbol: 24

Figura 3.3: Funcionamiento del algoritmo voraz.

La solución inicial representa el inicio de una búsqueda hacia soluciones potenciales, sin embargo también deben emplearse técnicas que guíen la búsqueda de manera estratégica con el fin de explorar en espacios prometedores. A continuación se muestra la técnica de búsqueda local empleada para este algoritmo.

3.2.2 Búsqueda local empleando un algoritmo de descenso

ILS-MP requiere de una heurística embebida que realice búsquedas de soluciones óptimas locales. El algoritmo de descenso se probó de manera independiente (ver Apéndice A) y demostró que podía obtener resultados competitivos por lo que se decidió implementarlo dentro de esta metaheurística a fin de mejorar la calidad de sus resultados.

Los algoritmos de descenso aprovechan la ventaja de tener una solución inicial, por lo que parten de ella explorando su vecindario en busca de una mejor solución. En este trabajo se implementó el

algoritmo de descenso mostrado en el Algoritmo 3.8.

Algoritmo 3.8 Algoritmo de descenso.

Entrada: Función de vecindad $\mathscr{N}$, función de evaluación f, solución inicial s''
Salida: La mejor solución encontrada s''
  mientras Condición de paro **hacer**
    Seleccionar $s \in \mathscr{N}(s'')$
    si $f(s) < f(s'')$ **entonces**
      $s'' \leftarrow s$
    fin si
  fin mientras

3.2.3 Funciones de vecindad

La metaheurística ILS-MP, requiere del uso de una función de vecindad, dentro de su heurística de búsqueda local embebida. En esta sección presentamos 11 diferentes funciones de vecindad, especialmente diseñadas para el problema MP.

En el primer grupo se implementaron las funciones de vecindad NNI ($\mathscr{N}_1$) y SPR ($\mathscr{N}_2$) introducidas en la Sección 2.5.2.3 y dos combinaciones de las mismas. El criterio de combinación emplea el contador CS de soluciones (aplicado en la heurística de búsqueda local) que no mejoran a la solución actual s. La Tabla 3.1 muestra dichas combinaciones.

Vecindario	$CS < maxCS/2$	$CS \geq maxCS/2$
$\mathscr{N}_3$	$\mathscr{N}_1$	$\mathscr{N}_2$
$\mathscr{N}_4$	$\mathscr{N}_2$	$\mathscr{N}_1$

Tabla 3.1: Funciones de vecindad para ILS-MP que usan como criterio de combinación el contador de soluciones CS del ciclo de búsqueda de la heurística embebida.

- La función de vecindad $\mathscr{N}_3$, inicia la búsqueda con el vecindario $\mathscr{N}_1$, cuando el contador de soluciones (CS) llega al 50 % de su límite permitido $maxCS$, el algoritmo mueve la búsqueda al vecindario $\mathscr{N}_2$. Con esta condición de aplicación, se permite buscar la mitad de las soluciones en cada uno de los vecindarios.

- La función de vecindad $\mathscr{N}_4$, inicia la búsqueda con el vecindario $\mathscr{N}_2$, cuando el contador de soluciones (CS) llega al 50 % de su límite permitido $maxCS$, el algoritmo mueve la búsqueda al vecindario $\mathscr{N}_1$.

El segundo grupo consta de cuatro combinaciones probabilísticas de las funciones de vecindad $\mathscr{N}_1$ y $\mathscr{N}_2$ mostradas en la Tabla 3.2, donde: $\mathscr{N}_1$ se aplica con probabilidad p y $\mathscr{N}_2$ con probabilidad $(1-p)$. Se obtiene un número aleatorio r en el intervalo $[0,1]$, si $r < p$ se aplica la función de vecindad $\mathscr{N}_1$, en caso contrario se emplea la función $\mathscr{N}_2$.

Vecindario	Valor de p
$\mathscr{N}_5$	0.50
$\mathscr{N}_6$	0.30
$\mathscr{N}_7$	0.40
$\mathscr{N}_8$	0.70

Tabla 3.2: Funciones probabilísticas de vecindad para ILS.

Las tres combinaciones restantes toman en cuenta el orden de aplicación de cada vecindario y el número de iteraciones permitidas. El proceso de búsqueda de una mejor solución inicia en alguno de los vecindarios $\mathscr{N}_1$ o $\mathscr{N}_2$, cuando el contador de soluciones vecinas CS alcanza un máximo permitido, la búsqueda se realiza en el otro vecindario. Cabe destacar que el contador CS se reinicia cada vez que encuentra una mejor solución. La Tabla 3.3 muestra las condiciones de aplicación de cada vecindario.

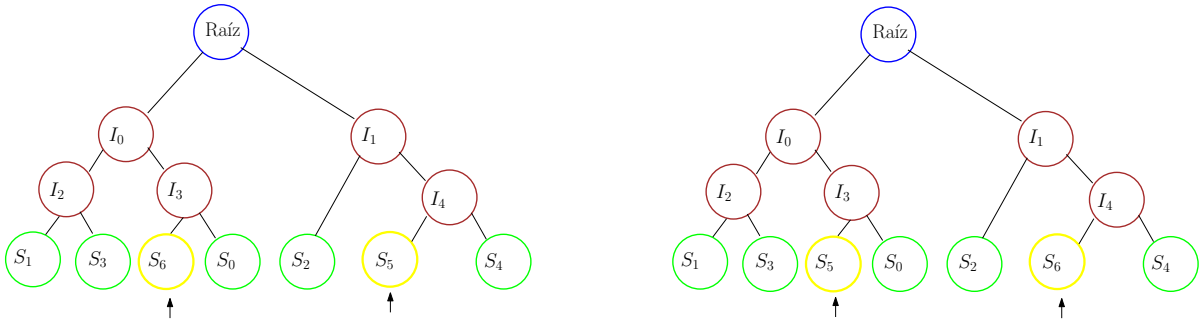
Vecindario	$\mathscr{N}_1$	$\mathscr{N}_2$
$\mathscr{N}_9$	$CS \geq 20$	$CS < 20$
$\mathscr{N}_{10}$	$CS < 20$	$CS \geq 20$
$\mathscr{N}_{11}$	$40 < CS < 60$	$CS \leq 40$ or $CS \geq 60$

Tabla 3.3: Funciones de vecindad combinadas usadas por ILS-MP, las cuales emplean el contador de número de soluciones vecinas que no mejoran CS.

En los algoritmos metaheurísticos es de suma importancia implementar mecanismos que eviten que la búsqueda permanezca atrapada en óptimos locales. Para el algoritmo ILS-MP se implementó una técnica en la que se hace una ligera perturbación a la solución actual. Este método se describe en la siguiente sección.

3.2.4 Perturbación

En la fase de perturbación del algoritmo se seleccionan de forma aleatoria dos hojas del árbol (en la Figura 3.4(a) se muestran en color amarillo) e intercambia su posición (ver Figura 3.4(b)).



(a) Las hojas S_5 y S_6 se seleccionaron como elementos de la perturbación

(b) Intercambio de posiciones de las hojas S_5 y S_6

Figura 3.4: Funcionamiento de la perturbación en ILS.

Después de realizar una perturbación a una solución no siempre es posible aceptar dicho cambio, por lo que hay que definir las condiciones que deberá cumplir para ser aceptado. En la siguiente sección se muestra la condición de aceptación y la condición de paro del algoritmo.

3.2.5 Criterio de aceptación y condición de paro

El asegurar siempre tomar la mejor ruta de búsqueda nos llevó a elegir un criterio de aceptación que solo permite aceptar los cambios que representan un mejor resultado.

Para determinar el mejor momento en el que el algoritmo debe finalizar su ejecución se sigue el siguiente proceso: mediante el contador CS se lleva el registro de las soluciones $s^{*'}$ que no mejoran con respecto al costo de la mejor solución s. Cuando se encuentra una solución $s^{*'}$ que mejora a s el contador CS toma el valor de cero y s se actualiza con el nuevo valor de $s^{*'}$. Cuando CS alcanza el valor de $maxCS$ se cumple la condición de paro y la solución s se reporta como la mejor encontrada por el algoritmo.

3.3 Recocido Simulado

Recocido Simulado (SA) es una técnica estocástica de optimización de propósito general, que ha probado ser una herramienta efectiva para aproximar soluciones globalmente óptimas de muchos problemas de optimización NP-dificiles (Kirkpatrick, Gelatt, y Vecchi, 1983; Cerny, 1985). La implementación de esta metaheurística se basa en el Algoritmo 3.9, el cual es llamado SA-MP en lo sucesivo.

Algoritmo 3.9 SA-MP

Entrada: Función de vecindad $\mathscr{N}$, función de evaluación f, temperatura inicial t_i, temperatura final t_f, numero máximo de soluciones visitadas por temperatura l, función de enfriamiento ω, condición de paro CP
Salida: Mejor solución visitada s^*

```
s_0 ← construir solución inicial
s* ← s_0
t_z ← t_i
mientras CP hacer
  c ← 0
  mientras c < l hacer
    c ← c + 1
    Obtener una solución s ∈ N(s_0)
    Δ ← f(s) − f(s_0)
    genera aleatorio r ∈ [0, 1]
    si (Δ < 0)||(e^(−Δ/t_z)) > r entonces
      s_0 ← s
      si s_0 < s* entonces
        s* ← s_0
      fin si
    fin si
  fin mientras
  z ← z + 1
  t_z ← ω(t_{z−1})
fin mientras
```

En esta tesis se decidió implementar esta metaheurística por diferentes razones. La primera, se busca probar su eficiencia frente al desempeño de ILS-MP. La segunda se debe a que en el estado del arte, solo existe la implementación LVB diseñada por Barker (2003). Esta implementación no

reporta resultados comparativos con ningún tipo de instancia y no menciona la manera en que se sintonizaron los componentes de la metaheurística.

3.3.1 Método de inicialización

Para esta implementación se probaron dos maneras de obtener una solución inicial:

Método de inicialización aleatoria. Se utilizó el método descrito en la Sección 3.2.1.1 con la idea de iniciar con una solución aleatoria que permitiera explorar espacios de búsqueda que quizás mediante un método de construcción estratégica nunca serían explorados.

Método de inicialización voraz. Debido a que este tipo de métodos construyen una solución de manera estratégica se analizó la posibilidad de iniciar la búsqueda partiendo de una buena solución, evitando explorar espacios de búsqueda no prometedores. Para ello se utilizó el método de inicialización descrito en la Sección 3.2.1.2.

Para continuar con la búsqueda de una solución partiendo de la solución inicial, se requiere definir el vecindario de soluciones donde se realizará dicha búsqueda, en la siguiente sección se muestran las funciones de vecindad implementadas para el algoritmo SA-MP.

3.3.2 Funciones de vecindad

Se implementaron once funciones de vecindad para el algoritmo SA-MP, esto se logró combinando las dos funciones básicas $\mathscr{N}_1$ y $\mathscr{N}_2$.

Las primeras tres funciones de vecindad intercalan la aplicación entre $\mathscr{N}_1$ y $\mathscr{N}_2$ dependiendo del valor de la temperatura actual t_z, tal como se describe en la Tabla 3.4, donde $t_m = (t_i - t_f)/2$ y $t_a = (t_i - t_f)/4$.

Vecindario	$\mathscr{N}_1$	$\mathscr{N}_2$
$\mathscr{N}_{12}$	$t_z \leq t_m$	$t_z > t_m$
$\mathscr{N}_{13}$	$t_z > t_m$	$t_z \leq t_m$
$\mathscr{N}_{14}$	$t_m \leq t_z > t_a$	$t_z > t_m$ ó $t_a > t_z \geq t_f$

Tabla 3.4: Funciones de vecindad combinadas usadas por el algoritmo SA-MP, empleando el valor de la temperatura actual t_z como un criterio de combinación.

Otras tres funciones de vecindad fueron implementadas combinando $\mathscr{N}_1$ y $\mathscr{N}_2$ dependiendo del número de soluciones vecinas visitadas c en una temperatura dada t_z. El criterio para combinar estas funciones de vecindad se detallan en la Tabla 3.5, donde l es el número máximo de soluciones visitadas en la temperatura t_z, $HM = l/2$ y $TM = 3l/4$.

Vecindario	$\mathscr{N}_1$	$\mathscr{N}_2$
$\mathscr{N}_{15}$	$c \geq HM$	$c < HM$
$\mathscr{N}_{16}$	$c < HM$	$c \geq HM$
$\mathscr{N}_{17}$	$TM > c \geq HM$	$c < HM$ or $l \geq c \geq TM$

Tabla 3.5: Funciones de vecindad usadas por SA-MP que emplean el número soluciones vecinas visitadas c en una temperatura dada t_z como un criterio de combinación.

Las últimas cinco combinaciones de funciones vecindad son probabilísticas. Aplican el vecindario $\mathscr{N}_1$ con probabilidad p y el vecindario $\mathscr{N}_2$ con una taza $(1.0 - p)$. La Tabla 3.6 muestra estas cinco funciones de vecindad y su valor de probabilidad asociado.

Vecindario	Probabilidad p
$\mathscr{N}_5$	0.50
$\mathscr{N}_{19}$	0.45
$\mathscr{N}_7$	0.40
$\mathscr{N}_{21}$	0.35
$\mathscr{N}_6$	0.30

Tabla 3.6: Funciones de vecindad probabilísticas usadas en la implementación del algoritmo SA-MP.

Durante la búsqueda de una solución generalmente encontraremos soluciones que empeoran en cierta medida la solución actual, por lo que se debe establecer un criterio de aceptación que permita para ciertas circunstancias aceptar soluciones que empeoran, en la siguiente sección se presenta el criterio utilizado para este trabajo.

3.3.3 Criterio de aceptación

Se utilizó el criterio de aceptación basado en el factor de probabilidad Boltzman (Glover y Kochenberger, 2003). Este criterio de aceptación es probabilístico y depende de las siguientes condiciones:

- Si $(f(s) - f(s_0)) \leq 0$ entonces s es aceptada, s_0 es la solución actual y s es la nueva solución vecina.

- Si $(f(s) - f(s_0)) > 0$ entonces s es aceptada si $r < e^{((f(s)-f(s_0))/t_z)}$, donde t_z es la temperatura actual y r un número generado de forma aleatoria en el intervalo $[0, 1]$.

Cada vecino tiene una probabilidad de reemplazar la solución actual. La temperatura t_z se modifica con cada iteración por lo que conforme el algoritmo progresa es menos probable aceptar soluciones que incrementen el costo de la solución actual. El decremento de la temperatura en el algoritmo se realiza empleando un esquema de enfriamiento dado.

3.3.4 Temperatura inicial

El algoritmo de SA-MP requiere definir una temperatura inicial t_i la cual debe permitir una alta probabilidad de aceptar movimientos que empeoren la solución actual (s_0) sin importar la diferencia $f(s) - f(s_0)$, por lo que al inicio el sistema debe tener un alto grado de libertad. En nuestra implementación se considera de gran importancia iniciar con una temperatura acorde a las características de cada instancia por lo que se tomó en consideración el número de taxones y la longitud de su secuencia. La siguiente ecuación se aplicó para calcular la temperatura inicial, esta inicia aceptando aproximadamente el 60 % de los movimientos realizados.

$$t_i = \sqrt[q]{n + k} \tag{3.1}$$

donde q puede tomar valores de: 2, 2.5 o 3

Además de esta forma adaptativa (para cada instancia) de inicializar la temperatura, también se implementó fijar el valor de dicha temperatura a un valor constante γ.

3.3.5 Esquema de enfriamiento

Una vez definida la temperatura inicial, se establece un esquema de enfriamiento considerando que si se hace descender bruscamente la temperatura el resultado final no será el deseado. En esta

investigación se analizaron dos diferentes esquemas esquemas de enfriamiento.

El esquema de enfriamiento geométrico (Laarhoven y Aarts, 1987) reduce la temperatura en cada iteración del algoritmo en función de un factor α, como se describe a continuación.

$$t_z = \alpha t_{z-1} \tag{3.2}$$

donde α es un número real positivo menor que 1. El esquema de enfriamiento adaptativo inspirado en el trabajo realizado por Abramson et al. (1994), reduce la temperatura en cada iteración del ciclo del algoritmo en función de un factor α usando la Ecuación 3.2. Si después de un determinado número de iteraciones ni no se observan mejoras en la solución actual, se incrementa la temperatura en razón de β a fin de incrementar la taza de aceptación y evitar estancamientos en óptimos locales, usando la siguiente relación:

$$t_z = \beta t_{z-1} \tag{3.3}$$

donde $1.0 < \beta < 2.0$

3.3.6 Longitud de la cadena de Markov

La longitud de la cadena de Markov establece el número de soluciones vecinas que serán visitadas durante cada cambio de temperatura. Puede ser calculada asignándole un valor fijo, o bien tomar en cuenta las características de la instancia, como el número de taxones n y la longitud de las secuencias k. Con esta información se establece una longitud constante, la cual se calcula de la siguiente manera.

$$(n + k) \times 50 \tag{3.4}$$

La longitud de la cadena de Markov también puede ser dinámica, basada en el contador de movimientos CS que no mejoran la solución actual. Para calcular la longitud de una cadena de Markov particular se emplea la siguiente expresión:.

$$maxCS = \frac{(n + k)}{d} \tag{3.5}$$

donde $1.0 < d \leq 1.5$

3.3.7 Temperatura final y condición de paro

La temperatura final utilizada para todas las instancias fue de 0.1, debido a que durante la experimentación se observó que en la mayoría de los casos, una vez que se alcanzaba esta temperatura, no había movimientos a mejores soluciones.

Una posible segunda condición de paro toma en cuenta el número de soluciones encontradas que no mejoran la solución actual s_0. El número máximo permitido de soluciones encontradas está dado por MI el cual toma valores superiores a 100.

3.4 Resumen del capítulo

En este capítulo se mostró el estudio realizado sobre las metaheurísticas ILS-MP y SA-MP. Se mostró el algoritmo de descenso como posible heurística embebida del algoritmo ILS-MP. Se estudiaron los métodos de inicialización aleatoria y voraz con el fin de ser aplicados en las dos metaheurísticas propuestas. Se analizaron diferentes funciones de vecindad, las cuales fueron aplicadas a los dos algoritmos. Para el algoritmo SA-MP se propusieron diferentes maneras de ajustar la temperatura inicial, la longitud de la cadena de Markov y la condición de paro. Para este mismo algoritmo se estudiaron diferentes esquemas de enfriamiento. Después de identificar los componentes que integran cada metaheurística, se propusieron diferentes combinaciones de las mismas a fin de identificar aquella que logra mejorar el desempeño de cada una

En el siguiente capítulo se identifica la mejor combinación de componentes para cada una de las metaheurísticas. Se realiza una comparación del desempeño mostrado por los algoritmos ILS-MP y SA-MP, se sintonizan los componentes del SA-MP mediante una técnica que hace uso de arreglos de cobertura (covering arrays), posteriormente se comparan los mejores resultados obtenidos durante la fase experimental con los mejores resultados reportados en la literatura. Por último se muestra una aplicación práctica del problema de MP.

4

Experimentación, resultados y aplicación práctica

El objetivo de la fase experimental, presentada en este capítulo, es validar el desempeño proporcionando por los dos métodos de resolución del problema de MP propuestos en este trabajo: ILS-MP y SA-MP. Adicionalmente, se demuestra la utilidad práctica del algoritmo SA-MP en la resolución de un problema real aplicado al área de epidemiología.

4.1 Introducción

Nuestro proceso de experimentación se dividió en cuatro etapas. En la primera se identifica la mejor combinación de componentes para cada una de las dos metaheurísticas propuestas en el Capítulo 3: ILS-MP y SA-MP. En la segunda etapa se realiza una comparación del desempeño mostrado por ambos algoritmos.

En la tercera etapa se realiza la sintonización de los componentes de SA-MP mediante una técnica que hace uso de arreglos de cobertura (*covering arrays*) y que está inspirada en las pruebas de interacción combinatoria, esto a fin de mejorar la eficiencia de SA-MP. Finalmente se hace una comparación de los resultados obtenidos en esta investigación contra los mejores resultados

reportados por tres diferentes métodos de la literatura. Una vez finalizada la fase experimental, se muestra una aplicación práctica del problema de MP orientada a resolver un problema real del área de epidemiología. Para este ejemplo se emplean los árboles filogenéticos generados por la mejor implementación resultante de este trabajo de investigación (SA-MP).

4.2 Instancias de prueba

En esta investigación, se emplearon dos grupos de instancias de prueba en formato phylip[1]. El primer grupo consta de ocho instancias reales proporcionadas por Goloboff (1997), Luckow y Pimentel (1985) y Platnick (1989), y utilizadas por Andreatta y Ribeiro (2002), Ribeiro y Vianna (2005) y Goëffon (2006). Las características de estas instancias se muestran en la Tabla 4.1

Instancia	Taxones (n)	Caracteres (k)	Mejor Costo	Reportado por:
ANGI	49	59	216	Hydra
CARP	117	110	548	Hydra
ETHE	58	86	372	Hydra
GOLO	77	97	496	Hydra
GRIS	47	93	172	Hydra
ROPA	75	82	325	Hydra
SCHU	113	146	759	Hydra
TENU	56	179	682	Hydra

Tabla 4.1: Primer grupo de instancias.

El segundo grupo consta de diez instancias generadas de forma aleatoria por Ribeiro y Vianna (2003), mostradas en la Tabla 4.2. Estas instancias fueron empleadas por Ribeiro y Vianna (2005). Durante esta investigación ambos grupos de instancias se utilizaron con el fin de evaluar la eficiencia de los algoritmos implementados.

[1]Es un formato de secuencias de nucleótidos y de residuos de aminoácidos que se utiliza frecuentemente como formato de entrada de diferentes programas, en el Apéndice B se puede ver el ejemplo de un archivo de este tipo.

Instancia	Taxones (n)	Caracteres (k)	Mejor Costo	Reportado por:
tst01	45	61	545	Hydra
tst02	47	151	1356	Hydra
tst03	49	111	833	Hydra
tst04	50	97	588	Hydra
tst05	52	75	789	Hydra
tst06	54	65	596	Hydra
tst07	56	143	1270	Hydra
tst08	57	119	853	Hydra
tst09	59	93	1145	Hydra
tst10	60	71	721	Hydra

Tabla 4.2: Segundo grupo de instancias.

4.3 Condiciones experimentales

En todos los experimentos se aplicaron las mismas condiciones experimentales para las dos metaheurísticas implementadas en esta investigación. Los algoritmos ILS-MP y SA-MP (incluyendo las variantes de cada uno) se ejecutaron cien veces con cada una de las instancias. De cada algoritmo surgieron variantes en razón de las diferentes opciones para cada uno de sus componentes.

Todos los algoritmos fueron codificados en lenguaje C y compilados con *gcc* usando la bandera -03. Estos se ejecutaron de forma secuencial en un cluster de 4 procesadores Xeon de seis núcleos X5650 a 2.66 GHZ, 32 Gb de memoria RAM y Sistema Operativo Linux.

Para cada algoritmo surgieron diferentes configuraciones en razón del valor de sus parámetros. Para ILS-MP y SA-MP se combinaron dos métodos de inicialización con once funciones de vecindad generando un total de 44 experimentos (2 algoritmos $\times$ 2 métodos de inicialización $\times$ 11 funciones de vecindad).

4.4 Identificación de los mejores componentes

En esta sección, mediante un proceso de experimentación, se analiza e identifica la mejor combinación de parámetros para cada algoritmo, con respecto al problema de MP y los valores de los parámetros probados. Se muestran los resultados obtenidos de cada experimento, se hace una

comparación de los dos algoritmos implementados en este trabajo en términos de calidad de solución con respecto a los mejores resultados reportados.

4.4.1 Métodos de inicialización

Los métodos de inicialización se aplicaron como componente esencial en los algoritmos ILS-MP y SA-MP. La Tabla 4.3 muestra los resultados obtenidos por los algoritmos de inicialización aleatoria y voraz para cada una de las instancias de prueba. La primera columna muestra el nombre de cada una de las instancias. En la segunda y tercera columna se muestra el menor costo reportado por el método de inicialización voraz y aleatoria. La última columna muestra el mejor costo reportado por estos métodos. En el reglón final se puede ver que el método de inicialización voraz obtuvo un costo promedio de 729.500 superando al método de inicialización aleatoria que obtuvo un promedio de 1068.667. El método de inicialización voraz al proporcionar el menor costo promedio, demuestra mayor eficiencia.

Instancia	Método Voraz	Método Aleatorio	Mejor
ANGI	**229**	394	*229*
CARP	**604**	1443	*604*
ETHE	**393**	815	*393*
GOLO	**538**	844	*538*
GRIS	**184**	414	*184*
ROPA	**351**	677	*351*
SCHU	**829**	2360	*829*
TENU	**726**	1442	*726*
tst01	**582**	684	*582*
tst02	**1419**	1588	*1419*
tst03	**891**	1036	*891*
tst04	**639**	762	*639*
tst05	**833**	988	*833*
tst06	**646**	787	*646*
tst07	**1356**	1548	*1356*
tst08	**921**	1080	*921*
tst09	**1209**	1412	*1209*
tst10	**781**	962	*781*
Promedio	**729.500**	1068.667	729.500

Tabla 4.3: Resultados de los algoritmos de inicialización.

La Figura 4.1 muestra los costos obtenidos por los dos métodos de inicialización. En el eje de las abscisas se muestran las instancias y en el de las ordenadas los mejores costos obtenidos por los métodos de inicialización aleatoria y voraz. Se puede notar que en todas las instancias el método de inicialización voraz proporciona los mejores resultados en relación con el método de inicialización aleatorio.

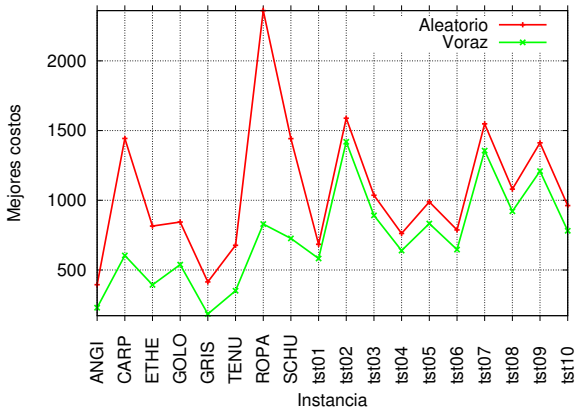


Figura 4.1: Gráfica comparativa de costos obtenidos por los métodos de inicialización.

Estos algoritmos muestran tiempos de ejecución menores a 0.0001 segundos, por lo que no se comparan sus tiempos de ejecución en la Tabla 4.3.

4.4.2 Funciones de vecindad para ILS-MP

Las funciones de vecindad aplicadas para este algoritmo fueron $\mathscr{N}_1$ y $\mathscr{N}_2$ como fueron definidas en la Sección 2.5.2.3 y nueve combinaciones de las mismas. Los mejores costos proporcionados por el algoritmo con las diferentes funciones de vecindad se pueden observar en la Tabla 4.4.

El último renglón de la Tabla 4.4, muestra el promedio de los costos proporcionados por ILS-MP con la función de vecindad correspondiente a cada columna. En esta tabla se puede notar que las funciones de vecindad $\mathscr{N}_{10}$ y $\mathscr{N}_{11}$ obtuvieron el mejor costo promedio de 687.389 y ambas obtuvieron los mejores resultados en 10 instancias. La función de vecindad $\mathscr{N}_1$ obtuvo el peor desempeño con un costo promedio de 707.667 y solo obtuvo el mejor costo en una de las instancias.

Se concluye que para el algoritmo ILS-MP las mejores funciones de vecindad son $\mathscr{N}_{10}$ y $\mathscr{N}_{11}$

ya que ayudan al algoritmo a mejorar su desempeño. Estas funciones de vecindad combinan los vecindarios $\mathscr{N}_1$ y $\mathscr{N}_2$ empleando como criterio de combinación el contador de número de soluciones vecinas que no mejoran CS.

Instancia	$\mathscr{N}_1$	$\mathscr{N}_2$	$\mathscr{N}_3$	$\mathscr{N}_4$	$\mathscr{N}_5$	$\mathscr{N}_6$	$\mathscr{N}_7$	$\mathscr{N}_8$	$\mathscr{N}_9$	$\mathscr{N}_{10}$	$\mathscr{N}_{11}$	Mejor
ANGI	217	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	*216*
CARP	568	556	562	560	549	549	**548**	**548**	551	**548**	549	548
ETHE	374	**372**	374	373	**372**	**372**	**372**	**372**	373	**372**	373	*372*
GOLO	513	500	505	503	**496**	**496**	**496**	**496**	498	**496**	**496**	*496*
GRIS	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	*172*
ROPA	328	327	328	328	**325**	326	**325**	**325**	326	**325**	**325**	*325*
SCHU	789	767	777	772	761	761	761	761	763	760	**759**	759
TENU	687	**682**	684	683	**682**	**682**	**682**	**682**	683	**682**	**682**	*682*
tst01	565	553	558	555	**550**	**550**	551	551	**550**	552	**550**	*550*
tst02	1398	1373	1372	1375	1365	1372	1369	1369	1376	1371	**1364**	*1364*
tst03	866	843	852	849	845	845	842	842	848	**839**	844	*839*
tst04	617	597	604	604	595	600	597	593	600	594	**592**	*592*
tst05	824	800	812	809	803	802	799	801	801	799	**796**	*796*
tst06	626	606	618	615	607	**602**	608	606	605	606	606	*602*
tst07	1330	1294	1305	1292	1291	**1284**	1291	1289	1293	1289	1290	*1284*
tst08	905	875	883	882	873	873	869	870	872	**864**	871	*864*
tst09	1198	1160	1170	1168	**1156**	1162	1161	1162	1164	**1156**	1157	*1156*
tst10	761	731	747	744	733	733	733	733	**730**	732	731	*730*
Promedio	707.667	690.222	696.611	694.444	688.389	688.722	688.444	688.222	690.056	687.389	687.389	

Tabla 4.4: Resultados de ILS-MP con diferentes funciones de vecindad.

4.4.3 Funciones de vecindad para SA-MP

Durante la fase de implementación del algoritmo SA-MP se aplicaron once combinaciones de las funciones de vecindad $\mathscr{N}_1$ y $\mathscr{N}_2$. Los resultados obtenidos por el algoritmo SA-MP con este tipo de funciones de vecindad se pueden observar en la Tabla 4.5.

El último renglón de la Tabla 4.5 muestra el costo promedio proporcionado por el algoritmo SA-MP con cada una de las funciones de vecindad. En este mismo renglón se observa que la función de vecindad $\mathscr{N}_7$ obtuvo el mejor promedio de 681.944 y en trece de las instancias proporcionó el mejor costo. La función de vecindad que mostró el más bajo desempeño fue $\mathscr{N}_{12}$ con un costo promedio de 682.944 y proporcionó el mejor resultado solo en 9 instancias.

Por lo tanto se concluye que la mejor función de vecindad para el algoritmo SA-MP es $\mathscr{N}_7$. Esta es una función de vecindad probabilística donde se aplica el vecindario $\mathscr{N}_1$ con probabilidad $p = 0.4$ y el vecindario $\mathscr{N}_2$ con una tasa de 0.6

Instancia	$\mathscr{N}_5$	$\mathscr{N}_6$	$\mathscr{N}_7$	$\mathscr{N}_{12}$	$\mathscr{N}_{13}$	$\mathscr{N}_{14}$	$\mathscr{N}_{15}$	$\mathscr{N}_{16}$	$\mathscr{N}_{17}$	$\mathscr{N}_{19}$	$\mathscr{N}_{21}$	Mejor
ANGI	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	**216**	216
CARP	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	**548**	548
ETHE	**372**	**372**	**372**	**372**	**372**	**372**	**372**	**372**	**372**	**372**	**372**	372
GOLO	**496**	**496**	**496**	**496**	**496**	**496**	**496**	**496**	**496**	**496**	**496**	496
GRIS	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	**172**	172
ROPA	**325**	**325**	**325**	**325**	**325**	**325**	**325**	**325**	**325**	**325**	**325**	325
SCHU	**759**	**759**	**759**	**759**	**759**	**759**	**759**	**759**	**759**	**759**	**759**	759
TENU	**682**	**682**	**682**	**682**	**682**	**682**	**682**	**682**	**682**	**682**	**682**	682
tst01	546	546	**545**	547	**545**	**545**	547	546	547	547	546	545
tst02	1357	1356	1356	1358	1356	1356	1355	1357	1356	1357	**1354**	1354
tst03	835	**833**	**833**	834	**833**	**833**	836	835	**833**	834	**833**	833
tst04	**588**	589	**588**	**588**	**588**	**588**	589	**588**	589	590	**588**	588
tst05	790	790	**789**	793	**789**	792	**789**	**789**	790	790	**789**	789
tst06	599	598	**596**	600	598	598	598	598	598	**596**	597	596
tst07	1274	1273	1275	1275	1273	**1271**	**1271**	1272	1272	1273	1274	1271
tst08	855	855	855	858	857	856	857	856	**854**	**854**	856	854
tst09	1147	1147	1147	1147	**1146**	**1146**	1148	**1146**	1147	1148	1147	1146
tst10	724	723	721	723	722	721	723	**720**	722	722	722	720
Promedio	682.500	682.222	681.944	682.944	682.055	682.000	682.389	682.056	682.111	682.278	682.000	

Tabla 4.5: Resultados proporcionados por el algoritmo SA-MP con diferentes funciones de vecindad.

4.5 Comparación entre ILS-MP y SA-MP

En la sección anterior se mostraron una serie de experimentos que ayudaron a determinar, para cada metaheurísctica propuesta (ILS-MP y SA-MP), los componentes que les permiten obtener sus mejores resultados. Dichos componentes se encuentran resumidos en la Tabla 4.6.

Algoritmo	Inicialización	Función de vecindad
ILS-MP	voraz	$\mathscr{N}_{11}$
SA-MP	voraz	$\mathscr{N}_7$

Tabla 4.6: Componentes que mostraron mejor desempeño en los diferentes algoritmos.

Con la finalidad de comparar el desempeño de ambas metaheurísticas, estas se ejecutaron nuevamente empleando sus mejores componentes. La Tabla 4.7 presenta los mejores resultados de esas 100 ejecuciones independientes, sobre cada una de las instancias descritas en la Sección 4.2. En el último renglón de la tabla se presenta el costo promedio proporcionado por cada uno de ellos, donde podemos notar que SA-MP obtuvo el mejor costo promedio de 681.778 comparado con 687.388 proporcionado por ILS-MP. En esta misma tabla se puede observar que el algoritmo SA-MP superó los resultados proporcionados por ILS-MP en la mayoría de las instancias.

Para hacer una comparación justa de los tiempos de ejecución, se les aplicó la misma condición

Instancia	ILS-MP	Tiempo	SA-MP	Tiempo	Mejor
ANGI	**216**	54.55	**216**	49.92	216
CARP	**548**	20.35	**548**	48.74	548
ETHE	**372**	19.95	**372**	25.55	372
GOLO	**496**	14.33	**496**	29.10	496
GRIS	**172**	152.84	**172**	21.53	172
ROPA	**325**	49.36	**325**	24.90	325
SCHU	760	38.18	**759**	81.68	759
TENU	**682**	30.20	**682**	156.35	682
tst01	552	22.39	**545**	17.61	545
tst02	1371	28.39	**1354**	41.35	1354
tst03	839	37.47	**833**	30.53	833
tst04	594	40.67	**588**	25.96	588
tst05	799	19.44	**789**	23.06	789
tst06	606	14.07	**596**	20.47	596
tst07	1289	17.41	**1275**	42.85	1275
tst08	864	34.08	**855**	38.56	855
tst09	1156	23.05	**1147**	31.31	1147
tst10	732	18.81	**720**	24.12	720
Promedio	687.388	35.308	681.778	40.755	

Tabla 4.7: Mejores resultados reportados por los algoritmos ILS-MP y SA-MP.

de paro: en el ciclo de búsqueda solo se permite encontrar 100 soluciones vecinas que no logran superar la solución actual. Este estudio se realizó con la instancia tst08. Los tiempos de ejecución se muestran en la Tabla 4.8. En esta tabla se puede observar que el algoritmo SA-MP proporcionó los mejores resultados, reportando menores tiempos de ejecución, menor número de evaluaciones de la función objetivo y menor costo.

Algoritmo	Costo	Tiempo	Evaluaciones
ILS-MP	881	97.36	1365220
SA-MP	867	27.00	357131

Tabla 4.8: Tiempos de ejecución de los algoritmos.

En la Figura 4.2 se muestra el número de evaluaciones a la función objetivo de los algoritmos ILS-MP y SA-MP bajo las condiciones descritas en la tabla de tiempos. En esta gráfica podemos notar que el algoritmo ILS-MP al inicio de la ejecución tiende a obtener mejores resultados que SA-MP, pero queda consistentemente atrapado en un óptimo local. A diferencia del algoritmo SA-MP,

que inicia avanzando de manera lenta hacia mejores soluciones. La condición de aceptación de SA-MP permite aceptar de manera probabilística soluciones que empeoren en cierta medida la solución actual, evitando permanecer atrapado en un óptimo local, proporcionando mejores resultados al final de la ejecución el algoritmo.

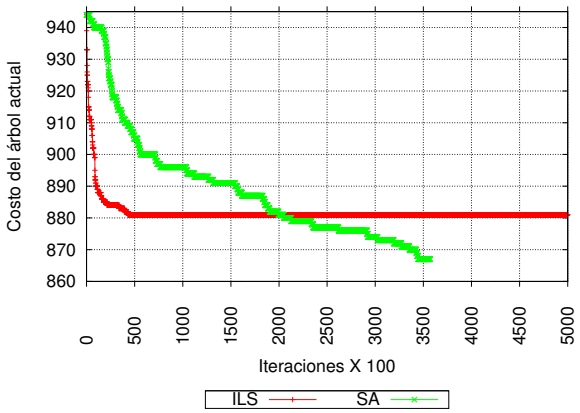


Figura 4.2: Gráfica de convergencia de los algoritmos ILS-MP y SA-MP.

De acuerdo a los resultados presentados, el algoritmo SA-MP muestra mejor desempeño en tiempos de ejecución y calidad de solución, por lo que se decidió mejorarlo probando diferentes combinaciones en los valores de sus parámetros. En la siguiente sección se muestra como se llevó acabo la sintonización.

4.6 Sintonización del algoritmo SA-MP

La sintonización de parámetros es una tarea importante en el diseño de algoritmos. En la literatura se han propuesto diferentes procedimientos con el finalidad de encontrar la combinación de parámetros más conveniente (Adenso-Diaz y Laguna, 2006; Gunawan y Lau, 2011).

Para este trabajo de investigación se utilizó una metodología de sintonización previamente reportada por Gonzalez-Hernandez y Torres-Jimenez (2010), la cual esta basada en pruebas de interacción combinatoria (Cohen, Dalal, Parelius, y Patton, 1996). Esta metodología reduce significativamente el número de pruebas o experimentos necesarios para determinar la mejor configuración de parámetros de un algoritmo. En lugar de hacer pruebas exhaustivas con todas

las combinaciones de parámetros solo se analizan las interacciones de f (o menos) parámetros de entrada, mediante la creación de pruebas de interacción que incluyen al menos una vez todas las f combinaciones entre los parámetros y sus valores.

Para implementar este enfoque se utilizó un Arreglo de Cobertura con Niveles Mezclados (*Mixed Level Covering Array*) que se define como MCA$(N : f, g, (v_1, v_2, ..., v_g))$, *i.e.* una matriz de $N \times g$ con v símbolos, donde $v = \sum_{i=1}^{g} v_i$, que tiene las siguientes propiedades:

- Cada columna i $(1 \leq i \leq g)$ contiene los elementos del conjunto G_i con $|G_i| = v_i$.
- Todas las filas de cada submatriz de tamaño $N \times f$ contienen todas las f-tuplas de valores al menos una vez.

El alfabeto v del MCA puede ser expresado en notación exponencial. Donde la base índica el número de valores diferentes que pueden asignarse a un parámetro en particular. Si dos parámetros diferentes pueden adquirir un valor de tres posibles, entonces se expresaría 3^2.

Para SA-MP se identificaron $g = 6$ parámetros de entrada: método de inicialización de la solución, temperatura inicial, funciones de vecindad, longitud de la cadena de Markov (l), condición de paro, esquema de enfriamiento. Con base en la experimentación preliminar se seleccionaron v_i posibles valores para cada parámetro de entrada que se detallan en la Tabla 4.9.

Si se realizaran todas las combinaciones posibles de parámetros, se tendrían que realizar $7 \times 6 \times 5 \times 5 \times 4 \times 2 = 8400$ experimentos. Para reducir este número de experimentos se construyó el arreglo de cobertura más pequeño posible MCA$(210; 3, 6(2^1, 4^1, 5^2, 6^1, 7^1))$ usando el AM reportado por Rodriguez-Tello y Torres-Jimenez (2010), el cual se puede ver en el Apéndice C. Este arreglo de cobertura puede mapearse fácilmente con los valores mostrados en la Tabla 4.9 al reemplazar cada símbolo de la columna con su valor correspondiente.

En la Tabla 4.9, la primera columna muestra que se tienen dos opciones para obtener una solución inicial: método aleatorio y método voraz.

La temperatura inicial del algoritmo (mostrada en la segunda columna) puede ser calculada de cuatro formas: En la primera opción se asigna una temperatura inicial fija $t_i = 10$, las otras tres opciones toman en cuenta el número de taxones n que contiene la instancia, así como la longitud k

Clave	Met. Ini.	Temp Ini	Func de vec	Long cad Markov	Cond de paro	Func de enf
0	Voraz	10	$\mathscr{N}_7$	fija 10,000	$T_f = 0.01$	Geo $\alpha = 0.99$
1	Aleatorio	$\sqrt[2]{n+k}$	$\mathscr{N}_{21}$	$maxCS = (n+k)/1.3$	$T_f = 0.001$	Geo $\alpha = 0.98$
2		$\sqrt[2.5]{n+k}$	$\mathscr{N}_{13}$	$maxCS = (n+k)/1.5$	$T_f = 0.0001$	Geo $\alpha = 0.97$
3		$\sqrt[3]{n+k}$	$\mathscr{N}_{14}$	$maxCS = (n+k)$	$MI \geq 100$	Adap $\alpha = 0.99\beta = 1.8$
4			$\mathscr{N}_{16}$	$(n+k) \times 50$	$MI \geq 200$	Adap $\alpha = 0.99\beta = 1.9$
5					$MI \geq 300$	Adap $\alpha = 0.98\beta = 1.8$
6						Adap $\alpha = 0.98\beta = 1.9$

Tabla 4.9: Posibles valores de entrada para SA-MP.

de las secuencias. Con estas condiciones se aplican las diferentes fórmulas mostradas en la segunda columna de la Tabla 4.9.

En los primeros experimentos realizados con el algoritmo de SA-MP se analizaron diferentes funciones de vecindad con lo que se pudo identificar cuáles serían las mejores para este nuevo experimento. En la tercera columna de la Tabla 4.9 podemos ver las cinco mejores funciones de vecindad para SA-MP.

El número de iteraciones permitidas para cada cambio de temperatura está definido como la longitud de la cadena de Markov. Para este experimento se propusieron cinco formas de calcular la longitud de la misma. La primera es una longitud constante de 10,000. En las opciones de la 1 a la 3 se identifica el número de soluciones vecinas CS que no logran mejorar la solución actual. Se permite encontrar un número máximo de soluciones $maxCS$ el cual se calcula empleando las fórmulas propuestas en las opciones 1 a 3 de la columna correspondiente a la longitud de la cadena de Markov de la Tabla 4.9. La opción 4 de la misma columna establece un número constante de iteraciones calculado con el número de taxones n de la instancia y la longitud k de sus secuencias.

La condición de paro del algoritmo SA-MP es determinante ya que permite finalizar el algoritmo al encontrar una solución óptima o cercana a ella, a la vez que evita continuar la ejecución del mismo por más tiempo del necesario. Para esta implementación se emplearon 6 opciones para sintonizar la condición de paro, las opciones de la 0 a la 2 detienen la ejecución del algoritmo cuando se alcanza la temperatura final, aplicando para ello diferentes valores de la misma. Las opciones de la 3 a la 5 finalizan el algoritmo tomando en cuenta el número de soluciones encontradas s que no mejoran la solución actual s_0. El número máximo permitido de soluciones que empeoran la solución actual está dado por MI, en la columna 5 de la Tabla 4.9 se pueden notar los diferentes valores que se le

asignaron.

La función de enfriamiento también es posible sintonizarla de diferentes maneras, durante este trabajo se utilizó el esquema de enfriamiento geométrico con diferentes valores de α y se implementó un esquema de enfriamiento adaptativo. En este se identifica el número de movimientos aceptados para cada cambio de temperatura, cuando se detecta que el sistema no muestra cambios hacia mejores resultados, incrementa la temperatura en razón de β, con el fin de permitir movimientos hacia soluciones que empeoran en cierta medida y evitar estancamientos en óptimos locales.

Como resultado de esta sintonización se ejecutaron 210 experimentos a fin de aplicar las mejores combinaciones de los componentes del algoritmo así como los valores de sus parámetros.

El algoritmo se ejecutó 100 veces con los parámetros sugeridos para las 6 instancias de prueba más difíciles pertenecientes al conjunto utilizado para esta investigación. Realizando un total de $100 * 6 * 210 = 126,000$ ejecuciones del algoritmo SA-MP.

En la Tabla 4.10 se muestran los diez experimentos que obtuvieron los mejores resultados. En la primera columna se puede ver el número de experimento, en la segunda columna los parámetros que proporcionaron los mejores resultados. La columna 3 muestra el costo promedio proporcionado por cada combinación de componentes. Por último se muestra el tiempo promedio de ejecución del algoritmo SA-MP con las diferentes combinaciones.

No.	Parametros	Costo promedio	Tiempo Promedio
129	0 2 0 2 3 4	895. 167	365.276
173	0 2 4 2 2 4	897.333	427.366
126	0 1 0 1 2 3	897.500	485.493
109	0 2 2 0 3 4	897.666	132.045
59	1 1 4 0 5 0	897.666	234.38
168	1 3 0 1 3 0	897.833	133.203
12	0 3 4 1 4 3	897.833	171.171
37	1 1 1 0 4 3	897.833	189.714
54	1 2 0 2 0 4	897.833	256.659
209	0 2 0 0 1 3	897.833	368.902

Tabla 4.10: Diez mejores casos de prueba ejecutados por SA-MP.

El experimento 129 con la combinación de parámetros 020234 obtuvo el mejor costo promedio, sin embargo, cabe mencionar la eficiencia del experimento 109 con combinación de parámetros 022034

al mostrar los mejores tiempos de ejecución.

De este experimento concluimos que la mejor combinación de elementos encontrada para el algoritmo SA-MP es la correspondiente al experimento 129, cuyos valores son:

Inicialización: Voraz

Temperatura Inicial: $\sqrt[2,5]{n+k}$

Función de vecindad : $\mathscr{N}_7$

Longitud de la cadena de Markov: ajustada de manera dinámica en base al contador de soluciones CS con un máximo $maxCS = (n+k)/1.5$

Condición de paro: ajustada de manera dinámica en base al contador de soluciones visitadas con un máximo $MI \geq 100$

Esquema de enfriamiento: Adaptativo con valores de $\alpha = 0.99$ y $\beta = 1.9$

En la siguiente sección se comparan los resultados proporcionados por SA-MP con los mejores resultados reportados en el estado del arte.

4.7 Comparación del algoritmo SA-MP con los mejores resultados reportados en el estado del arte

Se realizó una experimentación final en la que se ejecutó la mejor combinación de componentes de SA-MP 100 veces con todo el grupo de instancias de prueba, posteriormente estos resultados fueron comparados con Hydra (Goëffon, 2006), GRASP (Andreatta y Ribeiro, 2002) y TNT (Goloboff, Farris, y Nixon, 2008) considerados los mejores resultados reportados en el estado del arte.

En la Tabla 4.11 se muestran los mejores resultados obtenidos por el algoritmo SA-MP y los mejores resultados reportados en el estado del arte. En esta tabla se puede observar que el algoritmo SA-MP logró igualar los resultados reportados por Hydra considerados los mejores hasta el momento y en 4 instancias logró superarlos. La columna nombrada *Mejor* presenta los mejores resultados obtenidos antes de nuestro método de solución. En la última columna se muestra la diferencia Δ

de SA-MP con respecto a *Mejor*. Los valores de cero indican que se logró alcanzar los resultados reportados por Hydra, y los valores menores a cero indican que SA-MP logró superar dichos resultados. Se puede notar que en todas las instancias el algoritmo SA-MP fue capaz de igualar o superar los mejores resultados.

Instancia	GRASP	Hydra	TNT	Mejor	SA-MP	Δ
ANGI	216	216	216	216	216	0
CARP	548	548	548	548	548	0
ETHE	372	372	372	372	372	0
GOLO	497	496	496	496	496	0
GRIS	172	172	172	172	172	0
ROPA	326	325	325	325	325	0
SCHU	760	759	759	759	759	0
TENU	682	682	682	682	682	0
tst01	551	545	547	545	545	0
tst02	1364	1356	1361	1356	1354	-2
tst03	845	833	840	833	833	0
tst04	598	588	595	588	588	0
tst05	797	789	789	789	789	0
tst06	609	596	601	596	596	0
tst07	1291	1270	1272	1270	1269	-1
tst08	870	852	866	852	852	0
tst09	1152	1145	1146	1145	1144	-1
tst10	733	721	721	721	720	-1
Promedios	687.9444	681.4444	683.7778	681.4444	681.1111	

Tabla 4.11: Comparación de los resultados proporcionados por SA-MP con los reportados en la literatura.

En la siguiente sección se muestra una aplicación práctica del problema de MP.

4.8 Aplicación práctica del problema de MP

La influenza es una enfermedad infecciosa que se presenta en aves y mamíferos, afectando las vías respiratorias. En su fase inicial es semejante a un resfriado y frecuentemente se acompaña de dolor muscular, de garganta, estomacal, de cabeza y en las articulares, así como de fiebre y debilidad general. En algunos casos graves puede complicarse con pulmonía e incluso puede causar la muerte.

Se transmite entre individuos de diferentes maneras, entre las que se pueden mencionar: el contacto con gotas de secreción nasal o saliva de algún individuo afectado, heces de aves afectadas, etc.

La influenza se distribuye en epidemias estacionales; este tipo de virus tiende a mutar causando pandemias mundiales, provocadas por el trasvase[2] de cepas típicas de animales a humanos.

Cuando surge una nueva cepa producto de una mutación, es necesario identificar su procedencia con el fin de determinar la forma en que se pueden contrarrestar o curar los efectos del nuevo virus. Esta información es empleada en la generación de vacunas, las cuales pueden llegar a evitar pandemias mundiales en casos graves.

4.8.1 Planteamiento del problema

En marzo y principios de abril de 2009, un nuevo virus de origen porcino de influenza A(H1N1) y (S-OIV) surgió en México y los Estados Unidos. Durante las primeras semanas, el virus se propagó de humano a humano en todo el mundo en 30 países, lo que provocó que la Organización Mundial de la Salud elevara su alerta de pandemia del nivel 5 al 6, *i.e* epidemia mundial. Este virus tuvo el potencial de convertirse en la primera pandemia de gripe del siglo XXI. En nuestro ejemplo de aplicación práctica del problema de MP, empleamos un árbol filogenético construido con el algoritmo SA-MP, a fin de estimar los orígenes y el desarrollo de la epidemia A(HINI). El objetivo es demostrar el origen del virus definiendo una fecha aproximada de su aparición, así como determinar su ascendencia con el fin de prevenir nuevos brotes.

4.8.2 Datos experimentales

Para este experimento trabajamos con un grupo de 148 secuencias de ADN con una longitud de 1698 caracteres, estas instancias fueron utilizadas por Fraser et al. (2010) y se obtuvieron de las bases de datos de NCBI de su sitio *Influenza virus resource information search and analysis* (Cooper y Morris, 2011). Estas secuencias corresponden a diferentes cepas del virus de influenza entre las

[2]Transferencia del virus de un organismo a otro de diferente especie

que están del tipo humano, porcino y aviar, mismas que fueron previamente alineadas con BLAST (Basic Local Alignment Search Tool), que es un programa informático de alineamiento de secuencias creado por Altschul et al. (1997). Dichas secuencias se encuentran en un archivo de texto en formato philyp. En las Tablas de la D.1 a la D.5 del Apéndice D se muestran los datos correspondientes a estas secuencias, como son lugar, fecha de origen y grupo al que pertenecen.

4.8.3 Construcción del árbol filogenético de virus de influenza

Para la construcción del árbol filogenético se utilizó el archivo que contiene las secuencias de ADN de las diferentes cepas del virus de influenza. Los parámetros empleados por SA-MP para su ejecución fueron aquellos que se determinaron en el proceso de sintonización del algoritmo (Sección 4.6). Después de ejecutar el algoritmo durante 8211.940 segundos con estos parámetros obtuvimos un árbol que nos permite analizar las relaciones entre las diferentes cepas de virus de influenza. Este árbol se puede observar en el apéndice D. Cabe mencionar que las muestras corresponden a cepas presentadas en diferentes lugares del mundo, en diferentes épocas y correspondientes a diferentes especies afectadas (humanos, cerdos y aves). En la figura 4.3 podemos ver de forma sintetizada el árbol resultante proporcionado por el algoritmo SA-MP.

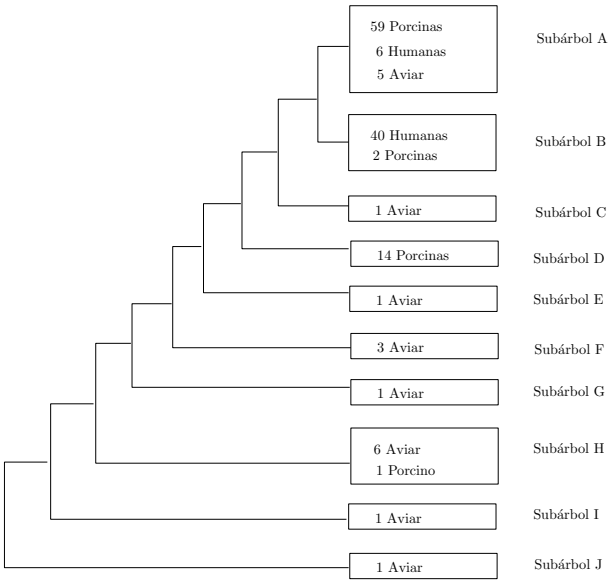


Figura 4.3: Árbol filogenético correspondiente a 148 diferentes cepas del virus de influenza.

En el subárbol A podemos notar la tendencia de agrupamiento de las cepas de influenza correspondientes a las especies porcinas. Este subárbol se puede ver completo en la Figura D.2 del Apéndice D. Estas cepas de especies porcinas tienen un ancestro común con 6 cepas de influenza humana y 5 de influenza del tipo aviar. En el subárbol B (ver Figura D.3 en el Apéndice D) se observa una concentración de 40 cepas del virus de influenza proveniente de humanos, que tienen un ancestro común con 2 cepas de virus porcino. El resto de los subárboles no muestran relación alguna con el virus de influenza humana.

4.8.4 Interpretación de los resultados

El análisis del árbol filogenético generado en este experimento muestra una fuerte tendencia de la cepa humana a relacionarse con la porcina, la mayoría de las cepas del tipo aviar tendieron a aislarse, por lo que podemos deducir que el virus de influenza humana (AH1N1) está fuertemente relacionado con el virus de influenza porcino y con solo 6 cepas del virus de influenza aviar. El resto de las cepas de influenza aviar se aislaron por lo que deducimos que no tuvieron relación con el brote de la cepa de influenza causante de la pandemia mundial del 2009.

Con la finalidad de corroborar nuestras conclusiones se obtuvo el análisis epidemiológico realizado por Nelson et al. (2008) con parte de las secuencias empleadas en nuestro ejemplo. Ellos tomaron en cuenta para su análisis la información adicional proporcionada en las Tablas D.1 a la D.5 del Apéndice D, para cada una de las secuencias de los virus, como el lugar y fecha de aparición de cada una de ellas. Nelson et al. (2008) concluyeron que las muestras correspondientes a la temporada epidémica de influenza estacional 2006-2007 revelan que múltiples cepas virales circularon en los E.U. durante este tiempo. Además, la amplia co-circulación de varias cepas, incluso en localidades geográficamente distantes, permitieron la redistribución entre los virus de influenza A del mismo subtipo. Se consideró también que hubo una redistribución genética de los virus A/H3N2 recobrando la sensibilidad a los medicamentos antivirales, y surgió una nueva variante antigénica A/H1N1 la que se distribuyó en el mundo de forma dominante. En resumen, estos resultados ponen de manifiesto la complejidad de la propagación del virus de influenza A en el tiempo y el espacio, y señalan la necesidad

de una intensificación en la vigilancia global que involucra a todo el genoma de la secuencia de datos.

En un trabajo más reciente Fraser et al. (2010) indican que el virus de influenza humana AH1N1 tiene su origen inmediato en la gripe porcina. Más específicamente, señala que es el producto de un evento de recombinación entre al menos dos variantes de cepas de virus pertenecientes a la influenza de tipo porcina. Una de estas variantes tiene algunos genes que incluyen enlaces a las cepas de H3N2 aviar y humana (tipos de gripe estacional). Estos enlaces se pueden observar en el árbol filogenético construido con SA-MP. Más específicamente en el subárbol A (Figura D.2 del Apéndice B).

Por otro lado Zhou et al. (2011) mencionan la gran posibilidad de que dichos genes entraron a la gripe porcina hace por lo menos 20 años (primer aislamiento). En su conclusión indican una muy baja probabilidad de que este brote del virus tenga una incidencia directa en la aparición del brote actual.

4.9 Resumen del capítulo

En este capítulo se presentaron los diversos experimentos realizados a lo largo de este trabajo. De estos experimentos se pudo concluir que el método de inicialización voraz demostró ser en promedio 58.51 % más eficiente que el método de inicialización aleatoria. Se analizaron también dos diferentes algoritmos metaheurísticos (ILS-MP y SA-MP) para obtener una solución de calidad al problema de MP. Durante la fase de experimentación se logró identificar que el algoritmo SA-MP proporciona resultados en promedio 0.99 % mejores que los entregados por ILS-MP. En relación con los mejores resultados reportados en la literatura por tres métodos de referencia, SA-MP permitió superar estos para cuatro de las instancias (*tst02, tst07, tst09* y *tst10*), e igualarlos en las catorce instancias restantes.

Para finalizar, en este capítulo se mostró un ejemplo del uso de SA-MP para generar árboles filogenéticos aplicados en el área de epidemiología. Con dicho ejemplo se mostró la eficiencia del algoritmo SA-MP para resolver instancias de gran tamaño ($n = 148$) que surgen en casos prácticos. En el siguiente capítulo se presentan las conclusiones de esta tesis y se plantean algunas posibilidades de trabajo futuro.

5

Conclusiones y trabajo futuro

En este trabajo de tesis se estudiaron los diferentes métodos existentes que ofrecen solución al problema de MP. Las metaheurísticas que hemos seleccionado para implementar y dar solución al problema de MP fueron la búsqueda local iterativa, denominado ILS-MP, y el recocido simulado, nombrado SA-MP. Se mostraron los componentes principales estos algoritmos, siendo para ILS-MP: método para generar la solución inicial, función de vecindad, criterio de aceptación de movimientos, mecanismo de perturbación y condición de paro. Los componentes para SA-MP fueron: método para generar la solución inicial, función de vecindad, criterio de aceptación de movimientos, inicialización de la temperatura, criterio para definir la longitud de la cadena de Markov, esquema de enfriamiento y la condición de paro. Además, se utilizó una metodología basada en pruebas de interacción combinatoria que utiliza arreglos de cobertura para sintonizar todos los parámetros de entrada de SA-MP, y así lograr obtener un mejor desempeño del algoritmo propuesto.

Posterior a la etapa experimental y de análisis de resultados, se realizó un estudio sobre la manera en que podría mejorarse la eficiencia de SA-MP, se identificaron diferentes formas de mejorar el desempeño de cada uno de sus componentes, lo cual se señala como trabajo futuro de esta investigación.

5.1 Conclusiones

Con los resultados obtenidos, se concluye que el recocido simulado implementado en esta tesis es un método muy competitivo con respecto a los algoritmos reportados en la literatura. Por lo anterior esta tesis aporta al estado del arte del problema de MP un nuevo algoritmo para su resolución. En la Figura 5.1 se observan todos los resultados concentrados en una gráfica circular. Se realizaron las pruebas con un total de 18 instancias. Es fácil visualizar que en $28\,\%$ de ellas (4 instancias de 18) SA-MP supera las mejores soluciones previamente conocidas y en el $72\,\%$ restante iguala estos resultados.

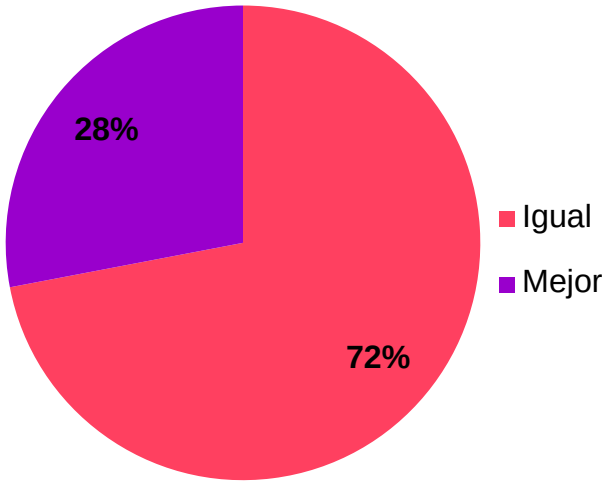


Figura 5.1: Resultados obtenidos por SA-MP sobre el total de instancias estudiadas.

5.2 Trabajo futuro

Una vez concluido el trabajo de tesis podemos proponer algunas posibilidades de trabajo futuro. Para que SA-MP pueda ser mejorado es necesario realizar un análisis adicional de sus componentes clave.

- Solución inicial: Se propone modificar el método de inicialización voraz a fin de obtener una mejor solución. Para lo cual se propone lo siguiente:
  - Durante el proceso de construcción del árbol filogenético se establece un porcentaje de construcción, en el que se tenga un subárbol que pueda ser reordenado mediante al un algoritmo *neighbor-joining* (NJ)
  - Realizar un estudio sobre la influencia de los métodos UPGMA y NJ en la obtención de una solución inicial
  - Analizar la posibilidad de hacer una mezcla de una construcción voraz siguiendo el orden de UPGMA o NJ
- Funciones de vecindad: Desarrollar una nueva función de vecindad basada en NNI empleando un criterio voraz
- Esquemas de enfriamiento: Implementar y experimentar con esquemas de enfriamiento diferentes a los utilizados como: el esquema propuesto por Lundy y Mees (1986), enfriamiento de Cauchy (Szu y Hartley, 1987), por mencionar algunos. Además de implementar esquemas de enfriamiento adaptativos, en donde se tome en cuenta el comportamiento del algoritmo durante la ejecución. Lo anterior con el propósito de realizar un estudio comparativo y observar la influencia que ejerce sobre la solución final
- Criterio de paro: Estudiar diferentes condiciones de paro, en especial aquellas que toman en cuenta el comportamiento de los movimientos aceptados durante la experimentación con el fin de reducir los tiempos de cómputo de SA-MP

A

Análisis y experimentación del algoritmo de descenso

En este apéndice se muestra el estudio y experimentación realizados con algoritmo de descenso .

Algoritmo de descenso

El algoritmo de descenso cuenta con 3 componentes principales los que sintonizados de forma adecuada ayudan al algoritmo a mejorar la calidad de sus resultados. A continuación se detallan los métodos aplicados en esta implementación:

- *Solución inicial:* Para obtener la solución inicial se emplearon los métodos de inicialización voraz y aleatorio descritos en las secciones 3.2.1.1 y 3.2.1.2 .
- *Funciones de vecindad:* Se implementaron las funciones de vecindad NNI ($\mathscr{N}_1$) y SPR ($\mathscr{N}_2$) mostradas en la Sección 2.5.2.3 y dos combinaciones de ellas donde se toma en cuenta el contador CS de soluciones que no mejoran a la solución actual s_0. La Tabla A.1 muestra las funciones de vecindad implementadas para este algoritmo.

Vecindario	$CS < maxCS/2$	$CS \geq maxCS/2$
$\mathscr{N}_3$	$\mathscr{N}_1$	$\mathscr{N}_2$
$\mathscr{N}_4$	$\mathscr{N}_2$	$\mathscr{N}_1$

Tabla A.1: Funciones de vecindad aplicadas en el algoritmo de descenso

- La función de vecindad $\mathscr{N}_3$, inicia la búsqueda con el vecindario $\mathscr{N}_1$, cuando el contador de soluciones (CS) llega al 50 % de su limite permitido $maxCS$, el algoritmo mueve la búsqueda al vecindario $\mathscr{N}_2$.
- La función de vecindad $\mathscr{N}_4$, inicia la búsqueda con el vecindario $\mathscr{N}_2$, cuando el contador de soluciones (CS) llega al 50 % de su limite permitido $maxCS$, el algoritmo mueve la búsqueda al vecindario $\mathscr{N}_1$.

- *Condición de paro:* Dentro del ciclo de búsqueda se maneja un contador de soluciones que no mejoran CS. Cuando se encuentra un solución s que supera a s_0 (la mejor encontrada hasta el momento), esta última se actualiza con la solución s y el contador CS se reinicia. El número máximo de soluciones vecinas permitidas se representa como $maxCS$, y cuando el contador de soluciones CS alcanza este valor *i.e.* $CS = maxCS$, el algoritmo cumple con la condición de paro y la solución s_0 se muestra como la mejor solución encontrada por el algoritmo.

Experimentación y resultados

Durante la fase de implementación del algoritmo de descenso se probó su desempeño con las funciones básicas de vecindad $\mathscr{N}_1$ y $\mathscr{N}_2$ y dos combinaciones de las las mismas ($\mathscr{N}_3$ y $\mathscr{N}_4$). En la Tabla A.2 se muestran los mejores resultados (costos asociados a cada topología de árbol) obtenidos para cada una de estas funciones.

En el último renglón de la Tabla A.2, se muestran los costos promedio obtenidos por el algoritmo con cada una de las funciones de vecindad, donde se puede observar que la función de vecindad $\mathscr{N}_2$ obtuvo el menor costo promedio a la vez que reportó el mejor resultado en 13 instancias.

Instancia	$\mathscr{N}_1$	$\mathscr{N}_2$	$\mathscr{N}_3$	$\mathscr{N}_4$	Mejor
ANGI	**220**	**216**	**216**	**216**	216
CARP	572	562	562	**560**	560
ETHE	376	374	374	**373**	373
GOLO	517	**503**	505	**503**	503
GRIS	173	**172**	**172**	**172**	172
ROPA	335	**328**	**328**	**328**	328
SCHU	788	777	777	**772**	772
TENU	694	**683**	684	**683**	683
tst01	569	**553**	558	555	553
tst02	1403	1376	**1372**	1375	1372
tst03	875	**846**	852	849	846
tst04	625	**602**	604	604	602
tst05	822	**806**	812	809	806
tst06	631	**609**	618	615	609
tst07	1326	1297	1305	**1292**	1292
tst08	907	**881**	883	882	881
tst09	1200	**1164**	1170	1168	1164
tst10	761	**737**	747	744	737
Promedio	710.778	**693.667**	696.611	694.444	692.722

Tabla A.2: Resultados obtenidos por las funciones de vecindad del Algoritmo de Descenso

B

Ejemplo de una instancia

En este apéndice se muestra el ejemplo de una instancia en formato Phylip.

Instancia en formato Phylip

```
10 35

Nicotiana       GTTGACAATTTAGCAATGCTTTTTGAGCCAAACCA
Galphimia       GTTGACAGTTTATTAATGCTCTTTGACTGGAACCA
Oenothera       GTTGACAACCTATTAATGCTTTTTGAACGAAACCA
Victoria        GTTGACAATTTATCAGTGCTTCTTGATTGAAACCC
Cypirapea       ??????????TATCAATGCTTTTTGACTGAAACCA
Barclaya        GTTGACAATTTATCAGTGCTTCTTGATTGAAACCC
Petunia         GTTGACAATTTAGCAATGCTTTTTGAGCCAAGCCA
Lycopersi       GTTGACAATTTAGCAATGCTTTTTGAGCCAAACCA
Adoxa           GCTGACAATTTATCAATGCTTTTTGACTGAAACCA
Lamiumpur       GTTGACAAGTTAGTAATGCTTTCTGAACGAAACCA
```

C

Combinaciones de componentes de SA-MP

En este apéndice se muestran las combinaciones resultantes de la sintonización del algoritmo SA-MP empleando una metodología previamente reportada en (Gonzalez-Hernandez y Torres-Jimenez, 2010), la cual está basada en pruebas de interacción combinatoria (Cohen et al., 1996).

1-44	45-88	89-132	133-176	177-210
1 1 3 2 0 1	1 1 0 4 4 6	1 0 3 0 4 5	1 3 4 1 3 1	1 2 0 1 5 2
0 2 3 3 1 0	1 0 2 4 2 6	1 0 3 2 2 2	0 1 2 3 2 4	0 0 2 3 3 5
1 0 0 4 1 4	1 3 4 4 4 4	0 0 2 3 5 2	1 0 2 2 5 3	1 2 2 1 3 6
0 1 2 2 0 2	1 1 2 1 1 0	1 1 3 2 2 0	1 0 4 3 2 3	0 2 0 3 2 2
0 3 1 3 5 0	0 0 1 4 0 1	1 2 1 1 1 2	1 0 4 0 4 1	1 1 0 2 1 2
1 0 2 0 0 6	1 2 1 3 2 1	0 1 0 0 4 4	1 3 0 2 2 6	0 0 3 4 5 0
1 3 4 4 1 1	1 1 2 0 0 3	0 0 1 2 4 4	0 3 2 0 0 1	1 2 1 1 5 1
1 2 1 3 5 3	0 3 4 4 3 3	1 0 0 3 5 6	0 3 1 2 0 6	1 2 3 0 3 1
0 0 3 0 0 2	0 2 4 0 5 3	0 3 3 3 2 6	0 2 0 0 5 1	1 2 4 2 4 2
1 0 3 3 1 1	1 2 0 2 0 4	1 2 4 3 0 1	1 1 3 3 3 2	0 0 0 0 2 4
0 2 2 1 1 1	0 3 1 2 3 2	1 3 1 4 4 2	1 1 4 0 1 5	0 0 0 3 2 0
0 3 4 1 4 3	0 3 2 4 2 2	1 1 4 3 3 6	1 2 4 4 1 6	1 0 2 4 3 0
1 3 0 0 3 2	0 3 4 1 2 0	1 3 3 1 0 0	1 1 2 4 1 5	0 2 2 1 2 5
0 2 1 3 3 0	0 0 3 4 5 1	0 3 1 0 1 4	0 3 3 2 4 1	1 1 4 0 2 2
1 2 1 4 1 3	1 1 4 0 5 0	0 0 1 1 5 6	0 0 1 1 4 0	1 1 4 4 0 3
1 2 4 2 3 0	1 0 0 3 4 2	1 1 4 4 5 2	0 3 1 2 2 3	0 0 0 2 4 3
1 0 4 2 4 6	1 2 0 4 0 2	1 3 2 0 5 5	1 2 2 1 0 5	1 2 3 3 5 5
1 3 1 4 3 6	1 3 0 1 4 5	0 2 4 4 4 5	1 3 2 2 4 0	1 2 0 2 1 1
0 0 1 1 2 2	0 0 1 2 3 1	1 0 4 1 5 5	1 2 3 3 4 4	0 1 3 3 1 3
1 2 1 0 0 0	0 0 0 1 0 1	0 0 1 0 5 2	1 1 3 0 3 0	0 3 3 0 2 3
1 0 3 1 1 5	0 1 4 3 4 0	0 2 2 0 3 4	0 2 0 3 0 6	1 1 2 3 5 1
1 2 2 1 5 0	1 0 2 1 4 4	0 1 1 3 4 5	0 0 3 1 3 4	1 2 2 2 5 4
1 2 2 3 1 4	0 0 0 0 1 0	0 1 3 4 0 6	0 2 3 2 0 3	0 1 4 1 0 6
1 2 2 3 4 3	0 1 2 2 4 5	0 1 1 0 2 6	1 2 1 0 4 6	1 1 1 4 5 4
1 1 1 0 1 1	0 2 3 2 2 5	0 3 4 3 1 2	0 1 3 1 5 3	1 3 0 3 3 4
0 1 3 1 0 4	0 2 3 4 1 2	1 2 4 0 3 5	0 1 1 2 1 0	1 3 0 1 5 4
1 0 4 2 1 3	1 0 3 4 4 3	1 0 4 2 0 0	0 3 3 4 3 5	0 2 4 1 2 6
0 1 3 1 2 1	1 0 2 3 0 0	1 1 0 2 5 0	0 1 4 2 0 5	1 3 4 4 1 0
0 2 3 2 3 6	1 3 0 0 0 5	1 0 1 4 2 0	1 0 1 1 0 3	1 1 1 3 0 2
0 2 2 0 2 0	0 2 2 1 3 2	0 2 0 3 1 5	1 3 0 4 5 3	1 2 4 4 3 4
0 2 2 4 5 6	1 1 0 4 3 1	1 2 3 1 3 3	0 3 3 4 2 4	1 0 1 3 1 6
0 3 3 0 5 4	0 0 1 0 2 5	0 1 1 1 3 5	0 0 0 2 3 5	0 3 2 4 0 4
1 1 2 2 3 3	0 2 2 4 4 1	1 2 2 0 4 2	1 3 3 2 5 2	0 2 0 0 1 3
1 3 0 1 1 6	1 3 3 0 4 0	1 0 1 4 0 5	1 2 1 1 2 4	1 2 0 3 3 3
0 3 2 3 4 6	0 3 4 1 0 2	0 3 1 1 4 1	0 3 4 2 5 1	
0 1 2 2 1 6	0 1 4 1 1 4	0 2 2 4 2 3	1 3 0 1 3 0	
1 1 1 0 4 3	1 2 1 3 0 4	0 3 2 1 1 3	1 0 1 0 3 3	
0 3 3 2 1 4	0 2 0 4 4 0	0 1 0 1 2 3	1 1 0 3 4 1	
1 1 3 3 0 5	0 1 3 2 5 6	1 1 0 4 2 5	0 2 1 2 5 5	
1 3 1 2 1 5	0 1 3 1 4 2	0 0 0 0 3 6	0 2 3 1 4 6	
1 1 0 4 5 5	0 3 0 3 0 3	0 0 4 3 5 4	0 2 4 2 2 4	
1 0 4 0 0 4	1 1 1 2 3 4	1 0 3 0 1 6	0 1 0 4 0 0	
1 0 2 2 2 1	1 0 4 4 3 2	1 3 4 3 2 5	1 3 0 0 2 1	
0 3 2 3 3 1	0 1 4 4 2 1	0 0 2 0 1 2	1 3 4 0 5 6	

Tabla C.1: Combinaciones de componentes y parámetros de SA-MP representadas por un MCA$(210 : 3, 6, (2^1 4^1 5^2 6^1 7^1))$.

D

Información relacionada al caso de estudio de la influenza

En este apéndice se muestran los árboles filogenéticos generados para el ejemplo práctico de aplicación del problema de MP, así como las tablas con la información referente a las secuencias de virus utilizadas para este mismo ejemplo.

Árbol filogenético para diferentes cepas del virus de influeza

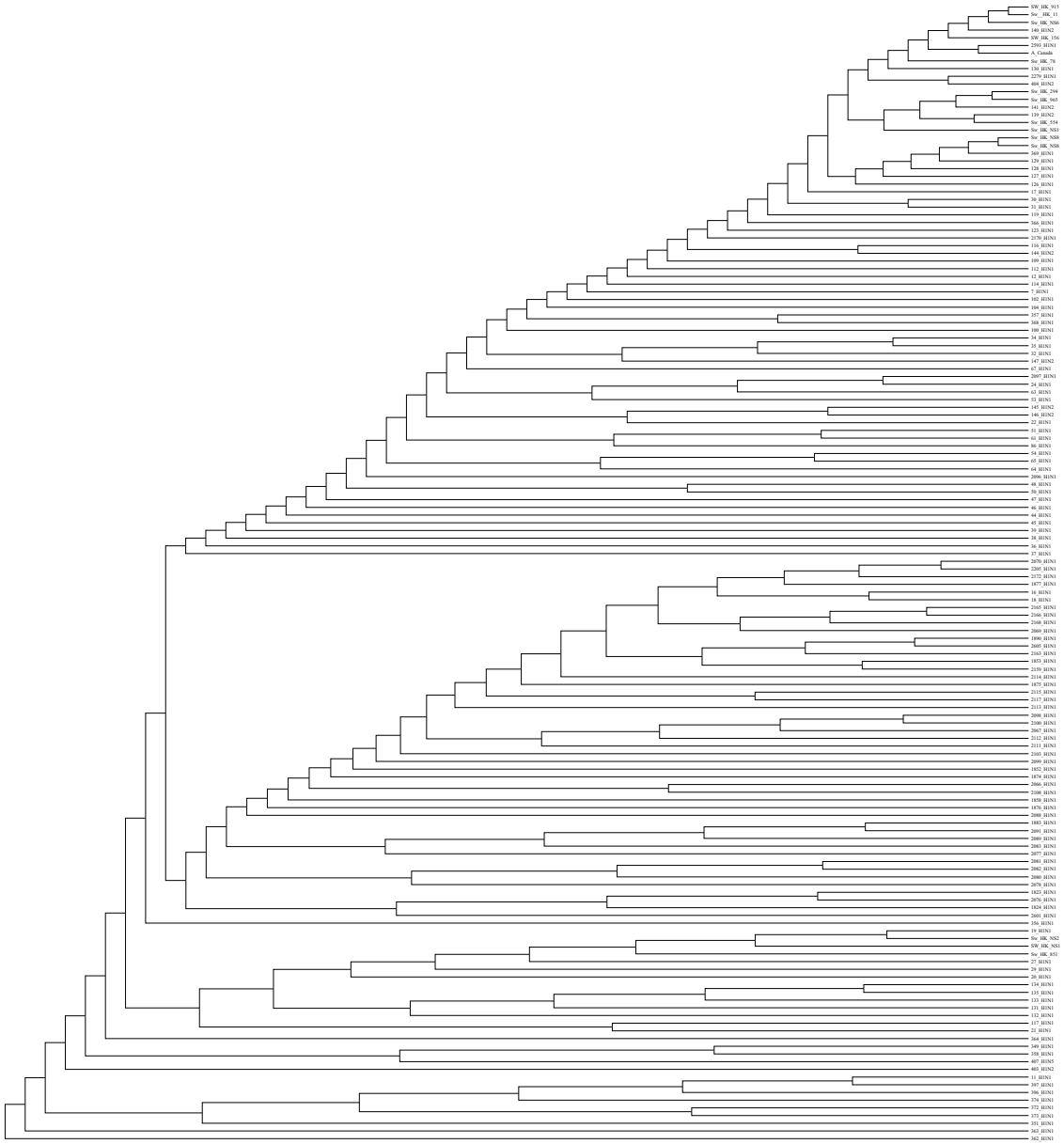


Figura D.1: Árbol filogenético completo de cepas de virus de influenza.

Subárbol filogenético A para diferentes cepas del virus de influeza

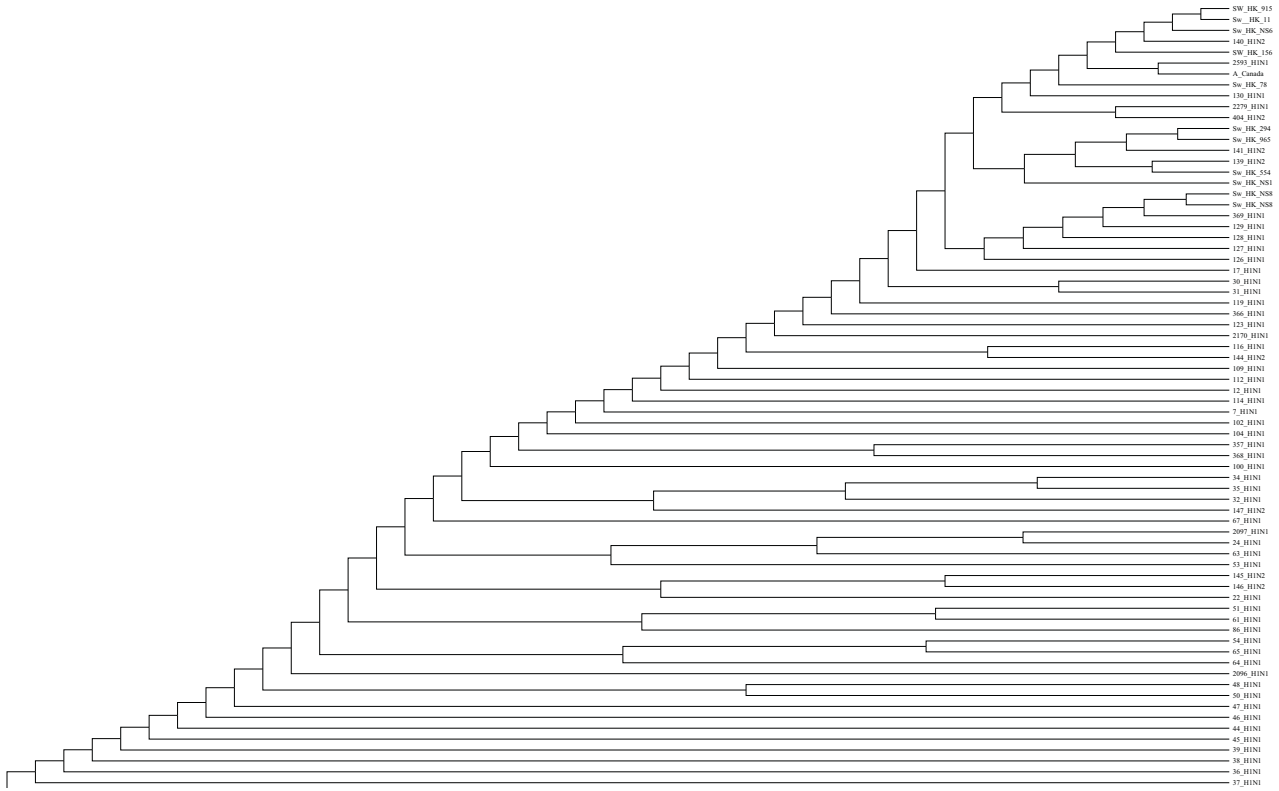


Figura D.2: Subárbol A tomado del árbol filogenético de la Figura B.1.

Subárbol filogenético B para diferentes cepas del virus de influeza

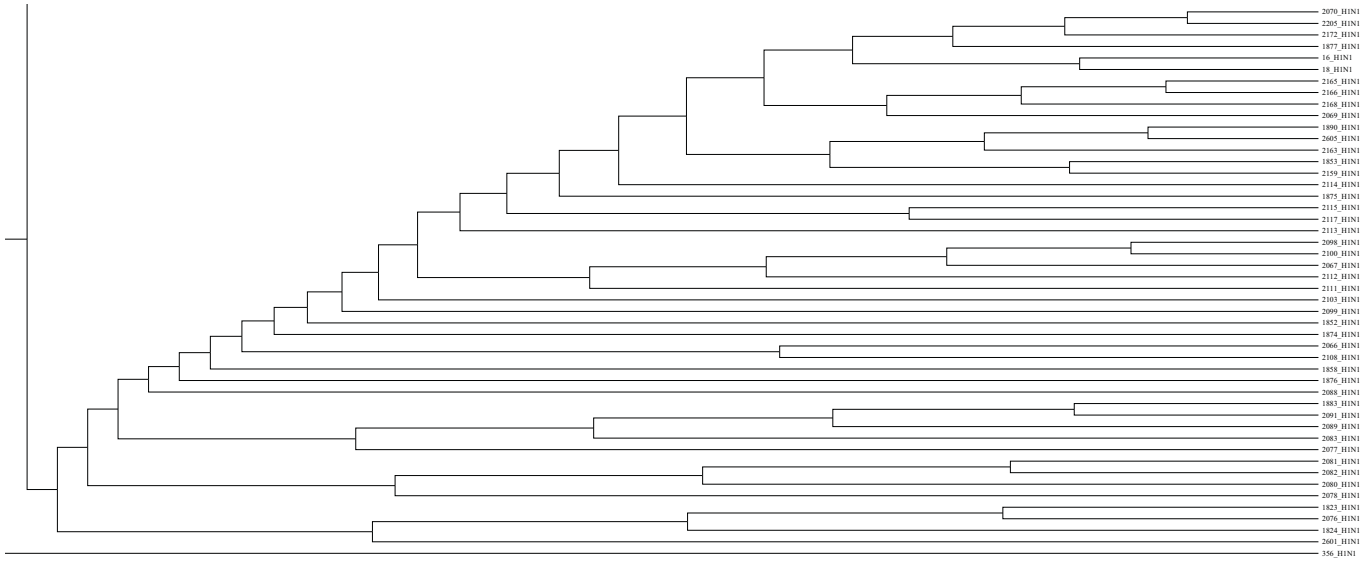


Figura D.3: Subárbol B tomado del árbol filogenético de la Figura B.1.

Tablas con información sobre los virus de influenza como lugar y fecha de aparición

Nombre	Tipo	Origen	Fecha
356_H1N1	Aviar	Italy	2007
364_H1N1	Aviar		1987
358_H1N1	Aviar	Pigeon	1981
407_H1N5	Aviar	Newyork	1978
349_H1N1	Aviar	Alb	1979
397_H1N1	Aviar	Mallard	2003
403_H1N2	Aviar	Newyork	1982
373_H1N1	Aviar	New York	1994
396_H1N1	Aviar	Mallard	2002
372_H1N1	Aviar	Quail	1993
363_H1N1	Aviar	sd	1986
374_H1N1	Aviar	nj	1995
351_H1N1	Aviar	Bluewingedteal	1992
362_H1N1	Aviar	Bluewingedteal	1986
404_H1N2	Aviar	nc	2001
369_H1N1	Aviar	la	1992
366_H1N1	Aviar	nc	1988
368_H1N1	Aviar	pa	1991
357_H1N1	Aviar	ks	1980

Tabla D.1: Información sobre las secuencias del virus de influenza de origen aviar originadas durante los años 1978-2007

Nombre	Tipo	Origen	Fecha
2601_$H1N1$	Humano	Wilsonsmith	1933
2038_$H1N1$	Humano	Puertorico	1934
2075_$H1N1$	Humano	Alaska	1935
1824_$H1N1$	Humano	Melbourne	1935
2076_$H1N1$	Humano	Phila	1935
1823_$H1N1$	Humano	Henry	1936
2077_$H1N1$	Humano	Hickox	1940
2078_$H1N1$	Humano	Bel	1942
2081_$H1N1$	Humano	Weiss	1943
2080_$H1N1$	Humano	Marton	1943
2082_$H1N1$	Humano	Huston	1945
2083_$H1N1$	Humano	Fortmonmouth	1947
2088_$H1N1$	Humano	Albany	1948
1876_$H1N1$	Humano	Roma	1949
2089_$H1N1$	Humano	Fortworth	1950
2091_$H1N1$	Humano	Albany	1951
1883_$H1N1$	Humano	Malaysia	1954
2094_$H1N1$	Humano	Denver	1957
2595_$H1N1$	Humano	wsn	1961
2096_$H1N1$	Humano	Newjersey	1976
2097_$H1N1$	Humano	Wisconsin	1976
2066_$H1N1$	Humano	Ussr	1977
1874_$H1N1$	Humano	Hongkong	1977
1858_$H1N1$	Humano	Tientsin	1977
2099_$H1N1$	Humano	Arizona	1978
2103_$H1N1$	Humano	Lackland	1978
1852_$H1N1$	Humano	Brazil	1978
2100_$H1N1$	Humano	California	1978
2098_$H1N1$	Humano	Albany	1978
2108_$H1N1$	Humano	Memphis	1978
2067_$H1N1$	Humano	Ussr	1979
2111_$H1N1$	Humano	Albany	1979
2112_$H1N1$	Humano	Memphis	1979
1875_$H1N1$	Humano	India	1980
2114_$H1N1$	Humano	Memphis	1980
2113_$H1N1$	Humano	Maryland	1980
2115_$H1N1$	Humano	Baylor	1981
2117_$H1N1$	Humano	Baylor	1982
2605_$H1N1$	Humano	Christshospital	1982
1890_$H1N1$	Humano	Newzealand	1983
1853_$H1N1$	Humano	Chile	1983
2159_$H1N1$	Humano	Memphis	1983
2163_$H1N1$	Humano	Memphis	1984
2069_$H1N1$	Humano	Taiwan	1986
2165_$H1N1$	Humano	Newyork	1986
2166_$H1N1$	Humano	Texas	1986
2168_$H1N1$	Humano	Memphis	1987
2170_$H1N1$	Humano	Ohio	1988
1877_$H1N1$	Humano	Siena	1989
2172_$H1N1$	Humano	Texas	1991
2202_$H1N1$	Humano	Charlottesville	1995
1859_$H1N1$	Humano	Beijing	1995
2205_$H1N1$	Humano	Memphis	1996
2070_$H1N1$	Humano	tw	1996
1873_$H1N1$	Humano	Nanchang	1996
2071_$H1N1$	Humano	Tw	1997
2072_$H1N1$	Humano	tw	1999
1884_$H1N1$	Humano	Newcaledonia	1999

Tabla D.2: Información sobre las secuencias del virus de influenza de origen humano originadas durante los años 1933-1999

Nombre	Tipo	Origen	Fecha
2217_H1N1	Humano	Memphis	2000
1935_H1N1	Humano	Southcanterbury	2000
1943_H1N1	Humano	Wellington	2000
1899_H1N1	Humano	Auckland	2000
1933_H1N1	Humano	Dunedin	2000
2223_H1N1	Humano	Memphis	2001
1986_H1N1	Humano	Canterbury	2001
2023_H1N1	Humano	Westcoast	2001
2004_H1N1	Humano	Southcanterbury	2001
2025_H1N1	Humano	Auckland	2005
2026_H1N1	Humano	Otago	2005
2279_H1N1	Humano	Iowa	2005
2068_H1N1	Humano	Stpetersburg	2006
2607_H1N1	Humano	England	2006
2282_H1N1	Humano	Michigan	2006
2280_H1N1	Humano	Kentucky	2006
2428_H1N1	Humano	Mississippi	2007
2592_H1N1	Humano	Washington	2007
2610_H1N1	Humano	England	2007
2557_H1N1	Humano	Vermont	2007
2462_H1N1	Humano	Oklahoma	2007
2319_H1N1	Humano	Florida	2007
2490_H1N1	Humano	Tennessee	2007
2288_H1N1	Humano	Alabama	2007
2360_H1N1	Humano	Kansas	2007
2581_H1N1	Humano	Virginia	2007
2437_H1N1	Humano	Northcarolina	2007
2302_H1N1	Humano	California	2007
2373_H1N1	Humano	Kentucky	2007
2450_H1N1	Humano	Ohio	2007
2431_H1N1	Humano	Newyork	2007
2471_H1N1	Humano	Oregon	2007
1880_H1N1	Humano	Japan	2008
2590_H1N1	Humano	Usa	2008
2037_H1N1	Humano	Managua	2008
2613_H1N1	Humano	England	2008
2593_H1N1	Humano	California	2009
$A/Canada-ON/RV1527$	Humano	Canada	2009

Tabla D.3: Información sobre las secuencias del virus de influenza de origen humano originadas durante los años 2000-2009

Nombre	Tipo	Origen	Fecha
36_H1N1	Porcino	Ohio	1935
37_H1N1	Porcino	Jamesburg	1942
38_H1N1	Porcino	Wisconsin	1957
39_H1N1	Porcino	Wisconsin	1961
44_H1N1	Porcino	Wisconsin	1966
45_H1N1	Porcino	Wisconsin	1967
46_H1N1	Porcino	Wisconsin	1970
47_H1N1	Porcino	Wisconsin	1971
48_H1N1	Porcino	Illinois	1975
50_H1N1	Porcino	Tennessee	1975
61_H1N1	Porcino	Tennessee	1976
51_H1N1	Porcino	Iowa	1976
63_H1N1	Porcino	Wisconsin	1976
53_H1N1	Porcino	Kentucky	1976
54_H1N1	Porcino	Minnesota	1976
86_H1N1	Porcino	Tennessee	1977
67_H1N1	Porcino	Nebraska	1977
64_H1N1	Porcino	Arizona	1977
22_H1N1	Porcino	Niigata	1977
65_H1N1	Porcino	Iowa	1977
100_H1N1	Porcino	Tennessee	1978
23_H1N1	Porcino	Kyoto	1979
102_H1N1	Porcino	Minnesota	1979
144_H1N2	Porcino	Ehime	1980
104_H1N1	Porcino	Wisconsin	1980
24_H1N1	Porcino	Hokkaido	1981
7_H1N1	Porcino	Ontario	1981
21_H1N1	Porcino	France	1985
109_H1N1	Porcino	Iowa	1985
112_H1N1	Porcino	Iowa	1986
117_H1N1	Porcino	Virginia	1987
114_H1N1	Porcino	Iowa	1987
116_H1N1	Porcino	Kansas	1987
123_H1N1	Porcino	Wisconsin	1988
119_H1N1	Porcino	Iowa	1988
126_H1N1	Porcino	Memphis	1990
127_H1N1	Porcino	California	1991
128_H1N1	Porcino	Iowa	1991
129_H1N1	Porcino	Maryland	1991
131_H1N1	Porcino	England	1992
20_H1N1	Porcino	Denmark	1993
132_H1N1	Porcino	England	1993
$Sw/HK/103$	Porcino		1993
133_H1N1	Porcino	England	1994
145_H1N2	Porcino	Saitama	1996
134_H1N1	Porcino	England	1997
135_H1N1	Porcino	England	1998

Tabla D.4: Información sobre las secuencias del virus de influenza de origen porcino, originadas durante los años 1935-1998

Nombre	Tipo	Origen	Fecha
30_H1N1	Porcino	Ratchaburi	2000
$Sw/HK/8512$	Porcino		2001
$Sw/HK/9656$	Porcino		2001
$Sw/HK/NS837$	Porcino		2001
$Sw/HK/NS857$	Porcino		2001
$Sw/HK/NS1659$	Porcino		2001
11_H1N1	Porcino	Saskatchewan	2002
$Sw/HK/NS623$	Porcino		2002
27_H1N1	Porcino	Spain	2003
$Sw/HK/78$	Porcino		2003
31_H1N1	Porcino	Ratchaburi	2003
$Sw/HK/554$	Porcino		2003
12_H1N1	Porcino	Alberta	2003
16_H1N1	Porcino	Tianjin	2004
137_H1N2	Porcino	Ontario	2004
29_H1N1	Porcino	Spain	2004
32_H1N1	Porcino	Chonburi	2004
$SW/HK/915$	Porcino		2004
139_H1N2	Porcino	Zhejiang	2004
142_H1N2	Porcino	Cloppenburg	2005
143_H1N2	Porcino	Doetlingen	2005
147_H1N2	Porcino	Saraburi	2005
35_H1N1	Porcino	Chonburi	2005
34_H1N1	Porcino	Chachoengsao	2005
$SW/HK/1562$	Porcino		2005
17_H1N1	Porcino	Shanghai	2005
18_H1N1	Porcino	Henan	2006
146_H1N2	Porcino	Miyazaki	2006
$Sw//HK/1110$	Porcino		2006
140_H1N2	Porcino	Guangxi	2006
19_H1N1	Porcino	Zhejiang	2007
$SW/HK/NS1179$	Porcino		2007
130_H1N1	Porcino	Oh	2007
141_H1N2	Porcino	Shanghai	2007
$Sw/HK/NS29$	Porcino		2009
$Sw/HK/294$	Porcino		2009

Tabla D.5: Información sobre las secuencias del virus de influenza de origen porcino, originadas durante los años 2000-2009

Bibliografía

D. A. Abramson, H. Dang, y M. Krisnamoorthy. Cooling schedules for simulated annealing based scheduling algorithms. In *Proceeding of the 17th Australian Computer Science Conference*, pages 541–550, 1994.

B. Adenso-Diaz y M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *OPERATIONS RESEARCH-BALTIMORE THEN LINTHICUM-*, 54(1):99, 2006.

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

S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, y D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

A. Andreatta y C. C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.

D. Barker. Lvb: Parsimony and simulated annealing in the search for phylogenetic trees. *Bioinformatics Applications Note*, 20(2):274–275, August 2003.

V. Cerny. A thermodynamic approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

D. M. Cohen, S. R. Dalal, J. Parelius, y G. C. Patton. The combinatorial design approach to automatic test generation. *IEEE Software*, 13:83–88, 1996.

P. Cooper y R. Morris. Ncbi news, august 2011. 2011.

Ch. Darwin. *On the Origin of Species*. John Murray (London), 1859.

K. Dose. Ernst haeckel's concept of an evolutionary origin of life. *Biosystems*, 13(4):253–258, 1981.

A.W.F. Edwards y C.L.L. Sforza. The reconstruction of evolution. *Heredity*, 18, 1963.

W. Fitch y E. Margoliash. Construction of phylogenetic trees. *Sience*, 155(3760):279–284, 1967.

P. G. Foster y D. A. Hickey. Compositional bias may affect both dna-based and protein-based phylogenetic reconstructions. *Journal of Molecular Evolution*, 48(3):284–290, 1999.

L. R. Foulds y R. L. Graham. The Steiner tree problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

C. Fraser, C. A. Donnelly, S. Cauchemez, y W. P. Hanage. Pandemic potential of a strain of influenza a (h1n1): Early findings, 2010.

G. Ganapathy, V. Ramachandran, y T. Warnow. On contract-and-refine transformations between phylogenetic trees. In *Proceedings of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 900–909, 2004. ISBN 0-89871-558-X.

M. Garey y D. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

F. Glover y G.A. Kochenberger. *Handbook of metaheuristics*. Springer, 2003.

F. Glover y M. Laguna. *Tabu Search*, volume 3 of *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, d-z edition, July-August 1999.

A. Goëffon. *Nouvelles Heuristiques de Voisinage et Mémétiques pour le Probléme Maximum de Parcimonie*. PhD thesis, Université D'Angers, november 2006.

A. Goëffon, J. M. Richer, y J. K. Hao. Local search for the maximum parsimony problem. *Lecture Notes in Computer Science*, 3612:678–683, 2005.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, volume 18 of *Artificial Intelligence*. Addison-Wesley, 1989. ISBN 978-0201157673.

P. A. Goloboff, J. S. Farris, y K. C. Nixon. Tnt, a free program for phylogenetic analysis. *Cladistics*, 24(5):774–786, 2008.

PA Goloboff. Nona, version 1.6, 1997.

L. Gonzalez-Hernandez y J. Torres-Jimenez. MiTS: A new approach of tabu search for constructing mixed covering arrays. *Lecture Notes in Artificial Intelligence*, 6438:382–392, 2010.

A. Gunawan y H. Lau. Fine-tuning algorithm parameters using the design of experiments approach. *Learning and Intelligent Optimization*, pages 278–292, 2011.

D. M. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. University of Cambridge, 1st. edition, 1997. ISBN 0-521-58519-8.

E. Haeckel. *Generelle Morphologie der Organismen*, volume bd. 1. Georg Riemer (Berlin), 1866.

P. Hansen y N. Mladenović. An introduction to variable neighborhood search. In *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer, 1999.

M. D. Hendy y D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59(2):277–290, 1982.

W. Hennig. *Phylogenetic Systematic*. Phylogeny. University of Illinois, 1966. ISBN 0-252-06814-9.

J. H. Holland. *Adaption in Natural and Artificial Systems*. Complex Adaptive Systems. The University of Michigan Press, 1975. ISBN 0-262-58111-6.

H. H. Hoos y T. Stützle. *Stochastic Local Search, Foundations and Applications*. Morgan Kaufmann Publishers, 2005.

S. Kirkpatrick, C. D. Gelatt, y M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983.

A. Kluge y J. Farris. Quantitative phyletics and the evolution of anurans. *Systematic Zoology*, 18 (1):1–32, 1969.

P. J. M. Laarhoven y E. H. L. Aarts. *Simulated annealing: Theory and applications*. Kluwer Academic Publishers, 1987.

A. H. Land y A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

P. O. Lewis. A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data. *Molecular Biology and Evolution*, 15(3):277–283, 1998.

M. Luckow y R.A. Pimentel. An empirical comparison of numerical wagner computer programs. *Cladistics*, 1(1):47–66, 1985.

M. Lundy y A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34: 111–124, 1986.

H. Matsuda. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. *Proceedings of 1st Pacific Symposium on Biocomputing*, 3(1):512–523, 1996.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, y A. H. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

N. Mladenović y P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24 (11):1097–1100, 1997.

P. Moscato. Memetic algorithms: A short introduction. In *New Ideas in Optimization*, pages 219–234. McGraw-Hill, London, 1999.

Y. Murakami y S. Jones. Sharp2: protein–protein interaction predictions using patch analysis. *Bioinformatics*, 22(14):1794–1795, 2006.

M. I. Nelson, L. Edelman, D. J. Spiro, A. R. Boyne, y J. Bera. Molecular epidemiology of a/h3n2 and a/h1n1 influenza virus during a single epidemic season in the united states. *PLosS Pathog*, 4:83–88, 2008.

N. R. Pace. A molecular view of microbial diversity and the biosphere. *Science*, 276(5313):734–740, 1997.

N.I. Platnick. An empirical comparison of microcomputer parsimony programs, ii. *Cladistics*, 5(2): 145–162, 1989.

Real Academia Española. *Ortografía de la lengua española*. Espasa, 2010.

C. C. Ribeiro y D. S. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path-relinking. pages 97–102, 2003.

C. C. Ribeiro y D. S. Vianna. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12(3):325–338, 2005.

J. M. Richer, A. Goëffon, y J. K. Hao. A memetic algorithm for phylogenetic reconstruction with maximum parsimony. In *EvoBIO*, volume 5483 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2009.

D. Robinson. Comparison of labeled trees with valency three. *Journal of Combinatorial Theory, Series B*, 11:105–119, 1971.

E. Rodriguez-Tello y J. Torres-Jimenez. Memetic algorithms for constructing binary covering arrays of strength three. *Lecture Notes in Computer Science*, 5975:86–97, 2010.

A. Rzhetsky y M. Nei. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10(5):1073–1095, 1993.

D. Sankoff y P. Rousseau. Locating the vertices of a Steiner tree in an arbitrary metric space. *Mathematical Programming*, 2:240–246, 1975.

R. R. Sokal y C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

D. L. Swofford y G. J. Olsen. Phylogeny reconstruction. In *Molecular Systematics*, pages 411–501. Sinauer Associates, Inc., Sunderland, USA, 1990.

D. L. Swofford, G. J. Olsen, P-J. Waddell, y D. M. Hillis. Phylogenetic inference. In *Molecular Systematics (2nd edition)*, pages 407–514. Sunderland, USA, 1996.

H. Szu y R. Hartley. Fast simulated annealing. *Physics Letters A*, 122(3-4):157–162, 1987.

N. L. J. Ulder, E. H. L. Aarts, H. Bandelt, P. J. M. van Laarhoven, y E. Pesch. Genetic Local Search Algorithms for the Traveling Salesman Problem. In *Parallel Problem Solving from Nature - PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 109–116. Springer, 1991.

P. Van Der Pas. A note on the bibliography of gregor mendel. *Medical History*, 3(4):331, 1959.

M. S. Waterman y T. F. Smith. On the similarity of dendrograms. *Journal of Theoretical Biology*, 73(4):784–900, 1978.

G. Weber, L. Ohno-Machado, y S. Shieber. Representation in stochastic search for phylogenetic tree reconstruction. *Journal of Biomedical Informatics*, 2006.

J. Xiong. *Essential Bioinformatics*. Cambridge University Press, 1st. edition, 2006. ISBN 978-0521600828.

J. J. Zhou, J. Tian, D. Y. Fang, Y. Liang, H. J. Yan, J. M. Zhou, H. L. Gao, C. Y. Fu, Y. Liu, H. Z. Ni, C. W. Ke, y L. F. Jiang. Analysis of antigen epitopes and molecular pathogenic characteristics of the 2009 h1n1 pandemic influenza a virus in china. *Acta Virol*, 55(3):195–202, 2011. ISSN 0001-723X.