UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

“DESARROLLO DE UN PROCEDIMIENTO PARA SOLUCIONAR EL PROBLEMA DE ALINEAMIENTO MÚLTIPLE DE SECUENCIAS”

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA

SISTEMAS - INVESTIGACIÓN DE OPERACIONES

P R E S E N T A :
ING. ROMAN A. MORA GUTIÉRREZ

T U T O R :
DR. JAVIER RAMIREZ RODRIGEZ

MÉXICO D.F. **2009**

JURADO ASIGNADO:

Presidente:	Dr. Miguel Ángel Gutiérrez Andrade.
Secretario:	Dra. Mayra Elizondo Cortes.
Vocal:	Dr. Javier Ramírez Rodríguez.
1er Suplente:	Dra. María Elena Larraga Ramírez.
2do Suplente:	Dr. Gerardo Eugenio Sierra Martínez.

Esta Tesis se realizo en
Ciudad Universitaria, Distrito Federal México.

TUTOR DE LA TESIS:

Dr. Javier Ramírez Rodríguez

_____
Firma.

Agradecimientos.

A la Universidad Nacional Autónoma de México (UNAM), y al programa de posgrado en ingeniería por haberme brindado la oportunidad de seguir formándome como profesionista y como ser humano.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por su apoyo, el cual hizo posible que me integrara como estudiante de tiempo completo al programa de maestría en ingeniería.

A los profesores: Dr. Javier Ramírez, Dra. Mayra Elizondo, Dr. Oscar Herrera y Dra. Ma. Elena Larraga, por su disposición permanente e incondicional en aclarar mis dudas, compartir su experiencia, por sus substanciales sugerencias durante mis estudios y en la elaboración de este trabajo, y en especial por su paciencia y amistad.

Al comité tutorial por la atención brindada y las observaciones hechas.

A los profesores de la maestría que compartieron con migo su tiempo y conocimiento, los cuales han sido fundamentales para la culminación de este peldaño.

A mis padres Anselmo Mora y Clara Gutiérrez por haberme brindado las herramientas y valores necesarios para transitar por la vida, por el afecto y confianza ilimitada otorgada en cada momento.

A mis hermanos Rodrigo, Fidel, Clara, Teresa y Beatriz por su compañía, consejos, palabras de aliento por su inmensa comprensión. A mis sobrinos que siempre han tenido una sonrisa en los labios, que me alienta a ser una mejor persona.

A mi familia por su comprensión, amor incondicional y el aliento brindado.

Al Dr. José Honorato, Dr. Irineo López y al Ing. Gonzalo Novelo por su apoyo y aprecio.

A todos mis amigos por cada instante compartido.

Les doy las gracias
Roman A. Mora Gutiérrez.

Dedicatoria.

Dedico este trabajo a:

A la memoria de mis abuelos.

A mis padres, hermanos y sobrinos.

A mi familia

A mis amigos

Tabla de contenido

I. ..... RESUMEN. IV

II. ABSTRACT..... V

III. INTRODUCCIÓN. VI

Objetivo de la tesis: viii

Estructura del trabajo: viii

CAPÍTULO 1 EL PROBLEMA DE ALINEAMIENTO MÚLTIPLE DE SECUENCIAS Y SUS CONCEPTOS BÁSICOS..... 1

1.1 Conceptos básicos..... 1
1.1.1 Subsecuencia. 5
1.1.2 Biosecuencia. 6
1.1.3 Familia de secuencias. 7
1.1.4 Similitud. 8
1.1.4.1 Métodos de cuantificación. 8
1.1.5 Alineamiento y patrones. 23

1.2 Características del alineamiento múltiple de secuencias. 26
1.2.1 Función objetivo. 26
1.2.1.1 Funciones de puntaje. 27
1.2.1.2 Aproximación por templetes. 30
1.2.2 Restricciones del problema. 32

1.3 Resumen. 33

CAPÍTULO 2 ESTADO DEL ARTE DE: LOS MÉTODOS DE SOLUCIÓN DEL AMS, LA BÚSQUEDA DE ARMONÍA Y EL RECOCIDO SIMULADO. 35

2.1 Conceptos básicos..... 37

2.2 Métodos de solución para el AMS. 37
2.2.1 Métodos de búsqueda exhaustiva. 41
2.2.1.1 Métodos basados en programación dinámica. 41
2.2.1.2 Ramificación y acotamiento. 67
2.2.2 Métodos aproximados. 68
2.2.2.1 Métodos progresivos.. 68
2.2.2.2 Alineamientos Híbridos 81
2.2.2.3 Métodos Iterativos. 84

2.3 Programas de cómputo..... 86
2.3.1 Método de comparación. 87
2.3.1.1 Métricas de cuantificación de eficiencia. 87
2.3.1.2 BALiBASE. 88

2.4 Metaheurísticas padres del método desarrollado..... 90
2.4.1 Búsqueda de armonía. 91
2.4.2 Algoritmo de búsqueda de armonía. 94
2.4.3 Aplicaciones a problemas de optimización. 108

2.5 Recocido simulado..... 109

2.6 Resumen..... 112

CAPÍTULO 3 ESTRATEGIA PROPUESTA PARA SOLUCIONAR EL AMS. 115

3.1 Desarrollo de la nueva estrategia. 116
3.1.1 Descripción del proceso. 116
3.1.1.1 Definición de los parámetros de entrada de entrada. 117
3.1.1.2 Creación de la memoria armónica. 123
3.1.1.3 Generación de una nueva armonía. 140
3.1.1.4 Actualización de la memoria armónica. 145
3.1.1.5 Criterio de paro. 147
3.1.2 Estructura de la estrategia propuesta y comparación con HS. 149

3.2 Validación de la estrategia desarrollada. 150
3.2.1 Referencia 1 152
3.2.2 Referencia 2 153
3.2.3 Referencia 3 154
3.2.4 Referencia 4 154
3.2.5 Referencia 5 155

3.3 Análisis de resultados 157

3.4 Resumen..... 159

CAPÍTULO 4 CONCLUSIONES Y TRABAJOS FUTUROS. 161

APÉNDICE A PROCESO EVOLUTIVO.....163

APÉNDICE B COMPLEJIDAD, ORDEN Y TIPO DE PROBLEMA. 166

APÉNDICE C BASES DE DATOS BIOLÓGICAS. 171

ANEXO A ALFABETOS DE BIOSECUENCIAS 173

ANEXO B RESULTADOS DE LAS PRUEBAS DE VALIDACIÓN BALIBASE VERSIÓN 1 174

B.1. Estimación de la función SSP 174

B.2. Iteración en la cual se alcanza el máximo valor 179

ANEXO C COMPARACIÓN CON OTROS MÉTODOS DE ALINEAMIENTO..... 184

C.1. Referencia 1 184

C.2. Referencia 2 186

C.3. Referencia 3 187

C.4. Referencia 4 187

C.5. Referencia 5 187

ANEXO D LISTA DE SÍMBOLOS..... 189

ANEXO E OTROS ÍNDICES. 192

E.1. Figuras..... 192

E.2. Tablas..... 195

E.3. Algoritmos..... 196

BIBLIOGRAFÍA..... 198

I. Resumen.

En el presente trabajo se desarrolló un algoritmo nuevo para resolver el problema de alineamiento múltiple de secuencias (AMS) basado en las metaheurísticas de búsqueda de armonía (HS) y recocido simulado. Para lo cual fue necesario realizar una investigación bibliográfica del estado del arte del problema de AMS y de las metaheurísticas utilizadas.

Como resultados del trabajo realizado se generó un algoritmo base para solucionar el AMS basado en HS y RS, el cual fue validado mediante la metodología de Julie Thompson. Y aunque los resultados obtenidos durante la etapa de validación no son muy alentadores, permiten apreciar las ventajas y desventajas del método desarrollado. Lo que podrá servir de base para trabajos futuros que ayuden a mejorar la estrategia.

Palabras clave: Alineamiento Múltiple de Secuencias, Algoritmo, Metaheurística, Búsqueda de la armonía.

II. Abstract.

In this paper we developed a new algorithm for solving the problem of multiple sequence alignment (AMS) based on the harmony search metaheuristics (HS) and simulated annealing. For which it was necessary to perform a literature search of the state of the art AMS problem and used metaheuristics.

As results of the work was generated a based algorithm to solve the HS-based AMS which the methodology was validated by Julie Thompson methodology. And although the results during the validation stage are not very elongated, permitting consideration of advantages and disadvantages of the method developed. Those who might form a basis for future work to help improve the strategy.

Keywords: Multiple sequence alignment, algorithm, Metaheuristic, A search for harmony.

III. Introducción.

Uno de los objetivos primordiales de la ingeniería de sistemas es la generación de nuevas ideas, conceptos, principios, métodos y dispositivos (Hall & D., 1983) a través de creatividad (habilidad de relacionar conceptos e ideas antes aislados) y conocimiento (conjunto organizado de datos e información) que permitan el modelado, análisis, solución de problemas. Son un campo fértil para el desarrollo de las habilidades creativas aquellos problemas que permiten una amplia forma de plantearse, admitan más de un procedimiento para atacarlos y tienen más de una respuesta correcta.

El presente trabajo surge al relacionar los conceptos del problema de alineamiento múltiple de secuencias y la metaheurística de búsqueda de armonía, antes disjuntos. Por lo anterior, se entiende que el presente trabajo radica en la generación un procedimiento para la resolución del problema de alineamiento de múltiples secuencias (AMS), para lo cual se utilizaron ideas y conceptos de las metaheurística de búsqueda de armonía (HS) y de recocido simulado (RS)

Como resultados del trabajo realizado se generó un algoritmo base para solucionar el AMS basado en HS y RS, el cual fue validado mediante la metodología de Julie Thompson. Y aunque los resultados obtenidos durante la etapa de validación no son muy alentadores, permiten apreciar las ventajas y desventajas del método desarrollado. Lo que podrá servir de base para trabajos futuros que ayuden a mejorar la estrategia.

A continuación esbozan imágenes generales sobre el AMS, la búsqueda de armonía y del recocido simulado, a fin de proporcionar al lector una idea sobre ellos:

El Alineamiento múltiple de secuencias en términos generales es un problema que trata de la búsqueda de patrones en un conjunto de objetos, (lo que se logra a través de un análisis comparativo) para identificar similitudes (elementos invariantes) y diferencias (scoring); con el objetivo de clasificar a los objetos del conjunto de interés. En otras palabras, el AMS es un problema que permite la comparación y análisis de estructuras de datos.

El AMS es un problema NP-duro[1], por lo cual se utilizan técnicas heurísticas y metaheurísticas para solucionarlo.

El AMS se encuentra dentro del grupo de problemas de la bioinformática, en virtud de que los mayores avances en técnicas y métodos para la comparación de secuencias se han dado al trabajar con secuencias biológicas (biosecuencias). Y su aplicaciones en la comparación de secuencias bilógicas (nucleótidos o proteicas), han servido para entender, analizar y rastreas el proceso evolutivo a través de los

[1] Véase anexo B.

patrones (motivos) e inferir características importantes de relación entre organismos[2].

Las aplicaciones del AMS no se restringen a la biología, ya que ha sido utilizado en áreas muy diversas del conocimiento donde en la comparación y el análisis de secuencias de datos han sido necesarios, como son: el reconocimiento de voz, el análisis musical, cromatografía de gases, etc.

Para ilustrar la idea del AMS obsérvese el siguiente ejemplo: "Al imaginar una charola llena de galletas en forma de hombres, se podría observar que los objetos en la charola son muy semejante entre sí, pero con sutiles diferencias que los hacen únicos. Y aunque ningún par de pastas sean iguales, es posible suponer un origen común (un molde) (Gaarder, 2004)." El problema anterior se reduce a inferir las características del molde a partir de las particularidades del conjunto de galletas. Se conjeturaría que los elementos comunes a todas las galletas son características del molde, y por tanto entre mayor sea el número de similitudes identificadas mejor será el constructo del molde.

La búsqueda de armonía es una metaheurística que imita el proceso de improvisación musical, ya que el proceso de innovación musical busca producir un estado ideal del sistema, a través de determinar la estimación musical (Geem & Choi, 2007). Esta metaheurística ofrece gran flexibilidad para el proceso de optimización, ya que requiere determinar un estado inicial del sistema, ocupa pocos parámetros para su ejecución, realiza una búsqueda aleatoria estocástica (Coelho & Mariani, 2009). Ha sido utilizada en problemas continuos y discretos con excelentes resultados. Hasta el momento no se encuentra reportada en la literatura alguna aplicación de dicha metaheurística al problema de AMS.

La metaheurística de recocido simulado imita el proceso termodinámico de recocido de materiales metálicos. Desde la óptica de la optimización el estado final del metal constituye la solución más adecuada al tratar de minimizar la energía empleada.
Esta metaheurística requiere una solución inicial del sistema y realiza una búsqueda estocástica en vecindades, aunque se aceptan todas aquellas soluciones que mejoren el valor de la solución actual, existe la posibilidad de aceptar soluciones peores a la actual.

Los procedimientos de búsqueda armónica y recocido simulado son técnicas metaheurísticas que han sido utilizado en una gran variedad de problemas de optimización ofreciendo buenos resultados. Sin embargo, la búsqueda armónica no ha sido utilizada como un procedimiento base para la resolución del AMS. Mientras que el procedimiento de recocido simulado ha ofrecido resultados de alta calidad a resolver problemas de AMS (Vélez & Montoya, 2007).

Dado este marco referencial, el presente trabajo crea y adapta una metaheurística hibrida entre búsqueda armónica y recocido simulado para la resolución del AMS en

[2] Véase anexo A.

secuencias biológicas. Para lo cual se adecuaron conceptos de ambas metaheurísticas, además se crearon y definieron elementos necesarios para tratar el problema con el híbrido mencionado.

Objetivo de la tesis:

Por lo anterior el objetivo general de este trabajo puede enunciarse como: “Desarrollar un método original para resolver el AMS, basado en las metaheurísticas de búsqueda de armonía y de recocido simulado”.

La existencia de varias formas para realizar el objetivo anterior, conllevan a delimitar el alcance y la justificación de este trabajo. En primer lugar el alcance de este trabajo es solamente desarrollar una metaheurística híbrida que permita solucionar el AMS, la cual será validada y esto servirá para determinar la calidad de sus resultados, y posteriormente se compararán estos resultados con los obtenidos por otros procedimiento. En segundo lugar este trabajo se justifica, ya genera un algoritmo base (el cual es susceptible de mejoras) que brinda un primer acercamiento para utilizar los conceptos de búsqueda armónica en la solución del AMS.

Para alcanzar el objetivo anterior se utilizaron las siguientes fases metodológicas (objetivos particulares):

1. Buscar, revisar, comprender y analizar la información disponible sobre el problema AMS y los métodos más usuales para su solución.

2. Entender y analizar las metaheurísticas de búsqueda de armonía y recocido simulado.

3. Adaptar las ideas y conceptos de búsqueda de armonía y recocido simulado para solucionar el AMS, desarrollando los preceptos necesarios para dicho propósito.

4. Comparar el método desarrollado con otros métodos utilizando en el alineamiento múltiple de secuencias utilizando la metodología de Julie Thompson (Thompson, Plewnlak, & Poch, BALiBASE: a benchmark aligment database for the evaluatios of multiple alignmet programs, 1999).

Estructura del trabajo:

El presente documento se encuentra organizado de la manera siguiente:

El capítulo 1 describe las características del problema de alineamiento múltiple de secuencias a través de analizar las ideas y conceptos con dicho problema.

El capítulo 2 se muestra algunos de los métodos desarrollados para solucionar el AMS; en él se analizan con mayor profundidad los procedimientos clásicos.

En el capítulo 3 se desarrolla la estrategia propuesta para solucionar el AMS en las primeras secciones se da una breve descripción de la estrategia de búsqueda de armonía y del recocido simulado, posteriormente se muestra el desarrollo del algoritmo.

En el capítulo 4 se presentan las conclusiones sobre el trabajo realizado ventajas y desventajas del método desarrollado así como sus limitaciones, también se realizan las recomendaciones para trabajos futuros.

Adicionalmente se incluyen 3 apéndices y tres anexos con la finalidad de proveer al lector de elementos que le permitan un mejor entendimiento del problema.

Capítulo 1 El problema de alineamiento múltiple de secuencias y sus conceptos básicos.

La función del presente capítulo es: “Definir y caracterizar el problema de alineamiento múltiple de secuencias (AMS), como un problema de optimización”, dicha función se relaciona directamente con el primer objetivo particular de la tesis.

Para alcanzar la función de este capítulo se utiliza la siguiente táctica:

- Definir y analizar los conceptos e ideas involucrados en el AMS, desde los puntos de vista matemático y biológico.

- Definir y analizar la estructura del AMS como un problema de investigación de operaciones.

Cabe resaltar que el tipo de secuencias que serán utilizadas en el desarrollo de este trabajo corresponden a secuencias biológicas (biosecuencias).

1.1 Conceptos básicos.

La búsqueda de técnicas y métodos eficientes para resolver problemas complejos no se restringe a alguna ciencia en particular, es un interés de todas las áreas del conocimiento. Generalmente la interacción entre varias ramas del conocimiento genera los métodos y procedimientos efectivos para resolver los problemas.

La interacción entre las ramas del conocimiento, frecuentemente genera un camino de ida y vuelta. Es de ida ya que algunas ciencias permiten entender, interpretar y predecir fenómenos o situaciones, y de vuelta, pues recorriendo ese camino se tiene una inagotable fuente de problemas que requieren solución a través de la creatividad y el conocimiento.

El problema de alineamiento múltiple de secuencias (AMS) es un claro ejemplo de la interacción entre las ciencias; ya que es un problema que nace en la biología y es abordado para su resolución por las ciencias de la computación y las matemáticas.

En la biología se entiende al AMS como el problema de comparar la similitud entre tres o más secuencias biológicas generalmente proteicas, de ADN o de ARN (Chan, Wong, & Chiu, 1992). Se aplica para: Encontrar subregiones altamente conservadas (patrones) de un conjunto de biosecuencias e inferir la historia

evolutiva de un conjunto de taxas[3] a través de la asociación de sus secuencias biológicas (Gusfieeld, 1993).
l
Un ejemplo de lo anterior es la comparación de proteínas muy conservadas evolutivamente que cumplen la misma función en distintos organismos, como es el caso de la insulina en los mamíferos y con base en la información resultante diseñar árboles evolutivos.

Antes de definir de manera formal el AMS resulta necesario exponer los conceptos de secuencia y comparación de secuencia, los cuales se dan a continuación.

De manera usual, se define a una secuencia como una serie ordenada de elementos, donde dichos elementos pertenecen a un conjunto de objetos (símbolos o letras) denominado alfabeto, $\mathcal{A}$. Es decir, una se es una sucesión de los objetos de un alfabeto. En lo siguiente se define formalmente secuencia y algunos conceptos relacionados.

Definición 1. Un **alfabeto** es un conjunto finito de símbolos, diferente al conjunto vacío.

$$\mathcal{A} = \{a_1, a_2\ , a_3, \cdots a_n\} \qquad (1).$$

Definición 2. Una **secuencia $\boldsymbol{s_i} = \{\boldsymbol{b_1}, \boldsymbol{b_2}\ , \boldsymbol{b_3}, \cdots \boldsymbol{b_l}\}$ es una secuencia formada a partir de un alfabeto $\boldsymbol{\mathcal{A}}$**, si y sólo si $b_i \in \mathcal{A} \quad \forall\ i = 1,2,3, \cdots, l.$

$$s_i\ es\ una\ secuencia\ de\ \mathcal{A} \Leftrightarrow b_i \in \mathcal{A} \qquad (2).$$

En la literatura frecuentemente se utilizan como sinónimos de secuencia los términos sentencia o cadena (Chan, Wong, & Chiu, 1992).

La cantidad de secuencias de una longitud (l) determinada, que pueden construirse a partir de un alfabeto dado permitiendo la repetición de elementos en la secuencia, se puede determinar al utilizar la regla del producto y el principio de disposición.

Definición 3. La **regla del producto** establece que si un procedimiento se puede descomponer en las etapas primera y segunda, y si existe m resultados posibles de la primera etapa y si para cada uno de estos resultados, existen n resultados posibles para la segunda etapa, entonces el procedimiento total se puede realiza, en el orden dado de $m * n$ formas (Grimaldi, 1998).

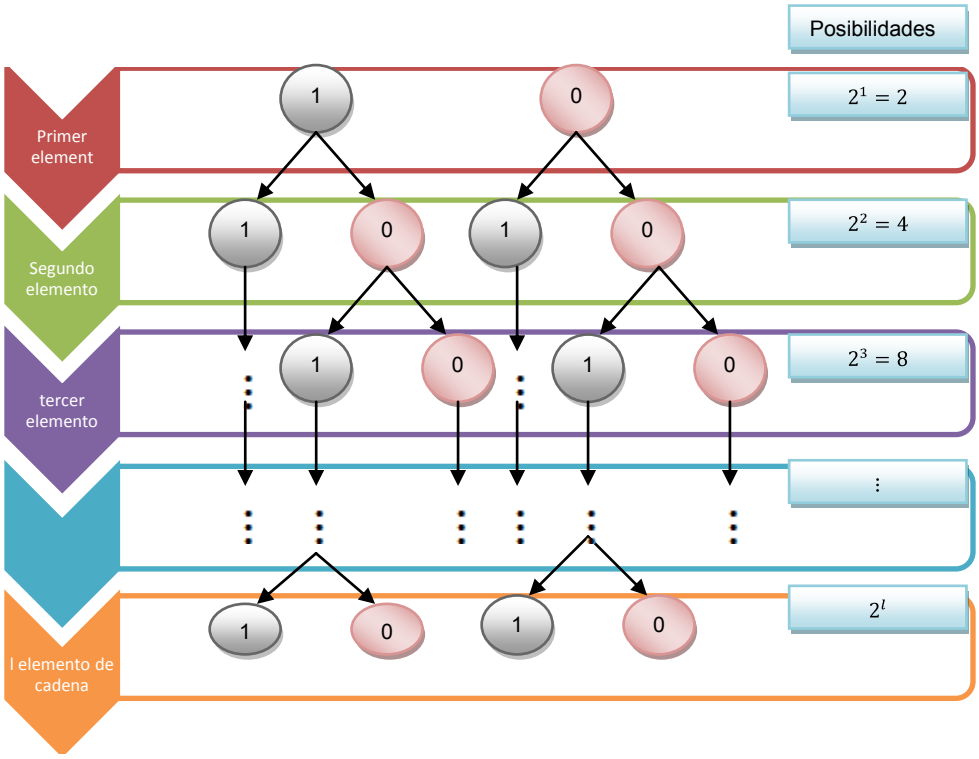
Definición 4. El **principio de disposición** establece que dada una colección de n objetos distintos, cualquier cadena lineal de r elementos, donde el orden es importante y se permita la repetición de los objetos n, se denomina disposición. Y existen n^r disposiciones posibles con $r \geq 0$.

[3] Una taxa es un grupo de individuos emparentados

Para el caso de las secuencias, los n objetos distintos son los elementos del alfabeto $\mathcal{A}$ y r coincide con la longitud l de la secuencias, entonces se deduce lo siguiente: si todo eslabón b_j y b_{j+1} de una secuencia s_i de longitud l pueden ser ocupados por cualquier elemento de $\mathcal{A}$, entonces existen $b_1 * b_2 * \ldots * b_j * \ldots, * b_l$ formas distintas de construir la secuencia s_i. Para ilustrar lo anterir considere el siguiente ejemplo.

Ejemplo 1. Dado el conjunto $z = \{1, 0\}$ ¿Cuáles cadenas con una longitud l se pueden formar? En la Figura1 se observa que existen 2^L cadenas posibles a formar. Por lo anterior, se podría inferir que, dado un conjunto de k elementos, la cantidad de cadenas (de longitud l) posibles es k^l.

Figura 1. Problema de permutación de objetos



Elaborada concepto de paralelismo Kuri Morales y Galaviz Casas 2002

Definición 5. Las **cadenas posibles de longitud *l*,** dado un alfabeto $\mathcal{A}$, se determinan con k^l, permitiendo la repetición de elementos (Kolman, Busy y Ross 1997). Lo cual se expresa mediante la siguiente igualdad.

$$S_l = k^l \qquad (3).$$

donde:
k: Número de elementos del alfabeto.
l: Longitud de las secuencia.
S_l: Conjunto de secuencias posibles.

La comparación de secuencias puede visualizarse como una forma de arqueología para descubrir las secciones de las secuencias que se han conservado a través del tiempo (Abascal, 2007). A continuación se define la comparación de secuencias en términos formales

Definición 6. La **comparación de secuencias**. Se define como la búsqueda de las zonas de similitud significativa entre dos o más secuencias para localizar características comunes o diferentes entre ellas. Dicha búsqueda servirá para determinar un arreglo entre las cadenas con el que se obtenga el número mayor de coincidencias entre los componentes (Hernández Valdemar, 2003). Comparar exhaustivamente un conjunto de secuencias implica verificar si en el i –ésimo eslabón de todas ellas, se encuentra el mismo caracter.

El alineamiento múltiple de secuencias implica la comparación entre secuencias para conformar una matriz con los eslabones de las mismas. Las columnas de dicha matriz permitirán localizar las zonas invariantes entre el conjunto de secuencias.

Definición 7. El **alineamiento múltiple de secuencias** de un conjunto de $N-secuencia$ ($N=\{s_1, s_2, \cdots, s_N\}$), implica generar una matriz $M_{\mathrm{N\,x\,k}}$ rectangular de caracteres sobre un alfabeto en unión con un guion (caracter vacío) ($\mathcal{A} \cup \{-\}$) de tal manera que ninguna columna de la matriz M conste completamente de caracteres vacios y el resultado de eliminar los guiones en cada una de las hileras de M sean las secuencias $\{s_1, s_2, \cdots, s_N\}$ correspondiente.

En el siguiente ejemplo, se muestra las implicaciones del problema AMS desde el punto de vista de la investigación de operaciones:

Ejemplo 2. Dadas las secuencias $s_1=\{L,I,D,I,A\}$, $s_2=\{L,I,S\}$, $s_3=\{E,L,I,S,A\}$ es posible construir los siguientes alineamientos múltiples:

1.

-	L	I	D	I	A
E	L	I	S	A	-
-	L	I	S	-	-

2.

-	L	I	D	I	A
E	L	I	S	-	A
-	L	I	S	-	-

Aunque ambos alineamientos son factibles, surge el problema de decidir ¿cuál alineamiento es mejor?

Para resolver la interrogante mencionada, se debe definir la función objetivo a utilizar, es decir se desea maximizar el número de similitudes entre las secuencias o se necesita minimizar las diferencias en la matriz de alineamiento.

El AMS es un problema combinatorio, lo que implica que el conjunto de soluciones factibles es finito.

1.1.1 Subsecuencia.

Por otra parte comparar exhaustivamente un conjunto de secuencias implica verificar si en el i –ésimo eslabón de todas ellas, se encuentra el mismo caracter e involucra determinar la subsecuencia común más larga.

Una subsecuencia es un subconjunto de los elementos que pertenecen a una cadena $s = \{a_1, a_2, \cdots a_n\}$ (pueden ser no consecutivos) y en el cual respeta el orden de aparición de dichos elementos tal como ocurre en s. Considérese el siguiente ejemplo.

Ejemplo 3. Dada $s = \{a, b, c, d, e\}$ se desea determinar si $s_{sub-1} = \{a, b, e\}$, $s_{sub-2} = \{b, c, a\}$ $s_{sub-3} = \{b, c, f\}$ son subsecuencias.

A) La primera subsecuencia se forma de los elementos de s y se respeta el orden de aparición de los elementos, por tanto, $s_{sub-1}\ es\ subsecuencia\ de\ s$ (figura 2, inciso a)

B) La segunda subsecuencia se forma de las elementos de s, pero no se respeta el orden, por tanto $s_{sub-2}\ no\ es\ subsecuencia\ de\ s$ (figura 2, inciso b)

C) La tercera subsecuencia no se forma de las elementos de s, y por ende $s_{sub-3}\ no\ es\ subsecuencia\ de\ s$ (figura 2, inciso c)

Figura 2. Subsecuencia.

a) Subsecuencia

S	a ↓	b ↓	c	d	e ↓
S_{sub-1}	a	b			e

b) No es Subsecuencia

S	a →	b ↓	c ↓	d	e	→ a
S_{sub-2}		b	c			

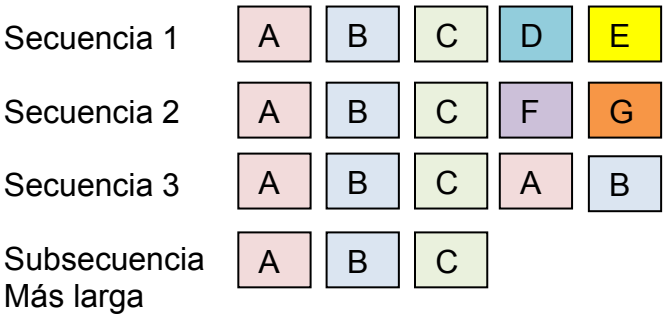
c) No es Subsecuencia

S	a	b ↓	c ↓	d	e	
S_{sub-1}		b	c			**f**

En resumen, una subsecuencia de s se obtiene al eliminar de cero a $l-1$ símbolos no necesariamente consecutivos de una secuencia s de longitud l, manteniendo el orden de s

Definición 8. La **subsecuencia común** dado $S = \{s_1, s_2, \ldots s_N\}$ conjunto de secuencias, s_{sub} es una subsecuencia común a ellas si y sólo si todos los elementos de s_{sub} también son elementos de cada una de las secuencias, y se mantiene el orden. Se dice que es la subsecuencia más larga si y sólo si todos los elementos comunes entre las secuencias del conjunto están contenida en ella.

Figura 3. Subsecuencia común.



En virtud que han sido revisados los conceptos de secuencia y alineamiento múltiple de secuencias, en la sección siguiente se analiza el concepto de biosecuencia.

1.1.2 Biosecuencia.

El análisis de secuencias biológicas (biosecuencias) mediante el AMS, se utiliza para identificar diferencias entre un conjunto de individuos con el fin de identificar los mecanismos de evolución. Es decir, la comparación entre biosecuencias sirve para identificar el patrón de las cadenas y predecir así su funcionalidad en individuos emparentados. Esto se basa en el hecho que estructuras similares tienen funciones similares, en otras palabras, se busca el motivo (regiones de secuencias que tienen una estructura específica y una funcionalidad significante) entre el conjunto de secuencias. Cabe mencionar que en teoría los cambios acumulados en una secuencia biológica se producen a una tasa relativamente constante e independiente de parámetros poblacionales.

Una biosecuencias es una representación simbólica de las cadenas de nucleótidos (Adenina, Guanina, Timina, Citosina y Uracilo) o proteicas.

Los nucleótidos[4] son bases nitrogenadas que conforman al acido desoxirribonucleico o ADN (A,C, G ,T) y al acido ribonucleico o ARN (A,C,G U). La disposición y secuenciación de los nucleótidos determina la codificación de la

[4] Véase anexo B Alfabetos de biosecuencias

información biológica. Los codones son uniones de tres nucleótidos. Y la unión de dos codones forma una proteína. El orden y disposición de los aminoácidos se rige por el código genético.

Por lo anterior, se conceptualiza a las secuencias biológicas (en su estructura lineal) como una cadena de símbolos discretos tomados de un alfabeto (que para el caso de los nucleótidos es un alfabeto de 4 caracteres y para el caso de las proteínas es un alfabeto de 20 caracteres). El número de cadenas posibles a formar de una longitud l dada para los nucleótidos y proteínas es de $4^l \ y \ 20^l$, respectivamente.

La secuencia lineal de los aminoácidos contiene la información necesaria para generar una molécula proteica con una estructura tridimensional particular. A esta secuencia se le llama estructura primaria de la proteína y cada posición es denomina residuo.

El comparar secuencias de nucleótidos o de aminoácidos depende de la información que se busque y los datos con que se dispongan. (Abascal, 2007).

1. La comparación entre secuencias de nucleótidos se usa cuando se desea determinar el grado de similitud entre individuos (estudios filogenéticos, genética de poblaciones, etc.). Lo que permite identificar genes variantes entre los individuos de una familia de secuencias.

2. La comparación de secuencias de aminoácidos es apropiado para buscar homólogos, (o semi-homólogo) ya que el parecido en la secuencia aminoácidos se pierde más lentamente y se sabe que algunos aminoácidos tienen propiedades más parecidas que otros, por lo que se puede inferir el sentido de los cambios (Abascal, 2007).

Las secuencias biológicas se pueden clasificar en: homólogas y análogas. Son homólogas si se relacionan evolutivamente por un ancestro común, es decir, han evolucionado desde la misma posición ancestral. Y son secuencias análogas si se relacionan funcionalmente pero no tienen antecesor común.

Debido al proceso evolutivo[5], con frecuencia las secuencias biológicas de individuos de una misma especie no coinciden en todos sus eslabones.

1.1.3 Familia de secuencias.

Toda comparación entre secuencias, debe ser realizada en cadenas de la misma familia. De manera general, se denomina como familia de secuencias a aquellas sentencias cuyos elementos provienen del mismo alfabeto (Chan, Wong, & Chiu, 1992). En la literatura se utiliza como sinónimo el término conjunto de secuencias.

[5] Véase apéndice A Proceso Evolutivo.

Definición 9. Una **familia de secuencias** dadas x secuencias de longitud l $s_1 = \{b_{1,1}, b_{1,2}, b_{1,3}, \cdots b_{1,l}\}$, $s_2 = \{b_{2,1}, b_{2,2}, b_{2,3}, \cdots b_{2,l}\}$, $\cdots, s_x = \{b_{x,1}, b_{x,2}, b_{x,3}, \cdots b_{x,l}\}$, se integra si y sólo si todo $b_{i,j} \in \mathcal{A} \quad \forall\ j = 1,2,3,\cdots,l\ \forall\ i = 1,2,3,\cdots,x$ (Chan, Wong, & Chiu, 1992).

1.1.4 Similitud.

La similitud de secuencias puede entenderse como el grado de proximidad existente entre ellas. Un conjunto de biosecuencias son similares por las siguientes razones:

1. **Filogenéticas**. Secuencias homólogas.
2. **Funcionales.** Generadas por convergencia evolutiva.
3. **Limitaciones físicas.** Por ejemplo, los dominios trans membrana tienen que ser hidrófobos a pesar de que tienen funciones muy distintas.
4. **Presencia de secuencias repetidas.** Su contenido informativo es bajo y su interpretación puede conducir a errores graves.

Dentro de la teoría de similitud son fundamentales los conceptos de homología y semi-homología ya que sirven para determinar el grado de proximidad entre cadenas. A continuación se describen los conceptos de homología y semi-homología

Definición 10. La **homología** es la relación existente entre dos individuos (o partes orgánicas) diferentes cuando sus determinantes genéticos tienen el mismo origen evolutivo. Y se utiliza para describir el porcentaje estimado de la similitud (porcentaje de posiciones idénticas de las secuencias en la comparación) (Leluk, Regularities in mutational variability in selected protein families and the Markovian model of amino acid replacement , 2000).

Definición 11. La **semi-homología** implica la posibilidad de substitución de un residuo x por otro residuo en un punto de mutación de los codones, de forma que las subsecuencias comparadas puedan transformarse una en la otra. Se trata entonces de codones semi-homólogos (Leluk, 1998).Existen tres tipos de semi-homología (Leluk, 1998).los cuales son:
A) Sustitución o cambio, (base purica por otra purina, una bases pirimidínicas por otra pirimidica o la sustitución de una base purica por una base pirimidica (o viceversa)).
B) Transformación del residuo en seis diferentes aminoácidos.
C) Conjunto de alternativas para formar los aminoácidos.

1.1.4.1 Métodos de cuantificación.

Existe una gran diversidad de métodos para cuantificar la similitud de las secuencias, a continuación se muestran los más utilizados.

A *Matrices de puntos.*

Es una representación gráfica para comparar dos biosecuencias en la que se pone de manifiesto las regiones de similaridad entre ambas, las cuales pueden ser apreciadas a simple vista por la detección de patrones (Valverde, 2006).

La idea base de estas comparaciones es usar dos secuencias como coordenadas de una gráfica bidimensional y comparar cada una de las posiciones del eje horizontal con todas las del eje vertical, señalando con un punto donde exista similitud (Valverde, 2006).

Es una técnica sencilla que utiliza un enfoque cualitativo, pero, consume mucho tiempo para análisis a gran escala. A continuación se muestra un algoritmo para la construcción de una matriz de puntos.

Algoritmo 1. Matriz de puntos.

Input: Un par de secuencias $s_a = \{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\}$ y $s_b = \{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\}$ de longitud l_a y l_b respectivamente

Output: Matriz de puntos.

1. Construir una matiz $m_{l_a+1\ x\ l_b+1}$
   // Colocar en la columna uno desde la celda $m_{2,1}$ los elementos s_b
   Colocar en la última fila desde la celda d $m_{1,2}$los elementos s_a //

2. For $(i = 1\,; i \leq l_a\,; i++)$
   a. For $(j = 1\,; j \leq l_b; j++)$
      i. If $(s_{a_i} = s_{b_j})$
         1. $m_{i+1,j+1} \leftarrow$ ■
      ii. Else
         1. $m_{i+1,j+1} \leftarrow$ ' '
      End if
   End for
End for

Fuente: Valverde, 2006.

La complejidad del algoritmo anterior es de $O(l_a * l_b)$. Para mostrar la aplicación del algoritmo anterior considere lo siguiente:

Ejemplo 4. Dadas las secuencias $s_a = \{S, O, L\}$ y $s_b = \{\mathrm{H}, \mathrm{O}, \mathrm{L}, \mathrm{A}\}$ se construye la matriz de puntos siguiente:

H			
O		■	
L			■
A			
	S	O	L

Las gráficas de puntos también pueden utilizarse para evaluar repetitividad en una sola secuencia, y muestran las regiones que comparten similitudes significativas como líneas fuera de la diagonal principal.

En el ejemplo anterior $s_{sub} = \{O, L\}$ es la subsecuencia común más larga. Y por lo tanto, puede alinearse el par de secuencias haciendo coincidir las zonas conservadas y sólo agregando espacios vacios cuando sea necesario, como se muestra a continuación.

s_a	S	O	L	-
s_b	H	O	L	A

B *Distancia de Hamming.*

Métrica desarrollada por Richard Hamming en 1951 para el análisis simultaneo de un par de secuencias. En términos genérales la distancia de Hamming cuantifica el número de posiciones donde las secuencias son diferentes. A continuación se define de manera formal.

Definición 12. La **distancia de Hamming** se refiere a dadas las secuencias $s_a = \{s_{a_1}, s_{a_2}, \dots, s_{a_l}\}\ y\ s_b = \{s_{b_1}, s_{b_2}, \dots, s_{b_l}\}$ si ambas son de la misma longitud l y si $s_a \epsilon\, \mathcal{A}$ y $s_b \epsilon\, \mathcal{A}$, entonces se encuentra definida la distancia entre $s_a\ y\ s_b$, que se denota como $d\,(s_a\,, s_b)$ y es el número de componentes tales que $s_{a_i} \neq s_{b_i}$ para $1 \leq i \leq l$ (Grimaldi, 1998).

$$d(s_a, s_b) = \sum_{i=1}^{l} d(s_{a_i}, s_{b_i}) \qquad (4).$$

donde:
$d(s_a, s_b)$: Distancia entre la secuencia s_a y la s_b.
$d(s_{a_i}, s_{b_i})$: Similitud entre el i –ésimo eslabón de ambas secuencias.

$$d(s_{a_i}, s_{b_i}) = \begin{cases} 1 \ \ si \ \ s_{a_i} \neq s_{b_i} \\ 0 \ \ si \ \ s_{a_i} = s_{b_i} \end{cases} \qquad (5).$$

Es quizá la forma más sencilla de distancia y prácticamente cualquier otra forma para la determinación de distancias la contiene. La distancia de Hamming sólo permite el remplazo de caracteres asociado a un costo, sin embargo tiene el inconveniente de estar solamente definida para la comparación de secuencias con

la misma longitud. (García, 2007). En el ejemplo siguiente se muestra la aplicación de la distancia de Hamming para casos particulares.

Ejemplo 5.

A) Se desea determinar la distancia de Hamming de las secuencias $s_a = \{a, a, b, i, o, n\}$ y $s_b = \{a, i, b, t, o, n\}$

Secuencia	Elemento					
	1	2	3	4	5	6
s_a	*a*	*a*	*b*	*i*	*o*	*n*
s_b	*a*	*i*	*b*	*t*	*o*	*n*
$d(s_{a_i}, s_{b_i})$	*0*	*1*	*0*	*1*	*0*	*0*

$$\sum_{i=1}^{6} d(s_{a_i}, s_{b_i}) = 0 + 1 + 0 + 1 + 0 + 0 = 2$$

La distancia de Hamming entre las secuencias es 2 lo que indica que existen dos eslabones en los cuales ambas cadenas no contienen el mismo caracter.

B) Se desea determinas la distancia de Hamming de las secuencias $s_a = \{a, a, b, i, o, n\}$ y $s_b = \{a, i, b\}$

Secuencia	Elemento					
	1	2	3	4	5	6
s_a	*a*	*a*	*b*	*i*	*o*	*n*
s_b	*a*	*i*	*b*			
$d(s_{a_i}, s_{b_i})$	*0*	*1*	*0*	*Función no definida*		

En virtud de la longitud de ambas secuencias es distinta la distancia de Hamming no se encuentra definida.

C *Distancia de Levenshtein.*

En la literatura la distancia de Levenshtein es sinónimo de la distancia de edición y se le considera una generalización de la distancia de Hamming. Es una métrica de distancia que se encuentra definida para secuencia de longitud igual o diferente.

La distancia de Levenshtein es el costo mínimo asociado con las operaciones necesarias para transformar una secuencia $s_a = \left\{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\right\}$ en la cadena $s_b = \left\{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\right\}$ (Lee, Tseng, Chang, & Tsai, 2007).

Las operaciones permitidas para transformar una secuencia en otra son:

- **Sustituir:** En este caso el elemento s_{a_i} en la cadena s_a es remplazado por el elemento s_{b_i}.
- **Eliminar:** En este caso el elemento s_{a_i} es borrado de la cadena s_a.
- **Insertar:** En este caso el elemento s_{b_i} y es agregado a la cadena s_a.

La distancia de Levenshtein se fundamenta en el concepto de Homomorfismo. El cual involucra la existencia de una función que permita transformar una cadena en otra, a continuación se describe dicho concepto.

Definición 13. **Homomorfismo**. Sean $s_a = \{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\}$ y $s_b = \{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\}$ secuencias de una familia. Un homomorfismo h de s_a en s_b es una función $h: |s_a| \rightarrow |s_b|$ tal que:

$$\{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\} \in s_a \leftrightarrow \{h(s_{a_1}), h(s_{a_2}), \dots, h(s_{a_{l_a}})\} \in s_b \quad (6).$$

$$h\left(f^{s_a}\left(s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\right)\right) = f^{s_b}\left(s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\right) \quad (7).$$

El homomorfismo preserva las relaciones y funciones de las cadenas (Enderton, 1987 & Grimaldi. 1998).

Para determinar la distancia de Levenshtein se ocupa un algoritmo de programación dinámica que implica comparar pares de cadena. A continuación se muestra el pseudocódigo.

Algoritmo 2. Distancia de Levenshtein.

Input: Un par de secuencias $s_a = \{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\}$ y $s_b = \{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\}$ de longitud l_a y l_b respectivamente.

Output: Distancia de Levenshtein entre ambas secuencias.

A) If $(l_a = 0)$
   a. $d(s_a, s_b) = l_b$
   b. Terminar el algoritmo.
B) Else if $(l_b = 0)$
   a. $d(s_a, s_b) = l_a$
   b. Terminar el algoritmo.
C) Else
   a. Construir una matiz $m_{l_a+2\ x\ l_b+2}$
   // Colocar en la columna uno desde la celda $m_{2,1}$los elementos $s_a \cup 0$
   Colocar en la fila uno desde la celda $m_{1,2}$ los elemento $s_b \cup 0$
   Colocar en el resto de las celdas ∞ //
   b. For $i = 2; i \leq l_a + 2; i++$
      i. $m_{2,i} \leftarrow i - 2$
   End For.
   c. For $i = 2; i \leq l_b + 2; i++$

i. $m_{i,2} \leftarrow i - 2$

End For.

d. For $i = 3, i \leq l_a + 2; i++$

i. For $j = 3 \leq l_b + 2; j++$

1. If $\left(s_{a_i} = s_{b_j}\right)$

a. $cost = 0$

2. Else

a. $cost = 1$

End if.

ii. $m_{j,i} = min \begin{cases} m_{j-1,i} + 1 & /* \textit{Eliminación de un caracter} */ \\ m_{j,i-1} + 1, & /* \textit{Inserción de un carácter} */ \\ m_{j-1,i-1} + cost & /* \textit{sustitución de un carácter} */ \end{cases}$

End For.

End For.

End if.

D) For $i = l_a + 2; i \geq 2 ; i--$

a. Buscar el a fila i el elemento más pequeños

b. Buscar en la fila $i - 1$ los elementos vecinos al conjunto anterior con valor más pequeño

c. Unir con una flecha los elementos de ambos conjuntos

End For

E) **While** $camino\ de\ flechas \neq \emptyset$

1. [**def** $d(s_a, s_b) = \sum m_{j,i}$ que se encuentran unidas con una flecha]

End while

F) **Return** $\boldsymbol{min}\ d\ (\mathrm{s_a}, \mathrm{s_b},)$

Elaborado a partir de García, 2007.

La complejidad del algoritmo de Levenshtein es de $O(l_a * l_b)$. Para ilustrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 6. Dadas las secuencias $s_a = \{A, M, O, R\}$ y $s_b = \{\mathrm{H}, \mathrm{O}, \mathrm{L}, \mathrm{A}\}$. Se desea determinar la cantidad mínima de operaciones para transformar s_a en s_b utilizando el algoritmo de distancia de Levenshtein.

Paso 3 inciso a. Construcción de la matriz.

	0	H	O	L	A
0	∞	∞	∞	∞	∞
A	∞	∞	∞	∞	∞
M	∞	∞	∞	∞	∞
O	∞	∞	∞	∞	∞
R	∞	∞	∞	∞	∞

Paso 3 incisos b y c.

	0	H	O	L	A
0	0	1	2	3	4
A	1	∞	∞	∞	∞
M	2	∞	∞	∞	∞
O	3	∞	∞	∞	∞
R	4	∞	∞	∞	∞

Paso 3 incisos d.

	0	H	O	L	A
0	0	1	2	3	4
A	1	1	2	3	3
M	2	2	2	3	4
O	3	3	2	3	4
R	4	4	3	3	4

Paso 4, 5 y 6.

	0	H	O	L	A
0	0	1	2	3	4
A	1	1	2	3	3
M	2	2	2	3	4
O	3	3	2	3	4
R	4	4	3	3	4

Puede verse con claridad varios caminos con una distancia de 8 operaciones, las cuales implican la eliminación e inserción de caracteres en las secuencias .En la matriz utilizada por el procedimiento de Levenshtein (y en todos aquellos basados en el) es posible determinar el alineamiento de las secuencia comparadas de forma indirecta, considerando que los movimientos diagonales indican que los caracteres de la fila y columna correspondientes deben colocarse en la misma columna de la matriz de alineamiento (M), mientras que los movimientos horizontales y verticales representan la inserción de un espacio vacío en la segunda y primera secuencia respectivamente. Considérese lo siguiente:

Ejemplo 7.

Matriz de Levenshtein:

	0	H	O	L	A
0	0	1	2	3	4
A	1	1	2	3	3
M	2	2	2	3	4
O	3	3	2	3	4
R	4	4	3	3	4

Alineamiento obtenido:

s_a	H	-	O	L	A
s_b	A	M	O	R	-

D *Distancia Indel.*

Es una variante de la distancia de Levenshtein en la que sólo se permiten las operaciones inserción y eliminación de caracteres, ya que se utiliza el supuesto de que la operación de sustitución es equivalente a insertar y eliminar un caracter en un punto específico de la secuencia. Por lo que, la función de distancia queda expresada solo en términos de dos operaciones básicas de edición.

Dentro de la literatura también se le conoce como distancia de la subsecuencia común más larga denotada como LCS por sus siglas en inglés, usando para ello la métrica de la subsecuencia más larga (García, 2007).

Algoritmo 3. Distancia LCS

Input: Un par de secuencias $s_a = \{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\}$ y $s_b = \{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\}$ de longitud l_a y l_b respectivamente

Output: Distancia de Indel entre ambas secuencias.

1. If $(l_a = 0)$
   a. $d(s_a, s_b) = l_b$
   b. Terminar el algoritmo.
2. Else if $(l_b = 0)$
   a. $d(s_a, s_b) = l_a$
   b. Terminar el algoritmo.
3. Else
   a. Construir una matiz $m_{l_a+2\ x\ l_b+2}$
   // Colocar en la columna uno desde la celda $m_{2,1}$los elementos $s_a \cup 0$
   Colocar en la fila uno desde la celda $m_{1,2}$ los elemento $s_b \cup 0$
   Colocar en el resto de las celdas ∞ //
   b. For $i = 2; i \leq l_a + 2; i++$
      i. $m_{2,i} \leftarrow i - 2$
      End For.
   c. For $i = 2; i \leq l_b + 2; i++$
      i. $m_{i,2} \leftarrow i - 2$
      End For.
   d. For $i = 3, i \leq l_a + 2; i++$
      i. For $j = 3 \leq l_b + 2; j++$
         1. If $\left(s_{a_i} = s_{b_j}\right)$
            a. $m_{ji} = 1 + m_{j-1,i-1}$
         2. Else
            a.
$$m_{j,i} = max\begin{cases} m_{j-1,i} + 1 & //Eliminación\ de\ un\ caracter// \\ m_{j,i-1} + 1, & //Inserción\ de\ un\ carácter// \end{cases}$$
            End if

End For.
End For.
End if.

4. $d\ (s_a, s_b) \leftarrow m_{l_b, l_a}$
5. $i = 2$
6. $j = 2$
7. While $i \leq l_a + 2$
   a. While $j \leq l_b + 2$
      i. Situarse en la celda $m_{j,i}$
      ii. $t \leftarrow max(m_{j+1,i}, m_{j+1,i+1},\ m_{j,i+1})$
      /*En caso de que las tres tengan el mimo valor se toma como elemento mayor el que este en diagonal.*/
      iii. If $t = m_{j+1,i}$
         1. Trazar una flecha de origen $m_{j,i}$ destino $m_{j+1,i}$
         2. $j = j + 1$;
      iv. Else if $t = m_{j+1,i+1}$
         1. Trazar una flecha de origen $m_{j,i}$ destino $m_{j+1,i+1}$
         2. $j = j + 1$;
         3. $i = i + 1$;
      v. Else
         1. Trazar una flecha de origen $m_{j,i}$ destino $m_{j,i+1}$
         2. $i = i + 1$;
      End if
   End While
End While.

Elaborado a partir de García, 2007.

La complejidad del algoritmo de LCS es de $O(l_a * l_b)$. Para ilustrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 8. Al determinar la distancia Indel de las secuencias $s_a = \{M.A, R, Y\}$ y $s_b = \{\mathrm{M, A, R, I, A}\}$ se obtiene

Paso 3 inciso a. Construcción de la matriz.

	0	M	A	R	Y
0	∞	∞	∞	∞	∞
M	∞	∞	∞	∞	∞
A	∞	∞	∞	∞	∞
R	∞	∞	∞	∞	∞
I	∞	∞	∞	∞	∞
A	∞	∞	∞	∞	∞

Paso 3 incisos b y c.

	0	M	A	R	Y
0	0	1	2	3	4
M	1	∞	∞	∞	∞
A	2	∞	∞	∞	∞
R	3	∞	∞	∞	∞
I	4	∞	∞	∞	∞
A	5	∞	∞	∞	∞

Paso 3 incisos d.

	0	M	A	R	Y
0	0	1	2	3	4
M	1	1	2	3	4
A	2	2	2	3	4
R	3	3	3	3	4
I	4	4	4	4	4
A	5	5	5	5	5

Paso 7.

	0	M	A	R	Y
0	0	1	2	3	4
M	1	1	2	3	4
A	2	2	2	3	4
R	3	3	3	3	4
I	4	4	4	4	4
A	5	5	5	5	5

La distancia Indel es de 5. Y el alineamiento encontrado entre ambas secuencia es:

s_a	M	A	R	Y	-
s_b	M	A	R	I	A

E *Distancia de Damerau (Damerau-Levenshtein).*

Se le considera como la generalización del procedimiento de Levenshtein, ya que su única diferencia es la incorporación de la operación de transposición de caracteres adyacentes.

Su objetivo es encontrar el número mínimo de operaciones básicas (trasponer, sustituir, insertar y eliminar) para transformar una secuencia s_a en otra s_b. A continuación se muestra el pseudocódigo para este método.

Algoritmo 4. Distancia de Damerau.

Input: Un par de secuencias $s_a = \{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\}$ y $s_b = \{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\}$ de longitud l_a y l_b respectivamente.

Output: Distancia de Damerau entre ambas secuencias.

1. If $(l_a = 0)$
   a. $d(s_a, s_b) = l_b$
   b. Terminar el algoritmo.
2. Else if $(l_b = 0)$
   a. $d(s_a, s_b) = l_a$
   b. Terminar el algoritmo.
3. Else
   a. Construir una matiz $m_{l_a+2\ x\ l_b+2}$
   // Colocar en la columna uno desde la celda $m_{2,1}$los elementos $s_a \cup 0$
   Colocar en la fila uno desde la celda $m_{1,2}$ los elemento $s_b \cup 0$
   Colocar en el resto de las celdas ∞ //
   b. For $i = 2; i \leq l_a + 2; i++$
      i. $m_{2,i} \leftarrow i - 2$
   End For.
   c. For $i = 2; i \leq l_b + 2; i++$
      i. $m_{i,2} \leftarrow i - 2$
   End For.
   d. For $i = 3, i \leq l_a + 2; i++$
      i. For $j = 3 \leq l_b + 2; j++$
         1. If $\left(s_{a_i} = s_{b_j}\right)$
            a. $cost = 0$
         2. Else
            a. $cost = 1$
         End if
         3. $m_{j,i} \leftarrow min \begin{cases} m_{j-1,i} + 1 & /* \textit{Eliminación de un caracter} */ \\ m_{j,i-1} + 1, & /* \textit{Inserción de un carácter} */ \\ m_{j-1,i-1} + cost & /* \textit{sustitución de un carácter} */ \end{cases}$
         4. If $\left((i > 3)\ \&\&\ (j > 3)\ \&\&\left(s_{a_i} = s_{b_{j-1}}\right)\ \&\&\left(s_{a_{i-1}} = s_{b_j}\right)\right)$
            a. $m_{j,i} \leftarrow min \begin{cases} m_{j,i} \\ m_{j-2,i-2} + cost & /* \textit{transposició} */ \end{cases}$
         End if
      End For.
   End For.
End if.
4. For $i = l_a + 2; i \geq 2; i--$
   d. Buscar el a fila i el elemento más pequeños

e. Buscar en la fila $i-1$ los elementos vecinos al conjunto anterior con valor más pequeño
f. Unir con una flecha los elementos de ambos conjuntos
End For

5. **While** $camino\ de\ flechas \neq \emptyset$
   a. [**def** $d(s_a, s_b) = \sum m_{j,i}$ que se encuentran unidas con una flecha]
   End while
6. **Return** $\boldsymbol{min}\ d\ (s_a, s_b,)$

Elaborado a partir de García, 2007.

La complejidad del algoritmo de Damerau es de $O(l_a * l_b)$. Al comparar el algoritmo 2 con el algoritmo 4 se observa que la diferencia existente entre ellos es la incorporación de la operación de transposición en el paso cuatro del segundo. Para ejemplificar lo anterior considere lo siguiente:

Ejemplo 9. Dadas las secuencias $s_a = \{0,0,1\}$ y $s_b = \{0,1,0\}$. Se desea determinar la distancia de Levenshtein y la distancia de Damerau.

Distancia de Levenshtein	Distancia de Damerau

Paso 1

	0	1	0	0
0	∞	∞	∞	∞
0	∞	∞	∞	∞
0	∞	∞	∞	∞
1	∞	∞	∞	∞

	0	0	1	0
0	∞	∞	∞	∞
0	∞	∞	∞	∞
0	∞	∞	∞	∞
1	∞	∞	∞	∞

Pasos 2, 3, 4

	0	0	1	0
0	0	1	2	3
0	1	0	1	2
0	2	1	1	2
1	3	2	1	2

	0	0	1	0
0	0	1	2	3
0	1	0	1	2
0	2	1	1	2
1	3	2	1	0

Pasos 5,6 7

	0	0	1	0
0	0	1	2	3
0	1	0	1	2
0	2	1	1	2
1	3	2	1	2

	0	0	1	0
0	0	1	2	3
0	1	0	1	2
0	2	1	1	2
1	3	2	1	0

La distancia de Levenshtein implica dos operaciones, las cuales son: substituir el elemento dos de la cadena s_b por un 1 y substituir el elemento tres de la misma cadena por un 0

La distancia de Damerau implica una operación, que es intercambiar los elementos dos y tres de la cadena s_b

Los alineamientos obtenidos por ambos algoritmos son:

Algoritmo de Levenshtein

s_a	-	0	0	1	0
s_b	0	0	0	1	-

Algoritmo de Damerau

s_a	0	0	1	0
s_b	0	0	0	1

Para una matriz de alineamiento denotada como M, el número de columnas en en dicha matriz corresponde a la longitud $\hat{l}$ de las secuencias después de la inserción de los espacios vacios.

El valor del alineamiento (valor de la función objetivo) en una matriz M se denota por $V\,(M)$ y se define utilizando alguna métrica de distancia de la manera siguiente:

$$V\,(M) = \sum_{i=1}^{\hat{l}} d(\,s_{a_i}, \mathrm{s_{b_i}}) \qquad (8).$$

donde:
$d(s_{a_i}, \mathrm{s_{b_i}})$ Es una métrica de comparación de elementos en la i –ésima columna de la matriz de alineamiento (Gusfieeld, 1993).

Definición 14. **Alineamiento óptimo**. Un alineamiento M con un $V(M)$ es óptimo si y sólo si no existe otro alineamiento M' con un $V(M')$ que mejore el valor de la función objetivo.

A continuación se señalan una serie de características de las funciones de distancia.

Definición 15. **Distancia entre secuencias iguales**

$$d\,(s_a, s_b) = 0 \quad \leftrightarrow\ s_a = s_b \qquad (9).$$

Lee, *et al*. 2007

La distancia entre la secuencia s_a y s_b es cero si y sólo si ambas secuencias son iguales $(s_a = s_b)$, es decir, ambas secuencias contienen los mismos elementos en la misma posición, por lo cual no haría falta ninguna operación para transformar una secuencia en la otra.

En los problemas de distancia entre secuencia se busca el mínimo valor de diferencias entre las cadenas del conjunto.

Definición 16. **Cota de distancia.** Dadas las secuencias $s_a = \left\{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\right\}$ y $s_b = \left\{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\right\}$ de longitud l_a y l_b, respectivamente, donde $l_a \neq l_b$, la

distancia entre ambas secuencias quedara contenida dentro del siguiente rango.

$$0 \leq d(s_a, s_b) \leq max(l_a, l_b) \qquad (10).$$

donde:
$d(s_a, s_b)$ es la distancia entre el par de secuencias.
$max(l_a, l_b)$ es la longitud de la secuencia más grande.

F *Matrices de Substitución.*

Son herramientas de comparación generalmente utilizados para secuencias de aminoácidos, y se basan en un análisis de la probabilidad de substitución de un caracter por otro en la secuencia s_a para generar una secuencia s_b.

Dentro de la construcción de las matrices de sustitución se implican los principios de mínima mutación, homología y semi-homología.

Definición 17. Principio de mínima mutación.

En el proceso evolutivo se involucra lo menos posible la ocurrencia de mutaciones idénticas, para individuos de diferentes líneas evolutivas. (Sankoff, Cedergren, & McKay, 1982)

F.1 Matrices de datos de mutación (MDM).

También conocidas como matrices de puntuación. En ellas se aceptan los principios de mutación (cambio puntual o generalizado) (Dayhoff, Barker, & Hunt, 1983). Las matrices de sustitución asemejan la regularidad de que un caracter en cambie por otro caracter en una secuencia s_a en función del tiempo. Estas matrices se utilizan como parámetros de los algoritmos de alineamiento en los cuales cumplen el papel de asignar una determinada puntuación a cada emparejamiento entre las secuencias a alinear.

Es decir en las matrices de puntuación se considera a las diferencias entre secuencias como el resultado de la divergencia evolutiva de los individuos a través del tiempo. Generalmente se utilizan al analizar secuencia proteicas, por lo que para cada par de aminoácidos i, j. (McLachlan, 1971) se determina la frecuencia de cambio a través del análisis de información empírica

Para su elaboración comúnmente se usan como datos de entrada para el análisis estadístico dos fuentes, las cuales son: A) A partir del comparar todos los segmentos de una secuencia s_a contra todos los segmentos de otra secuencia s_b , B) Se utilizan el mejor alineamiento de ambas secuencias. (Dayhoff, Barker, & Hunt, 1983).

Las probabilidades de cambio de un aminoácido en otro se sustenta en la hipótesis de que ambas secuencias son homólogas y sustitución se calcula mediante la siguiente expresión:

$$s(a,b) = \frac{1}{\lambda} log \frac{p_{ab}}{f_a * f_b} \qquad (11).$$

Sean 2004

donde:

$s(a,b)$: Diferencia para el alineamiento del caracter a con b.

$p_{a,b}$: La probabilidiad de sustitución del aminoacido a por el aminoácido b de acuerdo la experincia en observacíones.

f_a : La frecuencia de aparición del aminoácido a.

f_b : La frecuencia de aparición del aminoácido b

λ : Factor de escala.

Por lo tanto, el denominador es la probabilidad de que ambos aminoácidos queden alineados por casualidad (Sean, 2004). El cociente entre ambas probabilidades puede resultar:

- Mayor que uno, lo que indica que la probabilidad observada de sustitución entre aminoácidos es superior a la aleatoria y según su magnitud, podría asumirse, que en el proceso evolutivo se ha aceptando tal intercambio.

- Igual a uno lo que se refiere a que la sustitución entre uno y otro aminoácidos corresponde a un proceso aleatorio de mutación.
- Menor que uno lo que indica según la magnitud del cociente que esta sustitución no es aceptada evolutivamente.

Las matrices PAM y BLOTUS son ejemplos de las matrices de sustitución. Las matrices PAM son útiles para comparar especies cercanamente relacionadas, mientras las matrices BLOTOUS se usan para alineamientos de proteínas evolutivamente divergentes. A continuación se describe brevemente cada una de ellas

Se observan mejores resultados cuando la tabla refleja el patrón de equivalencia de residuos que se da en secuencias relacionadas estructural, funcional o evolutivamente.

Matrices PAM.

Desarrolladas por Dayhoff, donde se propone un esquema de puntuación evolutiva para el alineamiento de los elementos de la s_a con los elementos de la s_b utilizando información estadística.

A mayor distancia evolutiva, la frecuencia de los cambios conservados crece y las coincidencias entre elementos disminuye. Asemejando un proceso de diversificación evolutiva.

La puntuación para cada pareja de elementos se calcula con el logaritmo del cociente de la probabilidad de ocurrencia del par y la probabilidad esperada basada en la frecuencia de cada aminoácido. Estos cocientesfueron una importante contribución de Dayhoff.

Matrices BLOSUM.

Desarrolladas por Altschul en 1991, basadas en la premisa: "no se conoce *a priori* el grado de semejanza existente entre las secuencias que desea comparar, por lo que resulta indispensable diseñar un esquema de puntuación ad hoc".

Altschul demostró que la matriz PAM 250 resultaba poco útiles en algunas situaciones. La hipótesis básica en el modelo de Dayhoff establece que las relaciones lejanas entre secuencias pueden modelarse con suficiente precisión desde la información de secuencias estrechamente relacionadas.

La base para esta serie de matrices de puntuación es a partir de alineamientos de bloques de secuencias. Un *bloque* es una matriz cuyas filas representan segmentos de secuencias proteicas alineadas sin interrupciones, donde se indica el umbral de identidad utilizado para la seleccionar los bloques de alineamiento.
Otra diferencia significativa es que las matrices PAM asumen un proceso markoviano para la sustitución de aminoácidos (Leluk, 1998). Mientras que las matrices BLOSTUM no se basan en ningún modelo explícito de evolución, y consideran secuencias de proteínas empíricamente relacionadas que comparten un antepasado común.

1.1.5 Alineamiento y patrones.

En términos generales el alineamiento es un procedimiento de comparación de dos o más secuencias, lo cual se logra al determinar una serie de caracteres individuales o patrones de caracteres que se encuentren en el mismo orden en el conjunto de secuencias (Agüero, 2004).

Por lo tanto el caso más sencillo del AMS se presenta al trabajar con dos cadenas, y por ende se usara éste caso para explicar las implicaciones del alineamiento múltiple.

En términos generales el alineamiento de s_a y s_b, se lograr al insertar espacios vacíos (guiones) en una u otra de las secuencias según convenga, de forma tal

que se logre el mayor número de coincidencias y diferencias mínimas entre los caracteres.

Definición 18. El **alineamiento de dos secuencias** dadas $s_a = \left\{s_{a_1}, s_{a_2}, \dots, s_{a_{l_a}}\right\}$ y $s_b = \left\{s_{b_1}, s_{b_2}, \dots, s_{b_{l_b}}\right\}$ implica determinar una matriz M_{2*x} tal que $x \geq Max\,(l_a, l_b)$, cuyos elemento en la matriz M pertenecen al conjunto $\beta = \mathcal{A} \cup \{-\}$ y ninguna columna de M consta completamente de espacios vacios. Al eliminar todos los guiones del primer y segundo renglón de M son iguales a s_a y s_b respectivamente (Lee, Tseng, Chang, & Tsai, 2007). Puede observarse que la definición sobre el alineamiento de dos sentencias es un caso particular del problema AMS.

En el caso que un caracter u de la cadena s_a sea alineado con un guión de la secuencia s_b, se interpreta como la eliminación del elemento u de la primera secuencia o la inserción de u en la segunda secuencia (Gusfieeld, 1993). Lo anterior se formaliza mediante la siguiente definición.

Definición 19. **Inserción y eliminación de guiones**. Si en la secuencia $\widehat{s_a}$ la posición $\widehat{s_{a_i}}$ corresponde a un vacío mientas que en la cadena $\widehat{s_b}$ la posición $\widehat{s_{b_i}}$ es un elemento u entonces dicha operación es equivalente a insertar el caracter u de la secuencia $\widehat{s_a}$. Si en la secuencia $\widehat{s_a}$ la posición $\widehat{s_{a_i}}$ corresponde a un caracter u mientras que en la cadena $\widehat{s_b}$ la posición $\widehat{s_{b_i}}$ es un guion entonces dicha operación es equivalente a eliminar u de la secuencia $\widehat{s_a}$ (Ma, Wang, & Li, 2007).

El objetivo de la matriz resultante es encontrar una formación que garantice el mejor puntaje con base en una en función objetivo (Lee, Tseng, Chang, & Tsai, 2007)

Con base al procedimiento de alineación puede observarse que existe un conjunto amplio de posibilidades, entonces para lograr el mejor alineamiento posible, es necesario definir una función de evaluación.

Definición 20. **Alineamiento.** Dada una familia de N-secuencias ($s_1 = \left\{b_1^1, b_2^1, \dots, b_{l_1}^1\right\}$, $s_2 = \left\{b_1^2, b_2^2, \dots, b_{l_2}^2\right\}$,…, $s_x = \left\{b_1^x, b_2^x, \dots, b_{l_x}^x\right\}$), se obtiene un alineamiento $M = \{\widehat{s_1}, \widehat{s_2}, \cdots \widehat{s_x}\}$ de secuencias con caracteres sobre β. Donde todas las secuencia de M tienen la misma longitud y cada $\widehat{s_a}$ se obtiene a partir de insertar guiones (celdas vacías) a s_a. (Manthey, 2003). Y los elementos de M satiface la ecuación siguiente:

$$\bigcup_{i=max\{l_1,l_2,\cdots,l_x\}}^{l_1+l_2+\cdots+l_x} \prod_{j-1}^{x} \beta^i \qquad (12).$$

(Carrillo & Lipman, 1988)

donde:

β: $\mathcal{A} \cup \{-\}$.

l_i: Longitud de la secuencia i.
x: número de secuencias a comparar.

Sujeta a:

$$\max\{l_1, l_2, \cdots, l_x\} \leq \hat{l} \leq l_1 + l_2 + \cdots + l_x \qquad (13).$$

Un alineamiento de secuencias se obtiene insertando en cada secuencia un número (desde 0) de guiones de forma que: a) Las secuencias resultantes tengan la misma longitud y b) Cada columna tenga por lo menos un caracter diferente al vacío

Los alineamientos de secuencias pueden clasificarse a) de acuerdo con la cantidad de secuencias que analiza y b) de acuerdo al nivel de análisis. A continuación se describe cada uno de ellos **Fuente especificada no válida.**.

1. Número de secuencias analizadas:
    i. Alineamiento de un par de secuencias: en el sólo se analizan dos secuencias.
    ii. Alineamiento múltiple: Se analizan tres o más secuencia y el resultado es una secuencia consenso, esta secuencia media.
2. Nivel de análisis:
    i. Alineamiento global: Consiste en buscar la conservación de subsecuencias.
    ii. Alineamiento local: Consiste en buscar las coincidencias de eslabón a eslabón (ácidos nucleídos o aminoácidos).

De forma general, los patrones pueden ser divididos en determinísticos o probabilísticos.

Los determinísticos se definen como el ajuste para un determinado patrón, generalmente l son modelos generalmente binarios y se dividen en:

- **Los Oligos:** se reconocen como la función que corresponde a 1si ambos caracteres en el i −ésimo lugar de las secuencias comparadas son iguales y 0 en otro caso. Este modelo fueron comúnmente usado en los primeros métodos aplicados para el descubrimiento de motivos y actualmente se utiliza como método para conteo de letras.

- **Expresiones regulares:** retorna 1 si existe una subsecuencia idéntica a la expresión regular dada. Estos modelos son usados en el descubrimiento de motivos para composición de símbolos exactos, símbolos ambiguos, espacios fijos o flexibles.

- **Expresiones discordantes**: estos modelos evalúan si el número de discordancias (distancia Hamming) entre una subsecuencia y la subsecuencia consenso.

En los patrones probabilísticos, para cada secuencia se genera una probabilidad de ocurrencia a partir de un modelo (matrices MDM). Dichos patrones ofrecen la ventaja de representar de forma implícita las reglas de discriminación.

En la detección de patrones, existen tres diferentes niveles establecidos según su grado de dificultad (Restrepo-Montoya)

1. Detección de patrones para una secuencia dada, dicho patrón se debe describir por medio de un algoritmo.

2. Esquemas. Es un modelo de un patrón, se establece la estructura de los elementos invariantes, los cuales se ubican dentro del conjunto datos.

3. Probar patrones. En un conjunto de datos secuenciados de los que se desconoce el patrón, son comparados con un patrón x y se verifica el grado de adaptación de los datos. Esta categoría se reconoce como "aprendizaje no supervisado".

En la siguiente sección se ocuparan las ideas anteriores sobre secuencias, similitud, distancia y alineamiento para describir las formas de expresión del AMS como un problema de optimización.

1.2 Características del alineamiento múltiple de secuencias.

1.2.1 Función objetivo.

Dentro de la literatura no se ha generalizado el uso de una función única para medir la calidad del alineamiento en N-secuencias (Ma, Wang, & Li, 2007) & (Gusfieeld, 1993). Las funciones más conocidas son: A) Funciones de puntaje (penalización por espacios vacíos y suma por pares de las diferencias B) Alineamiento por templete (alineamiento por secuencia de consenso) y C) Alineamiento por secuencia media. A continuación se describen algunas de ellas.

Un rasgo común a todas las funciones objetivo es el uso de la métrica entre secuencias, con objeto de determinar el grado de similitud entre ellas.

1.2.1.1 Funciones de puntaje.

A *Penalización por espacios vacíos*

Es un problema NP-Duro[6]. Donde se considera un costo asociado a la introducción o expansión de espacios vacios en las secuencias, en el cual resulta ser más costoso introducir un nuevo espacio vació, que con sólo agrandar uno espacio ya existente (Agüero, 2004). El objeto de esta función objetivo es minimizar los costos asociados a los guiones.

Definición 21. **Función objetivo penalización por espacios vacíos**.

$$Min\ z\colon \sum_{\substack{a=1 \\ a\neq b}}^{x}\sum_{b=1}^{x} f(s_a, s_b) \qquad (14).$$

donde:

x: Número de secuencias.

s_a, s_b : Son un par de secuencias cualesquiera del conjunto de interés.

$f(s_a, s_b)$: Es la función asociada a los costos asociados por introducir y por extensión de espacios vacíos; dicho costos deben ser valores positivos.

Para ilustrar lo anterior considere el siguiente ejemplo.

Ejemplo 10. Dadas las secuencias $\widehat{s_a} = \{A, C, T, G, A\}$ Y $\widehat{s_b} = \{-, C, -, -, A\ , C\}$ y la función de similitud siguiente:

$$f(s_a, s_b)\begin{cases} 2 & costo\ por\ introducir\ un\ nuevo\ espacio\ vacio. \\ 1 & costo\ por\ extención\ de\ un\ espacio\ vacio. \\ 0 & en\ otro\ caso. \end{cases}$$

Se desea determinar el valor de la función objetivo.

Alineamiento

Secuencia 1	A	C	T	G	A	-	Costo por vacío= 1*(2)+2*(1)=4
Secuencia 2	-	C	-	-	A	C	Costo por vacío= 2*(2)+1(3)=7

Sea el costo total es igual a la suma del costo de la primera secuencia más el costo de la segunda. Por lo tanto el costo de la alineación anterior es de 11.

B *Suma por pares de las diferencias.*

Es una función de puntaje y se le conoce en la literatura como SP-score (por sus siglas en inglés). I. Elías demostró que el problema de AMS es NP-duro para toda formulación con una función SP-diferencias (Manthey, 2003).

[6] Vease apéndice B Complejidad orden y tipo de problemas.

Un alineamiento múltiple de un conjunto $\mathcal{F}$ de secuencias N, M es la matriz $\mathcal{F}\ x\ \hat{l}$, donde la i-ésima fila contiene a la cadena i-ésima con espacios los espacios insertados. La suma por pares de la diferenciaría para una alineación múltiple es la suma de las distancias para todo par de hileras en la alineación. Por lo que el problema es encontrar un mínimo de la alineación (Elias, 2003). En la definición de distancia se establece las simetría como una condición del problema.

Definición 22. **Suma por pares de las diferencias**.

$$Min\ z\colon \sum_{\substack{a\neq b\\ a,b\in\mathcal{F}}}^{x}\sum_{j=1}^{\hat{l}} d\left(s_{a,j}, s_{b,j}\right) \qquad (15).$$

donde:
x: número de secuencias.
$\hat{l}$: longitud de las secuencias después de insertar las celdas vacías.
$d\left(s_{a,j}, s_{b,j}\right)$: Distancia entre un par de secuencias.

Para ilustrar lo anterior considere lo siguiente:

Ejemplo 11. Dada la siguiente función de distancia

$$d\left(s_{a,j}, s_{b,j}\right)\colon \begin{cases} 0 & si\ \ s_{a,j} \neq s_{b,j}\ y\ ambos\ caracteres. \\ 1 & si\ \ s_{a,j} = s_{b,j}\ y\ ambos\ caracteres. \\ 2 & si\ \ s_{a,j}\ \text{ó}\ s_{b,j}\ son\ guiones. \\ 3 & si\ \ s_{a,j} = s_{b,j}\ son\ guiones \end{cases}$$

Se desea determinar cuál de los siguientes alineamientos es mejor, alineamiento con base al valor de $V(M)$ para el problema del ejemplo 2.

A)

-	L	I	D	I	A
E	L	I	S	A	-
-	L	I	S	-	-

B)

-	L	I	D	I	A
E	L	I	S	-	A
-	L	I	S	-	-

El valor de alineamiento para ambos casos se expresa mediante la siguiente relación.

$$V(M) = \sum_{j=1}^{6} d(s_{1,j}, s_{2,j}) + \sum_{j=1}^{6} d(s_{1,j}, s_{3,j}) + \sum_{j=1}^{6} d(s_{2,j}, s_{3,j}) + \sum_{j=1}^{6} d(s_{2,j}, s_{1,j}) + \sum_{j=1}^{6} d(s_{3,j}, s_{1,j}) + \sum_{j=1}^{6} d(s_{3,j}, s_{2,j})$$

Obtención del $V(M)$ para ambos casos.

Caso A

$d(s_{1,j}, s_{2,j})$= 2+0+0+1+1+2=6
$d(s_{1,j}, s_{3,j})$= 3+0+0+1+2+2=8
$d(s_{2,j}, s_{1,j})$= 2+0+0+1+1+2=6
$d(s_{2,j}, s_{3,j})$= 2+0+0+0+2+3=7
$d(s_{3,j}, s_{1,j})$= 3+0+0+1+2+2=8
$d(s_{3,j}, s_{2,j})$= 2+0+0+0+2+3=7

Caso B

$d(s_{1,j}, s_{2,j})$= 2+0+0+1+2+0=5
$d(s_{1,j}, s_{3,j})$= 3+0+0+1+2+2=8
$d(s_{2,j}, s_{1,j})$= 2+0+0+1+2+0=5
$d(s_{2,j}, s_{3,j})$= 2+0+0+0+3+2=7
$d(s_{3,j}, s_{1,j})$= 3+0+0+1+2+2=8
$d(s_{3,j}, s_{2,j})$= 2+0+0+0+3+2=7

La información anterior se organiza en matrices de distancia, obteniéndose lo siguiente:

Matriz de distancia caso A

	s_1	s_2	s_3
s_1	0	6	8
s_2	6	0	7
s_3	8	7	0

Matriz de distancia caso B

	s_1	s_2	s_3
s_1	0	5	8
s_2	5	0	7
s_3	8	7	0

$V(M)_{caso\ A}$=6 +8 +7+6+8+7=42

$V(M)_{caso\ B}$=5 +8 +7+5+8+7=40

Dado que $V(M)_{caso\ B} < V(M)_{caso\ A}$ se puede inferir que el alineamiento B es mejor que el alineamiento A. En virtud del alineamiento B es dos unidades más económico que A.

Definición 23. Función objetivo de la suma por pares de las diferencias.

$$Min\ z: \sum_{\substack{a=1 \\ a \neq b}}^{x} \sum_{b=1}^{x} d(s_a, s_b) \qquad (16).$$

La suma de las parejas de un alineamiento múltiple es la suma de las puntuaciones de todos los, pares implicados; es decir es la suma de las celdas de la matriz de distancia excluyendo los elementos de la diagonal.

1.2.1.2 Aproximación por templetes.

Estas aproximaciones utilizan la idea de subcadenas comunes a todas las secuencias de un conjunto, las cuales se han preservado a través del proceso evolutivo de los organismos. Es decir, los templetes son elementos invariantes a todas las cadenas del conjunto.

A *Alineamiento por consenso.*

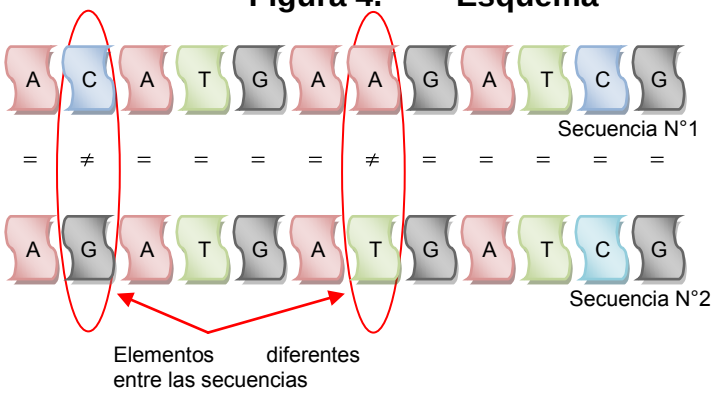
El alineamiento por consenso es un problema NP-duro para cualquier esquema arbitrario y para donde las diferencias tengan un costo de 1 y las coincidencias un costo de 0. (Elias, 2003). Antes de continuar se definirá el concepto de esquema.

Definición 24. **Esquema.** Dado el conjunto $N = \{s_1, s_2 \cdots s_x\}$ de secuencias que poseen elementos comunes en algunas posiciones. Es posible construir una secuencia representativa (s_*) de dicho conjunto tal que contenga un caracter en la i –ésima posición si y sólo si dicho caracter es invariante para todo el conjunto y un comodín $(*)$ en caso contrario. (Kuri Morales & Galaviz Casas, 2002). Lo cual se ilustra en la figura 4.

El número de posiciones invariantes se denomina orden del esquema (H) y es la distancia entre la primera y última posición explícitas que denota la longitud.

Un esquema determina el grado de convergencia de un conjunto de cadenas, la similitud entre estas aumenta a medida que los datos son más parecidos entre sí y permiten denotar los rasgos comunes de una familia de secuencias.

Figura 4. Esquema



Cadena esquema de las secuencias.

Para ilustrar lo anterior considera lo siguiente:

Ejemplo 12. Se desea representar mediante un esquema la familia de cadenas de longitud 3 de caracteres binarios donde la primera posición la ocupa un cero y la segunda sea un uno.

Por lo tanto el esquema correspondiente es {0,1,*} donde los caracteres (numéricos) representan los elementos invariantes, y el comodín * representa a los elementos variantes del conjunto.

Definición 25. **Función objetivo para el alineamiento por consenso**

$$min\ z: \sum_{\substack{b=1 \\ s_b \epsilon\, x}}^{x} D(s_*, s_b) \qquad (17).$$

donde:
x: número de secuencias
$D(s_*, s_b)$: Distancia entre el esquema y una secuencia del conjunto.

El esquema debe cumplir con la desigualdad del triángulo que implica que dadas las cadenas s_*, s_a, s_b entonces:

$$d(s_a, s_b) + d(s_a, s_*) \geq d(s_b, s_*) \qquad (18).$$

(Lee, Tseng, Chang, & Tsai, 2007)

B *Alineamiento por secuencia media.*

Uno de los principales problemas en la construcción del esquema exacto para un conjunto de secuencias, se debe a la exigencia de examinar todas las secuencias de la familia de interés. Para evitar lo anterior, se utiliza la secuencia media que es una cadena de la familia y que satisface la desigualdad del triángulo, además de minimizar la distancia entre las secuencias.

Definición 26. **Secuencia media**. Dada una familia de secuencia $N = \{s_1, s_2 \cdots s_x\}$ la secuencia media es aquella $s_c \in N$ y que satisface la siguiente relación:

$$min\, z: \sum_{\substack{b=1 \\ s_c \in N \setminus (s_b)}}^{x} D(s_c, s_b) \qquad (19).$$

(Lee, Tseng, Chang, & Tsai, 2007), (Elias, 2003) & (Gusfieeld, 1993)
donde:
$D(s_m, s_b)$ Distancia entre un esquema arbitrario (secuencia media) y una secuencia del conjunto.
x: número de secuencias

Dado que la secuencia media es un esquema arbitrario del conjunto, esta función objetivo induce a un problema NP-duro.

Existe un problema con el alineamiento por secuencia media, ya que al imponer un esquema arbitrario a todas las secuencias no puede garantizarse el alineamiento óptimo.

Definición 27. **Error por secuencia media**.

$$\frac{V(M_c)}{V(M^*)} \leq \frac{2(x-1)}{x-2} \qquad (20).$$

(Gusfieeld, 1993)

donde:
$V(M_c)$: Valor del alineamiento por secuencia media.
$V(M^*)$: Valor óptimo del alineamiento.
x: Número de secuencia del conjunto que debe ser mayor a 2.

1.2.2 Restricciones del problema.

En la definición de alineamiento, distancia y valor de alineamiento se incluyen restricciones explícitas para las secuencias del conjunto, las cuales son:

1. Todas las secuencias que se va alinear debe pertenecer a una familia.(definición de familia de secuencias).

$$\forall\, a \qquad s_a \in N \qquad \forall\, s_a = 1,2,\dots,x \qquad (21).$$

2. La longitud de las cadenas después de agregar los espacios vacíos debe ser la misma para todas que integren la matriz M.

$$\forall\, \widehat{s_a}\ y\ \widehat{s_b} \in \mathrm{M} \qquad \widehat{l_a} = \widehat{l_b} \qquad (22).$$

3. La distancia entre cualquier par de secuencias es no negativa

$$\forall\, s_a\ y\ s_b \qquad d(s_a, s_b) \geq 0 \qquad (23).$$

4. La distancia entre cualquier par de secuencias es simétrica.

$$\forall\, s_a\ y\ s_b \qquad d(s_a, s_b) = d(s_b, s_a) \qquad (24).$$

5. La distancia para dos secuencias iguales es cero, si y solo sí ambas secuencias son idénticas.

$$d\,(s_a, s_b) = 0 \qquad \leftrightarrow\ s_a = s_b \qquad (25).$$

(Lee, Tseng, Chang, & Tsai, 2007)

9)

6. Al seleccionar la distancia entre dos secuencias debe satisfacerse la desigualdad del triángulo.

$$\forall\ s_a, s_b\ y\ s_c \qquad d(s_a, s_b) + d(s_b, s_c) \geq d(s_a, s_c) \qquad (26).$$

7. El procedimiento para determinar la métrica de distancias dentro de un experimento, debe ser el mismo para todas las secuencias.

Pueden agregarse restricciones que representen los conocimientos biológicos con los que se cuenten, es decir, si se conoce que dentro del conjunto existen subconjuntos estrechamente relacionados biológicamente (géneros, especies etc.), se imponen restricciones de pertenecía a subgrupos. Con lo anterior se consigue acotar mejor el universo de posibilidades. Sin embargo el realizar suposiciones erróneas (sin evidencia tangible) puede llevarse a soluciones aberrantes. (Sankoff, Cedergren, & McKay, 1982)

1.3 Resumen.

En la sección 1.1 de este capítulo se presentaron algunas de las ideas y conceptos involucrados en el problema de alineamiento múltiple de secuencias AMS, el cual se define como aquel problema que involucra acomodar un conjunto de $N - secuencia$ con la finalidad de generar una matriz $M_{\mathrm{N\ x\ k}}$ cuyas celdas contienen caracteres sobre $\mathcal{A} \cup \{-\}$ de tal manera que ninguna columna de la matriz M conste completamente de caracteres vacios y el resultado de eliminar los guiones de la i −ésima hilera de M sea la i −ésima secuencia correspondiente.

El AMS es un problema de optimización combinatoria del tipo NP-completo[7]. Generalmente, las métricas de la similitud o la distancia total del conjunto de secuencias ya alineadas se utilizan como criterios de decisión al comparar dos o más alineamientos factibles de un conjunto de $N - secuencia$, . Puede entenderse como similitud entre dos secuencias como el nivel de proximidad existente entre ellas y como distancia el número de operaciones básicas necesarias para transformas una secuencia s_a en otra s_b.

En la sección 1.1.4.1 se expusieron algunos de los métodos para determinar la similitud o distancia entre pares de secuencias los cuales son:

1. Matriz de puntos.
2. Distancia de Hamming.
3. Distancia de Levenshtein.
4. Distancia Indel.
5. Distancia de Damerau.
6. Matrices de sustitución.

[7] Véase apéndice B

En virtud de que no existe una representación estándar del AMS como un problema de optimización en la sección 1.2 se presentaron las formulaciones posibles del AMS. En el conjunto posible de modelos del AMS existe como rasgo común el uso de una función objetivo, que utilicé algún tipo de métrica, para establecer del grado de similitud entre las secuencias.

Las funciones objetivo más usuales para el AMS se clasifican en: A) Funciones de puntaje (penalización por espacios vacíos y suma por pares de las diferencias B) Alineamiento por templete (alineamiento por secuencia de consenso y C) Alineamiento por secuencia media.

En el capítulo siguiente se utilizarán lo conceptos e ideas presentados en este apartado con la finalidad de analizar los métodos desarrollados para resolver el AMS, así como presentar la metodología desarrolladla por Julie Thompson para la comparación de los procedimientos de solución.

Capítulo 2 Estado del arte de: los métodos de solución del AMS, la búsqueda de armonía y el recocido simulado.

A medida que crece el número de secuencias biológicas conocidas, la necesidad de disponer de mecanismos para analizar dichas estructuras también se incrementa. A raíz de esta necesidad han surgido una gran variedad de métodos. Los modelos desarrollados hasta la actualidad implican minimizar una función distancia o maximizar una función de similitud.

En este capítulo se introducen las ideas y conceptos relacionados con los métodos de solución del AMS, así como una descripción de los métodos heurísticos que se utilizarán como base para el desarrollo del nuevo procedimiento de solución, por lo tanto, la función del presente capítulo es: "Presentar y analizar algunos de los métodos para resolver el AMS para biosecuencias, mostrar el procedimiento de validación desarrollado por Julie Thompson y analizar la metaheurísticas de búsqueda de armonía y recocido simulado", dicha función se relaciona directamente con los objetivos particulares primero y segundo de la tesis.

Para alcanzar la función de este capítulo se utiliza la siguiente táctica:

1. Definir y clasificar los métodos de solución del AMS de acuerdo con sus características.

2. Analizar algunos de los procedimientos desarrollados para solucionar el AMS, enfatizando el los métodos de programación dinámica, ya que estos son la base del funcionamiento de las técnicas heurísticas y metaheurísticas generadas para solucionar el AMS.

3. Presentar algunos de los programas de cómputo utilizados para resolver el AMS relacionándolos con el tipo de algoritmo de solución que utilizan.

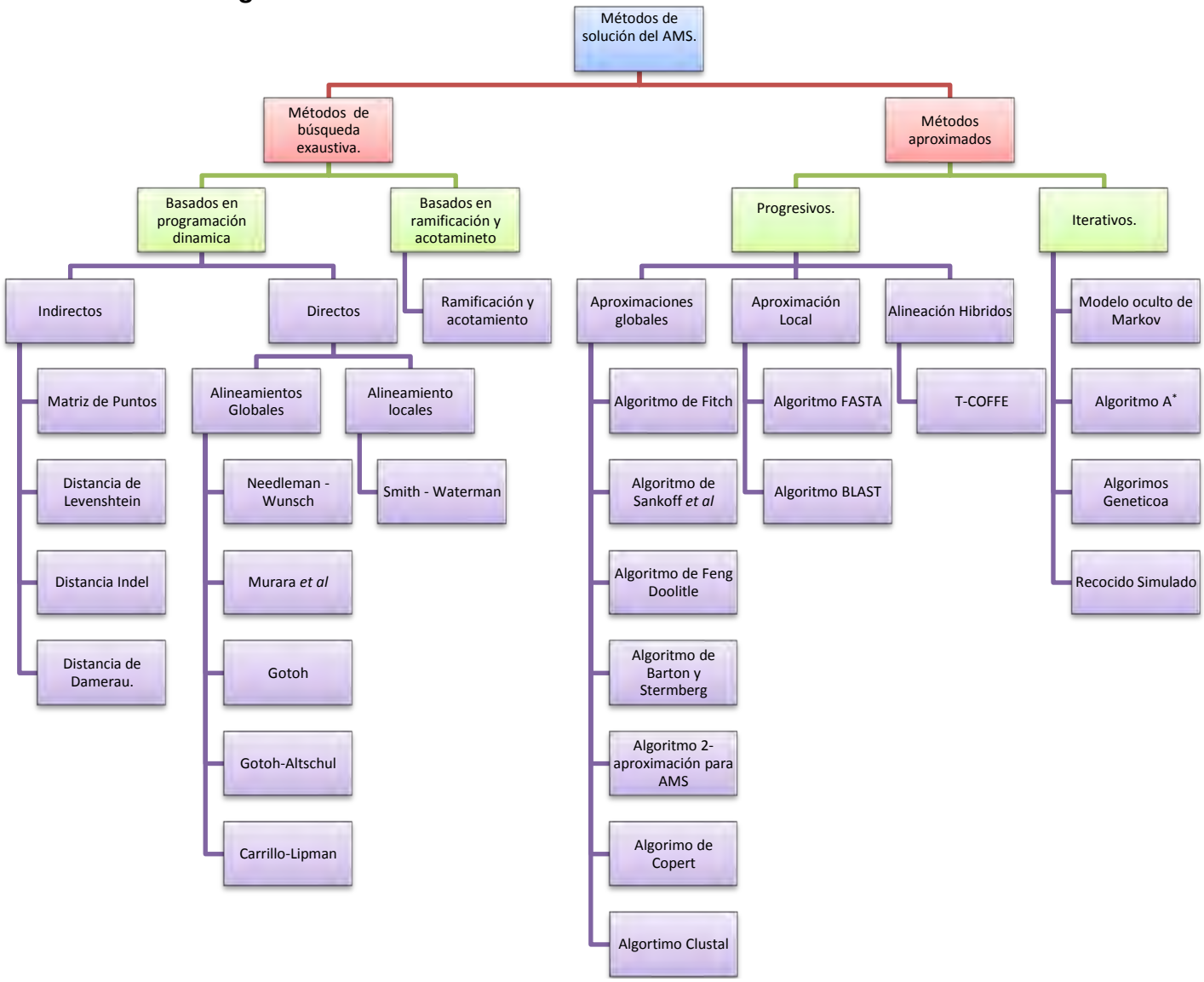
4. Caracterizar el método de Julie Thompson para la comparación de algoritmos y programas de computó a través de BALiBASE.

5. Describir y analizar la metaheurística de búsqueda de armonía.

6. Describir y analizar la metaheurística de recocido simulado.

En la figura 5 se muestran los métodos de solución analizados en el presente capítulo, reiterando que no son los únicos pero sí son los más usuales para el caso de biosecuencias.

Figura 5. Métodos de solución del AMS analizados.



2.1 Conceptos básicos.

El problema AMS es NP-duro[8], por lo que, se deben realizar relajaciones en la formulación de la función objetivo o utilizar alternativas de solución aproximada. Por ende, generalmente se hace uso de técnicas Heurísticas o Metaheurísticas para resolverlo.

Solamente se justifica el uso de técnicas heurísticas:

a. Por la naturaleza del problema: pues no se conoce ningún método exacto de solución.
b. El uso del método exacto resulta muy costoso (tiempo o espacio)
c. Los datos son poco fiables, o se hace uso de un modelo demasiado simplificado de la realidad.
d. Como paso intermedio para otras aplicaciones

Los métodos heurísticos usan dos pasos básicos: suponer y comprobar. El objeto de usar técnicas heurísticas y metaheurísticas en problemas de optimización es encontrar buenas soluciones en un tiempo razonable.

Definición 28. Las **técnicas heurísticas:** son métodos o procedimientos para resolver un problema, que no son producto de un riguroso análisis formal. En la investigación de operaciones es un procedimiento para el que se tiene un alto grado de confianza en que encontraran soluciones de alta calidad con un costo computacional razonable, aunque no garantiza optimalidad o factibilidad (Melián, Moreno Peréz, & Moreno Vega, 2003).

Definición 29. Las **técnicas metaheurísticas** son estrategias inteligentes para delinear mecanismos y mejoras a procedimientos heurísticos generales para resolver problemas con un alto rendimiento (Melián, Moreno Peréz, & Moreno Vega, 2003). Es decir, son métodos aproximados para resolver problemas de optimización, en los cuales las heurísticas clásicas no son efectivas (Vélez & Montoya, 2007).

En lo subsecuente se utilizaran estos conceptos ya que la gran parte de los métodos de solución del AMS son técnica heurísticas y metaheurísticas.

2.2 Métodos de solución para el AMS.

Existen varias formas de clasificar a los algoritmos utilizados para resolver el AMS, las más conocidas son las de Wang y Omar (clasifica de acuerdo a la estrategia de búsqueda) y Chan (clasifica de a cuerdo con el tipo de búsqueda).

[8] Véase apéndice B

Wang y Li clasifican a los algoritmos para alinear secuencias en progresivos e iterativos.**Fuente especificada no válida.**.

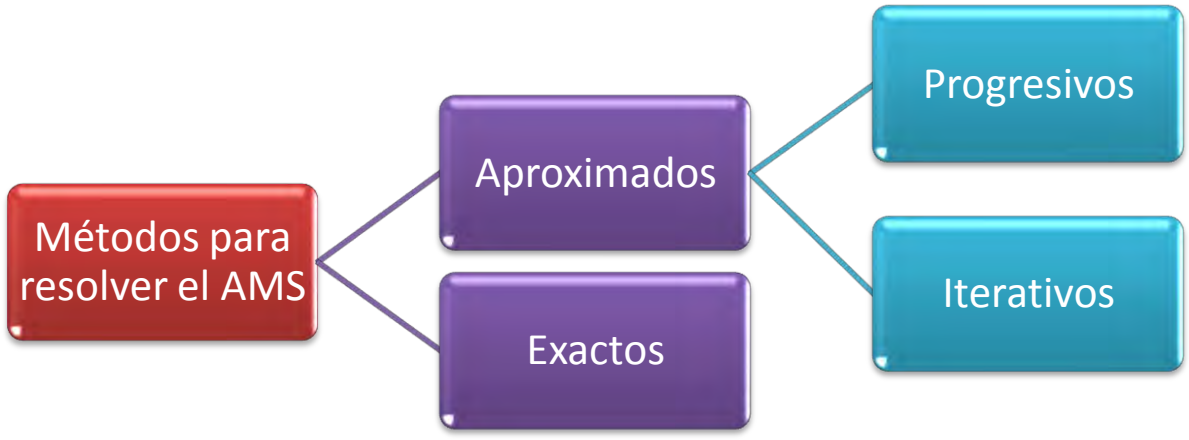
- **Los método progresivos**: también denominados jerárquicos, o por árbol, son aquellos que generan un AMS a través de alinear primero las secuencias más parecidas entre sí, para ir añadiendo sucesivamente al alineamiento secuencias o grupos menos relacionados, hasta que el total de las secuencias ha sido incorporado a la solución.

  Los métodos progresivos consisten en minimizar la distancia entre secuencias medias (o templetes). Los resultados del alineamiento dependen de la clasificación inicial de las secuencias en grupos, por lo que, son sensibles a imprecisiones en los alineamientos iniciales. Para mitigar esta desventaja la mayoría de los métodos progresivos de AMS ponderan adicionalmente, las secuencias en el conjunto problema de acuerdo a su parentesco, lo que reduce la probabilidad de efectuar una pobre elección de las secuencias iniciales y así se mejora la precisión del alineamiento.

- **Los métodos iterativos**: no requieren mucha precisión del alineamiento inicial en virtud de que estos métodos optimizan una función objetivo basada en estrategias de puntuación del alineamiento. El alineamiento global inicial se calcula mediante la función de puntuación y considera los alineamientos de subconjuntos de secuencias. Entonces, se realinean cada uno de los subconjuntos para producir la siguiente iteración de AMS.

Por otra parte Omar *et al* (Omar, Salam, Abdullah, & Rashid, 2005) clasifican a los algoritmos utilizados para resolver el AMS en : exhaustivos o exactos , progresivos e iterativos. En la figura siguiente se esquematiza la clasificación de Omar.

Figura 6. Clasificación de los Algoritmos utilizados para resolver el AMS.



Elaborado a partir de Omar, *et al*. 2005

A continuación se da una breve descripción de cada una de estas categorías:

- **Los métodos exactos:** son procedimientos que considera el problema de AMS como una generalización del problema de alineación de pares de secuencias. Dichos métodos se basan en programación dinámica donde se comparan simultáneamente n secuencias a través del uso de una matriz $n - dimensional$ de programación dinámica. Y el alineamiento es resultado de la comparación uno a uno de los elementos de cada una de las secuencias. Con el objetivo de determinar la secuencia media del conjunto de interés, sin embargo este también es un problema NP-duro. Por lo que sólo resultan útiles al trabajar con un número reducido de secuencias, aunque garantizan llegar al óptimo (Omar, Salam, Abdullah, & Rashid, 2005). Ejemplos de este tipo de algoritmos son: Needleman-Wunsch , Gotoh, Lipman, Smith-Waterman, etc.

- **Los métodos aproximados:** estos métodos a su vez se dividen en progresivos e iterativos. Los **métodos progresivos** son las técnicas más sencillas para encontrar una solución al AMS. Generalmente utilizan métodos de programación dinámica para llevar a cabo la alineación de las secuencias del conjunto de interés, en función del grado de relación entre ellas. Varios de los programas de cómputo utilizan este tipo de algoritmos por la velocidad y la simplicidad que presentan. Sin embargo, tienen el inconveniente que dependen de la solución inicial. (Omar, Salam, Abdullah, & Rashid, 2005). Ejemplos de este tipo de algoritmos son: Barton – Sternberg y 2 aproximado.

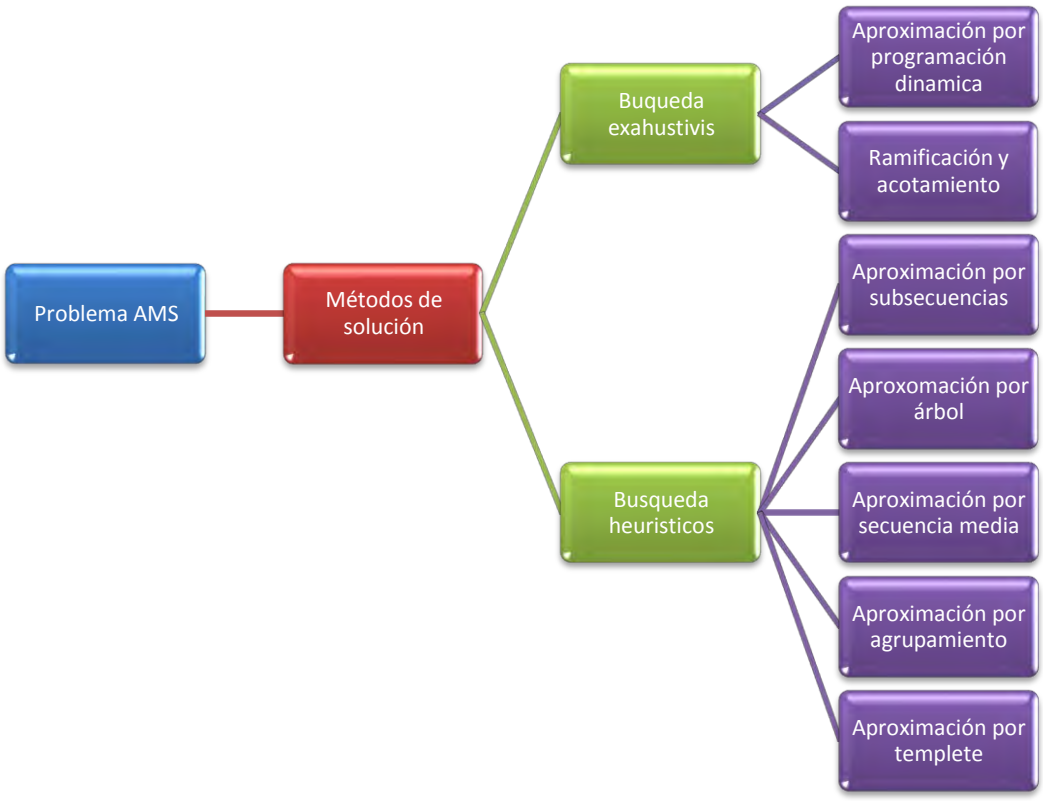
  Mientras tanto, los **métodos iterativos** son algoritmos que utilizan una alineación inicial y la perfeccionan a través de una serie de ciclos (iteraciones) hasta el punto donde no puede hacerse mejora alguna.. Ejemplos de estos algoritmos son el modelo oculto de Markov (HMM), recocido simulado, computación evolutiva, etc. Los métodos iterativos pueden clasificarse en aleatorizados y simulados. Los procedimientos a**leatorizados** son métodos iterativos inspirados en procesos biológicos aleatorios y son: algoritmos genéticos (AG), búsqueda por colonia de hormigas, computación evolutiva, el procedimiento glotón aleatorio de búsqueda (GRASP por sus siglas en inglés), etc. Los métodos aleatorizados han mostrado resultados prometedores al alinear secuencias, que han superado en algunos casos las alineaciones obtenidas por métodos progresivos tradicionales como (Clustal W). Son generalmente de caracter estocástico y básicamente cuentan con los siguientes componentes:

  A) Una representación de la solución al problema.
  B) Una forma de crear una población a partir de una solución inicial.
  C) Una función de evaluación de las soluciones en términos de su aptitud.
  D) Un operadores que alteran la composición de la soluciones.
  E) Valores de los parámetros de algoritmos. (Omar, Salam, Abdullah, & Rashid, 2005)

En contraste los procedimientos **simulados** utilizad el procedimiento Monte Carlo para examinar las ecuaciones de estado y los n-estados del sistema. Ejemplo de estos son el recocido simulado, la búsqueda tabú, etc. Estos procedimientos tratan de imitar el comportamiento de sistemas dinámicos, por lo que dado un estado inicial del sistema este se perturba y se analizan los cambios causados en él, con lo cual se trata de mejorar la solución actual. Lo cual se repite un número determinado de veces que representan el número de veces que es perturbado el sistema, con lo cual se obtiene estadísticas del comportamiento del sistema

Por otro lado, de acuerdo con Cahan, Wong y Chiu es posible clasificar a los métodos de solución en: exhaustivos (analizan todo el universo posible de soluciones para determinar un alineamiento óptimo), y heurísticos (solamente se busca en una región reducida del universo posible en búsqueda de una aproximación al óptimo). En la figura siguiente se muestra esta clasificación.

Figura 7. Métodos de solución al AMS



Elaborado a partir de Chan, Wong y Chiu 1992.

A continuación se describirán, analizaran algunos de los procedimientos para solucionar el AMS, utilizando la clasificación de Chan, Wong y Chiu para categorizarlos.

2.2.1 Métodos de búsqueda exhaustiva.

Los métodos exactos pueden dividiré de acuerdo con la estrategia de búsqueda que utilizan en: métodos basados en programación dinámica (indirectos y directos) y métodos basados en ramificación y acotamiento. A continuación se describen cada uno de ellos.

2.2.1.1 Métodos basados en programación dinámica.

La programación dinámica (PD) es una estrategia para resolver problemas de optimización, cuyo principio fundamental cosiste en descomponer el problema en subproblemas más simples, la solución del problema se consigue aplicando de manera recurrente un método para resolver cada uno de los subproblemas (Lee, Tseng, Chang, & Tsai, 2007).

Se suele usar la PD cuando existe un espacio de búsqueda muy grande y éste puede ser estructurado en una serie o sucesión de estados tales que: 1) El estado inicial contiene soluciones triviales de subproblemas; 2) Cada solución parcial de estados posteriores puede ser calculada por iteración sobre un número fijo de soluciones parciales de los estados anteriores y 3) El estado final contiene la solución final.

Los algoritmos basados en PD se componen básicamente de las siguientes fases:

1. **Inicialización:** consiste en dividir al problema en una serie de subproblemas más sencillos, los definir el mecanismo de solución para los subproblemas, la cual se aplicara de forma recurrente en el conjunto de subproblemas.

2. **Relleno de la matriz de PD**: ya que la matriz de PD sirve para guardar la puntuación de los subproblemas resueltos en cada iteración. El conjunto de subproblemas debe comenzarse a resolver el subproblema más pequeño.

3. **Un rastreo:** dentro de la matriz de PD se realiza una búsqueda de datos a fin de recuperar la estructura de la solución óptima.

Con base en la PD el alineamiento óptimo se obtiene al utilizar una valoración a partir de una matriz de puntuación para los posibles pares de eslabones y la introducción de un las penalizaciones por espacios vacíos en aquellas regiones no coincidentes del alineamiento **Fuente especificada no válida.**.

El analizar todos los pares posibles de la secuencias de interés, lo que conlleva a un crecimiento exponencial de las comparaciones necesarias para el alineamiento.

Con base en lo anterior el tiempo necesario para alinear n-secuencias tiene una complejidad de $O(2^n \prod_{i=1\ldots n} l_i)$.

Entonces, la PD sólo resulta útil al comparar menos de diez secuencias de manera simultánea y donde cada una de ellas con una longitud menor a 300 caracteres (Chan, Wong, & Chiu, 1992). Con base en lo anterior se infiere que a medida que se añaden secuencias al conjunto de interés se incrementa exponencialmente el número de comparaciones a realizar.

A *Métodos indirectos*

Los algoritmos de PD indirectos son adaptaciones de los métodos tradicionales para la cuantificación de la similitud entre pares de secuencias[9], para lo cual, se realiza la fase de rastreo en la matriz de PD creada para determinar la distancia entre las secuencias. Ejemplos, de dichos algoritmos son: Matriz de puntos, Distancia de Levenshtein, Distancia Indel, Distancia de Damerau, etc.

En el capítulo uno se presentó varios de estos algoritmos con la adaptación de rastreo para encontrar el alineamiento, por lo cual en esta sección no se incluyen.

B *Métodos directos*

Los algoritmos de PD directos utilizados para resolver el problema de AMS y pueden clasificarse con base al tipo de alineamiento que se obttiene o al objetivo que persiguen. La categorización de acuerdo al alineamiento que se obtiene es en: globales y locales. Y la categorización de acuerdo al objetivo que persiguen es en: aquellos con función de similitud (se busca maximizar el grado de similitud) y aquellos con función de métrica (se busca minimizar la distancia entre las secuencia). A continuación se muestran los algoritmos de PD directos más utilizados para resolver el AMS, dividiéndolos en globales y locales.

B.1 Alineamiento global.

Los procedimientos se alineamiento global busca encontrar el mejor alineamiento posible en un conjunto de secuencias, tras analizar todos los eslabones de las secuencias. A continuación se analizan algunos de los procedimientos más usuales para obtener el alineamiento global.

El algoritmo de Needleman y Wunsch.

Este algoritmo fue desarrollado por Saúl Needleman y Christian Wunsch en el año 1969. Y es una de las estrategias más recurrentes para resolver el AMS, por medio

[9] Véase capitulo uno.

del alineamiento de pares de secuencias. Y es un mecanismo para realizar un alineamiento global de secuencias.

El objetivo del procedimiento es buscar el máximo número de similitudes entre las cadenas considerando el factor de penalización por la inserción de espacios vacíos (que representa las opresiones de eliminación e inserción) (Needleman & Wunsh, 1970), en otras palabras, ocupa el criterio de máxima similitud.

Este algoritmo requiere definir un valor numérico a cada celda de la matriz de PD el cual depende de si los elementos en la i –ésima posición de ambas secuencias son iguales ($s_{a_i} = s_{b_i}$) o bien si son diferentes ($s_{a_i} \neq s_{b_i}$).

Además, es necesario definir los parámetros:

A) **Factor de penalización por espacios vacíos** $(\boldsymbol{w})$**,** el cual es un valor numérico que tiene el objeto de servir como elemento de contención para la inserción de espacios vacíos. (Needleman & Wunsh, 1970).

B) **Una matriz de PD** que almacena los resultados parciales de cada posible alineamiento, cuyas dimensiones corresponden al número de elementos en la secuencia comparadas.

Las variantes de este algoritmo pueden clasificarse con base en el mecanismo para estimar la función de similitud en: estimación estadística ó bien estimación por ecuaciones. La estimación estadística utiliza la frecuencia observada de substitución (MPD ó PAM), de los eslabones de las cadenas comparadas, también se puede utilizar las similitudes características químicas de cada elemento. No requiere una función de similitud de forma explícita. Mientras que en la estimación por ecuaciones se asigna un valor numérico al elemento $M_{i\,j}$, que representa la puntuación obtenida al alinear el elemento $i \in s_a$ con el elemento $j \in s_b$ y utilizando para ello una función de similitud y un factor de penalización por espacios vacíos. El algoritmo termina cuando se alinean los últimos elementos de $s_a \ y \ s_b$.

A continuación se muestra el pseudocódigo del algoritmo Needleman y Wunsch con las siguientes consideraciones:

Función de similitud.

$$f\,(s_i, s_j)\begin{cases} \mathrm{y}_1 & si\ \ s_{a_i} = s_{b_j} \\ \mathrm{y}_2 & en\ caso\ contrario \end{cases} \qquad (27).$$

Algoritmo 5. Algoritmo Needleman y Wunsch.

Input: Un par de secuencias a alinear, $s_a \ y \ s_b$ de longitud $l_a \ y \ l_b$, respectivamente, una función de similitud ($f\,(s_i, s_j)$) y un factor de penalización (g).

Output: Alineamiento de Needleman y Wunsch de un par de secuencias.

1. If $(l_a = 0)$
    a. $d(s_a, s_b) = l_b$
    b. Terminar el algoritmo.
2. Else if $(l_b = 0)$
    a. $d(s_a, s_b) = l_a$
    b. Terminar el algoritmo.
3. Else
    a. Construir una matiz $m_{l_a+1\ x\ l_b+1}$
    /* Etiquetar cada fila(i) con su correspondiente elemento s_i de la secuencia s_a.
    Etiquetar cada columna (j) con su correspondiente elemento s_j de la secuencia s_b */
    b. Construir una matriz $F_{l_a+1\ x\ l_b+1}$.
    /* Etiquetar cada fila(i) con su correspondiente elemento de la secuencia s_a.
    Etiquetar cada columna (j) con su correspondiente elemento de la secuencia s_b. */
    c. For $j = 1; j \leq l_b; ++$
        i. For $i = 1; i \leq l_a; ++$
            1. If $s_{a_i} = s_{b_j}$
                a. $F_{i,j} \leftarrow y_1$
            2. Else if
                a. $F_{i,j} \leftarrow y_2$
            End if
        End for;
    End for;
4. Buscar en la matriz F la celda más cercana a la celda F_{l_a, l_b} que contenga un valor similar a y_1
5. $y \leftarrow columa$ de la celda localizada
6. $z \leftarrow hilera$ de la celda localizada
7. For $j = c; j \leq l_b; ++$
    a. $i = h; h \leq l_a; ++$
        i. $m_{i,j} \leftarrow F_{i,j}$
    End for;
    End for;
8. While $y \geq 1$
    a. While $z \geq 1$
        i. $m_{z,y} \leftarrow max \begin{cases} m_{z+1,y} - w + F_{z+1,y} \\ m_{z+1,y+1} + F_{z+1,y+1} \\ m_{z,y+1} - w + F_{z,y+1} \end{cases}$
        ii. $z = z - 1$
    End while
    b. $y = y - 1$

End While

9. Buscar la celda con el valor más grande en la primera columna y en la primera fila de la matriz m.
10. $y \leftarrow número\ de\ \ la\ columna\ donde\ \ se\ localiza\ la\ celda\ anterior.$
11. $z \leftarrow número\ de\ \ la\ fila\ donde\ \ se\ localiza\ la\ celda\ anterior$
12. While $y \leq l_b$
    c. While $z \leq l_a$
        i. Localizar la celda $\max(m_{z+1,y}, m_{z+1,y+1},\ \ m_{z,y+1})$.
        Nota: En caso de que las tres tengan el mimo valor se toma como elemento mayor el que este en diagonal.
        ii. Trazar una flecha en la celda $m_{z,y}$ en dirección a la celda contigua con mayor valor.
        iii. $z = z - 1$
    End while
    d. $y = y - 1$

End While

13. $r = 1$
14. Construir una matriz M_{2xr}
15. $While\ (l_a > 1)\ \&\ (l_b > 1)$
    a. Colocarse en la celda $m_{z1,y1}$ y observar la dirección de la flecha
    b. If la dirección de la flecha es horizontal
        i. $M_{2,,r} \leftarrow s_{b_r}$ el r –ésimo elemento de la secuencia s_b
        ii. $M_{1,r} \leftarrow$ " $-$ "
        iii. $l_b = l_b - 1$
        iv. $r = r + 1$
    c. Else If la dirección de la flecha es vertical
        i. $M_{1,,r} \leftarrow s_r$ el r –ésimo elemento de la secuencia s_a
        ii. $M_{2,r} \leftarrow$ " $-$ "
        iii. $l_a = l_a - 1$
        iv. $r = r + 1$
    d. Else If la dirección de la flecha es diagonal
        i. $M_{1,,r} \leftarrow s_r$ el r –ésimo elemento de la secuencia s_a
        ii. $M_{2,,r} \leftarrow s_r$ el r –ésimo elemento de la secuencia s_b
        iii. $l_b = l_b - 1$
        iv. $l_a = l_a - 1$
        v. $r = r + 1$
    End if
    End While

Elaborado a partir de Needleman y Wunsh 1970.

La matriz M_{2xr} contendrá el alineamiento de ambas secuencias. La complejidad del algoritmo anterior es de $O(l_a * l_b)$. Para ilustrar el funcionamiento del algoritmo anterior considere lo siguiente

Ejemplo 13. Dadas las secuencias $s_a = \{\text{a t c a c a g a t t}\}$ y $s_b = \{\text{t a c a t a c a a t}\}$ se desea determinar su alineamiento óptimo por el

algoritmo de Needleman y Wunsch. Considerando la función de similitud $f\ (s_i, s_j)\begin{cases}1 & si\ \ s_{a_i} = s_{b_j} \\ 0 & en\ caso\ contrario\end{cases}$ y factor de penalización $w = 1$.

Fase 1

a) Matriz F

	t	a	c	a	t	a	c	a	a	t
a	0	1	0	1	0	1	0	1	1	0
t	1	0	0	0	1	0	0	0	0	1
c	0	0	1	0	0	0	1	0	0	0
a	0	1	0	1	0	1	0	1	1	0
c	0	0	1	0	0	0	1	0	0	0
a	0	1	0	1	0	1	0	1	1	0
g	0	0	0	0	0	0	0	0	0	0
a	0	1	0	1	0	1	0	1	1	0
t	1	0	0	0	1	0	0	0	0	1
t	1	0	0	0	1	0	0	0	0	1

b) Matriz m

	t	a	c	a	t	a	c	a	a	t
a										0
t										1
c										0
a										0
c										0
a										0
g										0
a										0
t										1
t	1	0	0	0	1	0	0	0	0	1

Fase dos.

Relleno de la matriz m

	t	a	c	a	t	a	c	a	a	t
a	6	6	7	7	6	4	2	2	2	0
t	6	5	5	6	6	4	3	1	1	1
c	4	5	5	5	5	5	3	2	1	0
a	4	4	4	4	5	5	4	2	1	0
c	3	3	3	3	3	4	4	3	1	0
a	3	3	2	2	3	3	3	3	1	0
g	1	2	2	2	1	2	2	2	2	0
a	1	1	2	2	1	1	2	2	2	0
t	1	0	0	1	1	0	0	0	1	1
t	1	0	0	0	1	0	0	0	0	1

Fase tres

a) Colocación de flechas.

	T	a	c	a	t	a	c	a	a	t
a	6	6	7	7	6	4	2	2	2	0
t	6	5	5	6	6	4	3	1	1	1
c	4	5	5	5	5	5	3	2	1	0
a	4	4	4	4	5	5	4	2	1	0
c	3	3	3	3	3	4	4	3	1	0
a	3	3	2	2	3	3	3	3	1	0
g	1	2	2	2	1	2	2	2	2	0
a	1	1	2	2	1	1	2	2	2	0
t	1	0	0	1	1	0	0	0	1	1
t	1	0	0	0	1	0	0	0	0	1

b) Camino del alineamiento

	t	a	c	a	t	a	c	a	a	t
a										
t										
c										
a										
c										
a										
g										
a										
t										
t										1

De acuerdo a la matriz de PD existen varios alineamientos globales (bifurcaciones del camino de alineamiento) con el máximo de similitud (siete coincidencias) para las secuencias anteriores, algunos de ellos son:

c) Alineamiento global

t	a	c	a	t	-	a	c	a	-	a	-	t
-	-	-	a	t	c	a	c	a	g	a	t	t

t	a	c	a	t	-	a	c	a	-	a	t	-
-	-	-	a	t	c	a	c	a	g	a	t	t

En la construcción de la matriz de PD implícitamente se involucra el concepto de desplazamiento de secuencias que en términos generales se refiere a ordenar una secuencia de forma vertical y horizontal (es decir, etiquetar las filas y columnas de

la matriz), y recorrer a cada diagonal de la matriz, acumulando el número de coincidencias que se produzcan.

La diagonal en la que se presenta el mayor número de coincidencias representa el desplazamiento relativo del mejor alineamiento de las secuencias. Para ilustrar lo anterior observe lo siguiente:

Ejemplo 14. De las secuencias $s_1 = \{a, c\ g, t, a, c, c, t, g, t, g, a, c\}$ y $s_2 = \{t, a, a, g, t, a, c, c, t, g, a\}$ se desea determinar el desplazamiento de secuencias, considerando la siguiente función de similitud $f\ (s_1, s_2)\begin{cases} 1\ si\ \ s_{1_i} = s_{2_j} \\ 0\ en\ caso\ contrario. \end{cases}$

Matriz de coincidencias

	a	c	g	t	a	c	c	t	g	t	g	a	c
t				1				1		1			
a	1				1							1	
a	1				1							1	
g			1						1		1		
t				1				1		1			
a	1				1							1	
c		1				1	1						1
c		1				1	1						1
t				1				1		1			
g			1						1		1		
a	1				1							1	

Desplazamiento de secuencias.

	A	C	G	T	A	C	C	T	G	T	G	A	C
T				1				1		1			
A	1				1							1	
A	1				1							1	
G			1						1		1		
T				1				1		1			
A	1				1							1	
C		1				1	1						1
C		1				1	1						1
T				1				1		1			
G			1						1		1		
A	1				1							1	

Diagonal con el mayor Número de coincidencias

En caso de colocar solamente puntos en lugar del valor de la función de coincidencias se obtendría la matriz de puntos[10],en la cual los fragmentos diagonales ofrecen información visual de la relación existente entre las secuencias comparadas.

El algoritmo de Needleman y Wunsch ha sido modificado por otros autores a lo largo del tiempo. A continuación se muestra el pseudocódigo de la variante propuesta por Vinuesa. En donde se destaca el uso de valores negativos y un rastreo simplificado en la matriz de PD.

Algoritmo 6. Variante del algoritmo Needleman y Wunsch variante propuesta por Vinuesa.

Input: Un par de secuencias a alinear, $s_a \ y \ s_b$ de longitud $l_a \ y \ l_b$, respectivamente, y un factor de penalización (w).

Output: Alineamiento de Needleman y Wunsch variante propuesta por Vinuesa.

1. If $(l_a = 0)$
   a. $d(s_a, s_b) = l_b$
   b. Terminar el algoritmo.
2. Else if $(l_b = 0)$
   a. $d(s_a, s_b) = l_a$
   b. Terminar el algoritmo.
3. Else
   a. Construir una matiz $m_{l_a+1 \ x \ l_b+1}$

   /* Etiquetar cada fila(i) con su correspondiente elemento de la secuencia s_a.

   Etiquetar cada columna (j) con su correspondiente elemento de la secuencia s_b */

   a. For $i = 1; i \leq l_a; i++$
      i. $m_{1,i} \leftarrow i$

      End For.
   b. For $i = 1; i \leq l_b; i++$
      i. $m_{i,1} \leftarrow i$

      End For.
   c. Colocar sobre cada uno de los elementos de la primera y segunda fila una flecha con dirección al origen.
   d. For$j = 1; j \leq l_b; ++$
      i. For $i = 1; i \leq l_a; ++$
         1. If $s_{a_i} = s_{b_j}$
            a. $s_{ij} = 0$
         2. Else

[10] Ver capituló uno métrica de secuencias.

a. $s_{ij} = 1$

End if

3. $m_{ij} = max \begin{cases} m_{i-1,j} + w \\ m_{i-1,j-1} + s_{ij} \\ m_{i,j-1} + w \end{cases}$
4. Colocar una flecha en dirección a la celda vecina con mayor valor

End for

End for

e. $d(s_a, s_b) = 0$
f. $r = 1$
g. Construir una matriz $M_{2xr} \leftarrow 0$
h. $While\ (l_a \geq 1)\ \&\ (l_b \geq 1)$
   1. Colocarse en la celda $d_{l_a+1\ ,l_b+1}$ y observar la dirección de la flecha
   2. If la dirección de la flecha es horizontal
      1. $M_{2,,r} \leftarrow s_{b_r}$ el r –ésimo elemento de la secuencia s_b
      2. $M_{1,r} \leftarrow " - "$
      3. $l_b = l_b - 1$
      4. $r = r + 1$

   ii. Else If la dirección de la flecha es vertical
      1. $M_{1,,r} \leftarrow s_r$ el r –ésimo elemento de la secuencia s_a
      2. $M_{2,r} \leftarrow " - "$
      3. $l_a = l_a - 1$
      4. $r = r + 1$

   iii. Else If la dirección de la flecha es diagonal
      1. $M_{1,,r} \leftarrow s_r$ el r-ésimo elemento de la secuencia s_a
      2. $M_{2,,r} \leftarrow s_r$ el r –ésimo elemento de la secuencia s_b
      3. $l_b = l_b - 1$
      4. $l_a = l_a - 1$
      5. $r = r + 1$

   iv. End if

i. End While.

4. $d(s_a, s_b) = m_{l_a\ ,l_b}$

Fuente: Vinuesa, 2007.

La complejidad de la variante del algoritmo Needleman y Wunsch propuesta por Vinuesa no cambia con respecto a la original es del orden $O(l_a * l_b)$.

Para ilustrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 15. Dadas las secuencias $s_a = \{m, q, n, r, k\}$ $y\ s_b = \{m, p, n, r\}$ se desea determinar su alineamiento óptimo por el algoritmo de Needleman y Wunsch variante de Vinuesa.

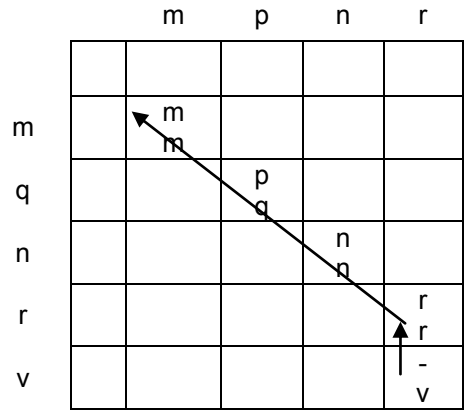
Fase de construcción de la matriz de PD.

		m	p	n	r
	0	-1	-2	-3	-4
m	-1				
q	-2				
n	-3				
r	-4				
k	-5				

Fase de relleno de la matriz

		m	p	n	r
	0	-1	-2	-3	-4
m	-1	0	-1	-2	-3
q	-2	-1	-1	-2	-3
n	-3	-2	-2	-1	-2
r	-4	-3	-3	-2	-1
v	-5	-4	-4	-3	-2

Fase de rastreo regresivo



Resultado del alineamiento

s_a	m	q	n	r	v
s_b	m	p	n	r	-

Con una distancia entre secuencias de dos operaciones, las cuales son la inserción de un espacio vacío en la s_b y el cambio de un caracter.

Con la comparación pareada de todas las secuencias de un conjunto de interés, es posible aplicar el algoritmo de Needleman – Wunsch al caso exhaustivo multidimensional, pero existe un crecimiento exponencial de la complejidad del método, lo cual limita las implementaciones a no más de tres dimensiones ($O(l^5)$). (Trelles, 1995).

El algoritmo de Murata *et al*.

Método desarrollado por Murata, Richardson y Sussman en 1985 para comparar simultáneamente tres secuencias. Cuyo objetivo es maximizar las similitudes del conjunto de interés y obtener un alineamiento global.

Este método utiliza las ideas del algoritmo de Needleman y Wunsch, pero con algunas diferencias, por lo cual e posible disminuir el tiempo de cálculo, a sólo $O(l_a * l_b * l_c)$ (Chan, Wong, & Chiu, 1992).

El mecanismo de Murata, requiere definir los siguientes parámetros al iniciar:

- Función de similitud (por estimación estadística o bien por estimación por ecuación). Esta función de similitud debe reflejar la relación biología de las secuencias que se comparan $\left(f\ (s_a, s_b, s_c)\right)$.
- Factor de penalización por espacios vacíos (w).
- Factor de ponderación entre pares de elemento de secuencias (γ).
- Relación de comparación simultánea, donde dicha ecuación debe garantizar que no se obtendrán valores negativos. (Murata, Richardson, & Sussman, 1985). Para ello se puede incluir la adición de una constante entera no negativa o la multiplicación por un factor $\left(\kappa(i,j,k)\right)$.

Tanto el factor de ponderación entre pares y la relación de comparación son expresiones determinadas por la relación biológica de las secuencias comparadas. El buen funcionamiento del algoritmo depende del establecimiento correcto de dichos parámetros.

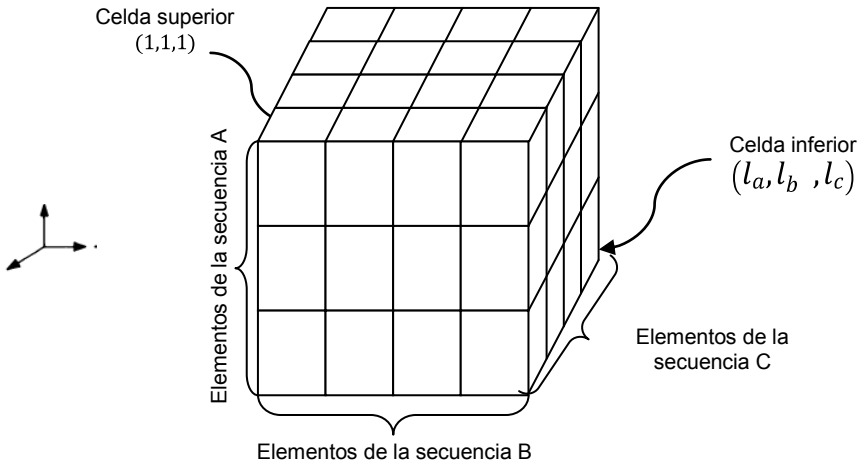
Las secuencias se analizan en una matriz tridimensional donde, se comparan simultáneamente las tres secuencias (s_a, s_b, s_c) con una longitud, $l_a, l_b\ y\ l_c$ respectivamente. Donde la celda $m_{i,j,k}$ se encuentran conectada con las celdas $(i+1, j, k)$, $(i, j+1, k)$,$(i, j, k+1)$,$(i+1, j+1, k+1)$; las cuales son determinadas por las combinaciones de los planos. Para la celda $m_{i,j,k}$ los planos son:

$$\{(i, y, z): j \leq y \leq l_b, k \leq z \leq l_c\} \qquad (28).$$

$$\{(x, j, z): i \leq x \leq l_a, k \leq z \leq l_c\} \qquad (29).$$

$$\{(x, y, k): i \leq x \leq l_a, j \leq y \leq l_c\} \qquad (30).$$

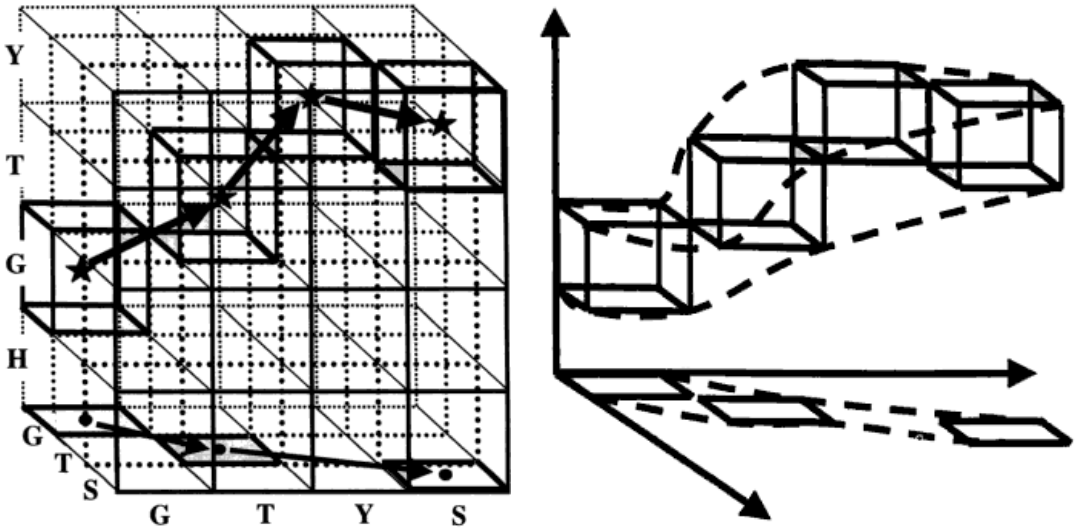
Figura 8. Matriz tridimensional para el algoritmo de Murata.



Elaborado a partir de Murata, Richardson y Sussman 1985.

La fase de inicialización del algoritmo de Murata es una extensión a tres dimensiones de la fase de inicialización del algoritmo de Needleman –Wunsch, con diferencia en la asignación de valores iniciales se realiza durante la fase de relleno de la matriz de PD. El camino de alineamiento de las tres secuencias se encuentra por la intersección de las proyecciones de los caminos en cada uno de los planos de la matriz, como lo muestra la figura siguiente.

Figura 9. Rastreo en una matriz de PD tridimensional.



Fuente: Gautham, 2006.

Algoritmo 7. Algoritmo de Murata *et al*

Input: Tres secuencias (s_a, s_b, s_c), de longitud $l_a, l_b\ y\ l_c$ respectivamente, una función de similitud $f\ (s_a, s_b, s_c)$, una relación de comparación simultánea $(\kappa(i,j,k))$ factor de ponderación de los pares posibles (γ) y un factor de penalización (w).

Output: Alineamiento de las tres secuencias.

1. If $\Big((l_a = 0)\big|\big|(l_b = 0)\ \big|\big|(l_c = 0)\Big)$.
   a. If $(l_a = 0)$
      i. Alinear las secuencias s_b, s_c por el algoritmo Needleman-Wunsch.
   b. Else if $(l_b = 0)$
      i. Alinear las secuencias s_a, s_c por el algoritmo Needleman-Wunsch.
   c. Else
      i. Alinear las secuencias s_b, s_a por el algoritmo Needleman-Wunsch.

   End if.

2. Else
   a. Construir la matriz tridimensional $m_{l_a * l_b * l_c}$

      /* Etiquetar el eje vertical con los elemento de la secuencia s_a.
      Etiquetar el eje horizontal con los elemento de la secuencia s_b
      Etiquetar el eje transversal con los elemento de la secuencia s_c.*/
   b. For $i = l_a; i \geq 1; --$
      i. For $j = l_b; j \geq 1; --$
         1. For $k = l_c; k \geq 1; --$
            a. If $(i+1) > l_a \parallel (j+1) > l_b \parallel (k+1) > l_c$
               i. $\lambda(i,j,k) \leftarrow \kappa(i,j,k)$
               ii. $\mu(i,j,k) \leftarrow \lambda(i,j,k)$
            b. Else if
               i. $\lambda(i,j,k) \leftarrow \kappa(i,j,k) + \max(\ \lambda\ (i+1, j+1, k+1), \mu(i+1, j+1, k+1) - w)$
               ii. $\mu(i,j,k) \leftarrow \max[\lambda(i,j,k), \mu(i+1,j,k), \mu(i,j+1,k), \mu(i,j,k+1)]$

            End if
            c. $m_{i,j,k} \leftarrow \lambda(i,j,k)$

         End For
      End For
   End For
   c. $contador = 1$
   d. $Q_l \leftarrow \emptyset$
   e. $(r,s,t) \leftarrow (1,1,1)$
   f. $b \leftarrow \lambda\ (1,1,1)$

g. While $\left((r \leq l_a)\ \&\&\ (s \leq l_b)\ \&\&\ (t \leq l_c)\right)$
   i. $b \leftarrow \max\left(\lambda\,(r+1,s,t), \lambda\,(r,s+1,t), \lambda\,(r,s,t+1), \lambda\,(r+1,s+1,t+1)\right)$
   ii. $if\ b = \lambda\,(r+1,s,t)$
      1. $r = r+1$
      2. $Q_{contador} \leftarrow b$
      3. $contador = contador + 1$
   iii. Else if $b = \lambda\,(r,s+1,t)$
      1. $s = s+1$
      2. $Q_{contador} \leftarrow b$
      3. $contador = contador + 1$
   iv. Else if $b = \lambda\,(r,s,t+1)$
      1. $t = t+1$
      2. $Q_{contador} \leftarrow b$
      3. $contador = contador + 1$
   v. Else
      1. $r = r+1$
      2. $s = s+1$
      3. $t = t+1$
      4. $Q_{contador} \leftarrow b$
      5. $contador = contador + 1$

3. Óptimo=$\max\{M\}$

End if

End While

End If

Fuente: Murata, Richardson y Sussman 1985 con modificación

La complejidad del algoritmo anterior es $O(l_a * l_b * l_c)$. Para ilustrar el funcionamiento del algoritmo anterior considere lo siguiente:

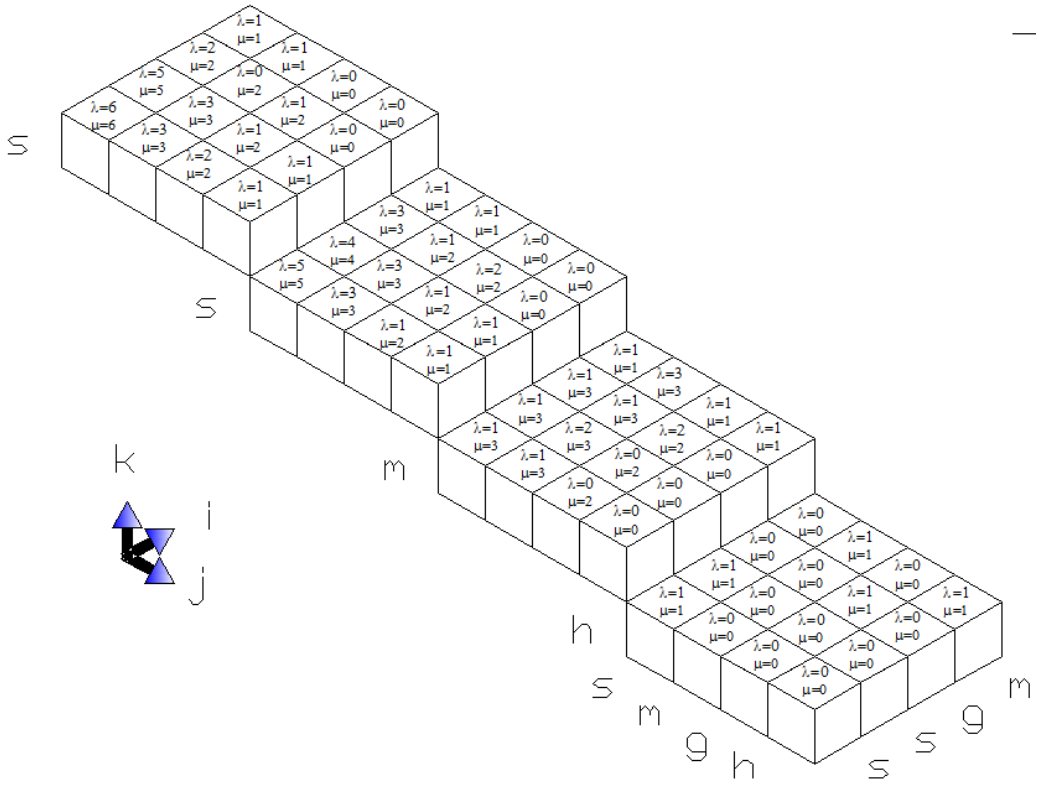
Ejemplo 16. Dadas las secuencias $s_a = \{s, m, g, h\}$, $s_b = \{s, s, g, m\}$ y $s_c = \{s, s, , m, h\}$ se desea determinar su múltiple alineamiento con el algoritmo de Murata *et al*. Considere los siguientes parámetros de entrada

Función de similitud

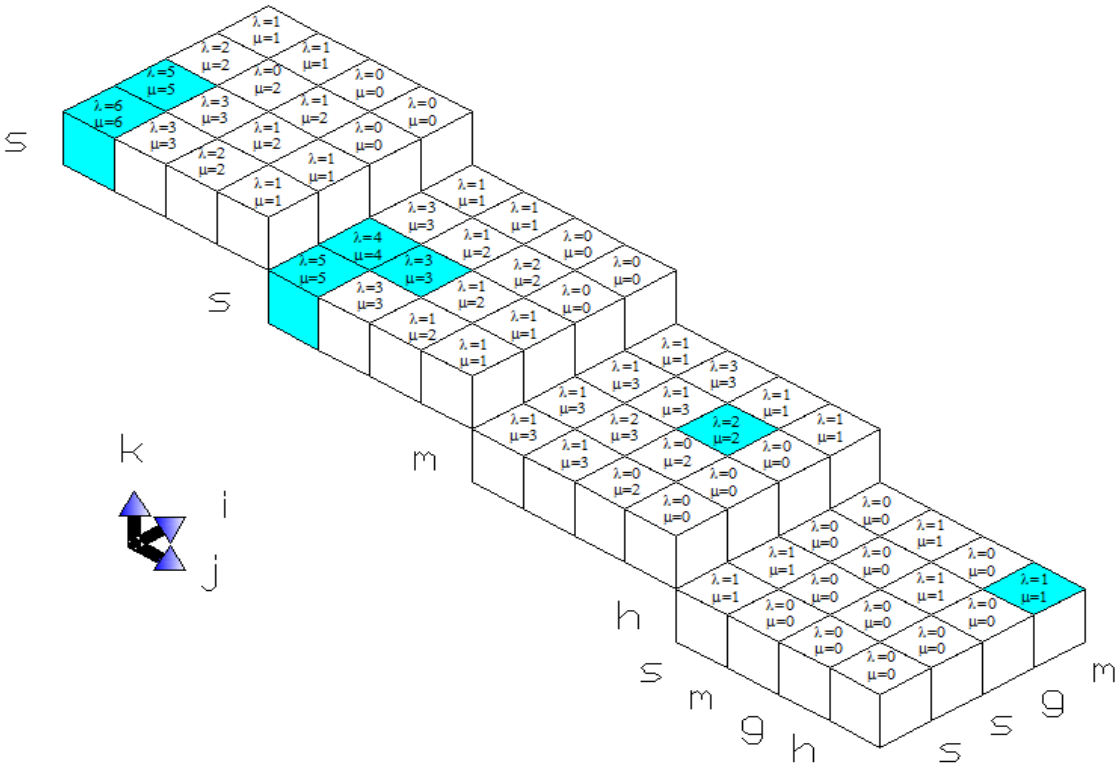
$$f\,(s_a, s_b, s_c)\begin{cases} 1\ si\ \ s_a = s_b = s_c\,, \\ 0\ en\ caso\ contrario. \end{cases}$$

Peso de los pares posibles $\gamma = 1$, factor de penalización $w = 0$, relación de comparación simultánea $\kappa(i,j,k) = \gamma * f\left(A_i, B_j\right) + \gamma * f\,(A_i, C_k) + \gamma * f\,(B_j, C_k)$.

Relleno de la matriz tridimensional.



Rastreo de la matriz de P.D.



Alineamiento obtenido.

s_a	-	s	m	g	h
s_b	s	s	-	g	m
s_c	s	s	m	-	h

La distancia entre las secuencias es de 5.

El algoritmo de Gotoh

Método desarrollado por Gotoh en 1982, para comparar dos secuencias y cuyo objetivo es minimizar el costo del alineamiento. La complejidad del algoritmo de Gotoh es del orden $O(l_a * l_b)$

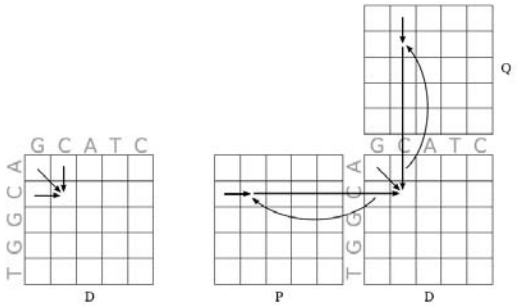
Se utilizan un factor de penalización que considera como eventos distintos inserción y expansión de espacios vacíos, lo que representa sistemas adaptativos de similitudes.

$$w = w_1 + w_2 * \mathrm{t} \qquad (31).$$

Donde:
w_1: Penalización por la introducción de un nuevo espacio vacío
w_2: Penalización por la extensión de un espacio vacío.
t: Magnitud de los guiones incertados.

Además se incluyen las matrices $P_{i,j}\ y\ Q_{i,j}$. La matriz $P_{i,j}$contiene el costo de convertir el elemento s_{a_i} en s_{b_j}, con objeto de eliminar el elemento s_{a_i} en s_a, mientras que la matriz $Q_{i,j}$ contiene el costo convertir el elemento s_{a_i} en s_{b_j}, con la finalidad de insertar el elemento s_{b_j} en s_a. (Maciel & Chiromatzo, 2004). El funcionamiento de las matrices adicionales $(P_{i,j}\ Q_{i,j})$ se representa en la siguiente figura.

Figura 10. Comparativo del método de Gotoh vs. Needleman



Needleman Gotoh

Fuente: (Bezerra, 2006)

Usualmente la función de similitud entre el i –ésimo elemento de la primera secuencia, con respecto al j –ésimo de la segunda debe cumplir con la siguiente función:

$$f\left(s_{a_i}, s_{b_j}\right)\begin{cases}\leq 0 & sí\ y\ sólo\ si\ \ s_{a_i} = s_{b_j}\\ > 0 & sí\ y\ sólo\ si\ \ s_{a_i} \neq s_{b_j}\end{cases} \qquad (32).$$

Algoritmo 8. Algoritmo de Gotoh

Input: Un par de secuencias a alinear, $s_a\ y\ s_b$ de longitud $l_a\ y\ l_b$, respectivamente, un factor de penalización por inserción de espacios vacios (w_1), un factor de penalización por extensión de espacios vacios (w_2) y una función de similitud.

Output: Alineamiento global del par de secuencias.

1. If $(l_a = 0)$
   a. $d(s_a, s_b) = l_b$
   b. Terminar el algoritmo.
2. Else if $(l_b = 0)$
   a. $d(s_a, s_b) = l_a$
   b. Terminar el algoritmo.
3. Else
   a. Construir una matiz m_{l_a+1, l_b+1}[11]
   b. For $i = 1; i \leq l_a + 1; ++$
      i. $m_{i,1} = w_1 + (i-1) * w_2$
      ii. $Q_{i,1} = m_{i,1} + w_1$

   End For
   c. For $j = 1; i \leq l_b + 1; ++$
      i. $m_{1,j} = w_1 + (j-1) * w_2$
      ii. $P_{1,i} = m_{1,j} + w_1$

   End For
   d. For $i = 2; i \leq l_a + 1; ++$
      i. For $j = 2; j \leq l_b + 1; ++$
         1. $P_{i,j} = \min\begin{cases} P_{i-1,j} + w_2 \\ m_{i-1,j} + w_1 + w_2 \end{cases}$
         2. $Q_{i,j} = min\begin{cases} Q_{i,j-1} + w_2 \\ m_{i,j-1} + w_1 + w_2 \end{cases}$
         3. $m_{i,j} = \min\begin{cases} P_{i,j} \\ Q_{i,j} \\ m_{i-1,j-1} + f\left(s_{a_i}, s_{b_j}\right) \end{cases}$

      End For
   End For

[11] Véase algoritmo de Needleman-Wunsch.

e. Colocarse en la celda a m_{l_a+1,l_b+1} con el valor más grande

f. $y \leftarrow l_a + 1$

g. $z \leftarrow l_b + 1$

h. $g \leftarrow m_{z,y}$

i. While $g > 0$

i. $g1 \leftarrow \min\{m_{z-1,y-1}, m_{z-1,y}, m_{z,y-1}\}$

ii. Trazar una fleche en dirección g1 desde g

iii. If $g1 = m_{z-1,y-1}$

1. $z = z - 1$
2. $y = y - 1$
3. $g \leftarrow m_{z,y}$

iii. Else if $g1 = m_{z-1,y}$

1. $z = z - 1$
2. $g \leftarrow m_{z,y}$

iv. Else if $g1 = m_{z,y-1}$

1. $z = z - 1$
2. $g \leftarrow m_{z,y}$

End If

End while

End if

4. Formar la matriz M con base en la dirección de las flechas colocadas en la matriz m

Fuente: Gotoh, 1982 con modificaciones.

Para ilustar el funcionamiento del algoritmo anterior considere lo siguiente

Ejemplo 17. Dadas las secuencias $s_a = \{a, c, d, e\}$ y $s_b = \{a, a, c, c\}$, $w_1 = 1$ y $w_2 = 0.5$ y la función de similitud $f(s_1, s_2)\begin{cases} 0 \ si \ s_{1_i} = s_{2_j} \\ 1 \ en \ caso \ contrario \end{cases}$. Determinar su alineamiento con el algoritmo de Gotoh.

Pasos a, b, c:

Matriz m

		a	a	c	a
	0	1.5	2	2.5	3
a	1.5				
c	2				
d	2.5				
e	3				

Matriz P

		a	a	c	a
	0	1.5	2	2.5	3
a					
c					
d					
e					

Matriz Q

		a	a	c	a
	0				
a	1.5				
c	2				
d	2.5				
e	3				

Paso d.

Matriz m

		a	a	c	a
	0	*1.5*	*2*	*2.5*	*3*
a	1.5	0	1.5	2.5	2.5
c	2	1.5	1	1.5	3
d	2.5	2.5	2.5	2	2.5
e	3	3.5	3.5	3.5	3

Matriz P

		a	a	c	a
	0	*1.5*	*2*	*2.5*	*3*
a	*1*	*2.5*	*3*	*3.5*	*4*
c	*2*	*1.5*	*3*	*4*	*4*
d	*3*	*2.5*	*2.5*	*3*	*4.5*
e	*4*	*3.5*	*3.5*	*3.5*	*4*

Matriz Q

		a	a	c	a
	0	*1*	*2*	*3*	*4*
a	*1.5*	*2.5*	*1.5*	*2.5*	*3.5*
c	*2*	*3*	*3*	*2.5*	*3*
d	*2.5*	*3.5*	*4*	*4*	*3.5*
e	*3*	*4*	*5*	*5*	*5*

Rastreo

		a	a	c	a
	0	*1.5*	*2*	*2.5*	*3*
a	1.5	0	1.5	2.5	2.5
c	2	1.5	1	1.5	3
d	2.5	2.5	2.5	2	2.5
e	3	3.5	3.5	3.5	3

Alineamiento obtenido

s_a	a	a	c	a
s_b	a	c	d	e

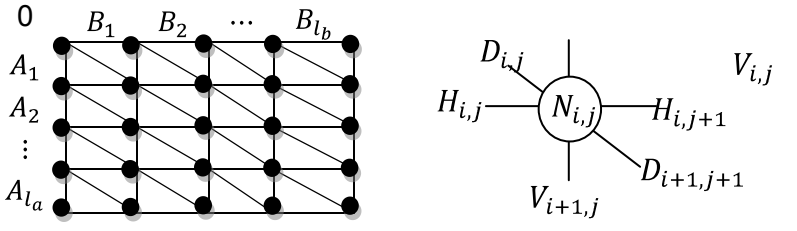
El costo del alineamiento anterior es de 3 operaciones.

El Algoritmo de Gotoh – Altschul.

Algoritmo desarrollado por S. F Altschul B. W. Erickson en 1986. Para compara dos secuencias, es una modificación al algoritmo de Gotoh y cuyo objetivo es determinar el mejor alineamiento al mínimo costo posible. Al igual que el algoritmo de Gotoh la complejidad de este algoritmo es es del orden $O(l_a * l_b)$.

Una de las aportaciones de este trabajo es la visualización del problema (alineamiento de pares de secuencias) como una gráfica, donde cada nodo se forma por alinear al i –ésimo elemento de la primera secuencia, con el j –ésimo elemento de la segunda cadena. Para ilustrar lo anterior véase la figura siguiente.

Figura 11. Representación gráfica del alineamiento de dos secuencias.



Gráfica completa

Arcos adyacentes al nodo N_{ij}

Fuente: Altschul y Erickson 1986

Cada arco tiene asociado un costo, y el problema se reduce a determinar la ruta de costo mínimo que une a los nodos $N_{0,0}$ hasta $N_{l_a l_b}$ a dicha ruta se le conoce como camino gráfico, es decir, cada alineamiento de dos secuencias puede ser representado en una gráfica en dos dimensiones, donde un camino consiste en la serie de arcos contiguos (horizontales, verticales y diagonales) entre los nodos de la gráfica tales que conectan al nodo superior izquierdo con el nodo inferior derecho. (Altschul & Erickson, 1986).

Una diferencia significativa con respecto al algoritmo de Gotoh es el uso de tres elementos $\left(P_{i,j}, Q_{i,j}, R_{i,j}\right)$ para almacenar el costo asociado con el nodo $N_{i,j}$. Y utilizar siete parámetros $(a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}, e_{i,j}, f_{i,j}, g_{i,j})$ para almacenar los datos asociados con los arcos de la gráfica.

El algoritmo se basa esencialmente en la comparación jerárquica de dos secuencias, que abarca todas las posibles rutas entre los nodos.

El Algoritmo de Carrillo –Lipman.

Método desarrollado por Carrillo y Lipman en 1988, que utiliza recursivamente el algoritmo de Needleman-Wunsch. En dicho método el objetivo es determinar el mejor alineamiento posible a mínimo costo.

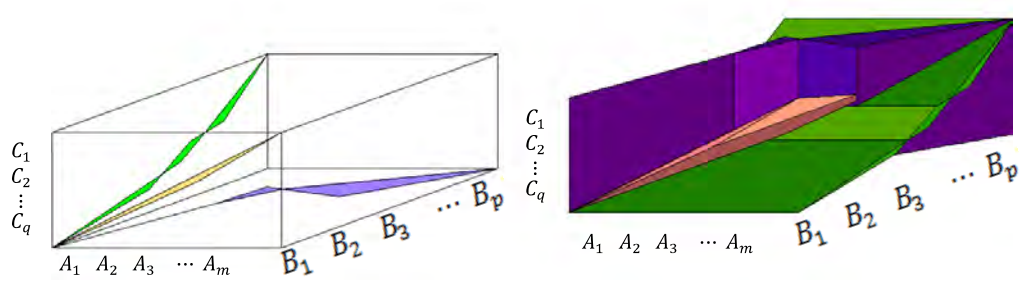
En el algoritmo de Carrillo – Lipman se introdujo el esquema de acotación del par sabio (pairwise en inglés) para limitar el volumen del espacio dimensional necesario (Trelles, 1995). Lo anterior mejoró la velocidad de procesamiento para el problema del AMS, y facilitó su implementación en programas de cómputo, (Lipman, Altschul, & Kececioglu, 1988).Desafortunadamente solamente resulta efectivo para alinear a lo más 10 secuencias en virtud de su complejidad es del orden $O(l^n)$.

La idea central del algoritmo de Carrillo-Lipman es calcular el parecido entre todos los pares de secuencias, para obtener una medida de la *distancia* evolutiva. Para lo cual se toman las dos secuencias más cercanas y se alinean entre sí. A partir de ese momento ambas secuencias se tratan como una sola. Se vuelven a alinear las dos más cercanas y así sucesivamente hasta obtener el alineamiento de todas las secuencias.

Vista en términos geométricos es determinar una ruta (alineamiento) que pase sólo por la zona de intersección de las proyecciones de los alineamientos por pares de las secuencias (lo que se ilustra en la Figura 10) esta idea se basa en que el alineamiento de N-secuencias implica crear un hípercubo N-dimensional y determinar un ruta que conecte la esquina superior izquierda con la esquina inferior derecha del mismo $\left(\gamma(s_1, s_2, \cdots, s_N)\right)$.

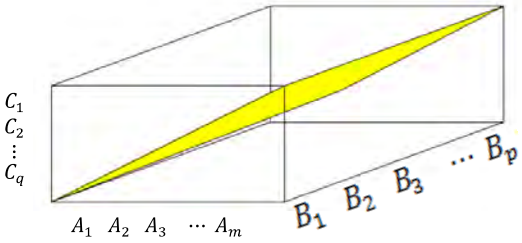
Cada ruta se relaciona con un único alineamiento de secuencias y cada alineamiento representa una ruta única.

Figura 12. Hipercubo al resultado del MSA para tres secuencias.



Alineamiento por pares de secuencias

Proyección de los pares de secuencias para obtener el alineamiento global



Alineamiento óptimo de las secuencias

Elaborado a partir de Trelles 1995.

Con lo anterior se puede esperar que la ruta óptima sea un poliedro cercano a la diagonal del hípercubo. Toda ruta puede ser proyectada a un plano formado por un par de secuencias $p_{a,b}\big(\gamma(s_1, s_2, \cdots, s_N)\big)$ denota la proyección de una ruta sobre el plano formado por la secuencia a con b (Carrillo & Lipman, 1988).

La proyección del alineamiento sobre un plano es la base para la restricción del espacio de búsqueda. En virtud de cada alineamiento óptimo por pares de secuencias impone una serie de rasgos característicos al alineamiento múltiple óptimo. Y por tanto al manejar N-secuencias se debe buscar que la ruta en el N-espacio dimensional este dentro de las restricciones impuestas por cada par alineado (Lipman, Altschul, & Kececioglu, 1988). Por lo que es posible calcular un límite superior del costo para el alineamiento múltiple de secuencias considerando un determinado par de secuencias.

Dado $p_{a,b}(\gamma)$ que representa un conjunto de rutas proyectadas en el plano (a, b) que la ruta optima es la intercesión entre las proyecciones.

$$p_{a,b}(\gamma *) = \bigcap_{(a,b),a<b} p_{a,b}(\gamma) \qquad (33).$$

Del alineamiento de dos secuencias $s_a \ y \ s_b$ se deriva el alineamiento múltiple a través de las proyecciones, ya que el costo de los espacios vacíos en los utilizados en la proyección influye en el costo total del alineamiento. Esto se logra a través de una función de relación y adaptación de los costos (Chan, Wong, & Chiu, 1992).

Otros métodos para alineamiento global.

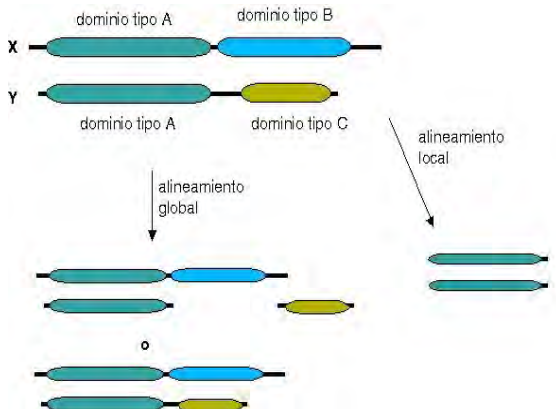
Además de los procedimientos expuestos con anterioridad para el alineamiento global, existen otros métodos exhaustivos para el alineamiento global de secuencias como son los siguientes:

1. Fickett (1984) sirve para alinear dos secuencias de manera simultánea, por medio de una matriz bidimensional cuyo objetivo es minimizar el costo. (Chan, Wong, & Chiu, 1992). La complejidad de este algoritmo es es del orden $O(l_a * l_b)$.
2. Sellers (1974) sirve para alinear dos secuencias de forma simultánea, en la construcción de la matriz de PD donde se busca la máxima similitud entre las secuencias. La complejidad de este algoritmo es del orden $O(l_a * l_b)$.
3. Fredman (1984) alinea dos secuencias de forma simultánea, cuyo objetivo es minimizar una relación de costo. (Chan, Wong, & Chiu, 1992). la complejidad de este algoritmo es del orden $O(l_a * l_b)$.

B.2 Alineamiento local.

El alineamiento local sólo resulta útil, cuando las secuencias a comparar no son homólogas, lo que permite despreciar aquellas regiones con escasa similitud entre las cadenas. Es decir, los procesamientos de alineamiento local identifican las regiones compartidas por dos secuencias que son más similares entre sí Lo que se esquematiza en la figura siguiente.

Figura 13. Alineamiento local vs. Global.



Fuente: Abascal, 2007.

Algoritmo de Smith y Waterman

El algoritmo se desarrolló en 1981 por Temple Smith y Michael Waterman, es un método de alineamiento local basado en programación dinámica, para identificar la máxima homología en subsecuencias, basado en el algoritmo de Needleman – Wunsch. Los motivos que impulsaron el desarrollo de alineamientos locales son: (González, 2008)

- **La dificultad de buenos alineamientos en regiones de baja similitud.**

En virtud de, se analizan biosecuencias poco emparentadas, o que los mecanismos evolutivos han agregaron un considerable "ruido" al conjunto de interés. Se dificulta una comparación significativa. Y los alineamientos locales evitan las regiones poco conservadas, concentrándose solo en aquellas invariantes durante la evolución. Para los alineamientos locales es necesario una expectativa de puntaje negativo, definida como el puntaje promedio que el sistema de puntaje (matriz de sustitución y penalidades por huecos) puede proporcionar para una secuencia aleatoria

- **La existencia de un modelo estadístico confiable.**

El alineamiento de secuencias no relacionadas tiende a producir puntajes de alineamiento local óptimos que siguen una distribución de valores extrema. Esta propiedad permite a producir un valor esperado para el alineamiento óptimo de dos secuencias, el cual es una medida de la frecuencia con que dos secuencias podrían producir un alineamiento óptimo cuyo puntaje es mayor o igual al observado.

En la tabla siguiente se indican alguna de las diferencias entre el algoritmo de Needleman-Wunsch con el algoritmo de Smith- Waterman.

Tabla 1. Diferencias entre algoritmos.

Algoritmo de Needleman-Wunsch	Algoritmo de Smith- Waterman
1. Alineamiento global	1. Alineamiento local
2. Requiere una función del tipo suma de pares de las diferencias con valores mayores o iguales a cero.	2. El valor de la función por pares de las diferencias puede ser positiva o negativa.
3. No requiere una función de penalización por espacios vacíos.	3. Requiere de una función de penalización por espacios vacíos.

Este procedimiento sólo permite alinear simultáneamente un par de secuencias. Y al igual que el algoritmo de Needleman - Wunsch se deben especificar los parámetros a) una función de similitud entre los elementos de las cadenas comparadas y b) un factor de penalización por espacios vacíos. Además debe ser determinado los valores de $t_1 \, y \, t_2$ que son números enteros positivos tales que $0 \leq t_1 \leq l_a$ y $0 \leq t_2 \leq l_b$. A continuación se muestra el pseudocódigo de método de Smith-Waterman.

Algoritmo 9. Algoritmo de Smith y Waterman.

Input: Un par de secuencias a alinear, s_a y s_b de longitud l_a y l_b respectivamente, un factor de penalización (w), una función de similitud $f\ (s_a, s_b)$.
Output: Alineamiento local del par de secuencias.

1. If $(l_a = 0)$
   a. $d(s_a, s_b) = l_b$
   b. Terminar el algoritmo.
2. Else if $(l_b = 0)$
   a. $d(s_a, s_b) = l_a$
   b. Terminar el algoritmo.
3. Else
   a. Construir una matiz $m_{l_a+1 * l_b+1}$
   // Etiquetar cada fila$(i+1)$ con su correspondiente elemento de la secuencia s_a.
   Etiquetar la fila y la columna uno con 0
   Etiquetar cada columna $(j+1)$ con su correspondiente elemento de la secuencia s_b//
   b. For $i = 1; i \leq l_a; ++$
      i. $m_{i,1} = 0$
   End For
   c. For $j = 1; j \leq l_b; ++$
      ii. $m_{1,j} = 0$
   End For
   d. For $i = 2{:}\,1{:}\,l_a$
      iii. For $j = 2{:}\,1{:}\,l_b$
         1. $m_{i,j} \leftarrow \max \begin{cases} m_{i-1,j-1} + f\ (s_{a_i}, s_{b_j}), \\ m_{i-t_1,j} - w \\ m_{i,j-t_2} - w \\ 0 \end{cases}$
      End For
   End for.
   e. Buscar la celda en M con el valor más grande
   f. $y \leftarrow número\ de\ \ la\ columna\ donde\ \ se\ localiza\ la\ celda\ anterior.$
   g. $z \leftarrow número\ de\ \ la\ fila\ donde\ \ se\ localiza\ la\ celda\ anterior$
   h. $g \leftarrow M_{z,y}$
   i. While $g > 0$
      i. $g1 \leftarrow \max\{m_{z-1,y-1}, m_{z-1,y}, m_{z,y-1}\}$
      ii. Trazar una fleche en dirección g1 desde g
      iii. If $g1 = M_{z-1,y-1}$
         1. $z = z - 1$
         2. $y = y - 1$
         3. $g \leftarrow m_{z,y}$
      iv. Else if $g1 = m_{z-1,y}$

1. $z = z - 1$
2. $g \leftarrow m_{z,y}$

v. Else if $g1 = m_{z,y-1}$

3. $z = z - 1$
4. $g \leftarrow m_{z,y}$

End if

End while.

End if

4. Formar la matriz de alineación con base en la dirección de las flechas colocadas

Fuente: Smith y Waterman 1981.

La complejidad del algoritmo de Smith- Waterman es del orden $O(l_a * l_b)$ .Para ilustrar el funcionamiento del método anterior considere lo siguiente:

Ejemplo 18.

Dadas $s_a = \{p, a, s, e, l, p, m, s, p, q, a, a, h\}$ y $s_b = \{m, a, s, e, l, p, m, s, p, h\}$, $k = l = 1$.Un factor de penalización $w = \frac{1}{2}$ Una función de similitud $f\ (s_a, s_b) \begin{cases} 1\ si\ \ \mathrm{s_a} = \mathrm{s_b} \\ 0\ en\ caso\ contrario. \end{cases}$

Matriz de programación dinámica.

	0	m	a	s	e	l	p	m	s	p	h
0	0	0	0	0	0	0	0	0	0	0	0
p	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a	0.00	0.00	1.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00
s	0.00	0.00	0.50	2.00	1.50	1.00	0.50	0.00	1.00	0.50	0.00
n	0.00	0.00	0.00	1.50	2.00	1.50	1.00	0.50	0.50	1.00	0.50
l	0.00	0.00	0.00	1.00	1.50	3.00	2.50	2.00	1.50	1.00	1.00
p	0.00	0.00	0.00	0.50	1.00	2.50	4.00	3.50	3.00	2.50	2.00
m	0.00	1.00	0.50	0.00	0.50	2.00	3.50	5.00	4.50	4.00	3.50
s	0.00	0.50	1.00	1.50	1.00	1.50	3.00	4.50	6.00	5.50	5.00
p	0.00	0.00	0.50	1.00	1.50	1.00	2.50	4.00	5.50	7.00	6.50
q	0.00	0.00	0.00	0.50	1.00	1.50	2.00	3.50	5.00	6.50	7.00
a	1.00	0.50	1.00	0.50	0.50	1.00	1.50	3.00	4.50	6.00	6.50
a	2.00	1.50	1.50	1.00	0.50	0.50	1.00	2.50	4.00	5.50	6.00
h	3.00	2.50	2.00	1.50	1.00	0.50	0.50	2.00	3.50	5.00	6.50

Alineamiento resultante.

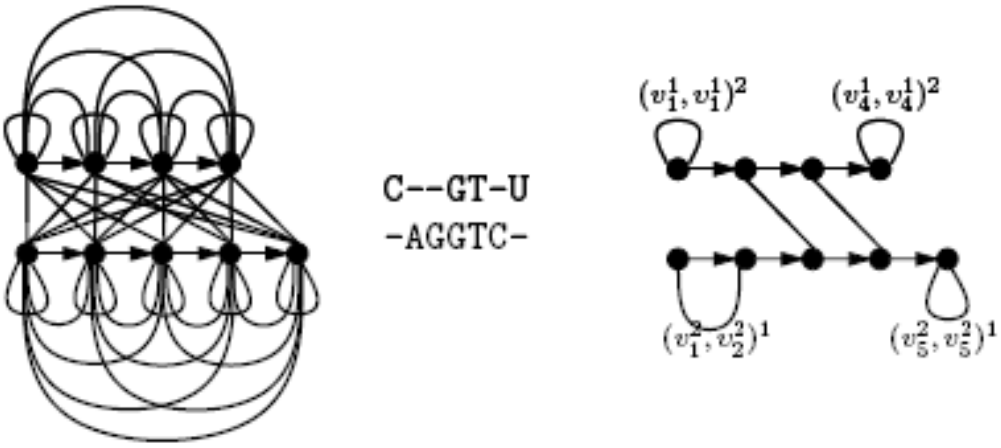
$s_a.$	m	a	s	e	l	p	m	s	p	h	-	-	-
$s_b.$	p	a	s	n	l	p	m	s	p	q	a	a	h

Subsecuencias alineada.
7 coincidencia

2.2.1.2 Ramificación y acotamiento.

Estrategia desarrollada por Althaus, Caprara, Lenhof y Reinert, el cual requiere la reformulación del problema AMS como un problema de programación lineal entera. Para ello se utiliza una gráfica múltiple mixta $\left(G = (V, E, A)\right)$ llamada gráfica de alineamiento por vacíos.

Figura 14. Gráfica de alineamiento por vacíos.



Fuente: Althaus, Caprara y Lenho 1997.

Donde V son los nodos y representan los caracteres de las cadenas, E es un conjunto de arcos no dirigidos y A arcos dirigidos en la gráfica.

Modelo binario para el AMS. (Althaus, Caprara, & Lenho, 1997)

$$max \sum_{e \epsilon E} w_e * x_e + \sum_{a \epsilon A} w_a * y_a \qquad (34).$$

Sujeto a:

$$\sum_{m=1}^{|s^j|} x_{\left\{v_l^i, v_m^j\right\}} + \sum_{a \epsilon A^{ij}\ (l \leftrightarrow l)} y_a = 1 \qquad \forall\ i, j = 1, \dots, k\ ;\ i \neq j\ ;\ \ l = 1, \dots, \left|s^i\right| \qquad (35).$$

$$\sum_{e \epsilon M \cap E} x_e \leq |M \cap E| - 1 \quad M \in \mathcal{M} \tag{36}$$

$$\sum_{a \in I} y_a \leq 1 \qquad I \in \mathcal{I} \tag{37}$$

$$x_{\left\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\right\}} + x_{\left\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\right\}} - x_{\left\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\right\}} \leq 1 \qquad i_r = 1, \ldots, k;\ i_r \neq i_{r+1};\ l_r = 1, \ldots, \left|s^{i_r}\right| \tag{38}$$

$$x_e, y_a \in \{1,0\}\ e \epsilon E\ a \epsilon A \tag{39}$$

Donde:
x_e: Variable de alineamiento.
y_a: Variable por la inserción de espacios vacíos
$\mathcal{M}$: Todos los ciclos en la gráfica G.
v_l^i: Nodo correspondiente en el lugar l de la i-esima secuencia.
$\left\{v_l^i, v_m^j\right\}$: Alineamiento del caracter l de la i –ésima secuencia con el caracter m de la j-ésima secuencia.
w_e w_a : Costos asociados a los arcos.

Un alineamiento, dada un gráfica se obtiene al determinar un ciclo crítico. El uso de las técnicas de ramificación y corte han permitido resolver el problema con buenos resultados. Hasta implementar técnicas modernas de ramificación y corte (ver (Althaus, Caprara, & Lenho, 1997)).

2.2.2 Métodos aproximados.

Son métodos heurísticos o metaheurísticos que anteponen el tiempo necesario para solucionar el problema sobre el grado de optimalidad de la solución encontrada. Es decir buscan resolver el problema con un consumo razonable de recursos para obtener un buen alineamiento (aun que no necesariamente el óptimo).

2.2.2.1 Métodos progresivos..

También conocidos como métodos jerárquicos o métodos de árboles evolutivos se caracterizan por:

- Son métodos más rápidos pero con mayor grado de incorrecciones que los métodos exhaustivos.
- Producen buenos alineamientos pero no necesariamente el óptimo.
- Alinean las secuencias más parecidas entre sí, para después ir incorporando el resto de acuerdo al grado de similitud entre sí.

Generalmente los métodos progresivos constan de tres fases básicas (Rech & Pilatti, 2004) las cuales son:

- **Alineamiento par a par**: En esta fase se construye la matriz de distancias de los pares posibles de las secuencias de entrada al algoritmo. Generalmente la distancia de alineamiento entre cada par secuencias es determinado por algún algoritmo de programación dinámica, el cual será invocado para un conjunto de $x - secuencias$ $\frac{x(x-1)}{2}$ veces. Para la construcción de la matriz de distancia par a par puede utilizarse el siguiente algoritmo.

Algoritmo 10. Matriz de distancia par a par.

Input: Un conjunto N de secuencias $(S = \{s_1, s_2, \dots, s_x\})$

Output: Matriz de distancia par a par del conjunto de secuencias (MD).

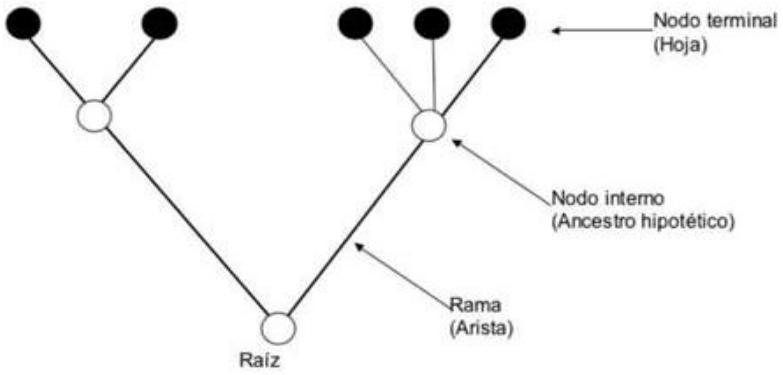
1. For $i = 1:1:x$
   a. For $j = 1:1:x$
      i. If $i < j$
         1. $d(s_i, s_j)$ ←la distancia entre la secuencia s_i y la s_j calculada a través de un algoritmo de programación dinámica.
         2. $MD_{i,j} \leftarrow d(s_i, s_j)$
      ii. Else if $i = j$
         1. $MD_{i,j} \leftarrow 0$
      iii. Else $i > j$
         1. $MD_{i,j} \leftarrow d(s_j, s_i)$
      End If
   End For
End For

La complejidad del algoritmo anterior es del orden $O\left(\frac{n(n-1)}{2} * (l^n)\right)$.

- **Construcción del árbol:** Una vez construida la matriz de programación dinámica, se crea un árbol guía, el cual se utilizara posteriormente para el alineamiento múltiple de secuencias. Uno de los métodos más utilizados para la generación del árbol guía es el algoritmo de Neighbor-Joining En este método se alinean sucesivamente las secuencias más cercanas entre sí (menor distancia). Sea S el conjunto de secuencias de entrada y s_a cualquier secuencia de dicho conjunto. Entonces se asigna a cada s_a como un nodo hoja de árbol evolutivo y se determinan grupos fuertemente emparentado (S_s).

Una vez que se ha determinado un conjunto de secuencias fuertemente emparentado (S_s) tal que $S_s \subseteq S$ debe determinarse la secuencia común más larga de S_s , la cual representa un nuevo nodo intermedio en el árbol, posteriormente se determina la distancia de este nodo a todas las secuencias tales que $s_a \notin S_s$ y se une por medio de aristas a los nodos que se agrupan en S_s, se sigue iterativamente este proceso hasta que el árbol guía conecta todos las secuencias de S.

Figura 15. Árbol evolutivo.



Fuente: Martínez Castilla 2008.

- **Alineamiento progresivo:** Una vez construido el árbol guía, se alinean progresivamente las secuencias de las ramas más próximas comenzando con las hojas del árbol hasta llegar a la raíz del mismo. Al alinear una secuencia con un grupo ya alineado se preservan los espacios vacios del conjunto ya alineado.

Los métodos progresivos se clasifican de acuerdo al tipo de alineamiento que se logra en:

- Aproximaciones globales.
- Aproximaciones locales,
- Aproximaciones hibridas.

Antes de analizar algunos de los métodos progresivos para solucionar el AMS se requiere definir el concepto de árboles evolutivos, por lo en la siguiente sección de analiza éste concepto.

A *Árboles evolutivos*

La idea central es la búsqueda de la relación ancestral entre las secuencias de interés. Para lo que se utilizan árboles filogenéticos que es una representación gráfica del conjunto de interés.

De acuerdo con la filosofía biológica, los métodos de reconstrucción filogenética asumen que el proceso evolutivo, que conecta al pasado con el presente, preserva (cuando menos en parte) la información del estado ancestral

Los objetivos de una reconstrucción filogenética son:

a) Reconstruir correctamente las relaciones de genealógicas.
b) Estimar correctamente las longitudes de las ramas, (distancia evolutiva).
c) Conocer la raíz del árbol.

Un árbol evolutivo es grafo compuesto de nodos y arcos con la cual se pretende representar las relaciones evolutivas entre unidades taxonómicas operacionales. Donde los nodos terminales representan secuencias de las que se poseen datos. Los nodos internos representan a los ancestros hipotéticos y el ancestro de todas las secuencias comprendidas en el árbol es la raíz del árbol. Y los arcos definen las relaciones de ancestaría y descendencia entre las unidades.

Definición 30. Un **árbol evolutivo** (T_N) para $N - secuencias$, es un árbol de peso mínimo, para una gráfica G=(V,E). La gráfica se construye con al menos x nodos donde cada secuencia se asocia exactamente a un vértice de la gráfica y a cada arco se le asigna el valor del alineamiento óptimo entre los nodos (secuencias) adyacentes. Por lo tanto el costo del árbol se relaciona directamente con el costo de los arcos incluidos en el árbol.

En la construcción de los árboles evolutivos se implica el principio de mínima mutación o principio de parsimonia.

Definición 31. El **principio de mínima mutación** implica que las secuencias más semejantes entre sí han tenido menos tiempo de evolución entre sí, que aquellas menos semejantes.

Desafortunadamente la construcción de un árbol evolutivo es un problema NP-duro, por lo tanto el tiempo necesario para su construcción crese exponencialmente en función del número de secuencias que se agregan al alineamiento. El concepto de árbol evolutivo e ideas implicadas servirán para analizar los métodos progresivos para solución del AMS.

B *Aproximaciones globales.*

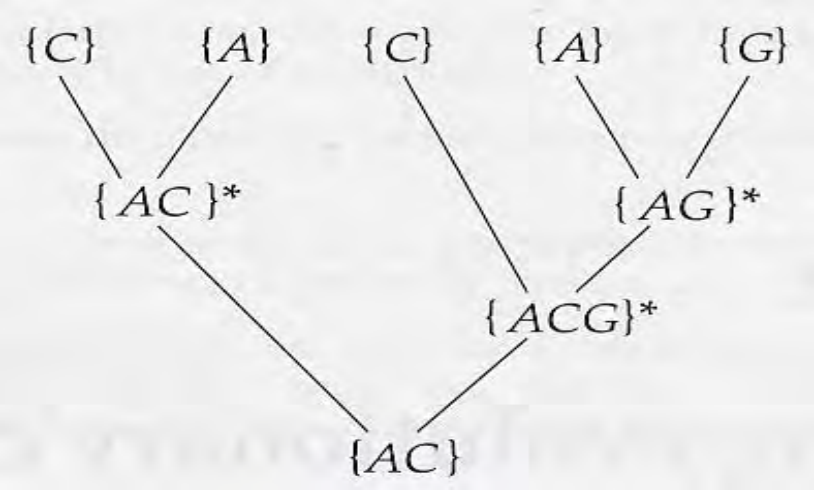
B.1 El algoritmo de Fitch.

El algoritmo de Fitch utiliza el cambio de estados en cada iteración, y es un algoritmo para la reconstrucción de la secuencia ancestral basada en la minimización de costos. Tiene una complejidad de $O(n_i(2n)^{n_e})$ donde n_i es el

número de nodos internos en el árbol, n_e el número de nodos externos en el árbol y n el número de secuencias.

El algoritmo funciona de forma general, en primer lugar coloca como hojas del árbol a las secuencias del conjunto de interés. Posteriormente, se agrupan los nodos en parejas, cada pareja representan dos secuencias emparentadas. En caso de que algún nodo quede sin pareja de intersección es necesario agregar una penalización. Para construir el siguiente estado se tiene que considerar los dos nodos que se intersecan para generar un nuevo nodo que es el resultado de comparar ambas secuencias. Lo cual se realiza de forma iterativa hasta que la raíz es alcanzada.

Figura 16. Algoritmo de Fitch.



Fuente: Huson, 2009.

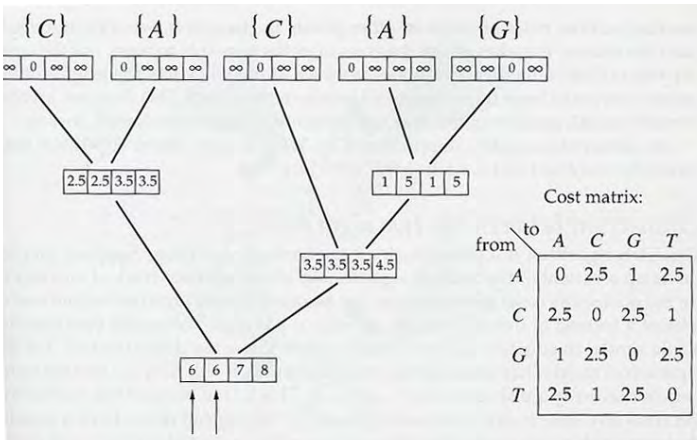
B.2 El algoritmo de Sankoff *et al*.

Método desarrollado por David Sankoff y R. J. Cedergren en 1982, es un algoritmo para la reconstrucción de la secuencia ancestral basada en la minimización de costos con una complejidad de $O(n_i(2N)^{n_e})$ donde n_i es el número de nodos internos en el árbol, n_e el número de nodos externos en el árbol y n el número de secuencias.. (Chan, Wong, & Chiu, 1992).

El algoritmo utiliza el método de Needleman – Wunsch en la construcción de la red, pero requiere determinar a priori las relaciones biológicas entre los individuos a analizar.

Es un algoritmo más complejo que el de Fitch, pero tiene la ventaja, que los cambios de un estado en otro estado pueden ser ponderados. Por que se utiliza una matriz de costos es para cambiar entre dos estados. Pero sólo resulta útil al alinear menos de 15 secuencias con una longitud no mayor a 50 caracteres.

Figura 17. Ejemplo del algoritmo deSankof.



Fuente: Huson, 2009.

B.3 El algoritmo de Feng-Doolittle

En este algoritmo fue desarrollado por Feng y Doolittle, 1987, la idea clave es encontrar distancia mínima entre las pareja posibles, para encontrar la información que puede extraerse de las cadenas.

Por lo tanto, cualquier espacio (vacío) que aparezca en la alineación óptima en cada par debe ser conservado en el conjunto de múltiples alineación. El algoritmo es el siguiente:

Algoritmo 11. Algoritmo de Feng Doolitle.

Input: Un conjunto de secuencias a alinear, $N = \{s_1, s_2, \dots, s_N\}$

Output: Alineamiento múltiple de las secuencias.

1. Calcular la distancia entre cada par de secuencias.
2. Utilizar un algoritmo de agrupamiento incremental (Fitch y Margoliash[12]) para construir un árbol a partir de las distancias.
3. Recorrer los nodos en el orden de adición al árbol, para obtener la alineación del conjunto.

Fuente: Huson, 2009

[12] El método Fitch-Margoliash es un procedimiento ponderado de mínimos cuadrados para el agrupamiento sobre la base de la distancia genética. A las secuencias más relacionadas se les da mayor peso en el proceso de construcción de árboles para corregir el aumento de inexactitud en la medición de distancias. Y las distancias utilizadas en el procedimiento deben ser normaliza para evitar gran consumo de recursos

B.4 El algoritmo de Barton y Sternberg

Algoritmo desarrollado por G. J Barton y M. J Sternberg en 1987 para alinear secuencias proteicas. Utiliza el alineamiento por pares de las secuencias por el algoritmo de Needleman – Wunsch. (Barton & Sternberg, 1987)

La aproximación se lleva a cabo por medio de la del agrupamiento jerárquico de secuencias. El alineamiento de las secuencias A, B, C y D se obtiene al alinear A con B (AB) , C se alinea con AB y D con ABC . (Chan, Wong, & Chiu, 1992). Por lo tanto el alineamiento de las secuencias se obtiene por el uso recurrente del alinear cada secuencia con el alineamiento previo. Para alinear n-secuencias con una longitud l se requieren $O(n(l^2))$ operaciones.

Algoritmo 12. Algoritmo de Barton y Sternberg

Input: Un conjunto de n secuencias a alinear, $S = \{s_1, s_2, \dots, s_n\}$

Output: Alineamiento múltiple de las secuencias.

1. Alinear la secuencia s_1 con s_2 usando el algoritmo de Needleman y Wunsch.
   Al terminar el alineamiento $|s_1| = |s_2|$
2. For i=1:1: $|s_2|$
3. $Q_i \leftarrow \emptyset$
   a. If s_{1_i} =-
      i. $s_{m_1} \leftarrow s_{2_i}$
   b. Else if $s_{2_i} = -$
      i. $s_{m_1} \leftarrow s_{1_i}$
   c. Else
      i. If $s_{1_i} = s_{2_i}$
         $s_{m_1} \leftarrow s_{2_i}$
      ii. Elseif $s_{1_i} \neq s_{2_i}$
         $s_{m_1} \leftarrow *$
         Añadir al vector Q_i los elementos s_{1_i} y s_{2_i}
      End if
   End if
End for;
4. Sustituir los elementos de s_{m_1}= * por el elemento s_{1_i}
5. $s_m = s_{m_1}$
6. For j=3:1:n
   a. Alinear con el algoritmo de Needleman y Wunsch la secuencia s_i con s_m
   b. For i=1:1: $|s_m|$
      i. If $s_{j_i} = -$

$$s_{m_{2_i}} \leftarrow s_{m_i}$$

ii. Else if $s_{m_i} = -$

$$s_{m_{2_i}} \leftarrow s_{j_i}$$

Agregar un espacio vacío en el lugar correspondiente a las secuencias anteriores a j

iii. Else

1. If $s_{j_i} = s_{m_i}$

$$s_{m_{2_i}} \leftarrow s_{j_i}$$

2. Elseif $s_{1_i} \neq s_{2_i}$

$$s_{m_{2_i}} \leftarrow *$$

Añadir al vector Q_i el elemento s_{j_i}

End if;

End if

7. Sustituir los elementos de s_{m_2}= * por el elemento más frecuente en el vector Q_i
8. $s_m = s_{m_1}$

End for

Fuente: Barton y Sternberg 1987 con modificaciones.

Para ilustrar el algoritmo anterior considérese lo siguiente:

Ejemplo 19. Dadas las secuencias $s_1 = \{\mathrm{m, q, n, r, v}\}$, $s_2 = \{\mathrm{m, p, n, r}\}$, $s_3 = \{\mathrm{m, q, n, r, k, t, y, d, d, w}\}$ y $s_4 = \{\mathrm{m, n, r, t, t, y, m, d}\}$ se desea determinar la matriz de múltiple alineamiento. Ya que las secuencias s_1 y s_2 se alinearon en el ejemplo 15, se inicia este ejemplo desde determinar la secuencia media.

s_1	m	q	n	r	v
s_2	m	p	n	r	-
$\boldsymbol{s_{m_1}}$	**m**	**p**	**n**	**r**	**v**

vector $Q_2 = \{p, q\}$
costo 2 operaciones

Se alinea la secuencia media obtenida con la secuencia s_3 y se obtiene lo siguiente: (existe un alineamiento alternativo con el mismo costo)

s_m	m	p	n	r	v	-	-	-	-	-
s_3	m	q	n	r	k	t	y	d	d	w
$\boldsymbol{s_{m_2}}$	m	q	n	r	v	t	y	d	d	w

vector $Q_2 = \{p, q, q\}$
vector $Q_5 = \{v, k\}$
costo 7 operaciones

Agregando los guiones correspondientes a las secuencias s_1 y s_2 e incluyendo s_3 al alineamiento parcial

s_1	m	q	n	r	v	-	-	-	-	-
s_2	m	p	n	r	-	-	-	-	-	-
s_3	m	q	n	r	k	T	y	d	d	w

Se alinea la secuencia media obtenida con la secuencia s_4 y se obtiene lo siguiente:

s_m	m	q	n	r	v	t	Y	-	d	d	w
s_4	m	-	n	r	t	t	Y	m	d	-	-
$\boldsymbol{s_{m_2}}$	m	q	n	r	v	t	Y	m	d	d	w

vector $Q_2 = \{p, q, q\}$
vector $Q_5 = \{v, k, t\}$
costo 4 operaciones

Incluyendo s_4 al alineamiento se obtiene

s_1	m	q	n	r	v	-	-	-	-	-	-
s_2	m	p	n	r	-	-	-	-	-	-	-
s_3	m	q	n	r	k	t	y	-	d	d	w
s_4	m	-	n	r	t	t	y	m	d	-	-

El costo de alineamiento total es de 13 operaciones

B.5 El algoritmo 2-aproximación para AMS.

En este método se utiliza una aproximación por secuencia media o por templete. El objetivo de este método es minimizar el costo total del alineamiento a través de la suma de pares de secuencia.

Una vez determinada la secuencia media [13]o el templete[14] se utiliza en algoritmo de distancia (Damerau o Levenshtein) para alinear las secuencias. Para alinear n-secuencias con una longitud l por templete $O(n(l^2))$ operaciones

---

Algoritmo 13. 2-aproximación para AMS con función objetivo de suma por pares de la diferencia.

Input: Un conjunto de n secuencias a alinear, $S = \{s_1, s_2, \dots, s_n\}$

Output: Alineamiento múltiple de las secuencias.

[13] Para determinar la secuencia media de n-secuencias se requiere determinar la distancia de los pares posibles $n(n-1)/2$ y buscar aquella secuencia con menor distancia al resto.
[14] Generalmente se utiliza la primera secuencia del conjunto de interés.

1. Determine la secuencia media o el templete.
2. Asigne s_m como la secuencia a utilizar.
   $i = 1$;
3. While $i \leq n$
   - Usar el algoritmo de Damerau o Levenshtein para alinear las secuencias $s_m \; con \; s_i$
   - Agregar los espacios a las secuencias anteriores a s_i en caso de ser necesario
   - I=i+1;
   End while.

Fuente: Lee, *et al*. 2007.

Para ilustrar el algoritmo anterior observe lo siguiente

Ejemplo 20. Dadas las secuencias $s_1 = \{\mathrm{m, q, n, r, v}\}$, $s_2 = \{\mathrm{m, p, n, r}\}$, $s_3 = \{\mathrm{m, q, n, r, k, t, y, d, d, w}\}$ y $s_4 = \{\mathrm{m, n, r, t, t, y, m, d}\}$ se desea determinar la matriz de múltiple alineamiento. Utilizando la distancia de Damerau.

Iteración 1

s_1	m	q	n	r	v
s_2	m	p	n	r	-

Costo 2 operaciones

Iteración 2

s_1	m	p	n	r	v	-	-	-	-	-
s_2	m	p	n	r	-	-	-	-	-	-
s_3	m	q	n	r	k	t	y	d	d	w

Costo 5 operaciones

Iteración 3

s_1	m	p	n	r	v	-	-	-	-	-
s_2	m	p	n	r	-	-	-	-	-	-
s_3	m	q	n	r	k	t	y	d	d	w
s_3	m	-	n	r	t	m	d	-	-	-

El costo del alineamiento es de 13 operaciones.

B.6 *Algoritmo de Corpet*

Desarrollado por Copert en 1988 que adopta un procedimiento jerárquico por grupos para la construcción del árbol de alineamiento. Cada grupo se compone por las secuencias más similares entre sí.

Fue desarrollado para el alineamiento de secuencias proteicas, y su objetivo es encontrar el máximo grado de similitud entre los grupos de secuencias. (Chan, Wong, & Chiu, 1992) mediante la el alineamiento pareado de las secuenciad del conjunto.

B.7 *Los algoritmos de la familia Clustal*

Contiene a un conjunto de algoritmos que se caracterizan por: alineas separadamente todos los pares de secuencias para calcular una matriz de distancias, para determinar el grado de divergencia entre cada par de secuencias. Son algoritmos de naturaleza glotona

Con base en la matriz se calcula el "árbol guía" (por método UPGMA o Neighbor-Joinning). Dicho árbol describe las relaciones existentes entre las secuencias que van a ser alineadas, es decir, es una guía para ir añadiendo las secuencias al alineamiento múltiple.

Las secuencias se alinean progresivamente siguiendo el orden de las ramas del árbol guía. En la figura siguiente se esquematiza el procedimiento general de los algoritmos de la familia Clustal.

Algoritmo 14. Algoritmo general para la familia Clustal

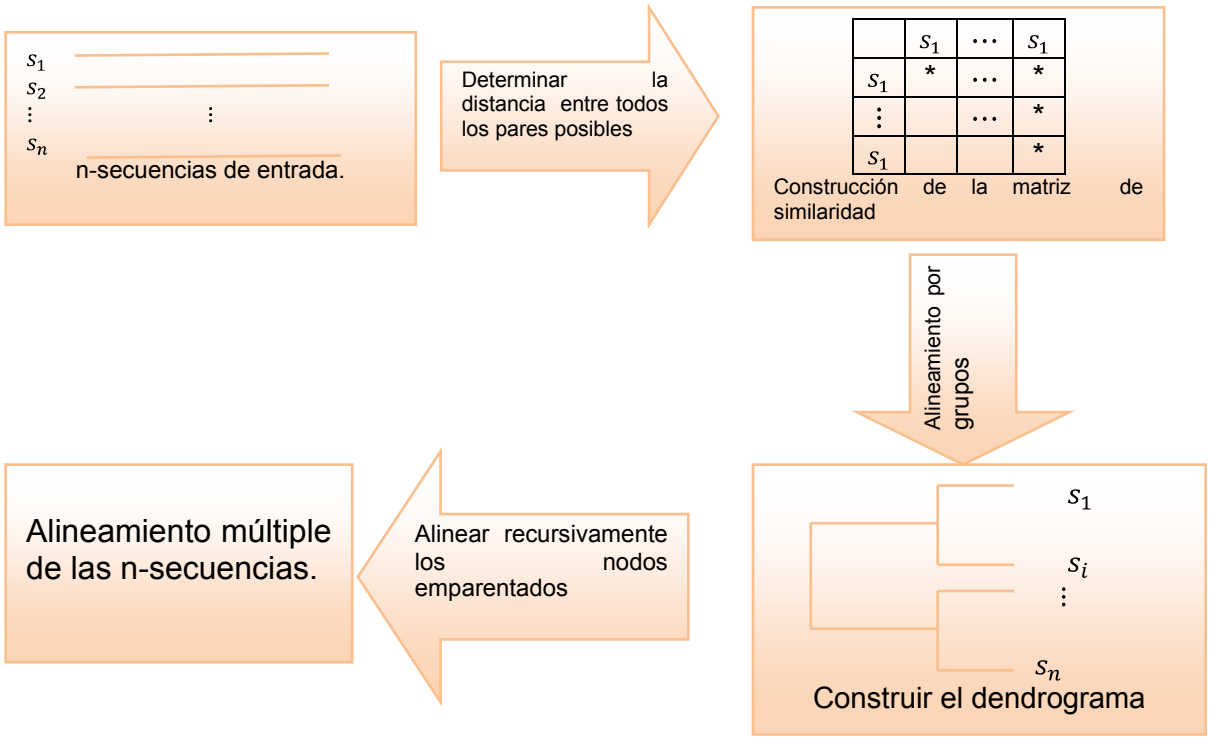
Input: Un conjunto de n secuencias a alinear, $S = \{s_1, s_2, \dots, s_n\}$

Output: Alineamiento múltiple de las secuencias.

1. Realizar un alineamiento global pareado mediante el algoritmo Needleman-
2. Wunsch para las secuencias incluidas en el alineamiento.
3. Cálculo del árbol guía a partir del puntaje (o distancia) de los alineamiento pareados realizados en el primer paso.
4. Creación del alineamiento múltiple. El alineamiento de las secuencias se lleva a cabo en el orden determinado por el árbol guía. El algoritmo selecciona primero las dos secuencias más relacionadas y crea un alineamiento pareado de estas, y de manera progresiva va sumando una secuencia al resultado de dicho alineamiento

Fuente: Bioinformática, 2005

Figura 18. Procedimiento para los algoritmos de la familia Clustal



Fuente: Solanilla, Gómez Teshima y Múnera 2004

El algoritmo Clustal W es la versión más actual de la familia de los algoritmos Clustal cuya diferencia con el resto es el uso de un parámetro de ponderación para el alineamiento de las secuencias. (Desmond & Thomson, 1996) Se utilizan dos factores de penalización por los espacios vacíos (inserción o extensión.). A continuación se muestra el pseudocódigo:

C *Aproximación local.*

C.1 Algoritmo FASTA.

El algoritmo FASTA fue desarrollado por Lipman y Pearson en 1985. Es de los algoritmos que no se basan en principios biológicos. Pero con una buena eficiencia para el alineamiento de secuencias.

En el procedimiento se analizan parejas de secuencias para aquellos segmentos que presentan las mejores uniones, las uniones significativas se seleccionan con base en un umbral mínimo de similitud y se desean encontrar coincidencias se pueden incluir espacios vacíos. Conceptualmente el procedimiento se asemeja a la búsqueda de diagonales significativas en dos secuencias.

Algoritmo 15. Algoritmo FASTA.

Input: Un conjunto de $n$secuencias a alinear, $S = \{s_1, s_2, \dots, s_n\}$

Output: Alineamiento múltiple de las secuencias.

1. Identificar regiones la k-tupla más larga. El parámetro de la k-tupla determina cuantas identidades consecutivas son requeridas en una coincidencia. Para lo cual se utiliza el método de la diagonal para hallar todas las regiones de similitud entre dos secuencias. Determinar las 10 regiones con mejor potencial.

2. Reescanear las regiones más prometedoras. Cada región es un alineamiento parcial sin interrupciones. Para el reescaneo se utiliza una matriz de puntaje que permite correr las identidades más cortas y realizar reemplazos conservativos para contribuir al puntaje de similitud.

3. Comprobar a si las regiones pueden ser unidas para formar un alineamiento aproximado con interrupciones.

4. Calcular el puntaje de similitud es la suma de las regiones iníciales unidas menos una penalización para cada interrupción.

5. Construir el alineamiento múltiple.

Fuente: Bioinformática, 2005.

C.2 Algoritmo BLAST.

Desarrollado a partir de 1990 por Alschultz. Realiza un alineamiento local sin permitir la presencia de huecos, utilizando el algoritmo de Smith – Waterman o el algoritmo de Sellers, aunque con una ligera modificación para no permitir la presencia de huecos en el segmento alineado.

Los algoritmos BLAST devuelven todos los pares de segmentos mejor alineados a los que les denomina “Pares de segmentos con alto puntaje” o HSP por sus siglas en inglés. Para lo cual se recurre al análisis estadístico de secuencias generadas aleatoriamente. Tomando dos secuencias de longitud m y n el valor de los pares de alto puntaje depende de dos parámetros.

D *Otros métodos de alineamiento progresivo .*

1. Algoritmo POA. Extiende la idea del móldelo de programación entero del AMS (grafo) para n- secuencias. La base central del algoritmo consiste en ir agregando secuencias en forma progresiva al grafo que crece.

2. Algoritmo de Predicción de espacios por secuencias hereditarias. También es conocido como GASP (Gapped Ancestral Sequence Prediction) por sus siglas en inglés. Fue desarrollado por R.J. Edwards y D. Shields (2004). Este método permite obtener el alineamiento múltiple de secuencias no utiliza un valor fijo del factor de penalización de espacios vacíos, lo que asemeja a un proceso muta genético sufrido por el material durante el proceso evolutivo.

2.2.2.2 Alineamientos Híbridos

El algoritmo intenta alinear regiones cortas de la secuencia desconocida con regiones encontradas en la base de datos. La fase inicial de búsqueda consiste en identificar fragmentos parecidos en base de datos y posteriormente se alinean los nucleótidos fragmentos con la secuencia desconocida, si un nucleótido de la secuencia desconocida se encuentra ubicado exactamente en la misma posición de la secuencia de la base de datos, se le adjudica un punto positivo. En el caso de que la correspondencia fuese buena pero no perfecta, entonces un se le adjudica un puntaje más bajo. Cuando no existe una correspondencia ni buena ni perfecta entre los nucleótidos se le adjudica un puntaje negativo. La suma de los puntajes es utilizada para determinar el grado de similitud.

A Algoritmo híbrido.

A.1 *Algoritmo basado en función de consistencia basada en la evaluación objetiva del alineamiento*

La función basada en la evaluación objetiva del alineamiento conocida como también como función COFFEE (por las siglas de “consistency based objective function for alignment evaluation”) fue desarrollada por Cedric Notredame, Holm y Higgins en 1998.

La función COFFEE presenta algunas similitudes con la función de 'sumas por pares de las diferencias", en virtud de en ambas se considera que todas las que las operaciones de transformación en cada par de secuencia debe reflejarse en las relaciones del conjunto total de secuencias.

Definición 32. La f**unción COFFEE** implica que dado un conjunto $S = \{s_1, s_2, \dots, s_x\}$ la objetiva evaluación del alineamiento múltiple a través minimizar el costo del alineamiento.

$$max\ z = \frac{\sum_{a=1}^{x-1}\sum_{\substack{b=a+1 \\ a\neq b}}^{x} W_{s_a,s_b} * similitud\left(A_{\widehat{s_a},\widehat{s_b}}\right)}{\sum_{a=1}^{x-1}\sum_{\substack{b=a+1 \\ a\neq b}}^{x} W_{\widehat{s_a},\widehat{s_b}} * l} \qquad (40).$$

Fuente especificada no válida.

donde:

$A_{\widehat{s_a},\widehat{s_b}}$: Proyección de los alineamientos pareados de las secuencias.

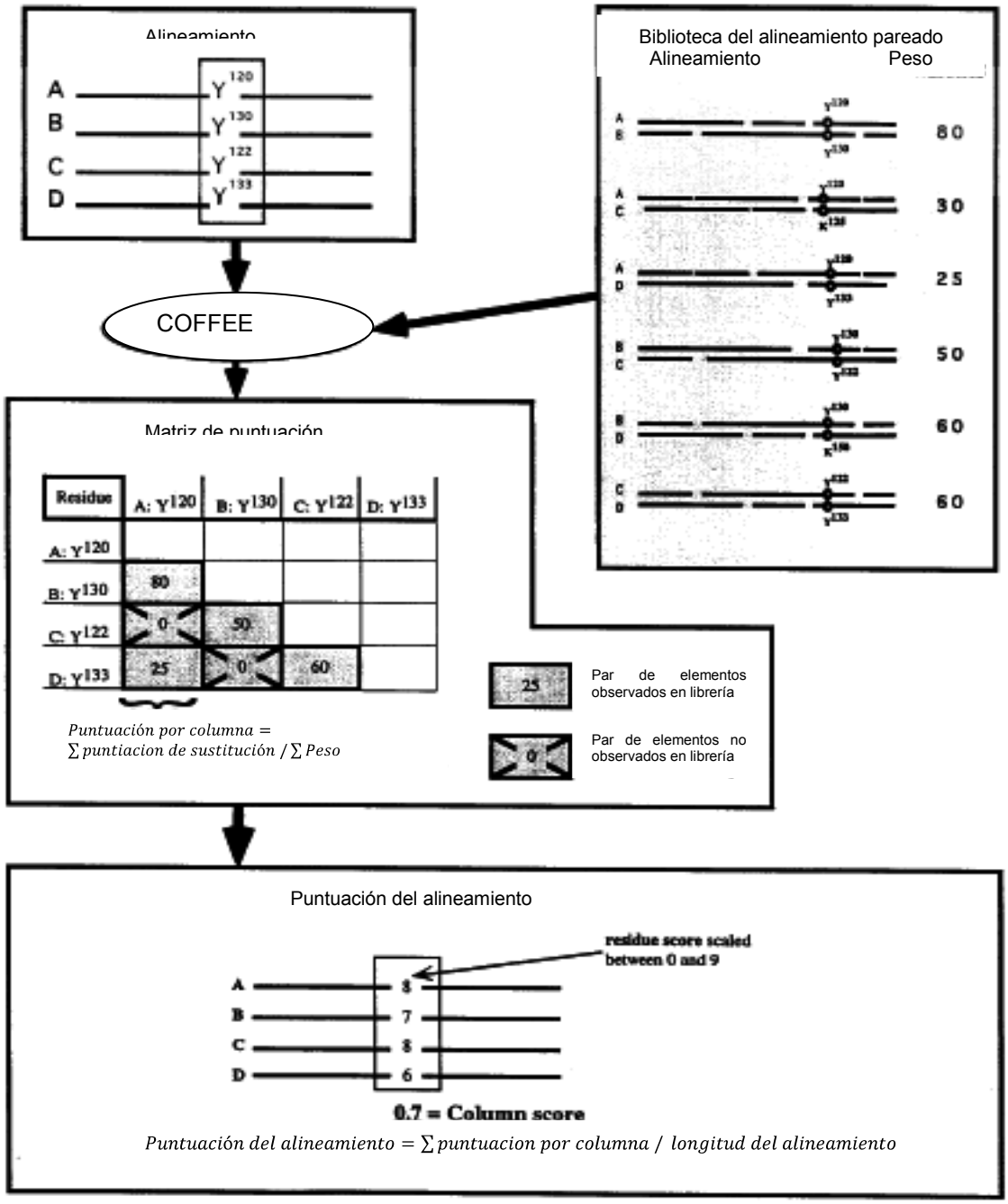
$similitud\left(A_{\widehat{s_a},\widehat{s_b}}\right)$: Valor del alineamiento entre $\widehat{s_a}, \widehat{s_b}$.

W_{s_a,s_b}: Ponderación del alineamiento entre $\widehat{s_a}, \widehat{s_b}$

l: numero de columnas de la matriz de alineamiento múltiple.

En la figura siguiente se muestra el esquema de puntuación de la función COFFEE.

Figura 19. Esquema de puntuación de la función COFFEE.



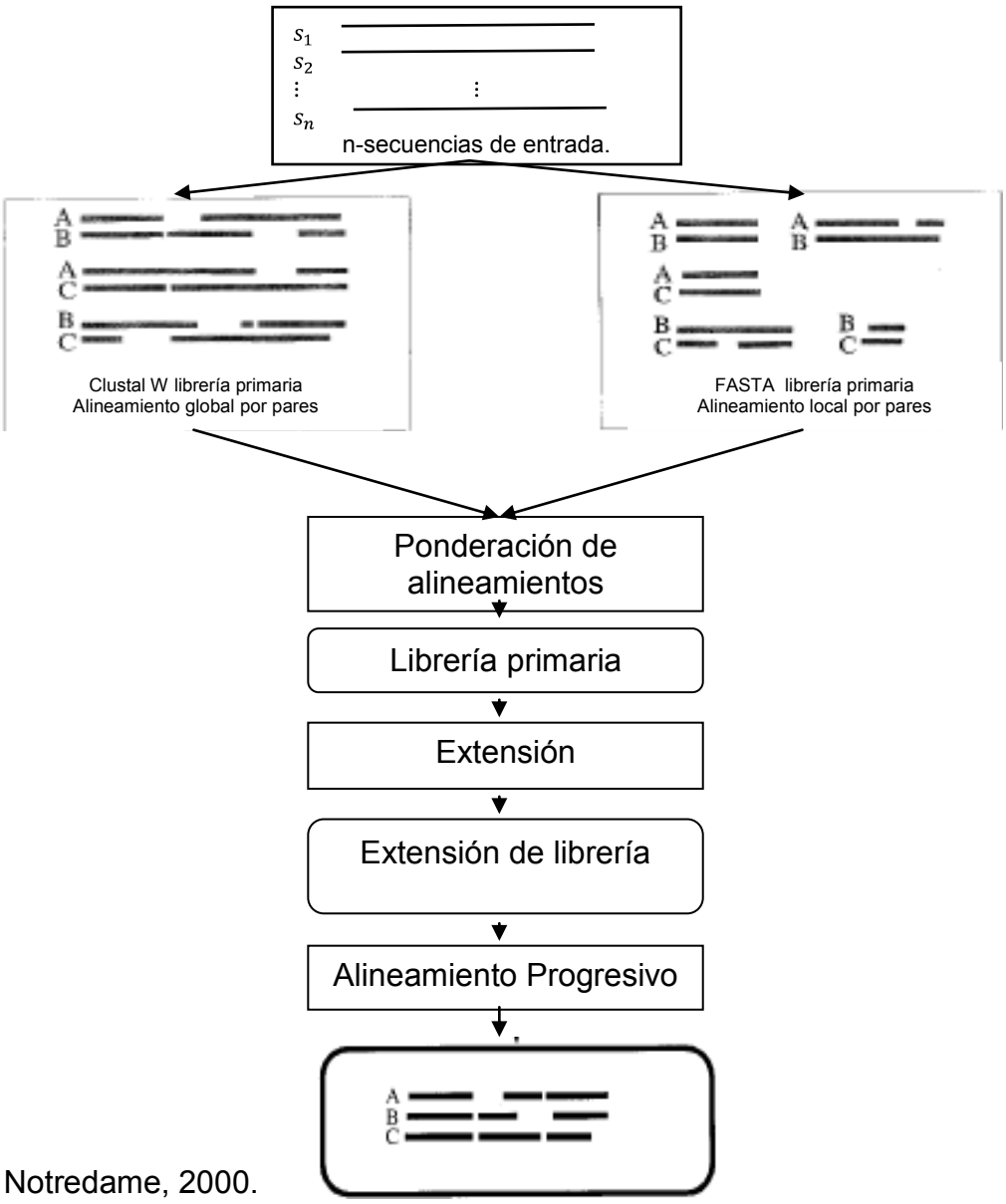
Fuente: Notredame, 2000.

El proceso de alineamiento por el método COFFEE es básicamente, la ejecución concatenada de dos procedimientos, los cuales son: obtener los alineamientos globales o locales pareados de las secuencias en el conjunto y optimizar el procedimiento al realizar una realineación progresiva para encontrar el alineamiento múltiple.

Algoritmo T-COFFE

Metodología desarrollada por Cedric Notredame, Desmond G. Higgins and Jaap Heringa en el 2000, la cual es una modificación del procedimiento COFFEE, en virtud de en este se utilizan dos librerías primarias. Lo anterior se esquematiza en la siguiente figura.

Figura 20. Método COFFEE.

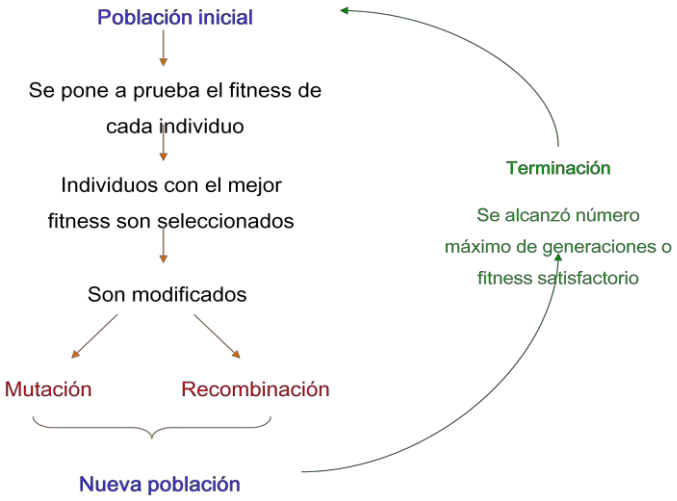


Fuente: Notredame, 2000.

2.2.2.3 Métodos Iterativos.

Son un conjunto de métodos para producir alineamientos múltiples de secuencias que reducen los errores inherentes en los métodos progresivos son los clasificados como “iterativos”, ya que trabajan de forma similar a los métodos progresivos, pero realinean repetidamente las secuencias iníciales además de añadir nuevas secuencias al MSA en crecimiento. Una razón por la que los métodos progresivos son tan fuertemente dependientes de la alta calidad del alineamiento inicial es el hecho de que estos alineamientos se incorporan siempre al resultado final; esto es, una vez que una secuencia ha sido alineada dentro del MSA, su alineamiento no vuelve a ser considerado. Este enfoque mejora la eficiencia a costa de la precisión.

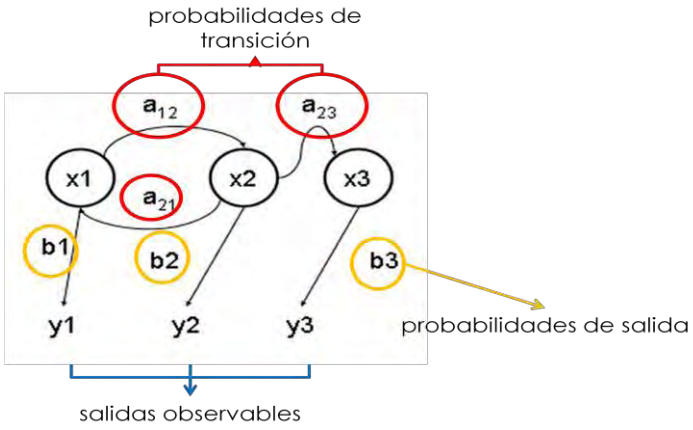
Figura 21. Características generales de los algoritmos iterativos.



A *Modelo oculto de Markov*

Es un modelo estadístico que asume que el sistema sigue un proceso de Markov de parámetros desconocidos.

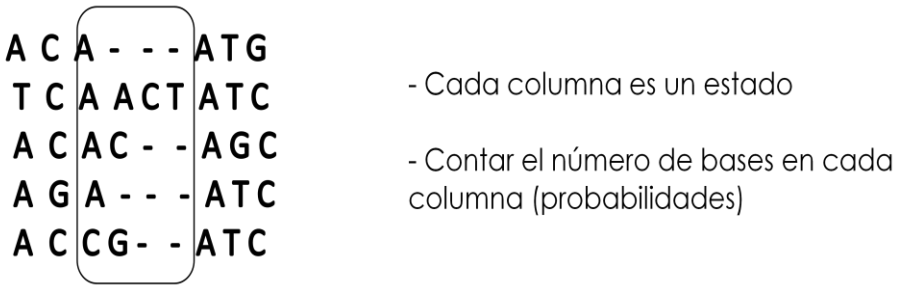
Figura 22. Modelo oculto de Markov



En este modelo existe una dependencia con el evento anterior. En este modelo el estado del sistema no es visible y cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida.

Dicho método que considera todas las posibles combinaciones de unidades y las transiciones para generar un alineamiento de un conjunto secuencial.

Figura 23. Elementos del HMM



B *Algoritmo A*.*

Método desarrollado por Ikeda e Imai en 1999. En él se utiliza la formulación IPL del problema del alineamiento múltiple de secuencias y estrategias de programación dinámica. Resulta conveniente para alinear menos de 100 secuencias largas. La complejidad de este procedimiento es $O(n^d)$ donde n es el número de secuencias del conjunto y d el tamaño de las secuencias. (Ikeda & Imai, 1999)

C *Algoritmos genéticos*

Los algoritmos genéticos se han utilizado en la producción de AMS intentando simular, el proceso evolutivo que da lugar a la divergencia en el conjunto problema. Este método trabaja rompiendo en fragmentos una serie de posibles AMS y reordenando repetidamente estos fragmentos con la introducción de huecos en diferentes posiciones. Donde se optimiza una función objetivo, normalmente una función de maximización “suma de pares” introducida en los métodos de AMS de programación dinámica.

D *Recocido simulado*

Mediante la técnica de recocido simulado, un alineamiento múltiple de secuencias inicial, producido por otro método, se mejora por una serie de reordenamientos diseñados para encontrar regiones más prometedoras en el espacio de alineamiento que la ya ocupada por el AMS previo. Aquí se maximiza una función objetivo como la suma de pares. Este método utiliza un “factor de temperatura” metafórico que determina el ritmo al cual los reordenamientos avanzan, así como la probabilidad de cada uno de ellos.

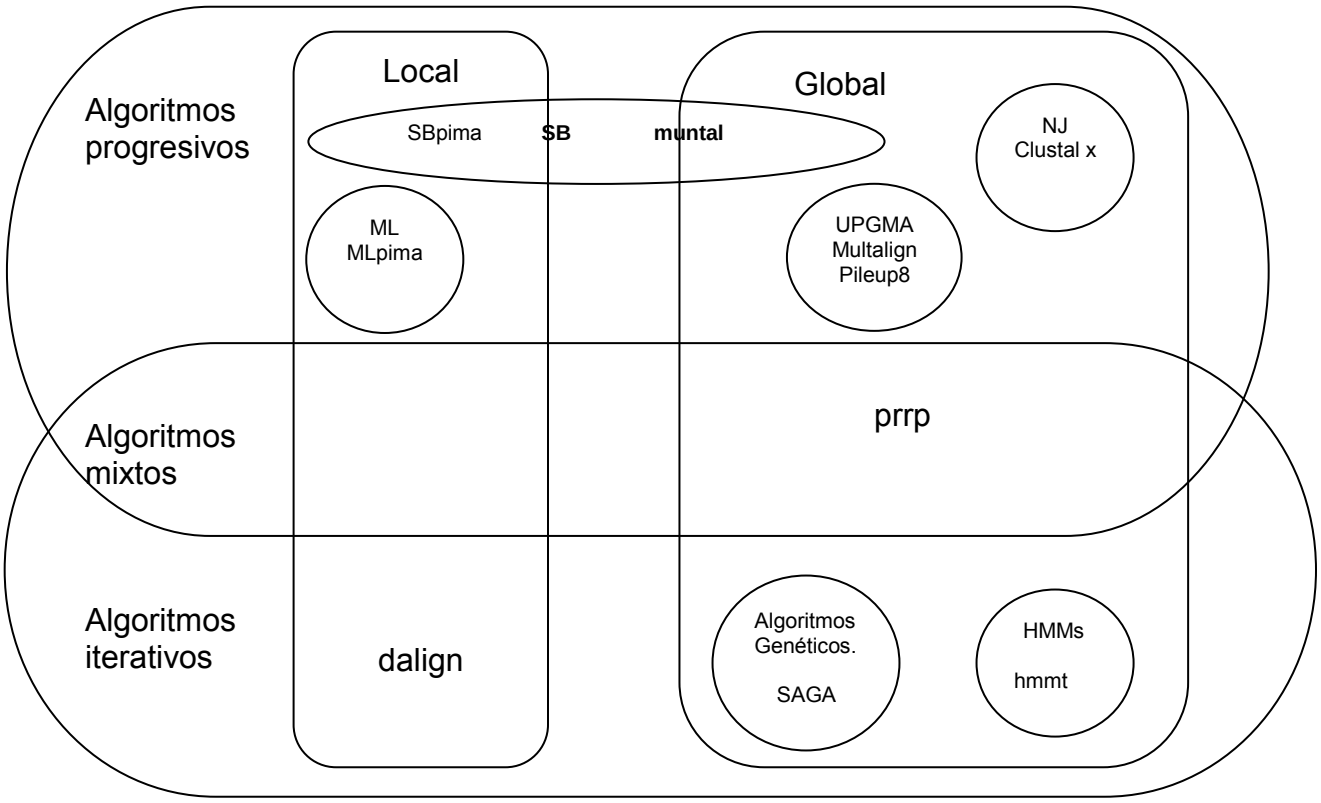
2.3 Programas de cómputo.

A medida que las técnicas (algoritmos) y tecnologías (aplicaciones computacionales) disponibles para resolver el problema AMS se incrementan, mejoran o se modifican, resulta importante conocer las características de eficiencia y eficacia de cada uno ellos con el objeto de establecer una elección correcta para la resolución de un caso particular, en virtud de cada uno de los algoritmos disponibles ofrecen una serie de ventajas y desventajas sobre el manejo que puede hacerse de la información que se desea analizar.

La elección correcta de las herramientas para enfrentar un problema resulta determinante en el grado de confianza que se puede tener en la solución encontrada.

En la figura siguiente se muestra una clasificación de los programas disponibles para resolver el AMS con base enl tipo de algoritmo que ocupan.

Figura 24. Esquema de relación entre programas y algoritmos



Fuente: Thompson, Plewniak, & Poch, 1999.

Los resultados encontrados por la ejecución de un algoritmo o programa de cómputo podrán variar por alguno de los siguientes motivos: (Thompson, Plewnlak, & Poch, BALiBASE: a benchmark aligment database for the evaluatios of multiple alignmet programs, 1999):

1. El grado de similitud entre las secuencias y el número de inserciones en la alineación.
2. La longitud de las secuencias.
3. La dominancia numérica de representación de algunos miembros de la familia de proteínas.

2.3.1 Método de comparación.

El primer estudio sobre la comparación de los programas desarrollados para resolver el AMS se realizo por Thompson, Plewniak & Poch (1999) en el trabajo "A comprehensive comparison of multiple sequence alignment programs", las aportaciones de dicho trabajo son[15]:

1. Desarrolla una metodología objetiva para la comparación de los algoritmos y programas de cómputo utilizados para resolver el problema de AMS
2. Crea una base de datos referencial sobre alineamientos múltiples de secuencias denominada BALIBASE (por sus siglas en inglés).
3. Establece dos métricas para cuantificar la eficiencia de los algoritmos y programas de cómputo, las cuales son: A) Puntaje de suma de pares y B) Puntuación entre columnas. Para la determinación se utiliza el programa bali_score .
4. Establece una base de datos sobre la eficiencia encontrada para algunos de los métodos más utilizados para resolver el problema de AMS.

2.3.1.1 Métricas de cuantificación de eficiencia.

Para evaluar el rendimiento de los algoritmos y programas de cómputo desarrollados para resolver el AMs se utilizan las métricas de cuantificación propuestas por Thompson, Plewniak, & Poch, las cuales son:

A) Puntuación de la suma de pares (SPS),

Dicho parámetro osilla entre 0 y 1, entre más cercano sea a uno mayor será la congruencia del alineamiento propuesto (M_p) con el alineamiento referencial (M_r) y se determina mediante la siguiente ecuación:

$$SPS = \frac{\sum_{i=1}^{M_p} S_i}{\sum_{i=1}^{M_r} S_{ri}} \qquad (41).$$

donde:
SPS: Puntuación de la suma de pares.
M_p: Número de columnas de alineamiento propuesto.
M_r: Número de columnas de alineamiento referencial.

[15] Las versiones de las bases de datos BAliBASE y el programa bali_score puede ser descargado desde el sitio web http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/ (Thompson, Plewnak, & Poch, 1999)

S_i: Puntuación de la i –ésima columna del alineamiento propuesto.
S_{ri}: Puntuación de la i –ésima columna del alineamiento referencial.

La puntuación de la i –ésima columna para cualquier alineamiento se determina mediante la siguiente expresión:

$$S_i = \sum_{\substack{j=1 \\ j \neq k}}^{x} \sum_{k=1}^{x} p_{ijk} \qquad (42).$$

donde:
p_{ijk}: Puntuación entre elementos de la i –ésima columna.

$$p_{ijk} = \begin{cases} 1 & si\ A_{ij} = A_{ik} \\ 0 & en\ caso\ contrario \end{cases} \qquad (43).$$

Se designa con $A_{i1}, A_{i2}, \ldots, A_{ix}$ a la i –ésima columna de la matriz de alineamiento, y sea $A_{ij}\ y\ A_{ik}$ un par de elementos de la i –ésima columna.

B) Puntuación entre columnas.
Determina el número de columnas de la matriz de alineamiento propuesto que son iguales al alineamiento referencial.

$$CS = \frac{\sum_{i=1}^{M_p} C_i}{M_p} \qquad (44).$$

donde:
CS: Puntuación entre columnas.
M_p: Número de columnas de alineamiento propuesto.
C_i= puntuación de la i –ésima columna.

$$C_i = \begin{cases} 1 & si\ M_{p_i} = M_{r_i} \\ 0 & en\ caso\ contrario \end{cases} \qquad (45).$$

donde:
M_{p_i}: i –ésima columna de la matriz de alineamiento propuesto.
M_{r_i}: i –ésima columa de la matriz de alineamiento referencial.

2.3.1.2 BALiBASE.

BALiBASE es una base de datos bilógica referencial[16] (alineamientos múltiples de secuencias) de caracter público sobre destinados a la evaluación y comparación de programas y algoritmos utilizados para resolver el AMS. Actualmente existen tres versiones de BALiBASE y continuación se da una descripción de cada una de estas versiones.

[16] Véase apéndice C Bases de datos biológicos.

A *BALiBASE versión 1.*

Es una colección de 141 alineamientos proteicos referenciales, que contiene más de 1000 secuencias[17] (Notredame 2000;Thompson, Plewnlak y Poch 1999 & Wang y Li 2004).

El conjunto de referencias se clasifican en**Fuente especificada no válida.**:

- **Referencia 1:** Contiene un conjunto de alineamientos, cada uno de los cuales se forma de a lo más 6 secuencias equi-distantes, de una longitud similar.
- **Referencia 2** Contiene un conjunto de alineamientos formado por secuencias estrechamente relacionas y secuencias. Dichos alineamientos contienen cuando menos 15 secuencias estrictamente relacionadas.
- **Referencia 3**. Contiene un conjunto de alineamientos formados por secuencias equidistantes de familias homólogas.
- **Referencia 4** Contienen un conjunto de alineamientos de hasta 20 secuencias, de extensión N/C terminal.
- **Referencia 5** Contienen un conjunto de alineamientos de hasta 20 secuencias de interna inserción.

El número de alineamientos de contenidos en cada una de las clases de este repertorio se muestra en la siguiente tabla.

Tabla 2. Número de alineamientos por cada referencia.

Referencia		Cortas (menos de 100 elementos por secuencia)	Medianas (de 200-300 elementos por secuencia)	Largas (más de 400 elementos por secuencia)
Referencia 1	V1 (menos del 25% identidad)	7	8	8
	V2 (del 20-40% identidad)	10	9	10
	V3 (más de 35% identidad)	10	10	8
Referencia 2		9	8	7
Referencia 3		5	3	5
Referencia 4		12		
Referencia 5		12		

Fuente: Thompson, Plewniak, & Poch, 1999.

[17] La versión 1 de BALiBASE puede consultarse en la página web http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/ (Thompson, Plewnlak, & Poch, 1999)

Esta es la única versión de BALiBASE que cuenta con los resultados del comportamiento de la eficiencia de los programas utilizados para resolver el AMS.

B *BALiBASE versión 2*

En esta versión se incluyen tres conjuntos de referencias (6-8) que contiene 26 familias de proteínas, en esta versión se representan 1100 secuencias. Las características de las referencias agregadas son[18]:

- **Referencia 6:** Contiene un conjunto de alineamientos proteicos construidos a partir de secuencias con alto grado de regiones repetidas.
- **Referencia 7:** Incluye casos de alineamientos de proteínas trans membranas,
- **Referencia 8:** Colección de alineamientos caracterizados por la permutación circular de las secuencias.

C *BALiBASE versión 3*

En esta versión se incluye un nuevo conjunto denominado referencia 9 en este se incluyen tres subconjuntos cuya característica es la existencia de motivos lineales en las secuencias del alineamiento. (Perrodou, Chica, Poch, Gibson, & Thompson, 2000)

Para la validación de la técnica desarrollada se utilizara la versión 1 de BALiBASE. A continuación se describirán las metaheurísticas que fueron tomadas como base para el desarrollo del procedimiento propuesto.

2.4 Metaheurísticas padres del método desarrollado.

La elección de las metaheurísticas búsqueda de armonía y recocido simulado se baso en:

1. Ambos procedimientos son metaheurísticas sencillas y coherentes, dado que, se basan en principios simples y claros, que permiten su fácil comprensión e implementación.
2. Son procedimientos generales y adaptables ya que en la literatura se han reportado un gran número de aplicaciones en diferentes problemas para contextos muy heterogéneos.
3. Son procedimientos efectivos y eficientes, con base a que en las aplicaciones reportadas en la literatura se han obtenido buenos resultados.
4. El recocido simulado ha si do utilizada para solucionar el AMS con excelentes resultados.
5. La metaheurística búsqueda de armonía es una técnica de reciente creación, por lo que no existe reportada una aplicación para resolver el AMS en la

[18] La versión 2 de BALiBASE puede consultarse en la página web: http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE2/ (Bahr, Thompson, Thierry, & Poch, 1999)

literatura, por ende la adaptación de este procedimiento para solucionar el AMS implica un aporte al conocimiento.

6. La combinación de métodos heurísticos permiten tomar mejores decisiones y proporcionan una mayor flexibilidad a la hora de definir diferentes estrategias de elección.

2.4.1 Búsqueda de armonía.

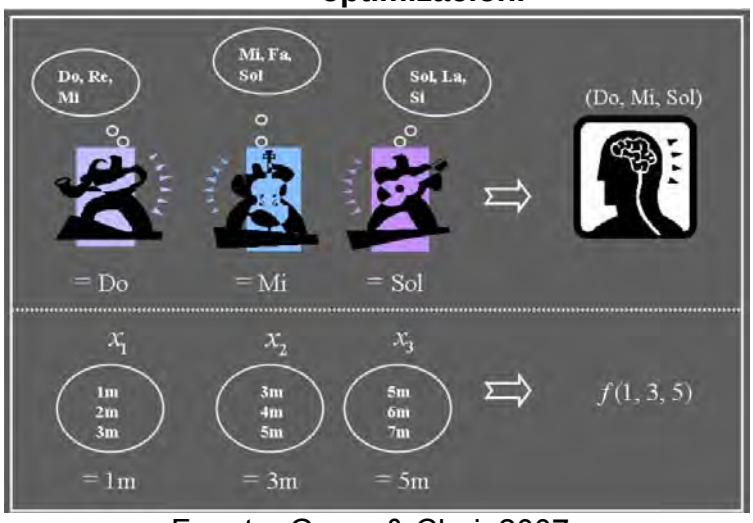
Zong Woo Geem en 2001 desarrolló esta metaheurística al combinar conceptos de ingeniería y música (Ingram & Zhang, 2009).

Antes de continuar se define de manera general a armonía como: el arte de combinar y organizar acordes, para conseguir la emoción deseada (alegría, melancolía, tristeza, etc.) en los espectadores. Es decir, es la habilidad de combinar sonidos desde un punto de vista artístico. En términos formales se refiere al estudio estético de la simultaneidad (estudio vertical) en secuencias musicales. Un acorde consiste en tres o más notas diferentes que suenan al mismo tiempo.

La búsqueda de armonía imita el proceso de perfeccionamiento del estado de armónico en la producción (innovación) musical, evaluando el estándar estético para cada innovación. Es decir, el grado en que se alcanza el estándar estético, permite evaluar a la combinación de sonidos, de cada uno de los instrumentos en cada ejecución. Lo que es similar a evaluar en la función objetivo los valores de las variables de decisión (Geem & Kim, A New heuristic Optimization Algoritm: Hamony Search, 2001).

Con objeto de ilustrar la analogía existente entre el proceso de innovación musical y el proceso de optimización considere la siguiente figura.

Figura 25. Analogía entre el proceso de innovación musical y la optimización.



Fuente: Geem & Choi, 2007.

En la figura siguiente se relaciona a cada uno de los tres músicos (saxofonista, contrabajista y guitarrita), con una variable de decisión $(x_1, x_2 x_3)$ del problema de optimización. A la gama de posibles notas que puede interpretar cada músico (saxofonista = {Do, Re, Mi}, contrabajista ={Mi, Fa, Sol} y guitarrista={ Sol, La, S}i) se le relaciona al rango de valor de cada una de las variables de decisión $(x_1 = \{1,2,3\}, x_2 = \{3,4,5\}\ y\ x_3 = \{5,6,7\})$. El conjunto de acordes posibles a ejecutar por los músicos se relaciona al conjunto de vectores factibles del problema de optimización. Para encontrar el mejor acorde de todos los acordes posibles se determina el grado de armonía del conjunto y se toma como mejor a aquel cuyo estándar estético es el más alto, por ejemplo, si el mejor acorde se alcanza cuando el saxofonista toca Do, el contrabajista toca Mi y el guitarrista toca Sol significa que esta combinación de sonido genera en mayor media la emoción deseada. Semejantemente en un problema de optimización la solución óptima es aquella en la que se alcanza el mejor valor de la función objetivo.

En la tabla siguiente se resume analogía existente entre la innovación musical y la optimización.

Tabla 3. Analogía entre optimización y innovación musical.

Proceso	Optimización	Innovación musical
Mejor estado	Óptimo global	Armonía fantástica
Estimación por	Función objetivo	Estándar estético..
Estimación con	Asignar valores a las variables	Ejecutar cada instrumento.
Proceso unitario	Cada iteración	Cada Práctica.

Fuente: Geem y Kim 2001.

En la innovación musical se utiliza una combinación de las siguientes operaciones básicas:

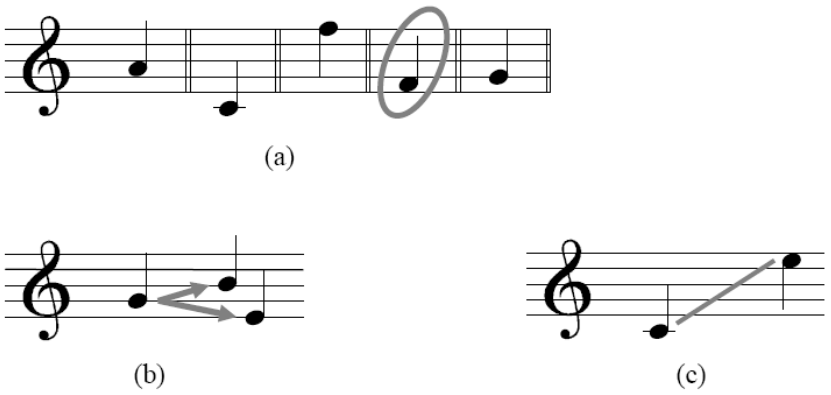
- **Recordar:** Utilizar algún acorde que se encuentre disponible en memoria.
- **Adaptar**: Realizar una modificación a un acorde disponible en memoria antes de ejecutarle.
- **Crear**: Implementar un nuevo acorde.

En la figura 26 se representan las operaciones de HS con base a una analogía musical.

En la búsqueda de armonía se incorporan estructuras y estrategias de otras metaheurísticas, como ejemplo, la memoria armónica es similar a la lista tabú de búsqueda tabú, la capacidad de variar y adaptar desde principio a fin del la ejecución el vector de búsqueda por medio de las consideración del ritmo de armonía es análogo a las estrategias de manejo de información utilizado por los algoritmos genéticos y el uso de bandas como el parámetro de ajuste del ritmo es

el símil de la búsqueda en vecindades del GRASP (adaptación de soluciones en memoria) (Geem & Kim, 2001).

Figura 26. Operaciones básicas del HS.



(a) Utilizar información disponible en memoria, (b) Adaptar información en memoria (c) Crear un nuevo acorde.
Fuente: Ingram y Zhang 2009.

La búsqueda armónica requiere que el problema de optimización que se desea resolver tenga la estructura siguiente: (Ingram & Zhang, 2009)

$$Optimizar\ \ f(x) \qquad (46).$$

Sujeta a: (47).

$$g(x) \geq 0$$

$$g''(x) \leq 0 \qquad (48).$$

$$g'(x) = 0 \qquad (49).$$

$$\begin{matrix} x_i^L \leq x_i \leq x_i^U & para\ variables\ continuas \\ x_i \in X_i & para\ variables\ discretas \end{matrix} \quad \forall\ i = 1,2,\dots,n \qquad (50).$$

donde:
$Optimizar\ \ f(x)$: Función objetivo a optimizar
x: Variables de decisión.
$g(x) \geq 0$: Restricciones de desigualdad del tipo mayor o igual.
$g''(x) \leq 0$: Restricciones de desigualdad del tipo menor o igual.
$g'(x) = 0$: Restricciones del tipo igualdad.
$x_i^L \leq x_i \leq x_i^U$: Restricción lógica para variables continuas que determinan el rango.
$x_i \in X_i$: Restricción lógica para variables discretas que indica el conjunto de valores posibles.

X_i: Conjunto de valores discretos para la variable x.
x_i^U: Valor superior del rango posible en variables continuas
x_i^L: Valor inferior del rango posible en variables continuas.

Además de ingresar al algoritmo la estructura del problema a optimizar como información de iniciol, la búsqueda de armonía requiere especificar los parámetros siguientes:

(a) $HMS \geq 1$ Tamaño de la memoria de armonía (valor discreto).
(b) $MaxImp > 1$ Máximo número de improvisaciones (valor discreto), el cual servirá como criterio de paro en el algoritmo.
(c) $0 \leq HMCR \leq 1$ Consideración del ritmo en la memoria de armonía.
(d) $0 \leq PAR \leq 1$ Parámetro de ajuste del ritmo.
(e) b Ancho de la banda usada en el parámetro de ajuste del ritmo.

2.4.2 Algoritmo de búsqueda de armonía.

El método de búsqueda armónica básicamente se compone de cinco pasos (Geem & Lee, 2004) los cuales se describen a continuación:

- **Paso 1 Inicializar proceso de optimización**: En él se ingresan los parámetros, y datos del problema de optimización necesarios para la ejecución del algoritmo.

- **Paso 2 Inicializar la memoria de armonía**: En esta fase se asignan aleatoriamente valores a las variables de decisión. Y se verifica el cumplimiento de restricciones; de no ser así se genera otra solución.

  Posteriormente se evalúa la función objetivo mediante el conjunto de valores factibles (x^i) y finalmente se forma la matriz de memoria armónica como se muestra en la ecuación siguiente:

$$HM = \left(\begin{array}{c|c} x^1 & f(x^1) \\ x^2 & f(x^2) \\ \vdots & \vdots \\ x^{HMS} & f(x^{HMS}) \end{array}\right) \qquad (51).$$

- **Paso 3 Improvisar una nueva armonía para la memoria armónica**: Se forma un nuevo vector de soluciones $x^i = \{x_1^i, x_2^i, \dots, x_N^i\}$, considerando la memoria de armonía y los parámetros de ajuste de ritmo (HMCR, PAR). Para lo cual el HMCR es la probabilidad de elegir un valor histórico contenido en la memoria HM para la i –ésima variable, mientras que, (1-HMCR) es la probabilidad de generar un nuevo valor aleatorio para dicha variable.

$$x_i^{'} \leftarrow \begin{cases} x_i^{'} \in \{x_i^1, x_i^2, \dots, x_i^{HMS}\} & con\ una\ probabilidad\ HMCR \\ x_i^{'} \in X_i & con\ una\ probabilidad\ (1 - HMCR) \end{cases} \qquad (52).$$

Entre más cercano sea el valor de $HMCR$ a uno, menor será la posibilidad de improvisar un nuevo el valor para las variable. La nueva armonía es examinada con el valor de ajuste del ritmo para cambiar el valor de la i –ésima variable con el objeto de escapar de óptimos locales, el cambio se realiza dentro de bandas. Esta operación es similar al proceso de mutación en los algoritmos genéticos.

$$= \begin{cases} Si & Ajuste\ del\ ritmo\ del\ vector\ x_i' \\ & se\ \ realiza\ el\ ajuste\ con\ la\ probabilidad\ PAR \\ No & e\ realiza\ el\ ajuste\ con\ la\ probabilidad\ (1 - PAR) \end{cases} \qquad (53).$$

Ajuste del valor de la i –ésima variable dentro de una banda.

Caso de variables continúas:

$$x_i' \leftarrow x_i' + \alpha \qquad (54).$$

$$\alpha = (br) * u \qquad (55).$$

donde:
x_i': Valor de la i –ésima variable.
α: Factor de ajuste del ritmo.
br: Distancia de ajuste del ritmo, y se calcula como:

$$br = b * rand \qquad (56).$$

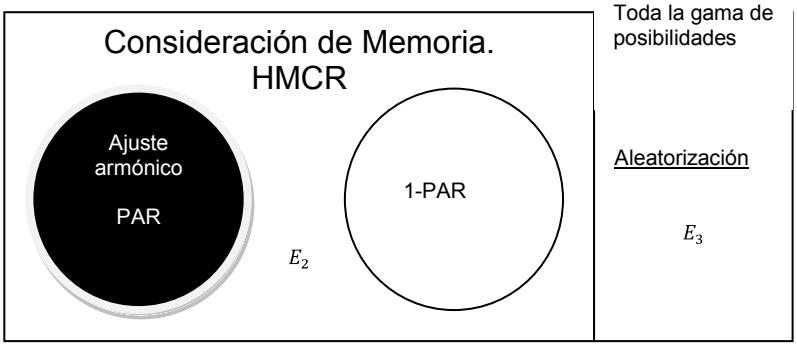
u: Sentido de ajuste del ritmo, el cual es un valor aleatorio generado de una distribución uniforme entre -1 y1.

Caso de variables discretas:

$$x_i' \leftarrow x_\alpha' \qquad (57).$$

donde:
x_α': Es un valor cercano a x_i' dentro de una banda tal que $x_a' \in X_i$
En la figura siguiente se da una representación gráfica de la nueva improvisación.

Figura 27. Concepto de nueva improvisación.



$$P(E_1) = HMCR * PAR \qquad (58).$$

$$P(E_2) = HMCR * (1 - PR) \qquad (59).$$

$$P(E_2) = 1 - HMCR \qquad (60).$$

Fuente: Geem y Lee 2004.

- **Paso 4 Actualizar memoria armónica**. Si el nuevo vector de armonía (x_i') al ser evaluado con la función objetivo es mejor que el peor vector armónico de memoria (x_{peor}), entonces x_i' remplazara a x_{peor}.

- **Paso 5 Cumplir criterio de paro**: Repetir el paso 3 y 4 hasta satisfacer el criterio de paro. Generalmente se utiliza el número de iteraciones (improvisaciones).

Para ilustrar el funcionamiento de algoritmo anterior considere lo siguiente:

Ejemplo 21. En una compañía manufacturera se desea determinar la secuenciación de operaciones que genere el costo mínimo total, cada uno de los trabajos se realiza una sola vez y el trabajo inicial debe coincidir con el trabajo final.

La matriz siguiente representa el costo de ir de trabajo a otro en pesos:

Del trabajo \ Al trabajo	A	B	C	D
A	--	300	250	492
B	765	--	121	431
C	982	350	--	103
D	706	689	921	--

Resolver el problema anterior mediante el algoritmo de búsqueda de armonía.

Paso 1 Inicializar el proceso de optimización

A) Formulación del problema de optimización:

Variables de decisión:

$$x_{ij} = \begin{cases} 1 & si\ el\ trabajo\ i\ se\ realiza\ antes\ que\ el\ trabajo\ j \\ 0 & en\ otro\ caso \end{cases}$$

Con la información anterior se construye la siguiente función objetivo

$$Min\ z: 300x_{AB} + 250x_{AC} + 492x_{AD} + 765x_{B4} + 121x_{BC} + 431x_{BD} + 982x_{C4} + 350x_{CB} + 103x_{CD} + 700x_{D4} + 689x_{DB} + 921x_{DC}$$

Sujeta a:

$x_{AB} + \mathrm{x_{AC}} + \mathrm{x_{AD}} = 1$
$x_{B4} + \mathrm{x_{BC}} + \mathrm{x_{BD}} = 1$
$x_{C4} + \mathrm{x_{CB}} + \mathrm{x_{CD}} = 1$
$x_{D4} + \mathrm{x_{DB}} + \mathrm{x_{DC}} = 1$
$x_{B4} + \mathrm{x_{C4}} + \mathrm{x_{D4}} = 1$
$x_{AB} + \mathrm{x_{CB}} + \mathrm{x_{DB}} = 1$
$x_{AC} + \mathrm{x_{BC}} + \mathrm{x_{DC}} = 1$
$x_{AD} + \mathrm{x_{BD}} + \mathrm{x_{CD}} = 1$
$-1 + 4x_{AB} \leq 3$
$-2 + 4x_{AC} \leq 3$
$-3 + 4x_{AD} \leq 3$
$-3 + 4x_{B4} \leq 3$
$-1 + 4x_{BC} \leq 3$
$-2 + 4x_{BD} \leq 3$
$-2 + 4x_{C4} \leq 3$
$1 + 4x_{CB} \leq 3$
$-1 + 4x_{CD} \leq 3$
$-1 + 4x_{D4} \leq 3$
$2 + 4x_{DB} \leq 3$
$1 + 4x_{DC} \leq 3$
$x_{A4} = 0$

B) Parámetros de entrada de la búsqueda de armonía. Para facilitar la implementación del algoritmo tómese los siguientes valores arbitrarios para los parámetros de entrada del algoritmo

(a) $HMS = 3.$
(b) $MaxImp = 10.$
(c) $HMCR = 0.90.$
(d) $PAR = 0.05.$
(e) $b = \lfloor 0.10 * |V| + 1 \rfloor$

Paso 2 Inicializar la memoria de armonía

Para la construcción de la memoria armónica inicial se debe considerar las características del problema que se desea resolver, ya que el vector x^i se construye a partir de los valores que asume cada una de las variables de decisión.

El problema a resolver en éste ejemplo es el agente viajero (TSP), por lo tanto para la formación del el vector x^k se involucra la definición de ciclo simple. Un **ciclo simple** para una $G = (V, E)$ es una trayectoria de longitud diferente a cero de v a v que no contiene aristas repetidas y en el que no hay vértices repetidos excepto por el inicio y fin que son iguales a v (Johnsonbaugh, 2005).

Con base a lo anterior el vector x^k, se representa con la sucesión de $|V| + 1$ vértices donde el vértice inicial y final de la sucesión son el mismo, es decir $x^k = \{x_1, x_2, x_3, x_4, x_1\}$. Por lo tanto la estructura de la matriz de memoria armónica correspondiente es:

$$HM = \left(\begin{array}{c|c} \{x_1, x_2, x_3, x_4, x_1\}^1 & f(x^1) \\ \{x_1, x_2, x_3, x_4, x_1\}^2 & f(x^2) \\ \vdots & \vdots \\ \{x_1, x_2, x_3, x_4, x_1\}^{HMS} & f(x^{HMS}) \end{array} \right)$$

Para la creación de la memoria armónica inicial del problema se utiliza el siguiente algoritmo.

Algoritmo 16. Procedimiento para la construcción de la HM inicial para el TSP del tipo binario.

Input: Matriz de costo asociado a cada arista del grafo $G = (V, E)$, tamaño de la memoria armónica HMS.

Output: Memoria armónica.

//Asignación de los parámetros de entrada//

1. $n \leftarrow |V|$

//Matriz de costos//

2. $C \leftarrow \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$
3. For $i = 1{:}\,1{;}\,n$
   a. $Q_{1,i} = v_i$

   End For
4. $h = length\,(Q)$
5. For $fila = 1{:}\,1{:}\,HMS$
   a. $contados \leftarrow \emptyset$
   b. $ciclo \leftarrow \emptyset$
   c. $c = n + 1$
   d. $x \leftarrow \emptyset$
   e. For $i = 1{:}\,1{:}\,n$
      i. $contador_{fila,i} \leftarrow 0$

      End For
   f. For $i = 1{:}\,1{:}\,n$
      i. $j = \lfloor 1 + ((h-1) * rand) \rfloor$
      ii. If $(i = 1)$
         1. $ciclo_{1,i} \leftarrow Q_{1,j}$
         2. $ciclo_{1,c} \leftarrow Q_{1,j}$
         3. $contador_{1,j} \leftarrow 1$
      iii. Else if $(contador_{1,j} = 0)$
         1. $ciclo_{1,i} \leftarrow Q_{1,j}$
         2. $contador_{1,j} \leftarrow 1$
      iv. Else
         1. While $(contador_{1,j} = 0)$
            a. $j = \lfloor 1 + ((h-1) * rand) \rfloor$

            End While

2. $ciclo_{1,i} \leftarrow Q_{1,j}$
3. $contador_{1,j} \leftarrow 1$

End If

End For

g. $cost = 0$

h. For $i = 1{:}\,1{:}\,c - 1$

i. $j = i + 1$

ii. $a = ciclo_{1,i}$

iii. $b = ciclo_{1,j}$

iv. $cost = cost + c_{a,b}$

End for

i. $ciclo_{1,c+1} \leftarrow cost$

j. For $columa = 1{:}\,1{:}\,c + 1$

i. $HM_{fila,columa} \leftarrow ciclo_{1,columan}$

k. End For

End for

A continuación se muestra la aplicación del algoritmo anterior para crear el primer renglón de la memoria armónica.

Pasos 1 al 4

//nodos en el grafo//.

$$n \leftarrow 4 = |V|$$

//Matriz de costos//

$$C \leftarrow \begin{bmatrix} \infty & 300 & 250 & 492 \\ 765 & \infty & 121 & 431 \\ 982 & 350 & \infty & 103 \\ 706 & 689 & 921 & \infty \end{bmatrix}$$

$$Q = [1 \quad 2 \quad 3 \quad 4]$$

$$h = 4$$

Paso 5 con $fila = 1$.

a. $contados \leftarrow \emptyset$

b. $ciclo \leftarrow \emptyset$

c. $c = 5$

d. $x \leftarrow \emptyset$

e. $contador = [0 \quad 0 \quad 0 \quad 0]$

f. Ciclo for

Para $i = 1$

$j = \lfloor 1 + (3 * rand) \rfloor = 2$

Dado que $i = 1$ entonces

1. $ciclo_{1,1} \leftarrow 2$
2. $ciclo_{1,5} \leftarrow 2$
3. $contador_{1,2} \leftarrow 1$

Para $i = 2$

$j = \lfloor 1 + (3 * rand) \rfloor = 2$

Dado que $i \neq 1$ entonces

Ya que $contador_{1,2} \neq 0$ entonces

1. Se genera otro valor para j

a. $j = \lfloor 1 + (3 * rand) \rfloor = 4$

2. $ciclo_{1,2} \leftarrow 4$
3. $contador_{1,4} \leftarrow 1$

Para $i = 3$

$j = \lfloor 1 + (3 * rand) \rfloor = 1$

Dado que $i \neq 1$ entonces

Ya que $contador_{1,3} = 0$ entonces

1. $ciclo_{1,3} \leftarrow 3$
2. $contador_{1,1} \leftarrow 1$

Para $i = 4$

$j = \lfloor 1 + (3 * rand) \rfloor = 3$

Dado que $i \neq 1$ entonces

Ya que $contador_{1,3} = 0$ entonces

1. $ciclo_{1,4} \leftarrow 3$
2. $contador_{1,3} \leftarrow 1$

Termina ciclo for y se obtiene lo siguiente:

$$ciclo = [2 \quad 4 \quad 1 \quad 3 \quad 2]$$
$$ciclo = [1 \quad 1 \quad 1 \quad 1]$$

g. $cost = 0$

h. Ciclo for

Para $i = 1$

i. $j = 2$
ii. $a = 2$
iii. $b = 4$
iv. $cost = 0 + (c_{2,4} = 431) = 431$

Para $i = 2$

i. $j = 3$
ii. $a = 4$
iii. $b = 1$
iv. $cost = 431 + (c_{4,1} = 706) = 1137$

Para $i = 3$

i. $j = 4$
ii. $a = 1$
iii. $b = 3$
iv. $cost = 1137 + (c_{1,3} = 250) = 1387$

Para $i = 4$

i. $j = 5$
ii. $a = 3$
iii. $b = 2$
iv. $cost = 1387 + (c_{3,2} = 350) = 1737$

Termina Ciclo for y se obtiene $cost = 1387$

i. $ciclo = [2 \quad 4 \quad 1 \quad 3 \quad 2 \quad |1387]$

j. Se genera la primera hilera de la matriz de armonía y se obtiene

i. $HM = [2 \quad 4 \quad 1 \quad 3 \quad 2 \quad |1387]$

Siguiendo el mismo procedimiento anterior para generar cada hilera de la matriz armónica.

Termina ciclo for y se obtiene la matriz de armonía

$$HM = \begin{bmatrix} 2 & 4 & 1 & 3 & 2 & |1387 \\ 2 & 1 & 3 & 4 & 2 & |1737 \\ 3 & 2 & 3 & 1 & 2 & |2634 \end{bmatrix}$$

Paso 3 Improvisar una nueva armonía para la memoria armónica

El improvisar una nueva armonía implica generar una perturbación del sistema en la $i-$ ésima variable de decisión a fin de generar una nueva solución factible para lo cual se ocupa el siguiente algoritmo.

Algoritmo 17. Generación de una nueva armonía para el TSP del tipo binario.

Input: Matriz de memoria armónica, $HMCR, PAR, c\ y\ b$.

Output: Nueva armonía.

1. $b = \lfloor (0.10 * n) + 1 \rfloor$
2. $nuevo\_ciclo \leftarrow \emptyset$
3. For $i = 1{:}\,1{:}\,c - 1$
   i. $contador_{1,i} \leftarrow 0$
   ii. $Q_{1,i} = v_i$

   End for
4. $h = length\ (Q)$
5. For $i = 1{:}\,1{:}\,c - 1$

   //Posibilidad de elegir información en memoria//

   i. If $(rand \leq HMCR)$

   //Elección aleatoria del información en la HM //

   1. $u = \lfloor 1 + ((HMS - 1) * rand) \rfloor$
   2. $j = HM_{u,i}$
   3. If $(i = 1)$
      a. $nuevo\_ciclo_{1,1} = j$
      b. $nuevo\_ciclo_{1,c} = j$
      c. $contador_{1,j} = 1$
   4. Else if $(contador_{1,j} = 0)$
      a. $nuevo\_ciclo_{1,i} = j$
      b. $contador_{1,j} = 1$
   5. Else
      a. While $(contador_{1,j} = 0)$
         i. $u = \lfloor 1 + ((HMS - 1) * rand) \rfloor$
         ii. $j = HMS_{u,i}$
      b. End
      c. $nuevo\_ciclo_{1,i} = j$
      d. $contador_{1,j} = 1$

   End
   6. If $(rand \leq PAR)$
      a. $banda = \lfloor rand * b \rfloor$

b. If $(rand \leq 0.5)$
    i. $ja = j + banda$
    ii. If $(ja > n)$
        1. $ja = n$

        End If
    iii. If $(i = 1)$
        1. $nuevo\_ciclo_{1,i} = ja$
        2. $nuevo\_ciclo_{1,c} = ja$
        3. $contador_{1,j} = 0$
        4. $contador_{1,ja} = 1$
    iv. Else if $\left(contador_{1,ja} = 0\right)$
        1. $contador_{1,j} = 0$
        2. $contador_{1,ja} = 1$
        3. $nuevo\_ciclo_{1,i} = ja$
    v. Else
        1. While $(contador_{1,ja} \neq 0)$
            a. $banda = \lfloor rand * b \rfloor$
            b. $ja = j + banda$
            c. If $ja > n$
                i. $ja = n$

                End if

            End while
        2. $contador_{1,j} = 0$
        3. $contador_{1,ja} = 1$
        4. $nuevo\_ciclo_{1,i} = ja$

c. Else
    i. $ja = j - banda$
    ii. If $(ja \leq 1)$
        1. $ja = 1$

        End If
    iii. If $(i = 1)$
        1. $nuevo\_ciclo_{1,i} = ja$
        2. $nuevo\_ciclo_{1,c} = ja$
        3. $contador_{1,j} = 0$
        4. $contador_{1,ja} = 1$
    iv. End
    v. $\left(contador_{1,ja} = 0\right)$
        1. $contador_{1,j} = 0$
        2. $contador_{1,ja} = 1$
        3. $nuevo\_ciclo_{1,i} = ja$
    vi. $\left(contador_{1,ja} \neq 0\right)$
        1. While $(contador_{1,ja} \neq 0)$
            a. $banda = rand * b$

b. $ja = j + banda$
c. If $ja \geq n$
i. $ja = n$
End if
End while
2. $contador_{1,j} = 0$
3. $contador_{1,ja} = 1$
4. $nuevo\_ciclo_{1,i} = ja$
End if
End if
ii. Else
1. $j = \lfloor 1 + ((h-1) * rand) \rfloor$
2. If $(i = 1)$
a. $nuevo\_ciclo_{1,i} \leftarrow Q_{1,j}$
b. $nuevo\_ciclo_{1,c} \leftarrow Q_{1,j}$
c. $contador_{1,j} \leftarrow 1$
3. Else if $(contador_{1,j} = 0)$
a. $nuevo\_ciclo_{1,i} \leftarrow Q_{1,j}$
b. $contador_{1,j} \leftarrow 1$
4. Else
a. While $(contador_{1,j} = 0)$
i. $j = \lfloor 1 + ((h-1) * rand) \rfloor$
End While
b. $nuevo\_ciclo_{1,i} \leftarrow Q_{1,j}$
c. $contador_{1,j} \leftarrow 1$
End If
End For
6. $cost = 0$
7. For $i = 1:1:c-1$
i. $j = i + 1$
ii. $a = nuevo\_ciclo_{1,i}$
iii. $b = nuevo\_ciclo_{1,j}$
iv. $cost = cost + c_{a,b}$
End for
8. $nuevo\_ciclo_{1,c+1} \leftarrow cost$
End for

A continuación se calcula una nueva armonía utilizando el algoritmo anterior.

Pasos 1 al 5

1. $b = \lfloor (0.10 * 4) + 1 \rfloor = 1$
2. $nuevo\_ciclo \leftarrow \emptyset$

3.

i. $contador = [0 \quad 0 \quad 0 \quad 0]$

ii. $Q = [1 \quad 2 \quad 3 \quad 4]$

4. $h = length\ (Q)$

Paso 5 ciclo for

$\boldsymbol{i = 1}$

Se genera un número aleatorio $rand = 0.98$ que es mayor a $HMCR$ y por lo tanto

$j = \lfloor 1 + ((3) * 1) \rfloor = 4$

Dado a que $i = 1$ entonces se obtiene

a. $nuevo\_ciclo_{1,1} \leftarrow 4$

b. $nuevo\_ciclo_{1,5} \leftarrow 4$

c. $contador_{1,4} \leftarrow 1$

$\boldsymbol{i = 2}$

Se genera un número aleatorio $rand = 0.15$ que es menor a $HMCR$ y por lo tanto

$u = \lfloor 1 + ((2) * 0.59) \rfloor = 2$

$j = HM_{2,2} = 1$

Dado a que $contador_{1,1} = 0$ entonces se obtiene

a. $nuevo\_ciclo_{1,2} \leftarrow 1$

b. $contador_{1,1} \leftarrow 1$

Se genera un número aleatorio $rand = 0.45$ que es mayor a PAR y por lo tanto no se aplica un ajuste del ritmo.

$\boldsymbol{i = 3}$

Se genera un número aleatorio $rand = 0.89$ que es menor a $HMCR$ y por lo tanto

$u = \lfloor 1 + ((2) * 0.15) \rfloor = 1$

$j = HM_{1,3} = 1$

Dado a que $contador_{1,1} \neq 0$ entonces se obtiene

a. $u = \lfloor 1 + ((2) * .58) \rfloor = 2$

b. $j = HM_{2,3} = 3$

c. $nuevo\_ciclo_{1,3} \leftarrow 3$

d. $contador_{1,3} \leftarrow 1$

Se genera un número aleatorio $rand = 0.04$ que es menor a PAR y por lo tanto se aplica un ajuste del ritmo.

$$banda = (1 * 1) = 1$$

Se genera otro número aleatorio $rand = 0.69$ que es mayor a 0.5 por lo cual:

$$ja = 3 - 1 = 2$$

Dado a que $contador_{1,2} = 0$ entonces se obtiene

a. $nuevo\_ciclo_{1,3} \leftarrow 2$
b. $contador_{1,3} \leftarrow 0$
c. $contador_{1,2} \leftarrow 1$

Se genera un número aleatorio $rand = 0.45$ que es mayor a PAR y por lo tanto no se aplica un ajuste del ritmo.

$\boldsymbol{i = 4}$
Se genera un número aleatorio $rand = 0.39$ que es menor a $HMCR$ y por lo tanto
$u = \lfloor 1 + ((2) * 0.20) \rfloor = 1$
$j = HM_{1,4} = 1$
Dado a que $contador_{1,1} = 0$ entonces se obtiene
a. $nuevo\_ciclo_{1,2} \leftarrow 3$
b. $contador_{1,3} \leftarrow 0$

Se genera un número aleatorio $rand = 0.95$ que es mayor a PAR y por lo tanto no se aplica un ajuste del ritmo.

Termina el ciclo for y se obtiene

$$nuevo\_ciclo = [4 \quad 1 \quad 2 \quad 3 \quad 4]$$

Por último se evalúa el valor del ciclo anterior, de forma semejante a lo aplicado en el método anterior, con lo que se obtiene:

$$cost = 706 + 300 + 121 + 103 = 1230$$
$$nuevo\_ciclo = [4 \quad 1 \quad 2 \quad 3 \quad 4 \quad |1230]$$

Paso 4 Actualizar memoria armónica.

En esta etapa se determina si el vector de la nueva armonía tiene un valor mejor que el peor vector contenido en la memoria armónica, para el ejemplo del TSP se verifica si el costo asociado al nuevo ciclo es menor, que el valor del ciclo con valor más alto contenido en la HM.

Para este propósito se utiliza el algoritmo siguiente.

Algoritmo 18. Actualización de la memoria armónica para el TSP binario.

Input: Matriz de costo, c**,** matriz de memoria y nueva armonía

Output: Matriz de memoria armónica actualizada.

1. $[peor\_costo, fila] = \max(MH_{:,c+1})$
2. If $(peor\_costo \geq nuevo\_ciclo_{1,c+1})$
    1. $MH_{fila,:} \leftarrow nuevo\_ciclo$

End if

Continuando con el ejemplo anterior, se tiene que dada la matriz de armonía siguiente: $\begin{bmatrix} 2 & 4 & 1 & 3 & 2 & |1387 \\ 2 & 1 & 3 & 4 & 2 & |1737 \\ 3 & 2 & 3 & 1 & 2 & |2634 \end{bmatrix}$ y el vector de nueva armonía $[4 \quad 1 \quad 2 \quad 3 \quad 4 \quad |1230]$ se desea actualizar la HM actualizada.

Dentro de HM el peor valor es 2634 que se encuentra en la tercera fila ya que $\left(MH_{3,c+1} = 2634\right) \geq \left(nuevo\_ciclo_{1,c+1} = 1230\right)$ se procede a remplazar la tercera fila de la HM por la nueva armonía con lo que se obtiene.

$$HM = \begin{bmatrix} 2 & 4 & 1 & 3 & 2 & |1387 \\ 2 & 1 & 3 & 4 & 2 & |1737 \\ 4 & 1 & 2 & 3 & 4 & |1230 \end{bmatrix}$$

Paso 5 Cumplir criterio de paro:

Este paso implica que deben realizarse los pasos 3 y 4 hasta satisfacer un número predeterminado el criterio iteraciones, lo cual se representa mediante el siguiente algoritmo.

Algoritmo 19. Cumplimiento del criterio de paro para el TSP binario.

Input: Matriz de memoria inicial, Matriz de costo, $MaxImp, HMCR, PAR, b \ y \ c,$

Output: Solución aproximada para el problema TPS.

1. For $v = 1{:}\,1{:}\,MaxImp$
   i. Generación de una nueva armonía.
   ii. Actualización de la memoria armónica.
   End For
2. $[costo\_solución, fila] = \min\left(MH_{:,c+1}\right.$
3. $solución \leftarrow MH_{fila,:}$

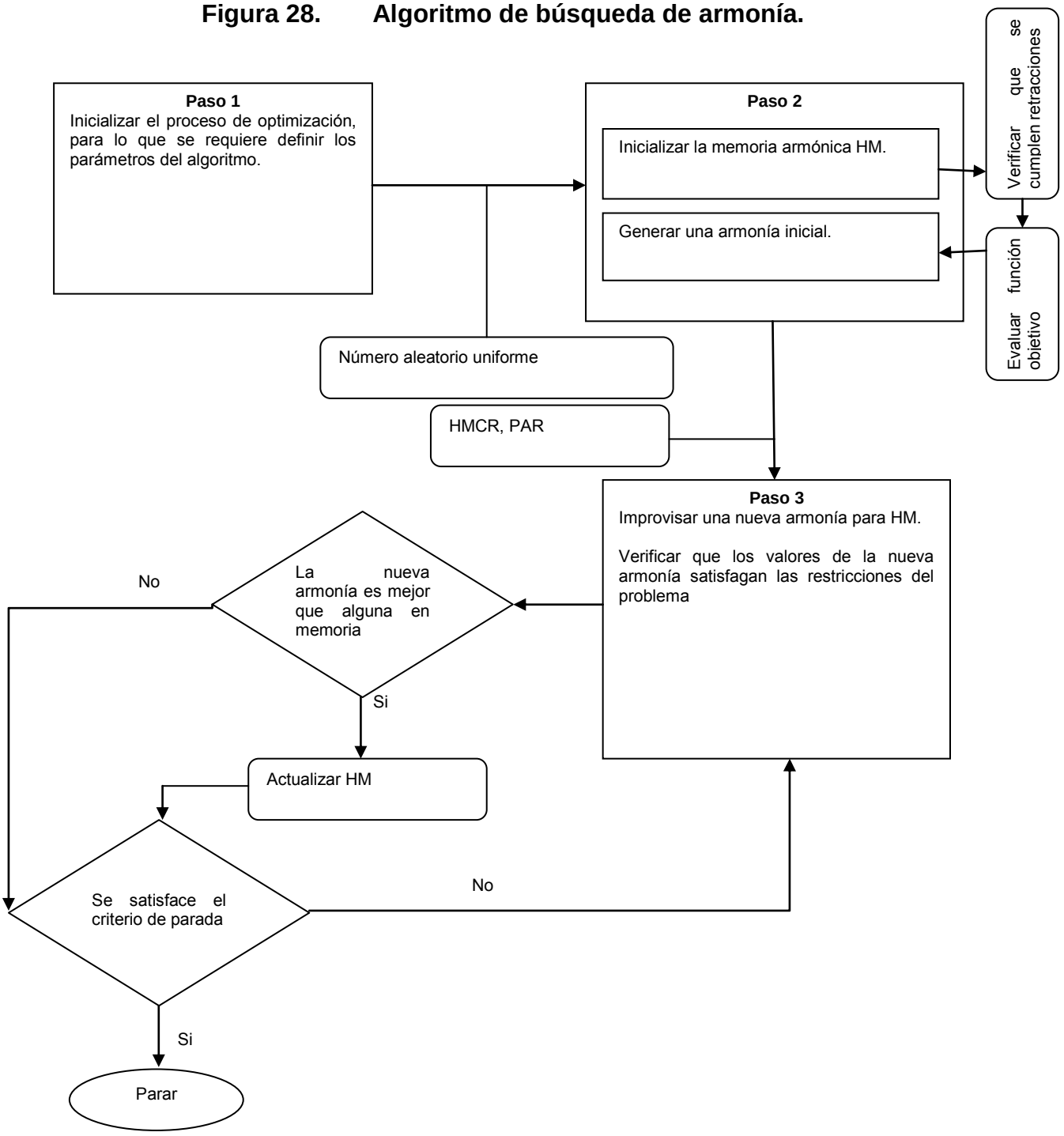
Al realizar el número de iteraciones determinadas para el ejemplo se determina la siguiente matriz armónica final.

$$HM = \begin{bmatrix} 2 & 4 & 1 & 3 & 2 & |1387 \\ 2 & 3 & 4 & 1 & 2 & |1230 \\ 4 & 1 & 2 & 3 & 4 & |1230 \end{bmatrix}$$

Por lo tanto el ciclo solución del problema involucra un costo de 1230 u.m.

Una vez analizado el ejemplo anterior, es posible entender las características del algoritmo de búsqueda de la armonía en conjunto, en la figura siguiente muestra el proceso de optimización a través de HS.

Figura 28. Algoritmo de búsqueda de armonía.



Fuente: (Geem & Lee, 2004)

2.4.3 Aplicaciones a problemas de optimización.

La búsqueda de armonía ha sido utilizado como estrategia para solucionar una gran variedad de problemas que van desde la optimización combinatoria, entera, de redes hasta programación no lineal. Como ejemplo se mencionan los siguientes problema: el agente viajero, flujo en redes (gaseoductos, agua potable etc.), determinación de estructuras, parámetros hidrológicos, composición musical, problemas de transporte (modelos de energía), scheduling, el problema de Proyección de retorno, entre otros.En la tabla siguiente se muestran el valor de los parámetros recomendado o usado al aplicar la HS para algunos de los problemas anteriores.

Tabla 4. Valor de los parámetros al aplicar de búsqueda armónica valor

Aplicación	*N*	*HMS*	*HMCR*	*PAR*	*MaxImp*	Autor referencia
Problema de Proyección de retorno	2-10	20	0.9	0.35	15-230000	Lee and Geem (2005)
	2-5,15,30,100	5	0.9	0.3	50000	Omran and Mahdavi (2008)
Designación estructural	10-29	20	0.8	0.3	5000	Lee and Geem (2004)
	8-27	20-40	0.8-0.9	0.3-0.45	30-80000	Lee and Geem (2005)
	2-13	30	0.9	0.4	4000	Saka (2007)
	2-20	50	0.8	0.5	5000	Degertekin(2007)
	2	50	0.9	0.5		Erdal and Saka (2008)
Distribución de redes de agua.	8	100	0.95	0.05	5000	Geem (2006)
	34	50	0.93	0.18	200,000	
	21	30	0.9	0.1	20,000	
	30	30-100	0.7-0.95	0.3-0.7	10,000	
	9	30-100	0.7-0.95	0.3-0.7	5000	
Estimación de parámetros hidrológicos	3	100	0.95	0.05	5000	Kim (2001)
Modelo de transporte de energía	4,7	10	0.8	0.4	100,000	Ceylan *et al* (2008)
Modelación de acuíferos	2-10	10	0.9	0.45	50000	Ayvaz (2007)
Scheduling múltiple	48	30	0.95	0.05	35000	Geem (2007)

Operaciones múltiples de bombas	40	19	0.97	0	3500	Geem (2005)
Ruteo de autobuses escolares	10	20	0.9-0.95	0	1000	Geem (2005)
Composición musical	22,48	10	0.9	0.3	3000	Geem and Choi (2007)

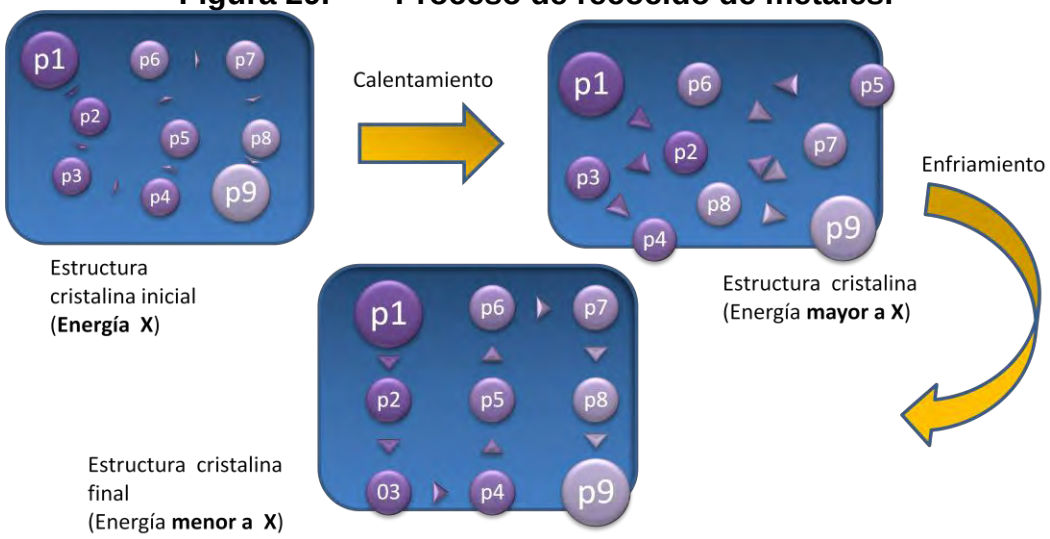
Fuente: Ingram y Zhang 2009

donde:
N: Tamaño o Número de entradas del problema a resolver.
HMS: Tamaño de la memoria armónica.
HMCR: Consideración del ritmo de la memoria armónica.
PAR: Parámetro de ajuste del ritmo.
MaxImp: Máximo número de improvisaciones.

2.5 Recocido simulado

El recocido simulado (RS) también es conocido como temple simulado, pertenece al grupo de metaheurísticas clásicas, se ha caracterizado por su simplicidad, robustez, adaptabilidad, eficiencia y eficacia. Esta metaheurística imita el proceso de temple de metales que ocurre con los cambios energéticos, en otras palabras se inicializa con un sistema de partículas excitadas, las cuales modifican su nivel de exaltación a medida que decrece la temperatura hasta alcanzar el estado deseable. Desde la óptica de la optimización, el estado final del metal constituye la solución óptima al objetivo de minimizar la energía. En la figura siguiente se muestra el proceso de recocido de metales.

Figura 29. Proceso de recocido de metales.



Elaborada a partir de Vélez & Montoya, 2007.

Kirkpatrick, Gelatt y Vecchi (1983) establecieron la analogía del recocido de metales y la búsqueda de la solución óptima en un problema de optimización. Para lo cual, se empleo el trabajo de Metrópolis (área de termodinámica estadística) como base en el desarrollo del RS. En la tabla siguiente se muestra la analogía entre el temple simulado de metales y el proceso de optimización.

Tabla 5. Analogía entre la optimización y el recocido metalúrgico de metales.

Proceso	Optimización	Metalurgia.
Estado inicial del sistema	Solución inicial	Configuración excitada de las moléculas.
Mejor estado	Solución optima	Configuración fundamental de las moléculas
Estimación por	Función objetivo	Estado de mínima energía
Estimación con	Costo de la solución	Energía de la configuración
Proceso unitario	Cada iteración	Cambio del estado energético

El recocido simulado se caracteriza por hacer uso de la búsqueda en vecindades, la probabilidad de aceptar soluciones de menor calidad, para permitir a esta metaheurística escapar de posibles óptimos locales. Es decir, este algoritmo acepta todos los movimientos que mejoren la solución de manera automática adicionalmente existe la probabilidad de admitir movimientos peores.

Siguiendo con la analogía física las leyes de la termodinámica establecen que a una temperatura t la probabilidad de incrementar la engría de las moléculas (dE), con objeto de alcanzar el estado deseable puede se aproxima por la formula:

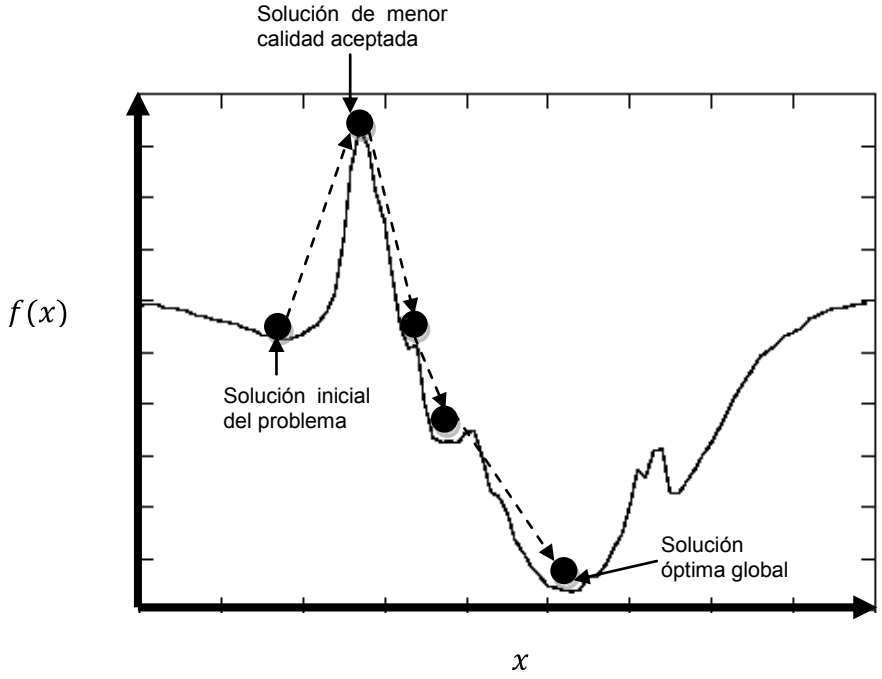
$$P(dE) = e^{-\frac{dE}{kt}} \qquad (61).$$

donde:
t= Temperatura del sistema.
$P(dE)$: Probabilidad de aceptar soluciones de peor calidad.
dE: Cambio de temperatura.
k: Constante de Boltzmann.

En las implementaciones del RS como procedimiento de solución a problemas de optimización, se ocupa las leyes de la termodinámica para determinar la probabilidad de estados peores, sin embargo en esta adaptación la constante de Boltzmann k en general no se considera, debido a que no tiene significado para los problemas de optimización. La probabilidad de aceptación de estados peores cambia a medida que avanza el proceso de optimización ya que en el recocido de metales a medida que decrece la temperatura es más difícil que un átomo se mueva a un estado de mínima energía, similarmente en el RS la probabilidad de aceptar soluciones de menor calidad desciende a medida que avanza el algoritmo.

En la figura siguiente se muestra la utilidad de aceptar soluciones peores, con el objeto de escapar de óptimos locales.

Figura 30. Particularidades del RS para escapar de óptimos locales.



Elaborada a partir de Vélez & Montoya, 2007.

Obsérvese que si no existiera la posibilidad de aceptar un estado peor al inicial, no existiría la forma escapar del óptimo local inicial. Este proceso es similar a mover una pelota en una tubería, cuando se aplica poca fuerza a la pelota esta quedara atrapada en alguno de los valles de la tubería sin que necesariamente este sea el valle más profundo, a medida que se aplica mayor fuerza a la pelota esta tendrá mayor posibilidad de alcanzar el óptimo global.

A continuación se muestra el pseudocódigo del RS.

Algoritmo 20. Recocido simulado.

Input: Solución Inicial, temperatura inicial del sistema (t), función objetivo del problema, criterio de paro $(número\ de\ iteraciones)$, tasa de enfriamiento $\big(a \in (0,1)\big)$.

Output: Solución aproximada del problema determinada por el RS.

1. S_1 ←solución inicial
2. f_0 ←valor de la función objetivo dado por S_1
3. t ←temperatura inicial
4. a ←tasa de enfriamiento

5. $número\ de\ iteraciones$ ← criterio de paro
6. $k \leftarrow 1$
7. For $i = 1{:}\,1{:}\,número\ de\ iteraciones$
    1. $t = t * a$
    2. Generar de forma aleatoria una nueva solución al problema en la vecindad de S_1.
    3. $S_2 \leftarrow$ nueva solución
    4. $f_1 \leftarrow$ valor de la función objetivo para la solución S_2
    5. If f_1 es mejor a f_0
        1. $S_1 \leftarrow S_2$
        2. $f_0 \leftarrow f_1$
    6. Else
        1. $dE = f_0 - f_1$
        2. If $\left(rand \leq e^{-\frac{dE}{k*t}}\right)$
            1. $S_1 \leftarrow S_2$
            2. $f_0 \leftarrow f_1$

            End if

        End If

End For

Fuente: Vélez & Montoya, 2007

Algunos estudios han mostrado que si t disminuye lo suficientemente lento, el proceso de solución por RS convergería a la solución óptima. Sin embargo, una función de reducción de temperatura que garantice la convergencia al óptimo global, requiere un tiempo de cálculo prohibitivo. Generalmente la determinación del parámetro t se realiza por medio de estudios empíricos.

En la literatura se ha reportado al RS como un método eficiente y eficaz de problemas combinatorios. Dado a la gran cantidad de literatura disponible sobre esta metaheurística no se incluye en este trabajo un ejemplo de aplicación.

2.6 Resumen.

En la sección 2.1 se examino que dada la naturaleza del AMS (NP-completo), se justifica el uso de técnicas heurísticas o metaheurísticas para resolverlo.

En la sección 2.2 se mostraron las clasificaciones diseñadas por Wang, Omar y Chan sobre los procedimientos de solución del AMS. Dichas sistematizaciones utilizan la estrategia y tipo de búsqueda como criterio de categorización. Cabe mencionar que la clasificación de Chan es la más utilizada dentro de la literatura.

Dentro de la misma sección se analizaron a detalle algunos de los métodos para resolver el AMS. Los procedimientos de Needleman – Wunsch, el de Smith - Waterman y el de Carrillo – Lipman son las formas clásicas para resolver el AMS,

sin embargo, su implementación queda limitada a comparar un número pequeño de cadenas. En la tabla siguiente se exhibe tanto la complejidad computacional como el número máximos de cadenas que pueden compararse de manera simultánea.

Tabla 6. Comparativo de algunos de los métodos de solución del AMS.

Nombre del algoritmo	Complejidad	Número de secuencias que alinea simultáneamente
Matriz de puntos	$O(l_a * l_b)$	2
Distancia de Levenshtein	$O(l_a * l_b)$	2
Distancia Indel	$O(l_a * l_b)$	2
Distancia de Damerau	$O(l_a * l_b)$	2
Needleman y Wunsch	$O(l_a * l_b)$	2
Murata	$O(l_a * l_b * l_c)$	3
Gotoh	$O(l_a * l_b)$	2
Algoritmo de Gotoh – Altschul	$O(l_a * l_b)$	2
Carrillo Lipman	$O(l^n)$	Hasta 10
Smith y Waterman	$O(l_a * l_b)$	2
Fredman	$O(l^3)$	3
Sankoff	$O(n_i(2l^n))$	5
Sankoff y Cedergren	$O(n_i(2l^n))$	3
Fitch	$O(n_i(2l^n))$	15
Jonson and Doolitle	$O(n(l - l_r)l_r^{n-1})$	3 o mas
Karlin	$O(\lvert\mathcal{L}\rvert^2)$	2 o más
Waterman y Jones	$O(n * {l_r}^2 * l * B)$	2 o más
Waterman y Perlwitz	$O(nl^2)$	
Barton y Sternberg	$O(n!)$	
Subbiah y Harrison	$O(n!)$	
Clustal	$O(n^2l^2)$	
T-Cofee	$O(n^2l^2) + O(n^3)$	
Algoritmo A*	$O(n^l)$	
Algoritmo de Viterbi	$O(n^3P)$	

donde:
n_i: Número de nodos internos en el árbol filogenético.
l: Longitud de la secuenica más larga.
n: Número de secuencias
l_r: Longitud del residuo.
$\mathcal{L}$: Tamalo de la lista de regiones.
B: Valor de función deacuerdo a la naturaleza de las cadenas-

De la gran cantidad de herramientas disponibles para la resolución del AMS, son los de aproximación progresiva los de uso más recurrente, sin embargo, estos métodos implican generar a priori algunas relaciones entre las secuencias.

En la sección 2.3 se presentaron varios de los programas de cómputo desarrollados para solucionar el AMA. También se examino el procedimiento desarrollado por Julie Thompson para la validación de algoritmos y programas de cómputo destinados a resolver el AMS, el cual se fundamenta en el uso de la base de datos referencial BALiBASE y la cuantificación de las métricas de eficiencia: a) Puntuación de la suma de pares y B) Puntuación entre columnas. La versión uno de BALiBASE es la única edición en la que se encuentran los resultados del comportamiento de algunos métodos para resolver el AMS.

En las secciones 2.4 y 2.5 se analizaron las metaheurísticas de búsqueda de armonía y recocido simulado, las cuales servirán como métodos padres para la generación del nuevo procedimiento para solucionar el AMS. La elección de estos métodos como padres de la técnica desarrollada se basa en; el alto grado de sencillez, generalidad, adaptabilidad, eficiencia y eficacia. En la tabla siguiente se da un breve resumen de las características de RS y HS

Tabla 7. Resumen de las metaheurísticas de RS y HS.

Metaheurística	Característica.
Recocido Simulado	• Se fundamenta en el trabajo de Metrópolis para el campo de la termodinámica estadística. En dicho trabajo se modeló el proceso de cambios energéticos en el recocido de los metales. • Es un procedimiento de búsqueda global, cuya táctica de búsqueda es del tipo probabilística. • Permite escapar de óptimos locales, al aprobar movimientos que no sean de mejora.
Búsqueda de armonía	• Se fundamenta en los trabajos de Zong Woo. La HS imita el proceso de innovación musical. • Es un procedimiento de búsqueda global, cuya táctica de búsqueda es del tipo aleatoria estocástica. • Permite la búsqueda en vecindades. • Es un procedimiento con memoria a corto plazo. • Ofrece una alta flexibilidad para su implementación en problemas de optimización. • Ocupa pocos parámetros para su ejecución. • Realiza una búsqueda aleatoria estocástica.

Elaborado a partir de Geem & Lee, 2004, Coelho & Mariani, 2009, Moreno Pérez & Melián Batista, 2005

Las ideas y conceptos analizados en esta sección serán ocupados en el capítulo siguiente como base para el desarrollo de la nueva estrategia AMS.

Capítulo 3 Estrategia propuesta para solucionar el AMS.

En virtud de que el objetivo general de este trabajo es: “Desarrollar un método original para resolver el AMS, basado en las metaheurísticas de búsqueda de armonía y de recocido simulado”, el presente capítulo constituye el punto medular de la tesis dado que en éste se adaptan, generan los conceptos e ideas para el manejo y solución del AMS, además de realizar la validación del nuevo método con objeto de determinar su eficiencia y eficacia. Por lo tanto, la función del actual capítulo es: “Generar y validar un método original para solucionar el AMS basado en las metaheurísticas de HS y RS”. Al mismo tiempo, existe una relación directa entre la función del presente capítulo con los objetivos particulares tercero y cuarto de la tesis y para alcanzar la función anterior se utiliza la siguiente táctica:

1. Desarrollar procedimiento híbrido del tipo iterativo para resolver el AMS a través de adaptar las ideas y conceptos de HS y RS.

2. Validar la estrategia propuesta por la metodología de Thompson.

3. Comparar la estrategia desarrollada con otros procedimientos del AMS.

4. Analizar las características del procedimiento desarrollado, identificando sus ventajas y desventajas.

En el capítulo anterior se expuso que dada la naturaleza NP-completo de problema AMS se utilizan técnicas heurísticas y metaheurísticas, con el objeto de encontrar una buena solución en un tiempo razonable. Así mismo se analizaron las metaheurísticas HS y RS, las cuales servirán como base para la generación del nuevo método.

El generar un nuevo procedimiento para resolver el AMS (pese a la existencia de métodos con dicho propósito) se fundamenta en utilizar el vigor híbrido (resultado de combinar el RS y la HS) para generar una estrategia viable para resolver el AMS.

En términos generales, el vigor híbrido es el nivel de mejora de aptitudes de un conjunto de procedimientos cuando se implementan unidos, en contraste a la implementación individual, para resolver un problema de optimización. Es decir, el vigor híbrido supone que al conjuntar dos o más métodos de solución (a los que se denomina padres) se potencializan las fortalezas de cada uno y a su vez se disminuyen sus debilidades.

La elección de las metaheurísticas búsqueda de armonía y recocido simulado como padres del método desarrollado se basa en las siguientes características:

1. Ambos procedimientos son metaheurísticas sencillas y coherentes, dado que, se basan en principios simples y claros, que permiten su fácil comprensión e implementación.
2. Son procedimientos generales y adaptables ya que en la literatura se han reportado un gran número de aplicaciones en diferentes problemas para contextos muy heterogéneos.
3. Son procedimientos efectivos y eficientes, con base a que en las aplicaciones reportadas en la literatura se han obtenido buenos resultados.
4. El recocido simulado ha sido utilizada para solucionar el AMS con excelentes resultados.
5. La combinación de métodos heurísticos permiten tomar mejores decisiones y proporcionan una mayor flexibilidad a la hora de definir diferentes estrategias de elección.

La implementación de este algoritmo híbrido en solución al problema AMS, implicará tomar decisiones sobre los conceptos rectores de procedimiento, al adaptar y generar ideas para la manipulación y solución del AMS, donde se aplicaron los conocimientos adquiridos. La estrategia HS fue utilizada como procedimiento rector, mientras que el RS permitirá que la estrategia desarrollada escape de óptimos locales y a su vez se mantenga un alto grado de diversificación en la memoria armónica .A continuación se describe el procedimiento desarrollado.

3.1 Desarrollo de la nueva estrategia.

Adicionalmente a las características consideradas para la elección de los procedimientos padres para el desarrollo de la nueva estrategia, se tomó en cuenta para la búsqueda de armonía que es una técnica de reciente creación, por lo que no existe reportado una aplicación para resolver el AMS en la literatura, por ende la adaptación de este procedimiento para solucionar el AMS implica un aporte al conocimiento.

3.1.1 Descripción del proceso.

La estructura del problema que se utiliza para el diseño de la estrategia incluye la función objetivo para COFFEE **Fuente especificada no válida.**. Lo anterior se debe a que dicha función ha sido utilizada en el diseño de otras técnicas heurísticas destinadas a resolver el AMS, como son: T-COFFEE (Notredame, 2000), Algoritmo iterativo y adaptativo para el mejoramiento de AMS **Fuente especificada no válida.** y Algoritmo genético con colonia de hormigas para el alineamiento múltiple de secuencia (Zne-Jung, Shun-Feng, Chen-Chia, & Kuan-Hung, 2008).

3.1.1.1 Definición de los parámetros de entrada de entrada.

En esta etapa del proceso deben alimentarse al algoritmo los parámetros requeridos para su funcionamiento. La selección apropiada del valor de los parámetros influye en los resultados obtenidos por la HS

1. **Un conjunto de N secuencias**, tal que $N = \{s_1, s_2, \dots, s_x\}\ y\ x \geq 3$.
   Por definición del problema AMS se requiere que el número mínimo de secuencias a alienar (mínimo tres secuencias).

2. **Tamaño de la memoria de armonía** $HMS \geq 1$ (valor discreto).
   El tamaño de la HMS influye de forma directa en la calidad de la solución final, sin embargo a medida que crese el valor de la HMS se incrementan el número de operaciones asociadas a la HM. Tamaños pequeños de memoria armónica intensifican la búsqueda mientras que tamaños grandes diversifican la búsqueda. Al analizar la información del tamaño de la memoria armónica asignados a los problemas resueltos con HS se observa que el valor mínimo utilizado es de 5 (problema de proyección de retorno). Se establece este valor dado a la cantidad de recurso memoria necesario para representar una fila de la memoria armónica.

3. **Criterio de paro** $MaxImp > 1$.
   El criterio de paro determina el número de iteraciones necesarias para alcanzar el resultado final. En la literatura se reporta que para procedimientos heurísticos desarrollados para solucionar el AMS tales como T-COFFEE y el algoritmo adaptativo e iterativo para el refinamiento de alineamientos múltiples de secuencia requieren a lo menos 5000 iteraciones. Sin embargo en virtud de se ocupa el valor mínimo de tamaño de memoria armónica y con base en experimentos previos se establece el valor de $MaxImp = 8000$.

4. **Consideraciones del ritmo en la memoria de armonía** $(HMCR)$
   Para determinar el valor de HMCR y PAR se les considera como un elemento variante en función al número de iteraciones, esto se debe a que la aplicación de HS para resolver el AMS es un nuevo método y la mayoría de los autores recomiendan un rango para parámetros cuando se analiza un nuevo problema.

$$HMCR = HMCRI + \left((HMCRF - HMCRI) * \left(\frac{v}{MaxImp} \right) \right) \qquad (62).$$

Adaptación de la fórmula de Ingram & Zhang, 2009
para el parámetro PAR.

donde:
$HMCR$: Adaptación del ritmo en la memoria de armonía.
v: Número de iteración.
$MaxImp$: Máximo número de iteraciones

$HMCRI$: Consideración inicial del ritmo en la memoria de armonía $0 \leq HMCRI \leq HMCRF$.
$HMCRF$: Consideración final del ritmo en la memoria de armonía $HMCRI \leq HMCRF \leq 1$.

Ingram y Zhang establecen que el valor de HMCR para analizar nuevos problemas sea $HMCR = (0.7 - 0.95)$. (Ingram & Zhang, 2009). Mientras Omran y Mahdavi sugirieren que el valor de $HMCR = 0.9$ **Fuente especificada no válida.**. Al analizar la información sobre el valor asignado en las aplicaciones del HS se obtiene el rango $HMCR = (0.7 - 0.97)$ con una media en 0.9.

En virtud de valores altos de $HMCR$ intensificarían la búsqueda en una región del espacio de soluciones y considerando que se estableció un memoria armónica con ese fin se opta por utilizar como límite superior el valor de Omran (que también corresponde a la media) y como límite inferior el valor mínimo establecido por Ingram. Por ende se fijan los valores $HMCRI = 0.7$ y $HMCRF = 0.9$.

5. **Parámetros de ajuste del ritmo** (PAR)

$$PAR = PARI + \left((PARF - PARI) * \left(\frac{v}{MaxImp} \right) \right) \qquad (63).$$

Ingram & Zhang, 2009.

donde:
PAR: Parámetro de ajuste del ritmo.
v: Número de iteración.
$MaxImp$: Máximo número de iteraciones
$PARI$: Parámetro de ajuste del ritmo inicial $0 \leq PARI \leq PARF$.
$PARF$: Parámetro de ajuste del ritmo final $PARI \leq PARF \leq 1$.

Ingram y Zhang establecen que el valor de PAR para analizar nuevos problemas sea $PAR = (0.2 - 0.5)$. (Ingram & Zhang, 2009). Mientras Omran y Mahdavi sugirieren que el valor de $PAR = 0.3$ **Fuente especificada no válida.**. Al analizar la información sobre el valor asignado en las aplicaciones del HS se obtiene el rango $PAR = (0 - 0.7)$ con una media en 0.3.

En el valor que se asigne al PAR está influido por: el valor establecido en la búsqueda en vecindades (valor mínimo valor del PAR) y el valor máximo establecido por Ingram . por lo tanto, el PAR el valor utilizado se fija $PARI = 0$ y $PARF = 0.5$.

6. **Temperatura inicial** $0 \leq T_o$. Y grado **de disminución de temperatura** $0 \leq \alpha \leq 1$
Williams, Gilbert y Westhead establece que el valor de temperatura inicial $\boldsymbol{t_0 = 10}$, y el grado de disminución de temperatura $\boldsymbol{\alpha = 0.7}$
.

7. **Ancho de banda para el ajuste del ritmo** $b \geq 0$ (valor discreto).
   De acuerdo con Cheng el ancho de banda puede calcularse mediante la fórmula siguiente:

$$b = \eta(l_{M_i}^{u} - l_{M_i}^{l}) \qquad (64).$$

Cheng, y otros 2008

donde:
η: Radio máximo de búsqueda de las vecindades. En la literatura se recomienda 0.1 sin embargo durante pruebas preliminares se obtuvo un mejor desempeño con 0.3
b: Ancho de banda.
$l_{M_i}^{u}$: Límite superior del número de columnas en la matriz de alineamiento
$l_{M_i}^{l}$: Límite inferior del número de columnas en la matriz de alineamiento

Para determinar el valor de l_M se realiza la siguiente inferencia tomando como base la definición de AMS.

Para cualquier secuencia $s_a \in N$ se cumple la siguiente relación

$$\forall s_a = 1,2,3 \dots s_x \;\; se\ satisface \;\; l_{s_a} \epsilon (l_{s_{min}}, l_{s_{max}}) \qquad (65).$$

donde:
l_{s_a}: Longitud de la cadena s_a
$l_{s_{min}}$: Longitud de la cadena más corta del conjunto N

$$l_{s_{max}} = min\{l_{s_1}, l_{s_2}, l_{s_3}, \dots l_{s_x}\} \qquad (66).$$

$l_{s_{max}}$ Longitud de la cadena más larga del conjunto N

$$l_{s_{max}} = max\{l_{s_1}, l_{s_2}, l_{s_3}, \dots l_{s_x}\} \qquad (67).$$

Con base en la relación anterior y la definición de AMS se infiere que el número mínimo de guiones insertados en la secuencia s_a durante el alineamiento se representa mediante la siguiente relación:

$$ns_a \geq l_{s_{max}} - l_{s_a} \qquad \forall s_a = 1,2,3 \dots s_x \qquad (68).$$

donde:
ns_a: Número de espacios vacios insertados a la secuencia s_a.
l_{s_a}: Longitud de la cadena s_a del conjunto N.
$l_{s_{max}}$ Longitud de la cadena más larga del conjunto N.

En la definición de AMS establece que no puede existir una columna de la matriz M formada únicamente por espacios vacios $\{-\}$, por lo tanto, el número de guiones insertados no puede ser infinito, quedando acotado en la siguiente relación:

$$ns_a \leq caracteres - l_{s_a} \qquad \forall s_a = 1,2,3 \dots s_x \qquad (69).$$

donde:
ns_a: Número de espacios vacios insertados a la secuencia s_a.
l_{s_a}: Longitud de la cadena s_a del conjunto N.
$caracteres$: Suma se la longitud de todas las secuencias del conjunto.

$$caracteres = \sum_{\substack{a=1 \\ s_a \in N}}^{x} l_{s_a} \qquad \forall s_a = 1,2,3 \dots s_x \qquad (70).$$

A partir de las relaciones anteriores se concluye que el número de guiones insertados en cualquier secuencia s_a queda comprendido dentro del rango siguiente:

$$ns_a \in \left(l_{s_{max}} - l_{s_a}, caracteres - l_{s_a}\right) \qquad (71).$$

La longitud de la secuencia $\widehat{s_a}$, se determina por la siguiente ecuación.

$$l_{\widehat{s_a}} = l_{s_a} + ns_a \qquad (72).$$

Relacionando las ecuaciones anteriores se establece que $l_{\widehat{s_a}}$ queda comprendida dentro del siguiente rango:

$$l_{\widehat{s_a}} \in (l_{max}, caracteres) \qquad \forall s_a \in N \qquad (73).$$

$$l_{max} = l_{s_a} + \left(l_{s_{max}} - l_{s_a}\right) = li \qquad (74).$$

$$caracteres = l_{s_a} + \left(caracteres - l_{s_a}\right) = ls \qquad (75).$$

donde:
$l_{\widehat{s_a}}$: Longitud de la secuencia $\widehat{s_a}$.
li: Mínima longitud posible de la secuencia $\widehat{s_a}$.
ls: Máxima longitud posible de la secuencia $\widehat{s_a}$.

La longitud de las secuencias después de insertarse los guiones (por definición del AMS debe ser la misma para todas las secuencias del conjunto), corresponde al número de columnas en la matriz M, por tanto se pueden establecer las siguientes relaciones:

$$l_M = l_{\widehat{s_a}} \qquad (76).$$

$$l_M \in (l_{max}, caracteres) \qquad (77).$$

donde:
l_M : Número de columnas de la submatriz M^P.
$caracteres$: Suma se la longitud de todas las secuencias del conjunto.
$l_{s_{max}}$: Longitud de la cadena más larga del conjunto N

Por lo tanto el ancho de banda se establece mediante la siguiente ecuación:

$$b = 0.3(caracteres - l_{max}) \qquad (78).$$

8. Matriz de ponderación de los alineamientos (W).

En la función COFFEE se utiliza un factor de ponderación de los alineamientos entre los pares posibles alineados de las secuencias del conjunto. Para ello se construye una matriz W cuyos elementos en cada celda representan el factor de ponderación cuyo valor oscila entre 1 y 0. Si el valor en la celda $w_{s_a s_b}$ tiende a cero significa poca similitud entre la secuencias y entre más cercano es a uno significa una mayor similitud entre secuencias.A continuación se muestra el algoritmo utilizado para la construcción de la matriz W.

Algoritmo 21. Construcción de la matriz de ponderación W.

Input: Un conjunto de secuencias $(N = \{s_1, s_2, \dots, s_x\})$.

Output: Matriz de ponderación.

1. For $a = 1{:}\,1{:}\,x$
    a. For $b = 1{:}\,1{:}\,x$
        i. If $(a < b)$
            1. $contador = 0$
            2. M_{s_a,s_b} ← Matriz de alineamiento de las secuencias s_a, s_b generado por el algoritmo de Needleman – Wunsch variante de Vinuesa $O(l_a * l_b)$.
            3. len ← Número de columnas de la matriz M_{s_a,s_b}.

4. For $j = 1{:}\,1{:}\,len$
   a. If $\left(M_{1,j} = M_{2,j}\right)$
      i. $contador = contador + 1$
   b. Else
      i. $contador = contador + 0$
      ii. End If
   End for
5. $W_{i,ii} \leftarrow \frac{contador}{len}$

ii. Else if $(a > b)$
   1. $W_{i,ii} \leftarrow W_{ii,i}$

iii. Else
   1. $W_{i,ii} \leftarrow 0$

End if

End for

End For

La complejidad del algoritmo anterior es del orden $O\left(n^2 * \left((l_a * l_b)\right) + \text{len}\right) \approx O(n^2 * l_a * l_b)$. Con objeto de ilustrar el funcionamiento del algoritmo anterior considere el siguiente ejemplo:

Ejemplo 22. Dado el siguiente conjunto de secuencias biológicas $N = \{s_1 = \{\text{g, g, a, t, t, c, c, c, c, t}\},\ s_2 = \{\text{a, c, c, c, g, c, g}\},\ s_3 = \{\text{g, g, g, g, t, c, c, c, c, t}\},\ s_4 = \{\text{g, g}\},\ s_5 = \{\text{g, g, a, t, c, t, t, t}\}\}$ se desea construir la matriz de ponderación W.

Construcción de la matriz de alineamiento de las secuencias s_1 y s_2.

$$M_{s_1,s_2} = \begin{bmatrix} g & g & a & t & t & c & c & c & c & t \\ - & - & a & - & c & c & c & g & c & g \end{bmatrix}$$

El número de columnas de la matriz de alineamiento es 10 por lo tanto el número de iteraciones para comparar los elementos en la matriz son 10.

Número de Iteración	1	2	3	4	5	6	7	8	9	10
Elemento $\left(M_{1,i}\right)$	g	g	a	t	t	c	c	c	c	t
Elemento $\left(M_{2,i}\right)$	-	-	a	-	c	c	c	g	c	g
$M_{1,i} = M_{2,i}$	no	no	si	no	no	si	si	no	si	no
Valor de contador en cada iteración	0	0	1	1	1	2	3	3	4	4

En virtud de el valor del contador es igual a cuatro unidades al terminar las iteraciones y que $len = 10$ se obtiene que el valor de $W_{1,2} = 0.4\ y\ W_{2,1} = 0.4$.

Repitiendo iterativamente el algoritmo anterior se obtiene la siguiente matriz de ponderación de los pares del conjunto.

$$W = \begin{bmatrix} 0 & 0.4 & 0.7273 & 0.10 & 0.4 \\ 0.4 & 0 & 0.1 & 0.2857 & 0.1429 \\ 0.7273 & 0.1 & 0 & 0.1 & 0.2 \\ 0.10 & 0.2857 & 0.10 & 0 & 0.25 \\ 0.4 & 0.1429 & 0.2 & 0.25 & 0 \end{bmatrix}$$

En la tabla siguiente se resume la información sobre el valor (o bien su forma de estimación) de los parámetros de entrada para el nuevo procedimiento desarrollado para resolver el AMS.

Tabla 8. Valor de los datos de entrada del algoritmo.

Parámetro		Valor
Un conjunto de secuencias que $N = \{s_1, s_2, \dots, s_x\}$		Datos de entrada
Tamaño de la memoria de armonía (HMS)		5
Criterio de paro $(MaxImp)$		8000
Consideración del ritmo en la memoria de armonía $HMCR$	$HMCRI$	0.7
	$HMCRF$	0.9
Parámetro de ajuste del ritmo (PAR)	$PARI$	0
	$PARF$	0.5
Temperatura inicial		10
Grado de disminución de temperatura (α)		0.7
Ancho de banda para el ajuste del ritmo. b		$0.3(caracteres - l_{max})$
Matriz de ponderación W		Generar con base enl algoritmo 16

3.1.1.2 Creación de la memoria armónica.

A *Matriz Auxiliar P.*

Para crear la memoria armónica es necesario desarrollar una matriz auxiliar P. Dicha matriz auxiliar, se forma por dos submatrices, las cuales son: submatriz M^P que almacenara el alineamiento múltiple propuesto para el conjunto $N = \{s_1, s_2, \dots, s_x\}$ secuencias, mientras que las segunda submatriz G^P contendrá dos vectores columna, donde , el i –ésimo elemento del primer vector representan el valor de la función COFEE de la i –ésima secuencia con el resto de las secuencias del conjunto, mientras que el i –ésimo elemento del segundo vector representa el valor de función de aptitud de la i –ésima secuencia. Lo anterior se representa en la siguiente relación.

$$P = \left(\begin{array}{cccc|cc} M^P_{\widehat{s_1},1} & M^P_{\widehat{s_1},2} & \dots & M^P_{\widehat{s_1},l_{\widehat{s_1}}} & COFFEE\ _{\widehat{s_1}} & fitness_{\widehat{s_1}} \\ M^P_{\widehat{s_2},1} & M^P_{\widehat{s_2},2} & \dots & M^P_{\widehat{s_2},l_{\widehat{s_1}}} & COFFEE\ _{\widehat{s_2}} & fitness_{\widehat{s_2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ M^P_{\widehat{s_x},1} & M^P_{\widehat{s_x},1} & \dots & M^P_{\widehat{s_x},l_{\widehat{s_x}}} & COFFEE_{\widehat{s_x}} & fitness_{\widehat{s_x}} \end{array}\right) \qquad (79).$$

donde:
P: Matriz auxiliar P
$M^P_{\widehat{s_a},j}$: j-ésimo elemento de la secuencia $\widehat{s_a}$.
$COFFEE\ _{\widehat{s_a}}$: Valor de la función COFFEE la secuencia $\widehat{s_a}$.
$fitness_{\widehat{s_1}}$: Valor de la función de aptitud de la secuencia $\widehat{s_a}$.

Las dimensiones de la matriz auxiliar P se relacionan el número de secuencias del conjunto que corresponden al número de filas de la matriz, y con el número de columnas de la matriz de alineamiento mas dos, corresponde al número de columnas de la matriz P.

$$Columnas\ de\ la\ matriz\ P = l_M + 2 \qquad (80).$$

l_M : Número de columnas en la matriz M de alineamiento.

En virtud del número de columnas posibles de la matriz de alineamiento queda comprendido en el rango $(l_{max}, caracteres)$ y considerando una distribución uniforme del valor de dicha variable (para evitar algún tipo de sesgo), se recurre al procedimiento de la transformada inversa para encontrar el número de columnas de la matriz de alineamiento:

$$l_M = \text{int}\left(l_{max} + \left((caracteres - l_{max}) * rand\right)\right) \qquad (81).$$

A.1 Construcción de la submatriz M^P.

La asignación de los elementos que integran a la submatriz M^P se realiza mediante el siguiente razonamiento: La probabilidad de asignar para la a –ésima hilera en la j –ésima columna un caracter (en caso de éxito) o un espacio vacío (en caso de fracaso) depende de las elecciones hechas con anterioridad en la a –ésima hilera, y se debe garantizar ingresar a la a-ésima hilera de M todos los caracteres de la a –ésima secuencia relacionada. Durante el proceso de asignación de elementos en la a –ésima fila el espacio muestral varía por los siguientes motivos:

1. El número de columnas disponibles cambia durante cada una de las asignaciones realizadas.
2. Puede haberse modificado el número de éxitos posibles, puesto que al asignar el j –ésimo caracter en alguna de las columnas, reduce el número posible de éxitos.

La probabilidad de asignar un caracter esta dado por la siguiente ecuación:

$$P\left(M_{i,j} \leftarrow caracter\right) = \frac{E_{s_i}}{\mathcal{S}_{s_i}} \quad (82).$$

$$E_{s_i} = l_{s_i} - E_u \quad (83).$$

$$\mathcal{S}_{s_i} = l_M - \mathcal{C}_A \quad (84).$$

donde:
$P\left(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}\right)$: Probabilidad de asignar un caracter en la j –ésima posición de la i –ésima fila
E_{s_j}: Número de eventos exitosos posibles.
$\mathcal{S}_{\mathrm{s_j}}$: Número de Columnas disponibles:
$\mathcal{C}_A$: Número de columnas ya asignadas.
l_{s_i}: Longitud de la secuencia i –ésima.
E_u: Número de caracteres ya asignados.
l_M: Número de columnas de la submatriz M^P.

Para la construcción de la submatriz M^P, se utiliza la siguiente rutina.

Algoritmo 22. Construcción de la submatriz M^P.

Input: Un conjunto de secuencias $(N = \{s_1, s_2, \dots, s_x\})$, número de columnas que integrarán a la a la submatriz M^P (l_M).

Output: submatriz M^P

1. $x \leftarrow$ Número de secuencias que integran al conjunto N
2. For $i = 1; i \leq x; ++$
    a. $l \leftarrow$ longitud de la secuencia s_i
    b. $u = 1$
    c. $\mathcal{S} \leftarrow l_M$
    d. For $j = 1; j \leq l_M; ++$
        i. $P\left(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}\right) = \frac{l}{\mathcal{S}}$
        ii. If $\left(rand \leq P\left(M_{\mathrm{i,j}}{}^{\mathrm{P}} \leftarrow \mathrm{caracter}\right)\right)$
            1. $M_{\mathrm{i,j}}{}^{\mathrm{P}} \leftarrow s_{i_u}$
            2. $\mathrm{u} = \mathrm{u} + 1$
            3. $l = l - 1$
        iii. Else
            1. $M_{\mathrm{i,j}}{}^{\mathrm{P}} \leftarrow '-'$
        End if
        iv. $\mathcal{S} = \mathcal{S} - 1$

End For

End For

//Eliminación de columnas que consten exclusivamente de guiones//

3. $u = 0$
4. For $j = 1; j \leq l_M; ++$
   a. If $M_{:,\mathrm{j}} = '-'$
      i. Eliminar la columna $M_{:,\mathrm{j}}$ de la submatriz M^P
      ii. $l_M = l_M + 1;$

   End if

End For

/*Actualizar el número de columnas de la submatriz M^P*/

5. $l_M = l_M - u,$

La complejidad del algoritmo anterior es $O(x * l_M)$, con objeto de ilustrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 23. Dadas las secuencias biológicas siguientes $N = \{s_1 = \{\mathrm{g,g,a,t,t,c,c,c,c,t}\},\ s_2 = \{\mathrm{a,c,c,c,g,c,g}\},\ s_3 = \{\mathrm{g,g,g,g,t,c,c,c,c,t}\},\ s_4 = \{\mathrm{g,g}\},\ s_5 = \{\mathrm{g,g,a,t,c,t,t,t}\}\}$ y $l_M = 15$, construir una submatriz M^P.

Inicia ciclo for $i = 1$

$j = 1$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{2}{3}$

$rand = .687$ y es mayor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,1} \leftarrow '-'$

Se actualiza $l \leftarrow 10, \mathcal{S} \leftarrow 14, u = 1$

$j = 2$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{5}{7}$

$rand = 0.81$ y es mayor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,2} \leftarrow '-'$

Se actualiza $l \leftarrow 10, \mathcal{S} \leftarrow 13, u = 1$

$j = 3$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{10}{13}$

$rand = .644$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,3} \leftarrow 'g'$

Se actualiza $l \leftarrow 9, \mathcal{S} \leftarrow 12, u = 2$

$j = 4$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{3}{4}$

$rand = 0.818$ y es mayor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,4} \leftarrow '-'$

Se actualiza $l \leftarrow 9, \mathcal{S} \leftarrow 11, u = 2$

$j = 5$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{9}{11}$

$rand = 0.83$ y es mayor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,5} \leftarrow '-'$

Se actualiza $l \leftarrow 9, \mathcal{S} \leftarrow 10, u = 2$

$j = 6$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{9}{10}$

$rand = 0.094$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M_{1,6} \leftarrow 'g'$

Se actualiza $l \leftarrow 8, \mathcal{S} \leftarrow 9, u = 3$

$j = 7$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{8}{9}$

$rand = 0.183$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,7} \leftarrow 'a'$

Se actualiza $l \leftarrow 7, \mathcal{S} \leftarrow 8, u = 4$

$j = 8$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{7}{8}$

$rand = 0.217$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,8} \leftarrow 't'$

Se actualiza $l \leftarrow 6, \mathcal{S} \leftarrow 7, u = 5$

$j = 9$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{6}{7}$

$rand = 0.005$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,9} \leftarrow 't'$

Se actualiza $l \leftarrow 5, \mathcal{S} \leftarrow 6, u = 6$

$j = 10$

$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = \frac{5}{6}$

$rand = 0.129$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$ $M^P{}_{1,10} \leftarrow 'c'$

Se actualiza $l \leftarrow 4, \mathcal{S} \leftarrow 5, u = 7$

$\boldsymbol{j = 11}$
$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = 4/5$
$rand = 0.989$ y es mayor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$
$M^P{}_{1,11} \leftarrow '-'$
Se actualiza $l \leftarrow 4, \mathcal{S} \leftarrow 4, u = 7$

$\boldsymbol{j = 12}$
$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = 1$
$rand = 0.747$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$
$M^P{}_{1,12} \leftarrow 'c'$
Se actualiza $l \leftarrow 3, \mathcal{S} \leftarrow 3, u = 8$

$\boldsymbol{j = 13}$
$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = 1$
$rand = 0.191$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$
$M^P{}_{1,13} \leftarrow 'c'$
Se actualiza $l \leftarrow 2, \mathcal{S} \leftarrow 2, u = 9$

$\boldsymbol{j = 14}$
$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = 1$
$rand = 0.400$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$
$M^P{}_{1,14} \leftarrow '-c'$
Se actualiza $l \leftarrow 1, \mathcal{S} \leftarrow 1, u = 10$

$\boldsymbol{j = 15}$
$P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter}) = 1$
$rand = 0.211$ y es menor a $P(M_{\mathrm{i,j}} \leftarrow \mathrm{caracter})$ $\therefore$
$M^P{}_{1,11} \leftarrow 't'$
Se actualiza $l \leftarrow 0, \mathcal{S} \leftarrow 0, u = 11$

Termina ciclo For

Por lo que se obtiene la fila:

$M^P_{1,j}$	-	-	g	-	-	g	a	t	t	c	-	c	c	c	t

Implementando el procedimiento en el resto de las secuencias, se obtiene la siguiente submatriz M^P:

$M^P_{1,j}$	-	-	g	-	-	g	a	t	t	c	-	c	c	c	t
$M^P_{2,j}$	-	-	a	c	c	g	-	-	c	g	-	-	-	-	-
$M^P_{2,j}$	-	g	g	-	g	g	t	c	-	c	-	c	c	t	-
$M^P_{2,j}$	-	-	-	-	g	-	-	-	-	-	-	g	-	-	-
$M^P_{2,j}$	-	g	g	-	a	t	-	-	c	t	-	-	t	-	t

Ya que la submatriz anterior contiene dos columnas formadas sólo por espacios vacíos (columnas uno y once), al aplicar el paso 3 del algoritmo dichas columnas son eliminadas resultando la siguiente submatriz M^P

$$M^P = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t \\ - & a & c & c & g & - & - & c & g & - & - & - & - \\ g & g & - & g & g & t & c & - & c & c & c & t & - \\ - & - & - & g & - & - & - & - & - & g & - & - & - \\ g & g & - & a & t & - & - & c & t & - & t & - & t \end{bmatrix}$$

Se actualiza el número de columnas de la submatriz M^P de acuerdo al último paso del algoritmo, dado a que se eliminan 2 columnas de la matriz origina se tiene que $l_M = 15 - 2 = 13$.

A.2 Construcción de la Submatriz G^P

Para construir la submatriz G^P se requieren generar una matriz de distancia entre los pares secuencias del conjunto a la cual se denominada D.

La matriz auxiliar D, contendra una cuantificación de la similitud entre los pares de secuencias posibles, es decir, la celda D_{ab} representa el grado de cercanía entre la secuencia $\widehat{s_a}\ y\ \widehat{s_b}$. Dicha matriz es simétrica, con elementos en la diagonal igual a cero. Lo anterior se representa en la relación siguiente.

$$D = \begin{pmatrix} 0 & f(\widehat{s_1},\widehat{s_2}) & f(\widehat{s_1},\widehat{s_3}) & \cdots & f(\widehat{s_1},\widehat{s_x}) \\ f(\widehat{s_1},\widehat{s_2}) & 0 & f(\widehat{s_2},\widehat{s_3}) & & \vdots \\ f(\widehat{s_1},\widehat{s_3}) & f(\widehat{s_2},\widehat{s_3}) & & & \\ \vdots & & & \ddots & f(\widehat{s_{x-1}},\widehat{s_x}) \\ f(\widehat{s_1},\widehat{s_x}) & \cdots & & f(\widehat{s_{x-1}},\widehat{s_x}) & 0 \end{pmatrix} \quad (85).$$

donde:

D: Matriz de distancia entre los pares de secuencias del cojunto N.

$f(\widehat{s_a},\widehat{s_b})$: Fusión de similitud en entre la secuencia $\widehat{s_a}\ y\ \widehat{s_b}$.

La función de similitud utilizada en el algoritmo es la siguiente, que es una particularización de la función COOFEE para el caso de dos secuencias.

$$f(\widehat{s_a},\widehat{s_b}) = \frac{\mathrm{s}_{\widehat{s_a},\widehat{s_b}} - \mathrm{scoring}_{\widehat{s_a},\widehat{s_b}}}{\mathrm{W_{a,b}} * \mathrm{l_M}} \quad (86).$$

donde:

$\mathrm{W_{a,b}}$: Ponderación de la relación entre las secuencias $\widehat{s_a}\ y\ \widehat{s_b}$

$\mathrm{s}_{\widehat{s_a},\widehat{s_b}}$: Similitud entre las secuencias $\widehat{s_a},\widehat{s_b}$

$\mathrm{l_M}$: Número de columnas de la matriz de alineamiento.

$$\mathrm{s}_{\widehat{s_a},\widehat{s_b}} = \sum_{\mathrm{i}=1}^{\mathrm{l_M}} \mathrm{s}(\widehat{s_{a_{(1,i)}}},\widehat{s_{b_{(1,i)}}}) \quad (87).$$

donde:

$\mathrm{l_M}$: Número de columnas de la matriz de alineamiento.

$\widehat{s_{a_{(1,i)}}},\widehat{s_{b_{(1,i)}}}$: Valor de similitud entre el i –ésimo elemento de ambas secuencias.

$$s\left(\widehat{s_{a_{(1,i)}}},\widehat{s_{b_{(1,i)}}}\right) = \begin{cases} y_1 & si\ y\ sólo\ si\ \widehat{s_{a_{(1,i)}}} = \widehat{s_{b_{(1,i)}}}\ y\ caracteres \\ y_2 & si\ y\ sólo\ si\ \widehat{s_{a_{(1,i)}}} = \widehat{s_{b_{(1,i)}}}\ y\ espacios\ vacios \\ y_3 & si\ y\ solo\ si\ \widehat{s_{a_{(1,i)}}}\ y\ \widehat{s_{b_{(1,i)}}} \in \Theta \\ 0 & en\ otro\ caso \end{cases} \quad (88).$$

donde:

Θ: Subconjunto de elementos del alfabeto con características comunes.

$\mathrm{scoring}_{\widehat{s_a},\widehat{s_b}}$: Costo por introducir y extender espacios vacios.

$$\mathrm{scoring}_{\widehat{s_a},\widehat{s_b}} = \mathrm{penalización}_{\mathrm{s_a}} + \mathrm{penalización}_{\mathrm{s_b}} \quad (89).$$

$$\mathrm{penalización} = \mathrm{c_1 w_{introducir}} + \mathrm{c_2 w_{extender}} \quad (90).$$

donde
$c_{introducir}$: Costo por introducir espacios vacios.
$c_{extender}$: Costo por extender espacios vacios

La dimensión de la matriz D se relacionan con el número de secuencias del conjunto N, donde el número de columnas y filas es igual al número de secuencias del conjunto. Para calcular la matriz D se utiliza la siguiente rutina.

Algoritmo 23. Construcción de la matriz D.

Input: Submatriz M^P .
Output: Matriz de distancia entre los pares posibles.

Calculo de similitud entre secuencias.

1. For $a = 1: a \leq x: ++$
    a. For $b = 1: b \leq x: ++$
        i. If $(a < b)$
        ii. $\text{similitud} = 0$
            1. For $j = 1: l_M: ++$
                a. If $(\widehat{s_b} = \widehat{s_a})$
                    i. If $\left((\widehat{s_b} =' -')||(\widehat{s_a} =' -')\right)$
                        1. $\text{similitud} = \text{similitud} + y_2$
                    ii. Else
                        1. $\text{similitud} = \text{similitud} + y_1$
                        2. End If
                b. Else
                    i. If $\left((\widehat{s_b} \in \theta)\&\&(\widehat{s_a} \in \theta)\right)$
                        1. $\text{similitud} = \text{similitud} + y_3$
                    ii. Else
                        1. $\text{similitud} = \text{similitud} + 0$
                        2. End If
                End if
            2. $s_{\widehat{s_a},\widehat{s_b}} \leftarrow similitud$
        iii. Else if $(a = b)$
            1. $s_{\widehat{s_a},\widehat{s_b}} \leftarrow 0$
        iv. Else
            1. $s_{\widehat{s_a},\widehat{s_b}} = s_{\widehat{s_b},\widehat{s_a}}$
        End if
    End for
End For.
2. For $a = 1: a \leq x: ++$
    a. $\text{penalización}_{s_a} \leftarrow$ Calcular penalización por inserción y extensión de espacios vacios $\widehat{s_a}$
End for

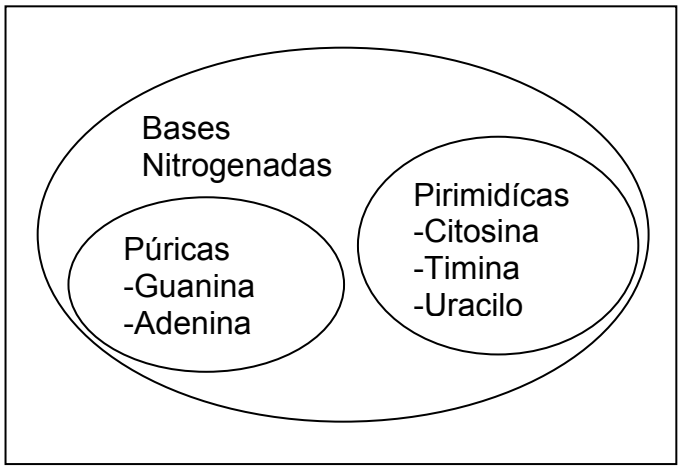
3. For $a = 1{:}\,x{:}++$
   a. For $b = 1{:}\,x{:}++$
      i. If $(a < b)$
         1. $\text{scoring}_{\widehat{s_a},\widehat{s_b}} = \text{penalización}_{\text{s}_\text{a}} + \text{penalización}_{\text{s}_\text{b}}$
         2. $D_{a,b} \leftarrow \frac{\text{s}_{\widehat{s_a},\widehat{s_b}} - \text{scoring}_{\widehat{s_a},\widehat{s_b}}}{\text{W}_{\text{a,b}} * \text{l}_\text{M}}$
      ii. Else if $(a = b)$
         1. $D_{a,b} \leftarrow 0$
      iii. Else
         1. $D_{a,b} \leftarrow D_{b,a}$
      End If
   End For
End For

La complejidad del algoritmo anterior es $O\left(\frac{x(x-1)}{2} * l_M\right)$, con objeto de ilustrar el funcionamiento del algoritmo anterior considere lo siguiente

Ejemplo 24. Retomando el ejemplo anterior se tiene una la submatriz $M^P = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t \\ - & a & c & c & g & - & - & c & g & - & - & - & - \\ g & g & - & g & g & t & c & - & c & c & c & t & - \\ - & - & - & g & - & - & - & - & - & g & - & - & - \\ g & g & - & a & t & - & - & c & t & - & t & - & t \end{bmatrix}$, considerando la función de similitud $s\left(\widehat{s_{a_{(1,\iota)}}}, \widehat{s_{b_{(1,\iota)}}}\right) = \begin{cases} 1 & si\ y\ solo\ si\ \ \widehat{s_{a_{(1,\iota)}}} = \widehat{s_{b_{(1,\iota)}}}\ y\ caracteres \\ 0.5 & si\ y\ solo\ si\ \ \widehat{s_{a_{(1,\iota)}}} = \widehat{s_{b_{(1,\iota)}}}\ y\ espacios\ vacios \\ 0.5 & si\ y\ solo\ si\ \ \widehat{s_{a_{(1,\iota)}}}\ y\ \widehat{s_{b_{(1,\iota)}}} \in \Theta \\ 0 & en\ otro\ caso \end{cases}$ un costo por inserción de espacios vacios de 0.01 y por extensión de 0.005, y la matriz de ponderación $W = \begin{bmatrix} 0 & 0.4 & 0.7273 & 0.10 & 0.4 \\ 0.4 & 0 & 0.1 & 0.2857 & 0.1429 \\ 0.7273 & 0.1 & 0 & 0.1 & 0.2 \\ 0.10 & 0.2857 & 0.10 & 0 & 0.25 \\ 0.4 & 0.1429 & 0.2 & 0.25 & 0 \end{bmatrix}$ encontrar la matriz D correspondiente.

Antes de comenzar la resolución del problema deben definir los subgrupos posibles del alfabeto. Dado a que el conjunto $N = \{s_1 = \{\text{g, g, a, t, t, c, c, c, c, t}\},\ s_2 = \{\text{a, c, c, c, g, c, g}\},\ s_3 = \{\text{g, g, g, g, t, c, c, c, c, t}\},\ s_4 = \{\text{g, g}\},\ s_5 = \{\text{g, g, a, t, c, t, t, t}\}\}$ se forma por biosecuencias de nucleótidos, entonces el alfabeto es $\mathcal{A} = \{A, G, C, T, U\}$ que representa las bases nitrogenadas. Se clasifican a los nucleótidos en : A) Bases púricas (Guanina y Adenina) y B) Bases pirimidínicas (Citosina Timina Uracilo). Con base a esta información biológica se determina los dos subgrupos posibles del alfabeto, lo cual se representa mediante la siguiente figura.

Figura 31. Subgrupos de las bases nitrogenadas.



Con base a la información anterior puede generarse la matriz D.

Inicia ciclo for

$a = 1$

$b = 1$

	$\widehat{s_1}$	$\widehat{s_2}$	$\widehat{s_3}$	$\widehat{s_4}$	$\widehat{s_5}$
$\widehat{s_1}$	0				
$\widehat{s_2}$					
$\widehat{s_3}$					
$\widehat{s_4}$					
$\widehat{s_5}$					

$b = 2$

$$\widehat{s_1} = [\ -\ \ g\ \ -\ \ -\ \ g\ \ a\ \ t\ \ t\ \ c\ \ c\ \ c\ \ c\ \ t\ ]$$

$$\widehat{s_2} = [\ -\ \ a\ \ c\ \ c\ \ g\ \ -\ \ -\ \ c\ \ g\ \ -\ \ -\ \ -\ \ -\ ]$$

$$s\left(\widehat{s_{a_{(1,l)}}}, \widehat{s_{b_{(1,l)}}}\right) = [\ 0.5\ \ 0.5\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0.5\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]$$

$$s(\widehat{s_a}, \widehat{s_b}) = 0.5 + 0.5 + 0 + 0 + 1 + 0 + 0 + 0.5 + 0 + 0 + 0 + 0 + 0 = 2.5$$

	$\widehat{s_1}$	$\widehat{s_2}$	$\widehat{s_3}$	$\widehat{s_4}$	$\widehat{s_5}$
$\widehat{s_1}$	0	2.5			
$\widehat{s_2}$	2.5				
$\widehat{s_3}$					
$\widehat{s_4}$					
$\widehat{s_5}$					

$b = 3$

$$\widehat{s_1} = [\ -\ g\ -\ -\ g\ a\ t\ t\ c\ c\ c\ c\ t\ ]$$
$$\widehat{s_2} = [\ g\ g\ -\ g\ g\ t\ c\ -\ c\ c\ c\ t\ -\ ]$$
$$s\left(\widehat{s_{a_{(1,\iota)}}}, \widehat{s_{b_{(1,\iota)}}}\right) = [\ 0\ 1\ 0.5\ 0\ 1\ 0\ 0.5\ 0\ 1\ 1\ 1\ 0.5\ 0\ ]$$

$$s(\widehat{s_a}, \widehat{s_b}) = 0 + 1 + 0.5 + 0 + 1 + 0 + 0.5 + 0 + 1 + 1 + 1 + 0.5 + 0 = 6.5$$

	$\widehat{s_1}$	$\widehat{s_2}$	$\widehat{s_3}$	$\widehat{s_4}$	$\widehat{s_5}$
$\widehat{s_1}$	0	2.5	6.5		
$\widehat{s_2}$	2.5				
$\widehat{s_3}$	6.5				
$\widehat{s_4}$					
$\widehat{s_5}$					

$b = 4$

$$\widehat{s_1} = [\ -\ g\ -\ -\ g\ a\ t\ t\ c\ c\ c\ c\ t\ ]$$
$$\widehat{s_2} = [\ -\ -\ -\ g\ -\ -\ -\ -\ -\ g\ -\ -\ -\ ]$$
$$s\left(\widehat{s_{a_{(1,\iota)}}}, \widehat{s_{b_{(1,\iota)}}}\right) = [\ 0.5\ 0\ 0.5\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ]$$

$$s(\widehat{s_a}, \widehat{s_b}) = 0.5 + 0 + 0.5 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 1$$

	$\widehat{s_1}$	$\widehat{s_2}$	$\widehat{s_3}$	$\widehat{s_4}$	$\widehat{s_5}$
$\widehat{s_1}$	0	2.5	6.5	1	
$\widehat{s_2}$	2.5				
$\widehat{s_3}$	6.5				
$\widehat{s_4}$	1				
$\widehat{s_5}$					

$b = 5$

$$\widehat{s_1} = [\ -\ g\ -\ -\ g\ a\ t\ t\ c\ c\ c\ c\ t\ ]$$
$$\widehat{s_2} = [\ g\ g\ -\ a\ t\ -\ -\ c\ t\ -\ t\ -\ t\ ]$$
$$s\left(\widehat{s_{a_{(1,\iota)}}}, \widehat{s_{b_{(1,\iota)}}}\right) = [\ 0\ 1\ 0.5\ 0\ 0\ 0\ 0\ 0.5\ 0.5\ 0\ 0.5\ 0\ 1\ ]$$

$$s(\widehat{s_a}, \widehat{s_b}) = 0 + 1 + 0.5 + 0 + 0 + 0 + 0 + 0.5 + 0.5 + 0 + 0.5 + 0 + 1 = 4$$

	$\widehat{s_1}$	$\widehat{s_2}$	$\widehat{s_3}$	$\widehat{s_4}$	$\widehat{s_5}$
$\widehat{s_1}$	0	2.5	6.5	1	4
$\widehat{s_2}$	2.5				
$\widehat{s_3}$	6.5				
$\widehat{s_4}$	1				
$\widehat{s_5}$	4				

Implementando el procedimiento en el resto de las secuencias, se obtiene lo siguiente:

	$\widehat{s_1}$	$\widehat{s_2}$	$\widehat{s_3}$	$\widehat{s_4}$	$\widehat{s_5}$
$\widehat{s_1}$	0	2.5	6.5	1	4
$\widehat{s_2}$	2.5	0	2	3	3.5
$\widehat{s_3}$	6.5	2	0	2.5	4
$\widehat{s_4}$	1	3	2.5	0	2.5
$\widehat{s_5}$	4	3.5	4	2.5	0

Termina ciclo for.

Cálculo del costo por inserción y extensión de espacios vacios.

$$\begin{matrix} & & & & & & & & & & & & & Secuencias & & Incertados & Extendidos \\ \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t \\ - & a & c & c & g & - & - & c & g & - & - & - & - \\ g & g & - & g & g & t & c & - & c & c & c & t & - \\ - & - & - & g & - & - & - & - & - & g & - & - & - \\ g & g & - & a & t & - & - & c & t & - & t & - & t \end{bmatrix} & & & & & & & & & & & & & 2 & 1 \\ & & & & & & & & & & & & & 3 & 4 \\ & & & & & & & & & & & & & 3 & 0 \\ & & & & & & & & & & & & & 3 & 8 \\ & & & & & & & & & & & & & 4 & 1 \end{matrix}$$

Se determina el valor de la penalización para cada una de las secuencias.

Secuencia	Calculo de penalización	Valor
s_1	(0.01*2)+(0.005*1)	0.025
s_2	(0.01*3)+(0.005*4)	0.05
s_3	(0.01*3)+(0.005*0)	0.03
s_4	(0.01*3)+(0.005*8)	0.07
s_5	(0.01*4)+(0.005*1)	0.045

Con la información anterior se calcula la matriz D y la fórmula de función $\left(\frac{\mathrm{s}_{\widehat{s_a},\widehat{s_b}}-\mathrm{scoring}_{\widehat{s_a},\widehat{s_b}}}{\mathrm{W_{a,b}}*\mathrm{l_M}}\right)$se calcula el valor para cada una de las celdas.

$$D_{1,1} = 0$$

$$D_{1,2} = \frac{2.5 - (.25 + 0.5)}{0.4 * 13} = 0.4663$$

Realizando iterativamente la información anterior sobre los datos anteriores se obtiene

$$D = \begin{pmatrix} 0 & 0.4663 & 0.6817 & 0.6962 & 0.7558 \\ 0.4663 & 0 & 1.4769 & 0.7754 & 1.8335 \\ 0.6817 & 1.7469 & 0 & 1.8462 & 1.5096 \\ 0.6962 & 0.7754 & 1.8462 & 0 & 0.7338 \\ 0.7558 & 1.8335 & 1.5096 & 0.7338 & 0 \end{pmatrix}$$

Cada una de las filas en la submatriz G^P en la primera columna se asocia con la similitud de la i –ésima secuencia con el resto de las secuencias del conjunto. Lo cual, se determina mediante la siguiente relación.

$$G_{i,1}^P = \sum_{j=1}^{x} D_{i,j} \qquad (91).$$

Para determinar la función de aptitud de cada una de las secuencias en el alineamiento se requiere calcular el valor del alineamiento propuesto. Lo cual se representa mediante la siguiente ecuación.

$$V(M^P) = \frac{\sum_{i,1}^{x} G_{i,1}^P}{2} \qquad (92).$$

El valor de la función de aptitud para la i –ésima secuencia dentro del alineamiento se determina mediante la siguiente relación:

$$fitness_{\hat{s}_i} = \frac{G_{i,1}^P}{2 * V(M^P)} \qquad (93).$$

Para la asignación de valores de la submatriz G^P se utiliza la siguiente rutina:

Algoritmo 24. Construcción de la submatriz G^P.

Input: Matriz D.

Output: Submatriz G^P.

1. $x \leftarrow$ Número de secuencia en el conjunto N.
2. For $i = 1; i \leq x; ++$
   a. $G_{i,1}^P = \sum_{j=1}^{x} D_{i,j}$
   End For
3. $V(M^P) = \frac{\sum_{i,1}^{x} G_{i,1}^P}{2}$
4. For $i = 1; i \leq x; ++$
   a. $G_{i,2}^P = \frac{G_{i,1}^P}{2 * V(M^P)}$
   End for.

La complejidad del algoritmo anterior es $O(2x)$, con objeto de ilustrar el funcionamiento del algoritmo anterior considere lo siguiente

Ejemplo 25. Continuando con el ejemplo anterior se tiene la submatriz $D = \begin{pmatrix} 0 & 0.4663 & 0.6817 & 0.6962 & 0.7558 \\ 0.4663 & 0 & 1.4769 & 0.7754 & 1.8335 \\ 0.6817 & 1.7469 & 0 & 1.8462 & 1.5096 \\ 0.6962 & 0.7754 & 1.8462 & 0 & 0.7338 \\ 0.7558 & 1.8335 & 1.5096 & 0.7338 & 0 \end{pmatrix}$ y se desea encontrar la sub-matriz G^P correspondiente.

Primer ciclo for

$i = 1$

$$G_{1,1}^P = 0 + 0.4663 + 0.6817 + 0.6962 + 0.7558 = 2.6$$

$i = 2$

$$G_{2,1}^P = 0.4663 + 0 + 1.4769 + 0.7754 + 1.8335 = 4.5521$$

$i = 3$

$$G_{3,1}^P = 0.6817 + 1.7469 + 0 + 1.8462 + 1.5096 = 5.8029$$

$i = 4$

$$G_{4,1}^P = 0.6962 + 0.7754 + 1.8462 + 0 + 0.7338 = 4.046$$

$i = 5$

$$G_{5,1}^P = 0.7558 + 1.8335 + 1.5096 + 0.7338 + 0 = 4.8327$$

Termina ciclo for.

$$V(M) = \frac{2.6 + 4.5521 + 5.8029 + 4.046 + 4.8327}{2} = 10.92115$$

Segundo ciclo for

$i = 1$

$$G_{1,2}^P = \frac{2.6}{21.8423} \approx 0.119$$

$i = 2$

$$G_{2,2}^P = \frac{4.5521}{21.8423} \approx 0.208$$

$i = 3$

$$G_{3,2}^P = \frac{5.8029}{21.8423} \approx 0.266$$

$i = 4$

$$G_{4,2}^P = \frac{4.0546}{21.8423} = 0.186$$

$i = 5$

$$G_{5,2}^P = \frac{4.8427}{21.8423} \approx 0.221$$

Termina ciclo for.

Con la información anterior se forma la siguiente submatriz G^P.

$$G^P = \begin{bmatrix} 2.6 & 0.119 \\ 4.5521 & 0.208 \\ 5.8029 & 0.266 \\ 4.0546 & 0.186 \\ 4.8327 & 0.221 \end{bmatrix}$$

A.3 Construcción de matriz auxiliar P.

Para formar la matriz auxiliar P se utilizan los algoritmos 15, 16 y 17 en la siguiente rutina.

Algoritmo 25. Construcción de la matriz P.

Input: Un conjunto de secuencias $(N = \{s_1, s_2, \dots, s_x\})$, número de columnas que integrarán la a la submatriz M^P (l_M)
Output: Matriz P.

1. Construir la submatriz M^P $O(x * l_M)$,
2. Construir la matriz D $O\left(\frac{x(x-1)}{2} * l_M\right)$
3. Construir la submatriz G^P $O(2x)$
4. $P = [M^P, G^P]$ $O(1)$

Al sumar la complejidad de las subrutinas asociadas a la construcción de la matriz P se obtiene:

$$O(x * l_M) + O\left(\frac{x(x-1)}{2} * l_M\right) + O(2x) + O(1)$$

Por lo tanto la complejidad de la rutina asociada a la construcción de la matriz P es del orden $O(x^2 * l_M)$. Para mostrar el funcionamiento del algoritmo anterior considere lo siguiente.

Ejemplo 26. En los ejemplos anterior se obtuvieron las matrices, $M^P = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t \\ - & a & c & c & g & - & - & c & g & - & - & - & - \\ g & g & - & g & g & t & c & - & c & c & c & t & - \\ - & - & - & g & - & - & - & - & - & g & - & - & - \\ g & g & - & a & t & - & - & c & t & - & t & - & t \end{bmatrix}$, $D = \begin{pmatrix} 0 & 0.4663 & 0.6817 & 0.6962 & 0.7558 \\ 0.4663 & 0 & 1.4769 & 0.7754 & 1.8335 \\ 0.6817 & 1.7469 & 0 & 1.8462 & 1.5096 \\ 0.6962 & 0.7754 & 1.8462 & 0 & 0.7338 \\ 0.7558 & 1.8335 & 1.5096 & 0.7338 & 0 \end{pmatrix}$

y $G^P = \begin{bmatrix} 2.6 & 0.119 \\ 4.5521 & 0.208 \\ 5.8029 & 0.266 \\ 4.0546 & 0.186 \\ 4.8327 & 0.221 \end{bmatrix}$ con las cuales se construye la siguiente matriz auxiliar P.

$$P = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t & 2.6 & 0.119 \\ - & a & c & c & g & - & - & c & g & - & - & - & - & 4.5521 & 0.208 \\ g & g & - & g & g & t & c & - & c & c & c & t & - & 5.8029 & 0.266 \\ - & - & - & g & - & - & - & - & - & g & - & - & - & 4.0546 & 0.186 \\ g & g & - & a & t & - & - & c & t & - & t & - & t & 4.8327 & 0.221 \end{bmatrix}$$

B *Creación de la memoria armónica*

En la memoria armónica el número de filas corresponde al valor del HMS, mientras que, el número de columnas es tres.

Cada una de las filas en HM representa un alineamiento propuesto, por lo que resulta necesario mantener en memoria las matices auxiliares P correspondientes. Las columnas de la HM corresponde a los siguientes valores: A) número de columnas de la submatriz M^P, B) valor del alineamiento propuesto $V(M^P)$ y C) Función de aptitud del alineamiento propuesto (fa). Lo anterior se representa en la siguiente igualdad.

$$HM = \begin{bmatrix} l_M^1 & V(M^1) & fa^1 \\ l_M^2 & V(M^2) & fa^2 \\ \vdots & \vdots & \vdots \\ l_M^{HMS} & V(M^{HMS}) & fa^{HMS} \end{bmatrix} \qquad (94).$$

donde:
l_M^i: Número de columnas del i –ésimo alineamiento propuesto
$V(M^i)$: Valor de la función objetivo del i – ésimo alineamiento propuesto.
fa^i: Función de aptitud del i – ésimo alineamiento propuesto.
Para encontrar el valor de la variable fa se utilizan las siguientes ecuaciones:

$$fa^i = \left(\frac{q - V(M^i)}{q}\right) \qquad (95).$$

$$q = \sum_{p=1}^{HMS} V(M^p) \qquad (96).$$

donde:
q: Suma del valor de las funciones objetivo dentro de la memoria armónica.
fa^i: Función de aptitud del i –ésimo alineamiento propuesto.

Algoritmo 26. Construcción de la memoria armónica inicial.

Input: Un conjunto de secuencias $(N = \{s_1, s_2, \dots, s_x\})$, tamaño de la memoria armonica HMS.

Output: Memoria armónica.

1. $q = 0$
2. For $P = 1; P \leq HMS; ++$
    1. ${l_M}^P = \text{int}\left(l_{max} + \left((caracteres - l_{max}) * rand\right)\right)$ $O(1)$
    2. Construir una matriz auxiliar P $O(x^2 * l_M)$
    3. Actualizar el valor de ${l_M}^P$

4. $HM_{P,1} \leftarrow {l_M}^P$ $O(1)$
5. $HM_{p,2} \leftarrow V(M^p)$ $O(1)$

End For.

3. $q = \sum_{p=1}^{HMS} HM_{p,2}$ $O(1)$
4. For $P = 1; P \leq HMS; ++$
    1. $HM_{i,3} = \frac{q - HM_{i,2}}{q}$ $O(1)$

End For

La complejidad de la rutina anterior es $O\left(x^2 * {l_M}^2\right)$, para mostrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 27. Dadas las secuencias $N = \{s_1 = \{\text{g, g, a, t, t, c, c, c, c, t}\},\ s_2 = \{\text{a, c, c, c, g, c, g}\},\ s_3 = \{\text{g, g, g, g, t, c, c, c, c, t}\},\ s_4 = \{\text{g, g}\},\ s_5 = \{\text{g, g, a, t, c, t, t, t}\}\}$, y un $HMS = 2$ se desea construir la memoria armónica inicial.

Sea $q = 0$

Primer ciclo For

$p = 1$

$n = \text{int}\big(10 + (27 * rand)\big) = 11$

/*Construir la matriz P*/

/*Construcción de la submatriz M^1 */

$$M^1 = \begin{bmatrix} \text{g} & \text{g} & \text{a} & \text{t} & \text{t} & \text{c} & - & \text{c} & \text{c} & \text{c} & \text{t} \\ \text{a} & - & \text{c} & \text{c} & - & \text{c} & \text{g} & \text{c} & - & \text{g} & - \\ - & \text{g} & \text{g} & \text{g} & \text{g} & \text{t} & \text{c} & \text{c} & \text{c} & \text{c} & \text{t} \\ - & \text{g} & - & - & \text{g} & - & - & - & - & - & - \\ \text{g} & - & \text{g} & \text{a} & - & \text{t} & \text{c} & - & \text{t} & \text{t} & \text{t} \end{bmatrix}$$

/*Construcción de la matriz D*/

$$D = \begin{pmatrix} 0 & 0.6705 & 0.7475 & 1.3 & 0.9 \\ 0.6705 & 0 & 1.3182 & 0.2864 & 1.2282 \\ 0.7475 & 1.3182 & 0 & 2.2091 & 2.4818 \\ 1.3 & 0.2864 & 2.2091 & 0 & 0.1491 \\ 0.9 & 1.2282 & 2.4818 & 0.1491 & 0 \end{pmatrix}$$

/*Construcción de la submatriz G^1*/

$$\boldsymbol{G^1} = \begin{bmatrix} 3.618 & 0.16 \\ 3.5033 & 0.155 \\ 6.7566 & 0.299 \\ 3.9446 & 0.714 \\ 4.7591 & 0.21 \end{bmatrix}$$

/* Matriz auxiliar P*/

$$P^1 = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 3.618 & 0.16 \\ a & - & c & c & - & c & g & c & - & g & - & 3.5033 & 0.155 \\ - & g & g & g & g & t & c & c & c & c & t & 6.7566 & 0.299 \\ - & g & - & - & g & - & - & - & - & - & - & 3.9446 & 0.714 \\ g & - & g & a & - & t & c & - & t & t & t & 4.7591 & 0.21 \end{bmatrix}$$

El valor del alineamiento propuesto es de 11.2908 unidades.
/* Primera hilera de la HM */

$$HM = \begin{bmatrix} 11 & 11.2908 & fa^1 \\ n^2 & V(M)^2 & fa^2 \end{bmatrix}$$

$p = 2$
$n = \text{int}\big(10 + (27 * rand)\big) = 13$

/*Construir la matriz P*/

/*Construcción de la submatriz M^1 */

$$M^2 = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t \\ - & a & c & c & g & - & - & c & g & - & - & - & - \\ g & g & - & g & g & t & c & - & c & c & c & t & - \\ - & - & - & g & - & - & - & - & - & g & - & - & - \\ g & g & - & a & t & - & - & c & t & - & t & - & t \end{bmatrix}$$

/*Construcción de la matriz D*/

$$D = \begin{pmatrix} 0 & 0.4663 & 0.6817 & 0.6962 & 0.7558 \\ 0.4663 & 0 & 1.4769 & 0.7754 & 1.8335 \\ 0.6817 & 1.7469 & 0 & 1.8462 & 1.5096 \\ 0.6962 & 0.7754 & 1.8462 & 0 & 0.7338 \\ 0.7558 & 1.8335 & 1.5096 & 0.7338 & 0 \end{pmatrix}$$

/*Construcción de la submatriz G^1*/

$$G^P = \begin{bmatrix} 2.6 & 0.119 \\ 4.5521 & 0.208 \\ 5.8029 & 0.266 \\ 4.0546 & 0.186 \\ 4.8327 & 0.221 \end{bmatrix}$$

/* matriz auxiliar P*/

$$P = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t & 2.6 & 0.119 \\ - & a & c & c & g & - & - & c & g & - & - & - & - & 4.5521 & 0.208 \\ g & g & - & g & g & t & c & - & c & c & c & t & - & 5.8029 & 0.266 \\ - & - & - & g & - & - & - & - & - & g & - & - & - & 4.0546 & 0.186 \\ g & g & - & a & t & - & - & c & t & - & t & - & t & 4.8327 & 0.221 \end{bmatrix}$$

El valor del alineamiento propuesto es de 10.92115 unidades.

/* Segunda hilera de la memoria armónica */

$$HM = \begin{bmatrix} 11 & 11.2908 & fa^1 \\ 13 & 10.92115 & fa^2 \end{bmatrix}$$

Termina ciclo for.

$$q = 11.2908 + 10.92115 = 22.21195$$

Se determina el valor de la función de aptitud de cada uno de los alineamientos propuestos.

$$HM = \begin{bmatrix} 11 & 11.2908 & 0.4916 \\ 13 & 10.92115 & 0.5083 \end{bmatrix}$$

3.1.1.3 Generación de una nueva armonía.

Una vez construida la memoria armónica inicial, se procede a generar una nueva improvisación, con el objeto de mejorar el valor de los alineamientos propuestos. Para lo cual se utiliza la siguiente rutina.

Algoritmo 27. Generación de una nueva improvisación.

Input: Conjunto de secuencias $S = \{s_1, s_2, \dots, s_x\}$, Vector de longitudes de las secuencias l, memoria armónica inicial HM, HMCRI,HMCRF, PARI,PARF,l_M, b, Matrices auxiliares asociadas $(P^1, P^2, \dots, P^{HMS})$.

Output: Nueva armonía.

/*Probabilidad de considera un alineamiento propuesto en memoria armónica*/

1. $PAR = PARI + \left((PARF - PARI) * \left(\frac{v}{MaxImp}\right)\right)$
2. $HMCR = HMCRI + \left((HMCRF - HMCRI) * \left(\frac{v}{MaxImp}\right)\right)$
3. If $rand \leq HMCR$
   a. $p = 1$;
   b. $elección = rand$;
   c. $fa \leftarrow HM_{1,3}$;

/*Elección aleatoria de un alineamiento*/

   d. While $elección < fa$.
      i. $p = p + 1$
      ii. $fa = fa + HM_{P,3}$
      End While
   e. $l_p = HM_{p,1}$
   f. $Auxiliar \leftarrow P^P$

/*Utilizar un acorde de memoria*/

   g. For $i = 1{:}\,x{:}\,++$
      i. If $\left(Auxiliar_{i,l_p} \leq rand\right)$
         1. For $j = 1{:}\,1{:}\,l_p$
            a. $A_{1_{i,j}} \leftarrow Auxiliar_{i,j}$
         End for
      ii. Else

1. $lu \leftarrow l_i$
2. $u = 1$
3. $\mathcal{S} \leftarrow l_p$
4. For $j = 1; j \leq l_p; ++$
    a. $P\left(A_{1_{\mathrm{i,j}}} \leftarrow \mathrm{caracter}\right) = \frac{l}{\mathcal{S}}$
    b. If $\left(rand \leq P\left(A_{1_{\mathrm{i,j}}} \leftarrow \mathrm{caracter}\right)\right)$
        i. $A_{1_{i,j}} \leftarrow s_{i_u}$
        ii. $\mathrm{u} = \mathrm{u} + 1$
        iii. $lu = lu - 1$
    c. Else
        i. $A_{1_{i,j}} \leftarrow '-'$

    End if
5. $\mathcal{S} = \mathcal{S} - 1$
6. End for

iii. End if

h. End For
i. $Auxiliar \leftarrow A_1$
j. Actualizar el valor de l_p
k. Creae la matriz D.
l. For $i = 1: x: ++$
    i. $Auxiliar_{i,l_p+1} \leftarrow \sum_{j=1}^{l_p} D_{ij}$

End For

m. $V(Auxiliar) \leftarrow \frac{\sum_{i=1}^{x} Auxiliar_{i,l_p+1}}{2}$
n. For $i = 1: x: ++$
    i. $Auxiliar_{i,l_p+2} \leftarrow \frac{\sum_{j=1}^{l_p} D_{ij}}{2 * V(Auxiliar)}$

End For

o. If $(rand \leq PAR)$
    i. $l_v \leftarrow \lfloor (l_p - b) + (2 * b * rand) \rceil$
    ii. If $l_v > caracteres$
        1. $l_v \leftarrow \lfloor caracteres - b \rfloor$
    iii. Else if $l_v < l_{max}$
        1. $l_v \leftarrow \lfloor l_{max} + b \rfloor$

    End if
    iv. If $(l_v \neq l_p)$
        1. If $(l_v < l_p)$
            a. For $i = 1: x: ++$
                i. $sde = l_p - l_v$
                ii. Remover aleatoriamente sde espacios vacios de la secuencia $Auxiliar_i$

            End For
        2. Else If

a. For $i = 1: x: + +$
   i. $sde = l_v - l_p$
   ii. Agregar aleatoriamente sde espacios vacios de la secuencia $\overset{s}{Auxiliar_i}$
End For
End If

v. Else If
1. For $i = 1: x: + +$
   a. If $\left(rand \leq Auxiliar_{\ i,l_p+2}\right)$
      i. Generar aleatoriamente un nuevo vector $Auxiliar_{i.l_p}$
   b. Else if
      i. Dejar intacto el vector $Auxiliar_{i.l_p}$
   End If
2. End For
3. For $i = 1: x: + +$
   a. $Auxiliar_{i,l_p+1} \leftarrow \sum_{j=1}^{l_p} D_{ij}$
   End For
4. $V(Auxiliar) \leftarrow \frac{\sum_{i=1}^{x} Auxiliar_{i,l_p+1}}{2}$
5. For $i = 1: x: + +$
   a. $Auxiliar_{i,l_p+2} \leftarrow \frac{\sum_{j=1}^{lp} D_{ij}}{2*V(Auxiliar)}$
   End For
End If
End for

4. Else
   a. $l_p = \lfloor (rand * (caracteres - l_{max})) + l_{max} \rfloor$
   b. Generar una matriz P con base en el parámetro l_p
   c. $Auxiliar \leftarrow P$
   d. For $i = 1: x: + +$
      i. $Auxiliar_{i,l_p+1} \leftarrow \sum_{j=1}^{l_p} D_{ij}$
   End For
   e. $V(Auxiliar) \leftarrow \frac{\sum_{i=1}^{x} Auxiliar_{i,l_p+1}}{2}$
   f. For $i = 1: x: + +$
      i. $Auxiliar_{i,l_p+2} \leftarrow \frac{\sum_{j=1}^{lp} D_{ij}}{2*V(Auxiliar)}$
   End For
End If

La complejidad del algoritmo anterior es de $O(x^2 * l_M)$, para mostrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 28. Dada, una memoria armónica inicial. $HM = \begin{bmatrix} 11 & 11.2908 & 0.4916 \\ 13 & 10.92115 & 0.5083 \end{bmatrix}$, $PARI = 0.05$, $PARF = 0.5$, un $HMCRF = 0.95$, $HMCRI = 0.7$, $v = 1$, $Masxlmp = 3$ y las matrices auxiliares

$$P^1 = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 3.618 & 0.16 \\ a & - & c & c & - & c & g & c & - & g & - & 3.5033 & 0.155 \\ - & g & g & g & g & t & c & c & c & c & t & 6.7566 & 0.299 \\ - & g & - & - & g & - & - & - & - & - & - & 3.9446 & 0.714 \\ g & - & g & a & - & t & c & - & t & t & t & 4.7591 & 0.21 \end{bmatrix}$$

$$P^2 = \begin{bmatrix} - & g & - & - & g & a & t & t & c & c & c & c & t & 2.6 & 0.119 \\ - & a & c & c & g & - & - & c & g & - & - & - & - & 4.5521 & 0.208 \\ g & g & - & g & g & t & c & - & c & c & c & t & - & 5.8029 & 0.266 \\ - & - & - & g & - & - & - & - & - & g & - & - & - & 4.0546 & 0.186 \\ g & g & - & a & t & - & - & c & t & - & t & - & t & 4.8327 & 0.221 \end{bmatrix}$$

se desea generar una nueva armonía

Se genera primero el valor de $HMCR\ y\ PAR$ correspondientes a la iteración.

$$HMCR = 0.7 + \left((0.95 - 0.7) * \left(\frac{1}{6} \right) \right) = 0.74$$

$$PAR = 0.05 + \left((0.5 - 0.05) * \left(\frac{1}{6} \right) \right) = .125$$

Se genera un aleatorio $(rand = 0.533) \leq (HMCR = 0.74)$ por lo que se usara información disponible en la memoria armónica.

Ahora se debe elegir aleatoriamente una hilera de la HM, para ello se genera un aleatorio $(rand = 0.279)$ y por lo que se elije la hilera uno.
A continuación debe generarse aleatoriamente $l_v = 11$ modificara el alineamiento en la memoria armónica. A continuación se muestra el método de modificación.

$$P^1 = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 3.618 & 0.16 \\ a & - & c & c & - & c & g & c & - & g & - & 3.5033 & 0.155 \\ - & g & g & g & g & t & c & c & c & c & t & 6.7566 & 0.299 \\ - & g & - & - & g & - & - & - & - & - & - & 3.9446 & 0.714 \\ g & - & g & a & - & t & c & - & t & t & t & 4.7591 & 0.21 \end{bmatrix}$$

$i = 1$
$rand = 0.374$ Que es menor a $(1 - P^1_{1,13})$; por lo cual se copian los elementos correspondientes a la matriz auxiliar.

$$Aux = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & a_{1,12} & a_{1,13} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} & a_{2,9} & a_{2,10} & a_{2,11} & a_{2,12} & a_{2,13} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} & a_{3,9} & a_{2,10} & a_{3,11} & a_{3,12} & a_{3,13} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & a_{4,9} & a_{2,10} & a_{4,11} & a_{4,12} & a_{4,13} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{4,5} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & a_{2,10} & a_{5,11} & a_{5,12} & a_{5,13} \end{bmatrix}$$

$i = 2$
$rand = 0.87$ que es mayor a $\left(1 - P_{2,13}^{1}\right)$; por lo cual deben de cambiarse la secuencia $\widehat{s_2}$.

$$Aux = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & a_{1,12} & a_{1,13} \\ - & a & c & - & c & c & - & g & c & - & g & a_{2,12} & a_{2,13} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} & a_{3,9} & a_{2,10} & a_{3,11} & a_{3,12} & a_{3,13} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & a_{4,9} & a_{2,10} & a_{4,11} & a_{4,12} & a_{4,13} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{4,5} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & a_{2,10} & a_{5,11} & a_{5,12} & a_{5,13} \end{bmatrix}$$

$i = 3$
$rand = 0.235$ que es menor a $(1 - P_{3,13}^{1})$; por lo cual se copian los elementos correspondientes a la matriz auxiliar.

$$Aux = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & a_{1,12} & a_{1,13} \\ - & a & c & - & c & c & - & g & c & - & g & a_{2,12} & a_{2,13} \\ - & g & g & g & g & t & c & c & c & c & t & a_{3,12} & a_{3,13} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & a_{4,9} & a_{2,10} & a_{4,11} & a_{4,12} & a_{4,13} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{4,5} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & a_{2,10} & a_{5,11} & a_{5,12} & a_{5,13} \end{bmatrix}$$

$i = 4$
$rand = 0.941$ que es mayor a $(1 - P_{4,13}^{1})$; por lo cual deben de cambiarse la secuencia $\widehat{s_4}$.

$$Aux = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & a_{1,12} & a_{1,13} \\ - & a & c & - & c & c & - & g & c & - & g & a_{2,12} & a_{2,13} \\ - & g & g & g & g & t & c & c & c & c & t & a_{3,12} & a_{3,13} \\ g & - & g & - & - & - & - & - & - & - & - & a_{4,12} & a_{4,13} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{4,5} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & a_{2,10} & a_{5,11} & a_{5,12} & a_{5,13} \end{bmatrix}$$

$i = 5$
$rand = 0.556$ que es menor a $(1 - P_{1,13}^{1})$; por lo cual se copian los elementos correspondientes a la matriz auxiliar.

$$Aux = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & a_{1,12} & a_{1,13} \\ - & a & c & - & c & c & - & g & c & - & g & a_{2,12} & a_{2,13} \\ - & g & g & g & g & t & c & c & c & c & t & a_{3,12} & a_{3,13} \\ g & - & g & - & - & - & - & - & - & - & - & a_{4,12} & a_{4,13} \\ g & - & g & a & - & t & c & - & t & t & t & a_{5,12} & a_{5,13} \end{bmatrix}$$

Creación de la matriz D correspondiente es:

$$D = \begin{pmatrix} 0 & 0.7841 & 0.7475 & 1.7591 & 0.9 \\ 0.7841 & 0 & 2.2273 & 0.4470 & 0.5918 \\ 0.7475 & 2.2273 & 0 & 0.85 & 2.4818 \\ 1.7591 & 0.4470 & 0.85 & 0 & 1.2418 \\ 0.9 & 0.5918 & 2.4818 & 1.2418 & 0 \end{pmatrix}$$

La matriz G correspondiente es:

$$\boldsymbol{G^2} = \begin{bmatrix} 4.1791 & 0.174 \\ 4.0502 & 0.168 \\ 6.3066 & 0.262 \\ 4.2979 & 0.179 \\ 5.2154 & 0.217 \end{bmatrix}$$

La matriz auxiliar correspondiente es:

$$Auxiliar = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 4.1791 & 0.174 \\ - & a & c & - & c & c & - & g & c & - & g & 4.0502 & 0.168 \\ - & g & g & g & g & t & c & c & c & c & t & 6.3066 & 0.262 \\ g & - & g & - & - & - & - & - & - & - & - & 4.2979 & 0.179 \\ g & - & g & a & - & t & c & - & t & t & t & 5.2154 & 0.217 \end{bmatrix}$$

$$V(Auxiliar) = \frac{4.1491 + 4.0502 + 6.3066 + 4.2979 + 5.2154}{2} = 12.0246$$

Se genera un aleatorio $\big((rand = .99) \geq (PAR)\big)$ por lo tanto no se realiza un ajuste del ritmo sobre la matriz auxiliar. La nueva improvisación genera un alineamiento cuyo valor de alineamiento es de 12.0246 unidades.

3.1.1.4 Actualización de la memoria armónica.

Una vez determinada la nueva improvisación el siguiente algoritmo, permitirá saber si esta remplazará un acorde en memoria.

Algoritmo 28. Actualización de la memoria armónica.

Input: Una nueva improvisación(matriz auxiliar), t_0, α, $MaxImp$, $V(Auxiliar)$ y la memoria armónica HM

Output: Memoria armónica actualizada.

1. $V_{max} = \min\big(HM_{1,2}, HM_{2,2}, \dots, HM_{HMS,2}\big)$ $O(HMS)$
2. $T = t_0$ $O(1)$
3. $i = 1$ $O(1)$
4. While $V_{min} \neq HM_{i,2}$ $O(HMS)$
   a. $i = i + 1$
   End while
5. If $(V(Auxiliar) \geq V_{min})$ $O(1)$
   a. $HM_{i,1} \leftarrow l_p$
   b. $HM_{i,2} \leftarrow V(Auxiliar)$
   c. $P^i \leftarrow Auxiliar$
6. Else
   /* Probabilidad de ingresar a la memoria armónica un acorde de menor calidad*/

a. $P(a) = e^{\left(\frac{V(Auxiliar)-V_{min}}{\alpha * T}\right)}$
b. $T = \alpha * T$
c. If $rand \leq P(a)$
    i. $HM_{i,1} \leftarrow l_p{}'$
    ii. $HM_{i,2} \leftarrow V(Auxiliar)$
    iii. $P^i \leftarrow Auxiliar$
End if
End if

7. $q = \sum_{p=1}^{HMS} HM_{p,2}$ $O(1)$
8. For $P = 1; P \leq HMS; ++$ $O(HMS)$
    a. $HM_{i,3} \leftarrow \frac{q-HM_{i,2}}{q}$

End For

La complejidad del algoritmo anterior es $O(3HMS)$ que es una complejidad constante. Para mostrar el funcionamiento del algoritmo anterior considere lo siguiente:

Ejemplo 29. Continuando con el ejemplo anterior se tiene la memoria armónica $HM = \begin{bmatrix} 11 & 11.2908 & 0.4916 \\ 13 & 10.92115 & 0.5083 \end{bmatrix}$, se generó una nueva armonía con un $V(Auxiliar) = 12.046$ con la siguiente matriz auxiliar, $Auxiliar = \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 4.1791 & 0.174 \\ - & a & c & - & c & c & - & g & c & - & g & 4.0502 & 0.168 \\ - & g & g & g & g & t & c & c & c & c & t & 6.3066 & 0.262 \\ g & - & g & - & - & - & - & - & - & - & - & 4.2979 & 0.179 \\ g & - & g & a & - & t & c & - & t & t & t & 5.2154 & 0.217 \end{bmatrix}$, $t_0 = 100$ y $\alpha = 0.9$ determinar si esta nueva armonía debe ser incluida en la matriz de armonía.

$V_{max} = \min(11.2908, 10.92115) = 10.92115$

En virtud de $V(Auxiliar) > V_{min}$ por lo tanto, el valor de alineamiento propuesto es mejor que el peor elemento en memora armónica, y por lo tanto.

$$HM_{2,1} \leftarrow 11$$
$$HM_{2,2} \leftarrow 12.046$$

$$P^2 \leftarrow \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 4.1791 & 0.174 \\ - & a & c & - & c & c & - & g & c & - & g & 4.0502 & 0.168 \\ - & g & g & g & g & t & c & c & c & c & t & 6.3066 & 0.262 \\ g & - & g & - & - & - & - & - & - & - & - & 4.2979 & 0.179 \\ g & - & g & a & - & t & c & - & t & t & t & 5.2154 & 0.217 \end{bmatrix}$$

Al sustituir los valores anteriores en la HM y recalcular el vector de aptitud de alineamiento se obtiene la siguiente matriz de memoria armónica.

$$HM = \begin{bmatrix} 11 & 11.2908 & 0.5161 \\ 11 & 12.046 & 0.4838 \end{bmatrix}$$

3.1.1.5 Criterio de paro.

El criterio que se utiliza esta adaptación es el de número máximo de iteraciones (nuevas improvisaciones), Los algoritmos de generación de una nueva armonía y el de actualización de memoria armónica son usados como subrutinas en el siguiente algoritmo de adaptación de la HS para el problema de AMS. Al terminar el número de ejecuciones máximas el algoritmo arrojara el valor de alineamiento más pequeño encontrado.

Algoritmo 29. Satisfacer el criterio de paro.

Input: Memoria armónica HM, HMCR, PAR, b, PASP, Matrices auxiliares asociadas $(P^1, P^2, \dots, P^{HMS})$, $MaxImp$.

Output: Alineamiento del conjunto N obtenido por la búsqueda armónica.

1. For $v = 1; v \leq MaxImp; ++$
   a. Generar una nueva armonía. $O(x^2 * l_M)$
   b. Actualizar la memoria armónica $O(10 + 2HMS)$

   End For

La complejidad del algoritmo es $O\big(MaxImp(x^2 * l_M))\big)$, para mostrar el funcionamiento del algoritmo anterior considere lo siguiente.

Ejemplo 30. Considerando un $MaxImp = 3$ y la información de los ejemplos anteriores, se utiliza el algoritmo anterior para determinar el alineamiento de las secuencias $N = \big\{s_1 = \{\text{g, g, a, t, t, c, c, c, c, t}\},\ s_2 = \{\text{a, c, c, c, g, c, g}\},\ s_3 = \{\text{g, g, g, g, t, c, c, c, c, t}\},\ s_4 = \{\text{g, g}\},\ s_5 = \{\text{g, g, a, t, c, t, t, t}\}\big\}$.

Ciclo For.

$v = 1$

$$P^2 \leftarrow \begin{bmatrix} g & g & a & t & t & c & - & c & c & c & t & 4.1791 & 0.174 \\ - & a & c & - & c & c & - & g & c & - & g & 4.0502 & 0.168 \\ - & g & g & g & g & t & c & c & c & c & t & 6.3066 & 0.262 \\ g & - & g & - & - & - & - & - & - & - & - & 4.2979 & 0.179 \\ g & - & g & a & - & t & c & - & t & t & t & 5.2154 & 0.217 \end{bmatrix}$$

$$M = \begin{bmatrix} 11 & 11.2908 & 0.5161 \\ 11 & 12.046 & 0.4838 \end{bmatrix}$$

$v = 2$
Nueva matriz auxiliar:

$$Auxiliar = \begin{bmatrix} g & g & a & t & t & c & c & c & c & t & 5.2363 & 0.170 \\ a & - & c & c & c & - & g & c & g & - & 4.4513 & 0.141 \\ g & g & g & g & t & c & c & c & c & t & 8.338 & 0.270 \\ g & g & - & - & - & - & - & - & - & - & 5.5828 & 0.181 \\ g & g & - & a & t & - & c & t & t & t & 7.27 & 0.2355 \end{bmatrix}$$

El valor del alineamiento es de $V(Auxiliar) = 15.4391$ unidades.
$V_{min} = \min(11.2908, 12.046) = 11.2908$

En virtud de $V(Auxiliar) > V_{max}$ significa el valor de alineamiento propuesto es mejor que el peor elemento en memoria armónica, y por lo tanto.

$$HM_{1,1} \leftarrow 10$$
$$HM_{1,2} \leftarrow 15.4391$$

$$P^1 \leftarrow \begin{bmatrix} g & g & a & t & t & c & c & c & c & t & 5.2363 & 0.170 \\ a & - & c & c & c & - & g & c & g & - & 4.4513 & 0.141 \\ g & g & g & g & t & c & c & c & c & t & 8.338 & 0.270 \\ g & g & - & - & - & - & - & - & - & - & 5.5828 & 0.181 \\ g & g & - & a & t & - & c & t & t & t & 7.27 & 0.2355 \end{bmatrix}$$

$$HM = \begin{bmatrix} 10 & 15.4391 & 0.4382 \\ 11 & 12.046 & 0.5617 \end{bmatrix}$$

$v = 3$
Nueva matriz auxiliar:

$$Auxiliar = \begin{bmatrix} g & g & - & a & t & - & t & c & c & - & - & c & c & - & t & 3.3784 & 0.1306 \\ a & - & c & - & c & c & - & - & - & g & c & - & - & g & - & 4.3339 & 0.1676 \\ g & g & - & g & g & t & - & c & c & c & - & - & c & t & - & 6.489 & 0.2509 \\ g & - & g & - & - & - & - & - & - & - & - & - & - & - & - & 6.288 & 0.2432 \\ g & - & g & - & a & t & - & - & c & t & - & t & - & - & t & 5.3699 & 0.2076 \end{bmatrix}$$

El valor del alineamiento es de $V(Auxiliar) = 12.9296$ unidades
$V_{min} = \min(15.4391, 12.046) = 12.046$

En virtud de $V(Auxiliar) > V_{max}$ significa el valor de alineamiento propuesto es mejor que el peor elemento en memora armónica, y por lo tanto sustituye.

$$P^2 \leftarrow \begin{bmatrix} g & g & - & a & t & - & t & c & c & - & - & c & c & - & t & 3.3784 & 0.1306 \\ a & - & c & - & c & c & - & - & - & g & c & - & - & g & - & 4.3339 & 0.1676 \\ g & g & - & g & g & t & - & c & c & c & - & - & c & t & - & 6.489 & 0.2509 \\ g & - & g & - & - & - & - & - & - & - & - & - & - & - & - & 6.288 & 0.2432 \\ g & - & g & - & a & t & - & - & c & t & - & t & - & - & t & 5.3699 & 0.2076 \end{bmatrix}$$

$$HM = \begin{bmatrix} 10 & 15.4391 & 0.45 \\ 15 & 12.9296 & 0.54 \end{bmatrix}$$

3.1.2 Estructura de la estrategia propuesta y comparación con HS.

En esta sección se muestra la estructura del método desarrollado para resolver el AMS a través del pseudocódigo (dado que, en las secciones anteriores sólo se analizó de manera independiente a cada una de las subrutinas).

Algoritmo 30. Alineamiento de múltiples secuencias por búsqueda armónica.

Input: Conjunto de secuencias ($N = \{s_1, s_2, \dots, s_x\}$)

Output: Alineamiento múltiple del conjunto de secuencias.

/*Asignación de Valores a los parámetros de entrada del algoritmo*/

1. $O(1)$
2. $l_{max} = \max(l_1, l_2, \dots, l_x)$. $O(x)$
3. $HMS = [10]$. $O(1)$
4. $MaxImp = 5000$. $O(1)$
5. $HMCR = 0.9$. $O(1)$
6. $PAR = 0.3$. $O(1)$
7. $t_o = 10$. $O(1)$
8. $\alpha = 0.7$. $O(1)$
9. $b = 0.1(caracteres - l_{max})$ $O(1)$
10. Determinar la matriz W $O\left(\frac{n(n-1)}{2} * l^2\right)$

/*Construir la memoria armónica inicial*/

11. For $P = 1; P \leq HMS; ++$ $O\big((HMS(4+x^2 * l_M^2) + (2)\big)$
    i. ${l_M}^P = \text{int}\left(l_{max} + \left((caracteres - l_{max}) * rand\right)\right)$
    ii. Construir una matriz auxiliar P
    iii. Actualizar el valor de ${l_M}^P$
    iv. $HM_{P,1} \leftarrow {l_M}^P$
    v. $HM_{p,2} \leftarrow V(M^p)$

    End For.

b. $q = \sum_{p=1}^{HMS} HM_{p,2}$

c. For $P = 1; P \leq HMS; ++$
    i. $HM_{i,3} = \frac{q - HM_{i,2}}{q}$

    End For

/*Satisfacer el criterio de paro*/

12. For $v = 1; v \leq MaxImp; ++$ $O\left(MaxImp\big(x^2 * l_M^2 + (10 + 3HMS)\big)\right)$
    a. Generar una nueva armonía.
    b. Actualizar la memoria armónica
        i. If $valor\ en\ memoria \leq$ valor encontrado
        ii. Else

1. if $(rand \leq e^{\left(\frac{V(Auxiliar)-V_{min}}{\alpha * T}\right)})$
   end If
   End If
End For

Al analizar la complejidad de las subrutinas asociadas se obtiene lo siguiente:

$$O\left(n + 9 + (HMS(4+x^2) + MaxImp\left(x^2 * l_M^2 + (10 + 3HMS)\right)\right)$$

Por lo tanto la complejidad de la rutina anterior es $O(n^2 * l_M^2)$.

En la tabla nueve siguiente se muestran un comparativo entre las etapas del algoritmo HS en contraste con las etapas del nuevo algoritmo híbrido desarrollado. En dicha tabla se muestra los conceptos adaptados y creados del HS y RS para la manipulación y resolución del AMS.

Tabla 9. Comparación de HS vs. Estrategia desarrollada.

Búsqueda armónica	Estrategia desarrollada para el AMS
Inicializar proceso de optimización	Inicializar proceso de optimización.
Inicializar la memoria amónica	Crear memoria armónica inicial 1. Construir la submatriz M^P. 2. Construir la matriz D. 3. Construir la submatriz G^P 4. $P = [M^P, G^P]$.
Improvisar una nueva armonía	Improvisar una nueva armonía.
Actualizar memoria armónica (solo en caso de mejorar la solución encontrada)	Actualizar memoria armónica (existiendo la posibilidad de aceptar acordes de peor calidad)
Cumplir con criterio de paro	Cumplir con criterio de paro

3.2 Validación de la estrategia desarrollada.

La validación, evaluación y comparación de los métodos desarrollados para el AMS requiere un gran número de alineaciones de referencia exacta que se puede utilizar como casos de prueba (Thompson, Plewnlak, & Poch, 1999). Por lo anterior la prueba de comparación entre los alineamientos resultantes de la estrategia hibrida desarrollada versus los alineamientos referenciales (contenidos en BALIBASE versión 1) consistió en determinar el porcentaje de diferencia existente entre ellos s mediante el programa bali_score mediante la cuantificación del parámetro SSP y la técnica de datos históricos.

Con dicho propósito se implemento este algoritmo en el programa MATLAB versión 7.6.0 (Ra2008) y se examinaron los 141 casos referenciales contenidos en BALiBASE versión uno; realizando cada una de las pruebas por triplicado.

Los conjuntos posibles se generaron a partir de la información biológica disponible es decir que bases son hidrofobias e hidrofilias, el tipo de carga que posen etc. Los valores numéricos tanto para el peso de las penalizaciones como para la función de similitud se obtuvo de (Solanilla, Gómez Teshima, & Múnera, 2004)[19].

Tabla 10. Valor de la similitud entre los alineamientos alcanzados y referenciales cuantificando la métrica SSP.

Referencia	Características de las secuencias del conjunto		Promedio
1	Cortas	Menor 25%	0.196809524
		20-40%	0.373566667
		Mayor 35%	0.514733333
	Medianas	Menor 25%	0.126166667
		20-40%	0.275592593
		Mayor 35%	0.411166667
	Largas	Menor 25%	0.070541667
		20-40%	0.236055556
		Mayor 35%	0.260625
2	Cortas		0.228407407
	Medianas		0.14547619
	Largas		0.155904762
3	Cortas		0.17075
	Medianas		0.076333333
	Largas		0.108333333
4			0.013416667
5			0.072361111

Con base en la información anterior se puede observar que el mejor desempeño del la heurística desarrollada se da al trabajar en conjuntos pequeños de secuencias equidistantes (referencia 1). En contraste, cuando se trabaja con secuencias poco emparentadas y de tamaños variados su desempeño es poco satisfactorio, obsérvese las referencias 4 y 5 en las cuales se obtiene una cercanía al alineamiento referencia inferior al 10%.Con base en la información obtenida durante la etapa de validación se puede afirmar que el número de improvisaciones necesarias para el funcionamiento correcto del algoritmo es de 6417[20].

A continuación se muestra el análisis comparativo entre los resultados obtenidos por el algoritmo híbrido desarrollado con respecto a otros procedimientos para la

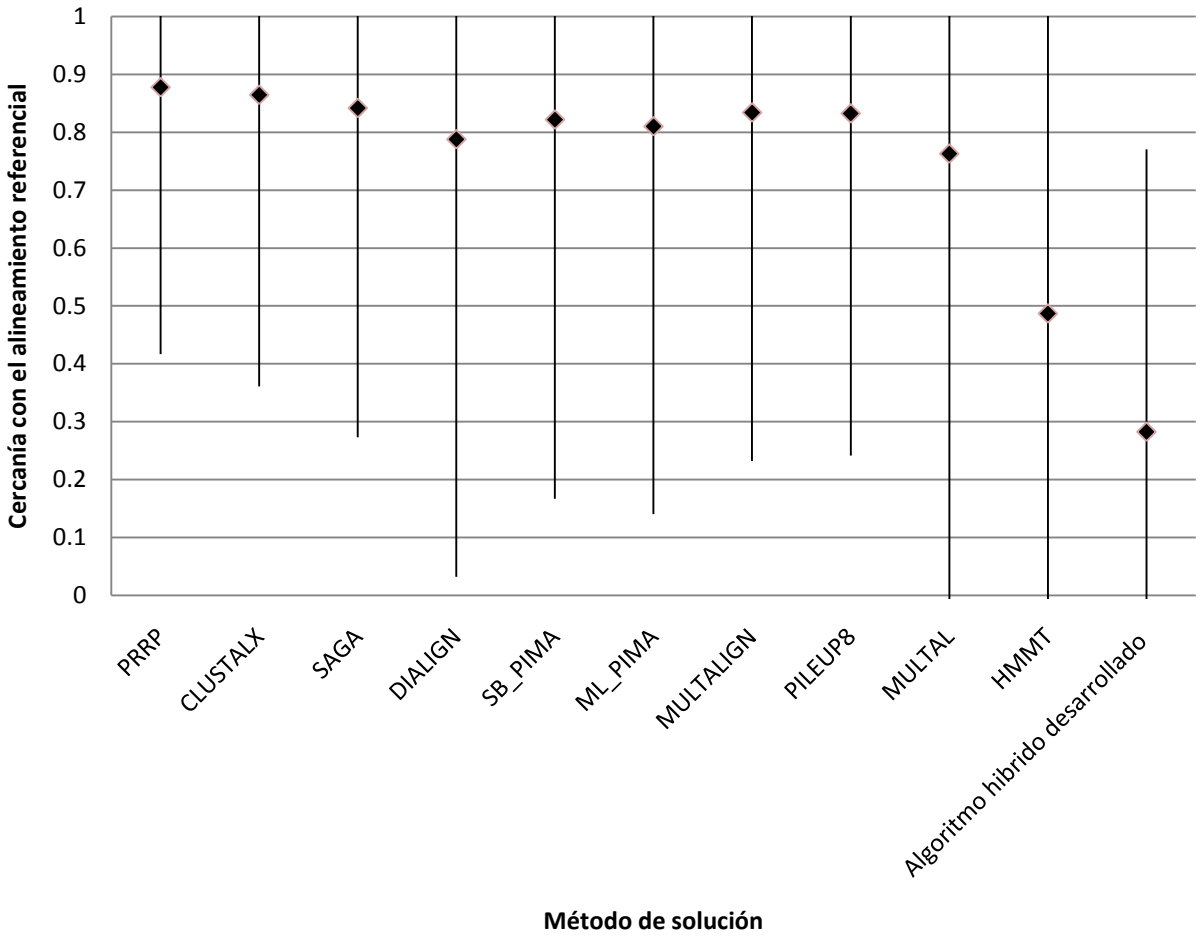
[19] Los datos de la experimentación se incluyen en el anexo B sección 1
[20] Los datos sobre la iteración se incluyen en anexo B sección 2

solución del AMS. Los resultados de los otros procedimientos fueron obtenidos de la página de BALiBASE.

3.2.1 Referencia 1

El comportamiento del algoritmo híbrido desarrollado en comparación con otros métodos de solución para el AMS para las secuencias de la referencia 1, en promedio ofrece una cercanía al alineamiento de referencial del 30%.[21]. El híbrido desarrollado da un rango de similitud que osilla entre 0 y 70%. Los resultados de similitud con respecto a los demás métodos de solución son inferiores solamente cercanos a los reportados por el modelo oculto de Markov (otro método iterativo).

Figura 32. Comportamiento de la búsqueda armónica con respecto a la referencia 1.



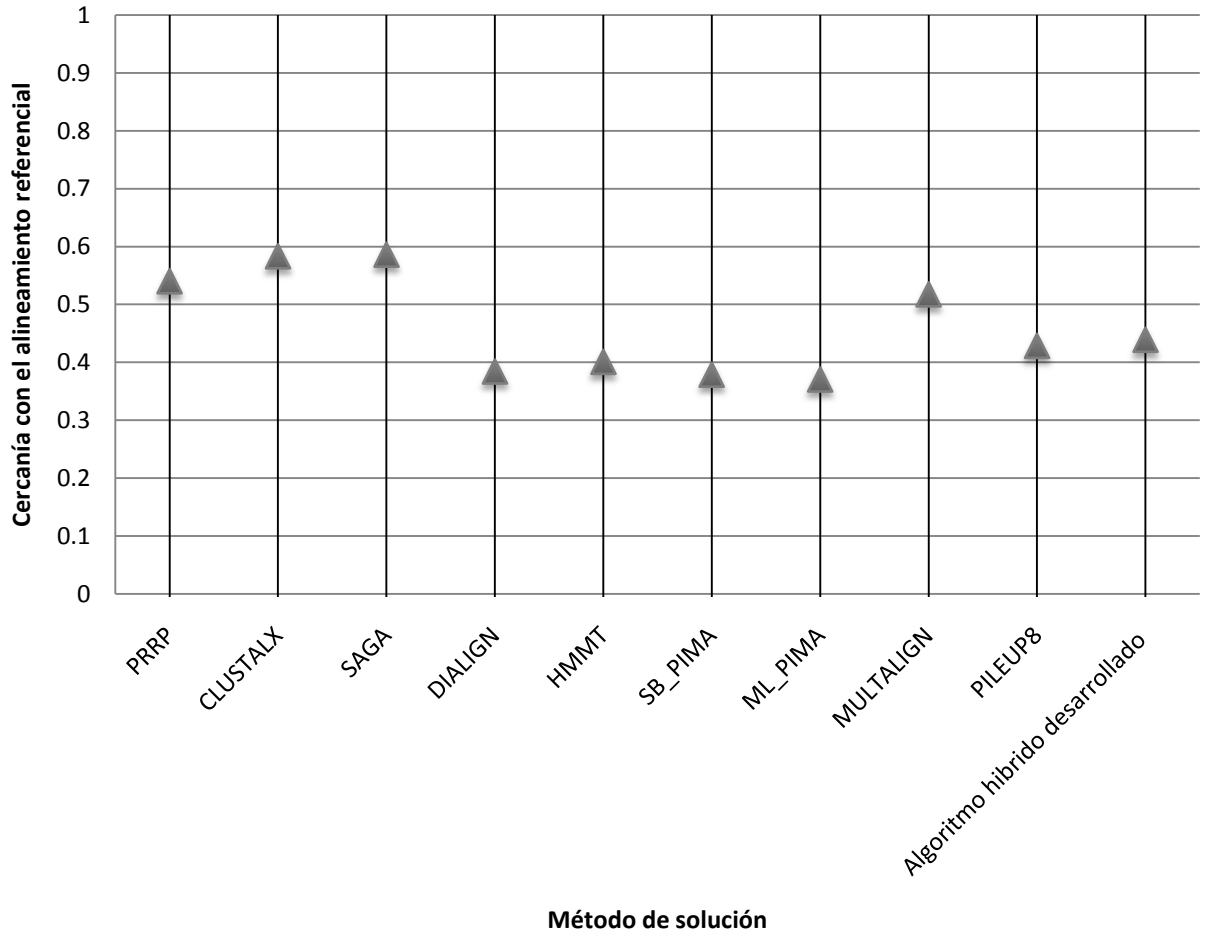
[21] Los datos se incluyen en el anexo C sección 1.

3.2.2 Referencia 2

El comportamiento del algoritmo híbrido desarrollado en comparación con otros métodos de solución para el AMS para las secuencias de la referencia es en promedio una mejor alternativa a los métodos DIALIGN, HMMT, SB_PIMA y ML_PIMAS, ya que estos ofreces una similitud promedio al alineamiento referencial inferior al que se obtiene con HS. Sin embargo, los resultados obtenidos por el híbrido son inferiores a los encontrados por SAGA, Clustal X y Multialing. Lo anterior se muestra en figura 33.

El rango de cercanía a los alineamientos referenciales que ofrece el algoritmo desarrollado osilla entre 0 al 100%. Los resultaos anterior pueden deberse a la naturaleza de los alineamientos referenciales, los cuales se forman por secuencias estrechamente relacionadas.

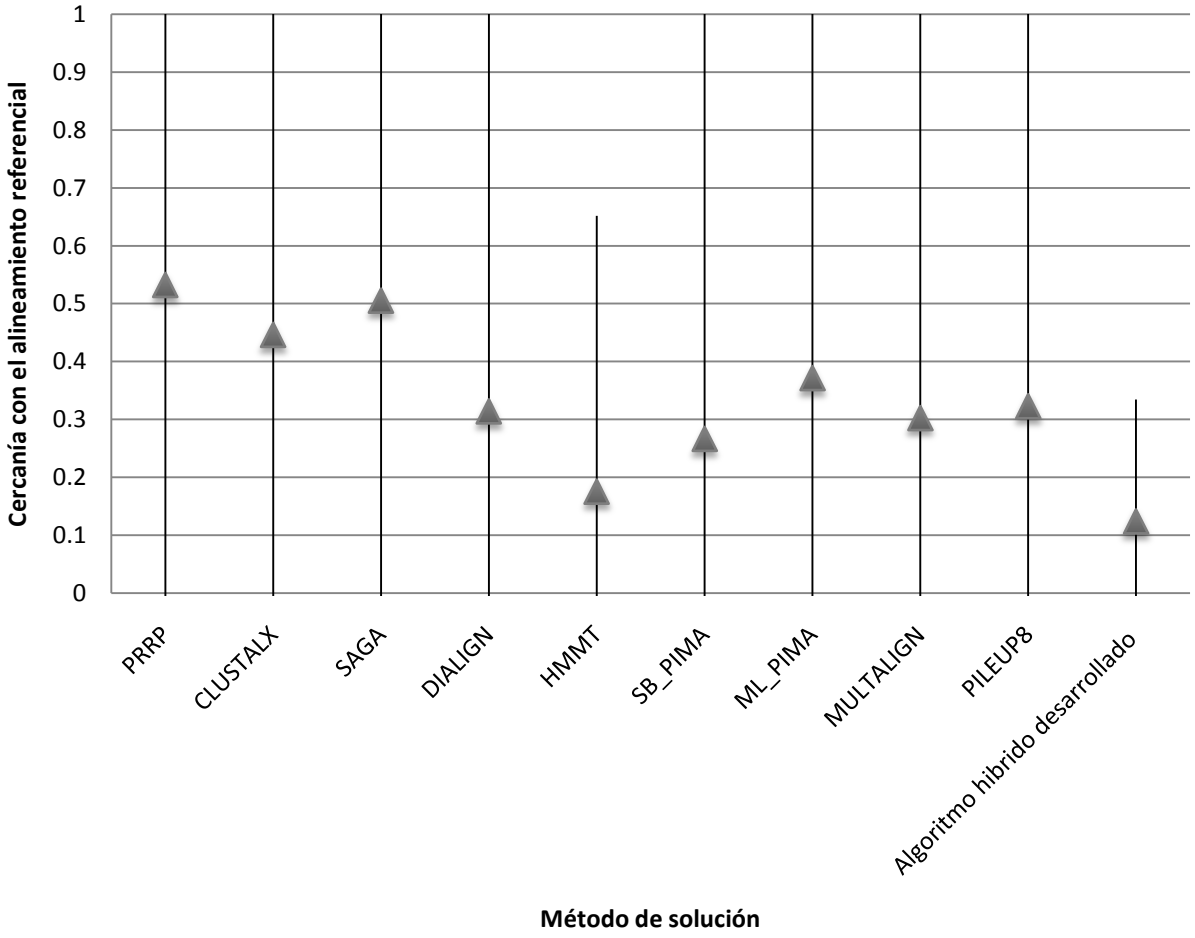
Figura 33. Comportamiento de la búsqueda armónica con respecto a la referencia 2.



3.2.3 Referencia 3

El comportamiento del algoritmo híbrido desarrollado en comparación con otros métodos de solución para el AMS para las secuencias de la referencia 3 se caracteriza por ser inferior que la mayoría de los demás métodos desarrollados para solucionar el AMS ya que en promedio ofrece una cercanía al alineamiento referencial del 10%, solamente cercano a los resultados obtenidos mediante el modelo oculto de Markov.

Figura 34. Comportamiento de la búsqueda armónica con respecto a la referencia 3



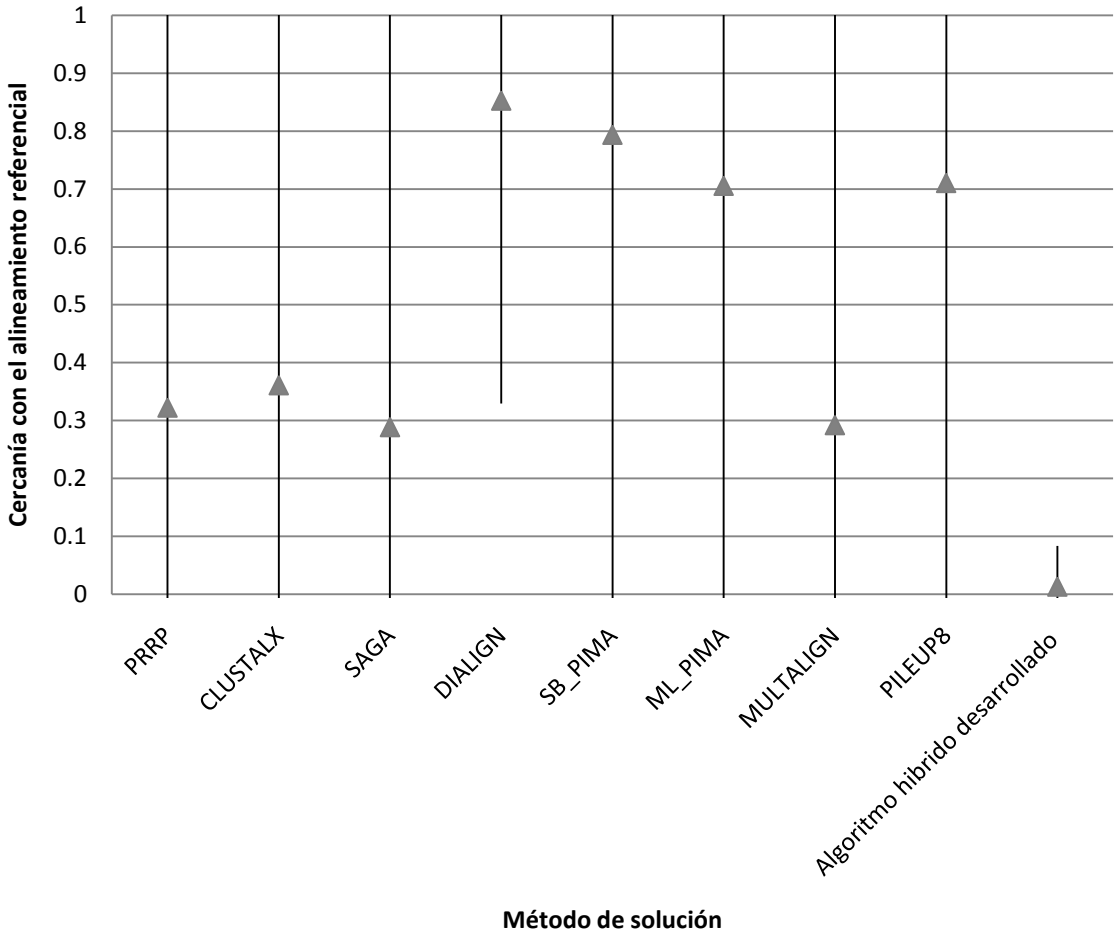
3.2.4 Referencia 4

El comportamiento del algoritmo híbrido desarrollado en comparación con otros métodos de solución para el AMS para las secuencias de la referencia 4 se caracteriza por ser poco satisfactoria ya que los demás métodos desarrollados para solucionar el AMS ofrecen un nivel de cercanía a los alineamientos referenciales

superior al obtenido por el híbrido que en promedio da un nivel de cercanía a los alineamientos referenciales inferior al 5%

.Este pobre nivel de cercanía puede atribuirse a que el manejo de sentencias poco emparentadas (de inserciones terminales de espacios vacios) realizado por el algoritmo híbrido no es el adecuado y por lo tanto esta debilidad es una oportunidad para mejorar el método desarrollado.

Figura 35. Comportamiento de la búsqueda armónica con respecto a la referencia 4



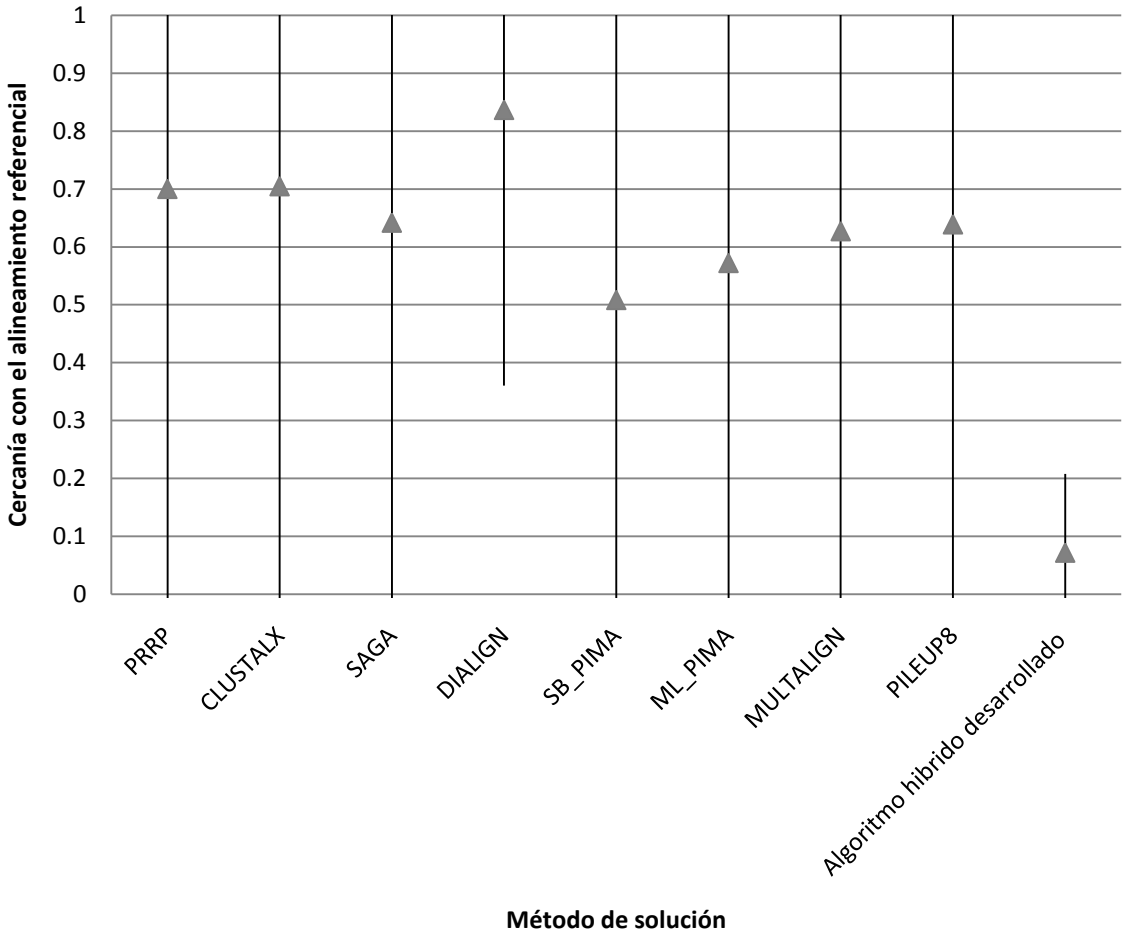
3.2.5 Referencia 5

El comportamiento del algoritmo híbrido desarrollado en comparación con otros métodos de solución para el AMS para las secuencias de la referencia 5 se caracteriza por ser poco satisfactoria ya que los demás métodos desarrollados para solucionar el AMS ofrecen un nivel de cercanía a los alineamientos referenciales

superior al obtenido por el híbrido que en promedio da un nivel de cercanía a los alineamientos referenciales inferior al 10%.

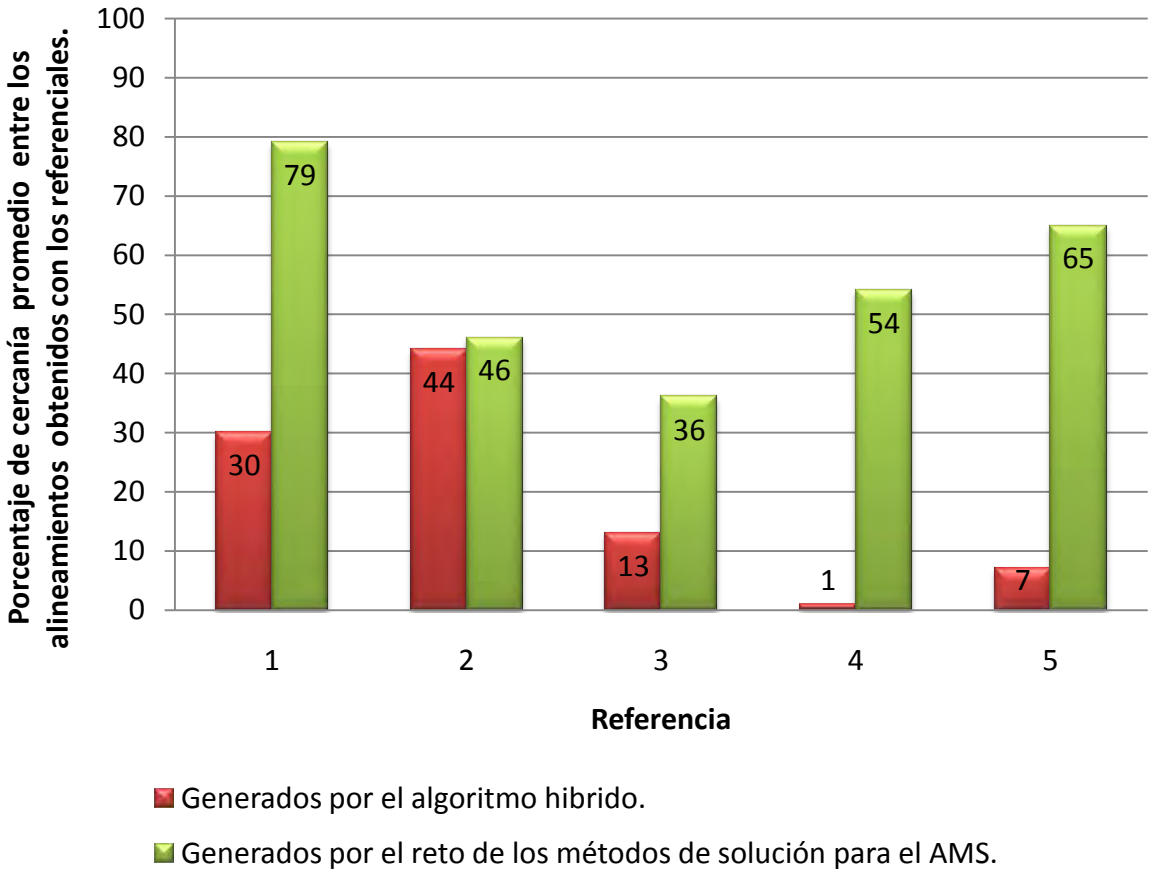
Este pobre nivel de cercanía puede atribuirse a que el manejo de sentencias poco emparentadas (de inserciones internas de espacios vacios) realizado por el algoritmo híbrido no es el adecuado y por lo tanto esta debilidad es una oportunidad para mejorar el método desarrollado.

Figura 36. Comportamiento de la búsqueda armónica con respecto a la referencia 5.



Con base a los resultados obtenidos de la validación del algoritmo híbrido desarrollado se puede concluir que ésta estrategia sólo es conveniente utilizar esta metodología cuando se trabaja con un conjunto de secuencias homologas, mientras que al analizar secuencias poco emparentadas resulta poco eficaz. En la figura siguiente se muestra una síntesis del nivel de cercanía obtenido por el híbrido desarrollado para cada una de las referencias de BALiBASE.

Figura 37. Comportamiento de algoritmo desarrollado para las referencias de BALiBASE.



En la grafica anterior se observa que el nivel de cercanía promedio obtenidos por el algoritmo híbrido para todas las referencias de BALiBASE se encuentra por debajo del promedio de similitud obtenido por el resto de los métodos de solución del AMS, sin embargo, son las referencias 4 y 5 con las que se consiguen peores resultados. Lo anterior puede deberse al método de generación de la matriz auxiliar P y a la estructura definida de la matriz de memoria armónica.

3.3 Análisis de resultados

Si bien en este trabajo se desarrollo un algoritmo híbrido para resolver el AMS, fundamentado en las metaheurísticas de RS y HS, también es cierto, que esta metodología debe ser mejorada con la finalidad de mejorar los resultados obtenidos de ella.

A continuación se caracteriza el algoritmo desarrollado a través de determinar la eficiencia y eficacia del mismo. En términos de eficiencia (complejidad temporal[22])

[22] Véase apéndice B.

el algoritmo desarrollado resulta atractivo dado a que su complejidad computacional es igual a los métodos Clustal (actualmente el procedimiento más utilizado para resolver el AMS) y T-COFFE. Adicionalmente que el número de secuencias que pueden ser alineados de manera simultánea por este procedimiento queda limitado por las características del programa computacional donde se implemente.

Tabla 11. Comparativo de la complejidad de los algoritmos para solucionar el AMS.

Tipo de algoritmo	Nombre del algoritmo	Número de secuencias que alinea simultáneamente	Complejidad
Exhaustivos.	Matriz de puntos	2	$O(l_a * l_b)$
	Distancia de Levenshtein	2	$O(l_a * l_b)$
	Distancia Indel	2	$O(l_a * l_b)$
	Distancia de Damerau	2	$O(l_a * l_b)$
	Needleman y Wunsch	2	$O(l_a * l_b)$
	Murata	3	$O(l_a * l_b * l_c)$
	Gotoh	2	$O(l_a * l_b)$
	Algoritmo de Gotoh – Altschul	2	$O(l_a * l_b)$
	Carrillo Lipman	Hasta 10	$O(l^n)$
	Smith y Waterman	2	$O(l_a * l_b)$
	Fredman	3	$O(l^3)$
Progresivo	Sankoff	5	$O(n_i(2l^n))$
	Sankoff y Cedergren	3	$O(n_i(2l^n))$
	Fitch	15	$O(n_i(2l^n))$
	Jonson and Doolitle	3 o mas	$O(n(l - l_r)l_r^{n-1})$
	Karlin	2 o más	$O(\lvert\mathcal{L}\rvert^2)$
	Waterman y Jones	2 o más	$O(n * {l_r}^2 * l * B)$
	Waterman y Perlwitz		$O(nl^2)$
	Barton y Sternberg		$O(n!)$
	Subbiah y Harrison		$O(n!)$
	Clustal		$O(n^2l^2)$
	T-Cofee		$O(n^2l^2) + O(n^3)$
Iterativos	Algoritmo A*		$O(n^l)$
	Algoritmo de Viterbi		$O(n^3P)$
	Algoritmo híbrido de HS y RS		$O(n^2 * l^2)$

Elaborado a partir de Chan, Wong y Chiu 1992; Ikeda y Imai 1999; Notredame 2000 y resultados del trabajo.

donde:
n_i: Número de nodos internos en el árbol filogenético.
l: Longitud de la secuenica más larga.
n: Número de secuencias
l_r: Longitud del residuo.
$\mathcal{L}$: Tamalo de la lista de regiones.
B: Valor de función deacuerdo a la naturaleza de las cadenas

$$B = \begin{cases} Cadenas\ de\ nucleotidos & 4^k \\ Cadenas\ proteicas & 20^k \end{cases}; \qquad (97).$$

En contraste en términos de eficacia el algoritmo desarrollado solamente resulta útil para el alineamiento de secuencias homologas y estrechamente relacionadas mientras que para alinear secuencias heterogéneas de longitud variable resulta una estrategia pobre. Lo anterior puede deberse al método de generación de la matriz auxiliar P y a la estructura definida de la matriz de memoria armónica.

Una oportunidad de mejora del procedimiento desarrollado radica en mejorar los procedimientos para el manejo de secuencias heterogéneas de tamaño variable.

3.4 Resumen.

En esta sección se expuso que el nuevo procedimiento híbrido para resolver el AMS trata de en utilizar el vigor híbrido resultado de la combinación de las metaheurísticas RS y la HS.

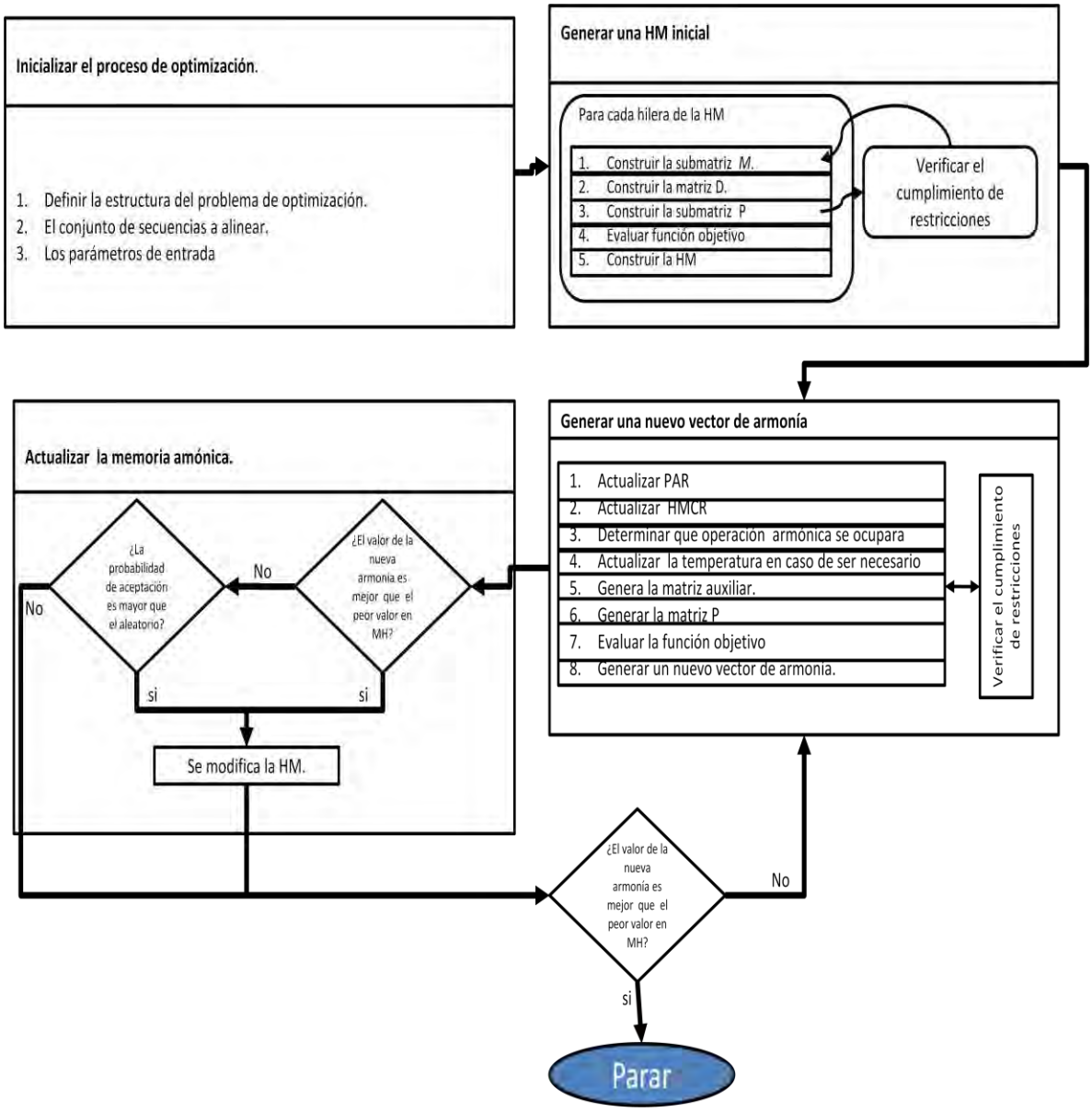
Definición 33. El **vigor híbrido** supone que al conjuntar dos o más métodos de solución (a los que se denomina padres) se potencializan las fortalezas de cada uno y a su vez se disminuyen sus debilidades.

La elección de los procedimientos padres de este híbrido se baso en que ambos métodos son: sencillos, adaptables, robustos, eficientes y eficaces. En el diseño del híbrido la estrategia HS fue utilizada como procedimiento rector, mientras que el RS se utilizo como un proceso de decisión sobre las soluciones que integran la matriz armónica que permite mantener un alto grado de diversificación en la HM.

Aunque se desarrolló un algoritmo híbrido para resolver el AMS, basado en las metaheurísticas de RS y HS (figura 38) también es cierto, que esta metodología debe ser perfeccionada con la finalidad de mejorar los resultados obtenidos de ella. En términos de eficiencia, el algoritmo desarrollado resulta atractivo dado que su complejidad computacional es igual a la de los métodos que actualmente son más utilizados para resolver el AMS, además que el número de secuencias que pueden ser alineados de manera simultánea por este procedimiento queda limitado por las características del programa computacional donde se implemente.

Sin embargo, en términos de eficacia el algoritmo desarrollado es pobre al alinear secuencias heterogéneas de longitud variable, en contraste resulta útil para el alineamiento de secuencias homólogas y estrechamente relacionadas. Lo anterior puede deberse al método de generación de la matriz auxiliar P y a la estructura definida de la matriz de memoria armónica

Figura 38. Algoritmo híbrido de HS y RS para solucionar el AMS.



En el siguiente capítulo se utilizara la información generada en esta sección con objeto de verificar el cumplimiento del objetivo de la tesis, además de identificar las ventajas y desventajas del híbrido desarrollado.

Capítulo 4 Conclusiones y trabajos futuros.

Como resultado del desarrollo de la tesis, fue generado un algoritmo híbrido para resolver el AMS, basado en las metaheurísticas de búsqueda de armonía y de recocido simulado, el cual fue validado mediante el procedimiento de comparación con datos referenciales propuesto por Julie Thompson en BALiBASE. Secundariamente, se crearon conceptos e ideas para la manipulación de secuencias como ejemplo, la matriz P que es un símil de la memoria de interpretación que un grupo musical tendría sobre la ejecución de una armonía determinada.

Sin embargo, este algoritmo híbrido debe ser corregido y perfeccionado con la finalidad de mejorar los resultados obtenidos de ella.

Las fortalezas del algoritmo radican en: su eficiencia computacional, a raíz que su complejidad ($O(n^2 * l^2)$) es similar a los métodos que actualmente son más utilizados para resolver el AMS, adicionalmente es un procedimiento útil para el alineamiento de secuencias homólogas y estrechamente relacionadas y también consigue alinear de manera simultánea a un número grande de secuencias en virtud de que este valor queda limitado por las características del programa computacional donde se implemente. En contraste las debilidades se encuentran en la pobre eficacia del algoritmo para alinear secuencias heterogéneas de longitud variable. Cabe mencionar, que este algoritmo sólo fue probado para aplicaciones biológicas del AMS.

Las debilidades de este procedimiento se causan por: el método de asignación de valores en la matriz P, el cual es totalmente aleatorio lo que conlleva a no considerar el grado de cercanía entre las secuencias del conjunto y la estructura considerada para la formación de la matriz armónica.

Por lo tanto las oportunidades de mejoras y futuros trabajos son:

Modificar la estructura de la matriz armónica asemejando la siguiente estructura.

$$HM = \left(\begin{array}{c|c} \{x_i, x_j, .., x_k\}^1 & f(x^1) \\ \{x_i, x_j, .., x_k\}^2 & f(x^2) \\ \vdots & \vdots \\ \{x_i, x_j, .., x_k\}^{HMS} & f(x^{HMS}) \end{array}\right)$$

En la que se considera la existencia de una semejanza del problema AMS como un problema de grafos donde cada nodo es equivalente a una cadena del conjunto, por lo tanto, el número de nodos es igual al número de secuencias del conjunto y el vector factible representa un árbol posible en el grafo. Esto conllevaría a la reformulación matemática del problema a la forma de un problema de redes.

Con base en lo anterior, la asignación de valores a la matriz P se realizaría adaptando algún procedimiento para el alineamiento pareado de las secuencias, donde el $i-$ ésimo renglón de la matriz P sea el resultado de alinear la $i-$ésima secuencia con la $(i-1)-$ésima secuencia del conjunto, utilizando para la primera secuencia una secuencia aleatoria del conjunto. Además debe de trabajarse en conjunto con especialistas con la finalidad de establecer una función de similitud más adecuada entre los elementos del alfabeto.

Por último, después del perfeccionamiento del algoritmo se considera importante realizar aplicaciones de este algoritmo para el AMS fuera del contexto biológico, como ejemplo podrían ser: lingüística, música y robótica.

Apéndice A Proceso evolutivo.

La evolución, es un proceso por medio del cual todos los organismos se han desarrollado a partir de otras formas de vida más antiguas (Otto & Towle, 1996), lo que quiere decir que todos los organismos actuales provienen de las células primigenias que poblaron la tierra y solo han sufrido alteraciones para adaptarse mejor a su medio.

Puede verse al proceso evolutivo como un problema de optimización natural, donde la función objetivo es maximizar la adaptabilidad de los organismos a su medio, sujeto a las restricciones que imponen el fenotipo, genotipo, disponibilidad de recursos (alimento, espacio etc.) y los cambios que se presenta en el ambiente para un periodo “t” de tiempo (sistema dinámico).

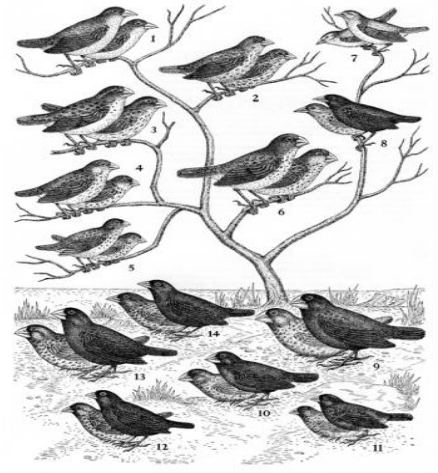
En forma análoga a los problemas de optimización puede existir solución factible, múltiple o no factible cada uno de los casos puede asociarse con alguna repercusión biológica. El primer caso se afilia a la continuidad de la especie. El segundo representa el surgimiento de nuevas variedades (y en ciertas circunstancias nuevas especies). Y el último se relacionaría a la extinción de las especies.

La diversidad entre los seres vivientes es casi infinita (no hay dos individuos exactamente iguales en todos sus detalles) y por otra parte, la variabilidad entre los entes biológicos vista como conjunto no es continua sino muestra lagunas de tamaño diverso. (Cronquist, 1978).

Pero, ¿Cómo ocurre la evolución? Este proceso surge en los organismos durante el proceso de sucesión de generaciones, donde ocurren cambios pequeños (insignificantes) entre padres e hijos en un periodo largo de tiempo, al final del periodo el resultado de la suma de los cambios minúsculos en el material genético da como resultado una nueva especie.

Las semejanzas y diferencias que se observan entre padres e hijos han interesado y desconcentrado a los hombres a través de la historia, es común identificar las similitudes entre individuos emparentados como lo manifiesta el proverbio “De tal palo tal astilla”, sin embargo, se reconoce también la diferencia entre hermanos. (Cronquist, 1978). Para aclarar lo anterior observe la figura siguiente.

Figura 39. Pinzones de Darwin



Fuente: http://dualjournal.wordpress.com/2008/01/22/los-pinzones-de-darwin/

Los pinzones de las Galápagos poseen una gran variedad de picos tanto en sus formas como en sus tamaños, cada uno adaptado a su dieta y estilo de vida particular. (R., 2008)

Dichas diferencias fueron causadas por selección natural, pero sus similitudes indican que descienden de un ancestro común.

Hay que recalcar que evolución biológica resulta de los cambios a través del tiempo en la constitución genética de las especies. A menudo, pero no siempre, los cambios genéticos producen cambios notables en la apariencia o en el comportamiento de los organismos. La evolución requiere tanto la producción de variación como la dispersión de variantes que reemplacen a otras. Por lo que resulta importante considerar los mecanismos de evolución (¿Como surge la variación?) y lo patrones evolutivos (¿Cómo se dispersa este cambio?).

Los mecanismos de la evolución son los procesos responsables de la evolución. Entre los que se pueden citar los siguientes: deriva genética, selección natural, especificación, mutación migración, combinación de códigos, a continuación se describe a cada uno de ellos:

4. **Deriva genética** es el cambio en la frecuencia de un gen (aumento o disminución) en las generaciones sucesivas de una población que resultan del azar. (Otto & Towle, 1996). La deriva genética tiende a reducir la heterogeneidad en el material genético por lo que forma poblaciones homocigóticas. El impacto de la deriva genética se manifiesta con mayor impacto en poblaciones pequeñas como lo demuestran los estudios de *Luigi Cavalli-Sforza* (Times, 1993)

5. **Selección natural** es la base de los cambios evolutivos. Es el proceso a través del cual, los organismos mejor adaptados desplazan a los menos adaptados mediante la acumulación lenta de cambios genéticos favorables en la población a lo largo de las generaciones. (Madrid, 2008). Es resultado de las presiones que ejercen las condiciones ambientales, sobre los individuos lo que favorece un rasgo particular el que aumenta en la siguiente

generación. El caracter sobre el cual actual la selección natural es la **eficiencia biológica** ya que los individuos más aptos tienen mayor probabilidad de sobrevivir hasta la edad reproductora y, por tanto, de dejar descendientes a las siguientes generaciones. (Madrid, 2008).

6. **Especificación** es la formación de una nueva especie a partir de otra ya existente . Una nueva especie se forma cuando ya no puede procrear con le especie general. (Otto & Towle, 1996).

7. **La mutación** es un cambio repentino en el material genético, el cual es permanente y trasmisible a las nuevas generaciones. Y puede ser producida por *errores de copia* en el material genético durante la división celular, por la exposición a radiación, agentes químicos o virus. En la teoría sintética, la mutación tiene el papel de generar diversidad genética sobre la cual actúa la selección natural, y también la deriva. Las mutaciones que afectan a la eficacia biológica del portador, y por tanto son objeto de la selección natural, pueden ser negativas o beneficiosa.

8. **La migración** es el movimiento de organismos de un lugar a otro, el cual permite el flujo de material genético.

9. **Recombinación genética** es el proceso mediante el cual la información genética se redistribuye por transposición de fragmentos de ADN entre dos cromosomas durante la meiosis, en otras palabras durante el proceso de reproducción el ADN de los progenitores se mezcla, y por lo tanto las características combinadas de los progenitores son heredados a los hijos. Reproducirse significa que, dados dos individuos seleccionados en función del grado de adaptabilidad, dichas características pasen a formar parte de la siguiente generación a partir de la mezcla de sus códigos genéticos. (Kuri Morales & Galaviz Casas, 2002).

Los patrones evolutivos son las formas en que un cambio en un individuo se manifiesta en toda una especie los cuales son:

- Radiación evolutiva: se presenta cuando existe un ancestro común para varias especies.
- Evolución convergente: se presenta cuando dos especies no relacionadas se vuelven parecidas en algunas características debido a que se adaptan a ambientes parecidos.
- Extinción de especies es la perdida de combinaciones genéticas.

Apéndice B Complejidad, orden y tipo de problema.

Un algoritmo es un conjunto de instrucciones paso a paso para solucionar un problema. (Ahuja, Magnanti, & Orlin, 1993). Es decir, un algoritmo es una lista completa de los pasos necesarios para realizar una tarea o cálculo (Kolman, Busy, & Ross, 1997).

Una algoritmo es correcto (exacto) si se obtiene de él una solución del problema para el que fue diseñado sin pérdida de ninguna de sus etapas fundamentales.

Los algoritmos se forman de operaciones elementales las cuales son:

- Asignación: Establecer un valor a alguna variable.
- Operaciones aritméticas: son operaciones básicas aritméticas (suma, resta, multiplicación y aritmética).
- Operaciones lógicas: Es la comparación entre dos números.

La complejidad algorítmica es una métrica teórica para estimar e costo en tiempo y/o espacio de algoritmo. La complejidad temporal de un algoritmo se define como el tiempo que emplea dicho algoritmo en ejecutarse dados unos datos de entrada, ellos se basan en el número de operaciones elementales necesarias para realizar la tarea. A continuación se muestra el costo de las instrucciones más relevante en los algoritmos como Número de pasos (Martínez Gil & Quetglás, 2003):

- Costo de constantes, comentarios y declaración de variables: 0 pasos.
- Costo de expresiones y asignaciones: Para casa una de las expresiones se requiere 1 paso, así que el costo de las expresiones utilizadas será la suma de las operaciones necesarias; sin embargo si se llama a una función se aplicará el costo de llamada.
- Sentencias condicionales (IF, Else) será el costo en pasos de la suma de las expresiones para uno u otro caso.
- Sentencias iterativas simples (for While). Tendrán el costo de los pasos multiplicado por el Número de ciclos necesarios.
- Sentencias condicionales múltiples (switch) será el número de comparaciones que debe hacer al buscar, sentencias por sentencia.

Los pasos no son unidades de tiempo real ya que estas varían dependiendo del programa, maquina y experiencia, se refieren al número de operaciones básicas necesarias para realizar una tarea.

La complejidad de espacio es una métrica que se centra en tratar de determinar la cantidad de memoria necesaria para ejecutar un algoritmo hasta llegar a la compleción de una tarea. El avance tecnológico proporciona hoy en día un incremento sustancial en la cantidad de memoria disponible, por lo cual generalmente el análisis de los algoritmos se centra fundamentalmente en

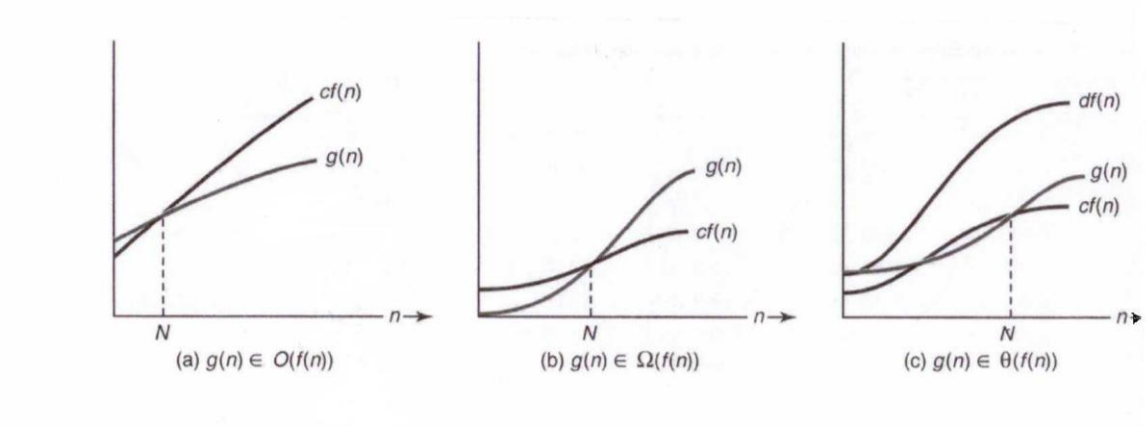
determinar el costo temporal de los procedimientos. (Jayanes Aguilar & Zahonero Martínez, 2004).

El análisis de la eficiencia temporal de los algoritmos consta de dos fases:

1. El análisis a posteriori ofrece una medida real, consistente en medir el tiempo de ejecución del algoritmo para unos valores de entrada dados y en un ordenador concreto.
2. El análisis a priori proporciona una medida teórica, que consiste en obtener una función de acotamiento (por arriba y/o por abajo) del tiempo de ejecución, para unos valores de entrada dados. Esta medida ofrece estimaciones del comportamiento de los algoritmos de forma independiente del ordenador .

El orden de los algoritmos es una función (f(n)) que acota asintóticamente, ya sea superior (O grande), inferior (Ω) o superior e inferiormente (Θ) a la función estimada de complejidad temporal de un algoritmo (g(n)), para una entrada determinada a partir de una n_0.. Observe la figura siguiente.

Figura 40. Cotas asintóticas de una función.



Generalmente se utiliza la notación O grande para referirse al orden de un algoritmo, la función f(n) es asintóticamente superior a la función de complejidad temporal g(n).

$$O\big(f(n)\big) = \{g(n):\ existe\ si\ y\ sólo\ si\ existe\ una\ cosntante\ \text{"c"}\ \text{y una}\ n_0\ tal\ que\}$$
$$0 \le g(n) \le cf(n) \quad \forall\, n \ge n_0$$

Propiedades de las funciones O grande.

Transitividad.

$$g(n) = O\big(f(n)\big) \quad y \quad f(n) = O\big(h(n)\big)\ \ implica \quad g(n) = O\big(h(n)\big)$$

Reflexividad.

$$g(n) = O\big(g(n)\big)$$

Figura 41. Crecimiento de funciones de complejidad computacional con relación al tamaño de la entrada.

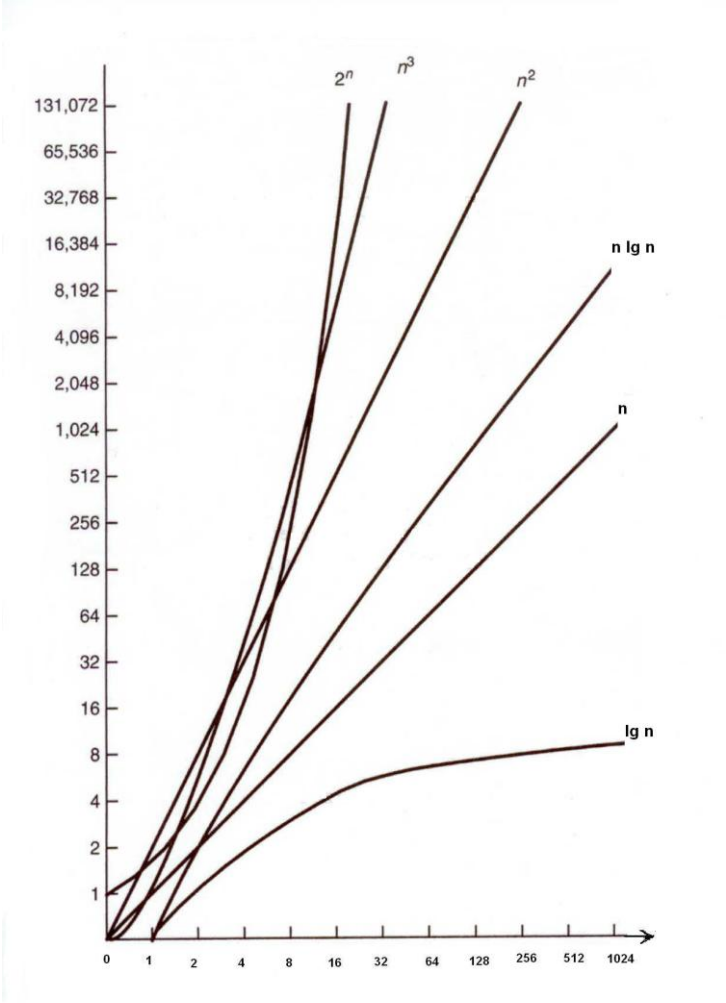


Tabla 12. Iteraciones del orden de algoritmos

Tamaño de n	O(1)	$O(log\ n)$	$O(n)$	$O(nlogn)$	$O(n^2)$	$O(n^3)$	$O(n^{100})$	$O(2^n)$	$O(n!)$
n=1	1	0	1	0	1	1	1	2	1
n=5	1	2.32192809	5	11.6096405	25	125	9765625	32	120
n=10	1	3.32192809	10	33.2192809	100	1000	1E+10	1024	3628800
n=50	1	5.64385619	50	282.192809	2500	125000	9.7656E+16	1.1259E+15	3.0414E+64
n=100	1	6.64385619	100	664.385619	10000	1000000	1E+20	1.2677E+30	9.333E+157

Clasificación de problemas.

Los problemas pueden clasificarse de acuerdo a su grado de complejidad en (Flores de la Mota, 1980):

1. **Indecidibles:** Denota una colección de problemas para los cuales no existe ningún algoritmo (polinomial, exponencial) para resolverlos.

2. **Intratables:** Denota una colección de problemas de problemas para los cuales solo se conocen algoritmos exponenciales para resolverlos.

3. **NP:** Esta clase incluye a los problemas que se pueden resolver en tiempo polinomial, siempre y cuando se adivine correctamente la ruta computacional. Es decir, denota la colección de todos los problemas de decisión para los cuales se tienen algoritmos de solución no-determinísticos en tiempo polinomial. (Castañeda Roldán, 2000).

4. **P :** Son aquellos problemas para los cuales existe un algoritmo polinomial para resolverlos. En otras palabras, Denota una colección de todos los problemas de decisión que pueden ser resueltos por medio de un algoritmo determinístico en tiempo polinomial (Castañeda Roldán, 2000)

En la definición del los problemas NP se incluye el concepto de **algoritmo no determinístico** que es una idea teórica que representa a un algoritmo con dos fases: suponer y comprobar y que además, implica siempre hacer una suposición correcta. Si al utilizar un algoritmo no determinístico para resolver un problema X, durante la fase de comprobación la complejidad temporal puede aproximarse de forma polinomial, entonces dicho algoritmo es polinomial no determinístico y por tanto el problema X es polinomial no determinístico (NP). (Lee, Tseng, Chang, & Tsai, 2007) . En otras palabras, un problema de decisión pertenece a NP si puede ser resuelto en una Máquina de Turing indeterminista en tiempo polinómico.

El teorema de Cook fue la primera prueba de existencia de problemas NP-completos. Este teorema establece que el problema SAT es NP-Completo.

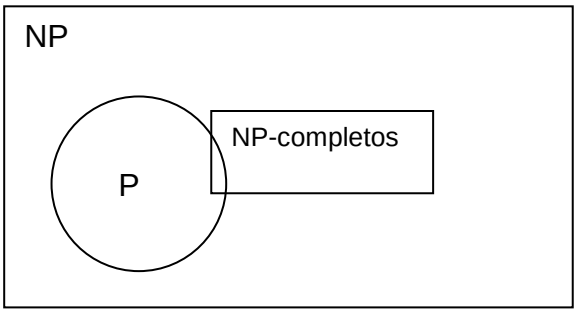
Definición 34. El **Teorema de Cook** establece que $NP = P$ si y sólo si el problema de satifactibilidad es un P. (Lee, Tseng, Chang, & Tsai, 2007)

La demostración intuitiva de él teorema consta de dos partes: A) “Si NP=P, entonces el problema de satifactibilidad (SAT) es un P” B) “ Si el problema SAT es un P, entonces NP=P” . (Lee, Tseng, Chang, & Tsai, 2007). En esencia el teorema de Cook indica que si el SAT puede resolverse con un algoritmo polinomial entonces todo problema NP puede ser resuelto en un número polinomial de pasos.

Definición 35. Un problemas X es **NP-completos** completo si $X \in NP$ y todo problema NP se reduce a X. (Lee, Tseng, Chang, & Tsai, 2007)

Los problemas $P \subseteq NP$, pero además los problemas $NP - completos \subset NP$ como se muestra en la siguiente figura:

Figura 42. Conjunto de problemas NP.



Fuente: (Lee, Tseng, Chang, & Tsai, 2007)

Mientras los problemas se clasifican de acuerdo al grado de complejidad, los algoritmos se clasifican en exactos y aproximados. (Vélez & Montoya, 2007)

1. **Exactos o completos:** Son aquellos con los que se garantiza encontrar la solución exacta a un problema después de un número finito de pasos.

2. **Aproximados o heurísticos:** Procedimientos simples, a menudo basados en el sentido común, que se supone ofrecen una buena solución (aunque no necesariamente la óptima) a problemas difíciles de un modo fácil y rápido. (De Alba Romenus, 2004). Es decir son algoritmos o estrategias para solucionar problemas complejos con tiempos de ejecución buenos y que arrojan buenas soluciones.

Apéndice C Bases de datos biológicas.

Una base de datos biológica es una biblioteca de información sobre ciencias de la vida, recogida de experimentos científicos, literatura publicada, tecnología de experimentación de alto rendimiento, y análisis computacional. Contiene información de áreas de investigación incluyendo genómica, proteómica, metabolómica, expresión génica mediante micro arreglos, y filogenética.La información contenida en bases de datos biológicas incluye funciones, estructura y localización (tanto celular como cromosómica) de genes, efectos clínicos de mutaciones, así como similitudes de secuencias y estructuras biológicas.

Para entender las bases de datos biológicas son importantes los conceptos de bases de datos relacionales de las ciencias de la computación, y los conceptos de recuperación de información de las bibliotecas digitales. El diseño de estas bases de datos, su desarrollo y su gestión a largo plazo, forman un área nuclear de la disciplina de la bioinformáticaEl contenido de los datos incluye secuencias génicas, descripciones textuales, atributos y clasificaciones ontológicas, anotaciones, y datos en forma tabular. Estos son descritos a menudo como datos semi-estructurados, y se pueden representar como tablas, registros delimitados por claves, y estructuras XML. Son comunes las referencias cruzadas entre bases de datos usando números de acceso (identificadores únicos de registros de secuencias de proteínas o ADN)

Uno de los tipos de bases de datos más usuales en bioinformática, son las bases de secuencias. Estas son una gran colección de secuencias de ADN, ARN y proteínas, que son almacenadas en computadoras. Una base de datos puede incluir secuencias de un sólo organismo, como ejemplo esta la base de datos que contiene las secuencias del organismo *Saccharomyces cerevisiae*.

Existen bases de datos primarias, que contienen información directa de la secuencia, estructura o patrón de expresión de ADN o proteína, y secundarias que contienen datos e hipótesis derivados del análisis de las bases de datos primarias, como mutaciones, relaciones evolutivas, agrupación por familias o funciones, implicación en enfermedades, etc.

Un problema fundamental en todas las bases de datos genómicas es que los registros provienen de una gran variedad de fuentes, desde investigadores individuales hasta grandes centros de investigación. Como resultado, las secuencias mismas y principalmente las anotaciones biológicas adjuntas a estas secuencias, varían notablemente en calidad. También hay mucha redundancia ya que muchos laboratorios ingresan a menudo secuencias que son idénticas o muy similares a otras en la base de datos.

Muchas anotaciones no están basadas en experimentos de laboratorio sino en resultados de búsquedas de secuencias similares de secuencias previamente anotadas. Por supuesto, una vez que una secuencia es anotada basándose en su

similitud con otra, puede servir como base para futuras anotaciones. Esto conduce al problema de las anotaciones transitivas, porque puede haber varias de esas secuencias transferidas por similitud de secuencia entre una base de datos de registro real y la información experimental de laboratorio. Por lo tanto, siempre hay que analizar el sentido biológico de las anotaciones en las principales bases de datos de secuencias con un considerable grado de escepticismo, a menos que pueda ser verificada por referencias a artículos publicados con la descripción de la alta calidad de los datos experimentales, o al menos por referencia a una secuencia de la base de datos arreglada por un humano.

Anexo A Alfabetos de biosecuencias

Tabla 13. Alfabeto de nucleótidos.

Símbolo	Significado	Explicación
G	G	Guanina
A	A	Adenina
T	T	Tiamina
C	C	Citosina
U	U	Uraxilo
Y	C o T	Primidina
R	A	Purina
M	A o C	Amino
K	G o T	Keto
S	C o G	Interacción fuerte
W	A o T	Interacción débil
H	A, C o T	H sigue a G en el alfabeto
B	C, G o T	B sigue a A en el alfabeto
V	A, C o G no T (no U)	V sigue a U en el alfabeto
D	A, G o T no C	D sigue a C en el alfabeto
N	A, C, G, U o T	Cualquier base

Fuente: (Solanilla, Gómez Teshima, & Múnera, 2004)

Tabla 14. Alfabeto de aminoácidos.

Código		Aminoácido
Simplificado	Extendido	
A	Ala	Alanina
C	Cys	Cysteína
D	Asp	Ácido aspártico
E	Glu	Ácido glutámico
F	Phe	Phenylalanina
G	Gly	Glicina
H	His	Histidina
I	Ile	Isoleucina
K	Lys	Lysina

Anexo B Resultados de las pruebas de validación BALiBASE versión 1

B.1. Estimación de la función SSP

	Referencia	Tipo	Porcentaje medio de identidad	Archivo	SPS			
					muestra 1	muestra 2	muestra 3	Promedio
1	Referencia 1	Cortas de <25% de identidad	15	1aboA	0.051	0.149	0.113	0.1043333
2	Referencia 1	Cortas de <25% de identidad	14	1idy	0.319	0.415	0.377	0.3703333
3	Referencia 1	Cortas de <25% de identidad	13	1r69	0.28	0.105	0.158	0.181
4	Referencia 1	Cortas de <25% de identidad	18	1tvxA	0.064	0.106	0.091	0.087
5	Referencia 1	Cortas de <25% de identidad	18	1ubi	0.202	0.139	0.186	0.1756667
6	Referencia 1	Cortas de <25% de identidad	17	1wit	0.333	0.321	0.282	0.312
7	Referencia 1	Cortas de <25% de identidad	17	2trx	0.146	0.107	0.189	0.1473333
8	Referencia 1	Cortas de 20-40% de identidad	30	1aab	0.27	0.22	0.56	0.35
9	Referencia 1	Cortas de 20-40% de identidad	28	1fjlA	0.485	0.557	0.477	0.5063333
10	Referencia 1	Cortas de 20-40% de identidad	31	1hfh	0.239	0.209	0.239	0.229
11	Referencia 1	Cortas de 20-40% de identidad	33	1hpi	0.482	0.387	0.362	0.4103333
12	Referencia 1	Cortas de 20-40% de identidad	30	1cys	0.493	0.515	0.488	0.4986667
13	Referencia 1	Cortas de 20-40% de identidad	28	1pfc	0.511	0.55	0.471	0.5106667
14	Referencia 1	Cortas de 20-40% de identidad	31	1tgxA	0.388	0.394	0.401	0.3943333
15	Referencia 1	Cortas de 20-40% de identidad	29	1ycc	0.159	0.157	0.147	0.1543333
16	Referencia 1	Cortas de 20-40% de identidad	31	3cyr	0.366	0.194	0.273	0.2776667
17	Referencia 1	Cortas de 20-40% de identidad	27	451c	0.368	0.468	0.377	0.4043333
18	Referencia 1	Cortas de >35% de identidad	44	1aho	0.5	0.435	0.537	0.4906667
19	Referencia 1	Cortas de >35% de identidad	51	1csp	0.343	0.33	0.452	0.375
20	Referencia 1	Cortas de >35% de identidad	46	1dox	0.265	0.27	0.276	0.2703333
21	Referencia 1	Cortas de >35% de identidad	44	1fkj	0.498	0.441	0.527	0.4886667
22	Referencia 1	Cortas de >35% de identidad	49	1fmb	0.608	0.66	0.674	0.6473333
23	Referencia 1	Cortas de >35% de identidad	45	1krn	0.698	0.78	0.722	0.7333333
24	Referencia 1	Cortas de >35% de identidad	46	1plc	0.501	0.589	0.473	0.521

25	Referencia 1	Cortas de >35% de identidad	51	2fxb	0.544	0.52	0.457	0.507
26	Referencia 1	Cortas de >35% de identidad	45	2mhr	0.55	0.629	0.451	0.5433333 3
27	Referencia 1	Cortas de >35% de identidad	57	9rnt	0.536	0.586	0.59	0.5706666 7
28	Referencia 1	Medianas, <25% de identidad	14	1bbt3	0.121	0.195	0.174	0.1633333 3
29	Referencia 1	Medianas, <25% de identidad	19	1sbp	0.074	0.109	0.11	0.0976666 7
30	Referencia 1	Medianas, <25% de identidad	15	1havA	0.063	0.084	0.093	0.08
31	Referencia 1	Medianas, <25% de identidad	17	1uky	0.048	0.069	0.072	0.063
32	Referencia 1	Medianas, <25% de identidad	18	2hsdA	0.121	0.15	0.113	0.128
33	Referencia 1	Medianas, <25% de identidad	18	2pia	0.03	0.057	0.023	0.0366666 7
34	Referencia 1	Medianas, <25% de identidad	17	3grs	0.185	0.274	0.193	0.2173333 3
35	Referencia 1	Medianas, <25% de identidad	20	kinase	0.198	0.246	0.226	0.2233333 3
36	Referencia 1	Medianas, 20-40% identidad	31	1ad2	0.262	0.277	0.277	0.272
37	Referencia 1	Medianas, 20-40% identidad	32	1aym3	0.424	0.416	0.352	0.3973333 3
38	Referencia 1	Medianas, 20-40% identidad	30	1gdoA	0.197	0.194	0.191	0.194
39	Referencia 1	Medianas, 20-40% identidad	27	1ldg	0.282	0.334	0.367	0.3276666 7
40	Referencia 1	Medianas, 20-40% identidad	33	1mrj	0.376	0.307	0.31	0.331
41	Referencia 1	Medianas, 20-40% identidad	26	1pgtA	0.271	0.298	0.253	0.274
42	Referencia 1	Medianas, 20-40% identidad	33	1pii	0.206	0.241	0.218	0.2216666 7
43	Referencia 1	Medianas, 20-40% identidad	30	1ton	0.268	0.262	0.313	0.281
44	Referencia 1	Medianas, 20-40% identidad	26	2cba	0.119	0.26	0.166	0.1816666 7
45	Referencia 1	Medianas, >35% de identidad	49	1amk	0.447	0.346	0.403	0.3986666 7
46	Referencia 1	Medianas, >35% de identidad	43	1ar5A	0.428	0.421	0.413	0.4206666 7
47	Referencia 1	Medianas, >35% de identidad	60	1ezm	0.448	0.456	0.463	0.4556666 7
48	Referencia 1	Medianas, >35% de identidad	43	1led	0.404	0.421	0.4	0.4083333 3
49	Referencia 1	Medianas, >35% de identidad	46	1ppn	0.477	0.504	0.47	0.4836666 7
50	Referencia 1	Medianas, >35% de identidad	44	1pysA	0.248	0.262	0.355	0.2883333 3
51	Referencia 1	Medianas, >35% de identidad	49	1thm	0.481	0.505	0.501	0.4956666 7
52	Referencia 1	Medianas, >35% de identidad	50	1tis	0.252	0.284	0.283	0.273
53	Referencia 1	Medianas, >35% de identidad	42	1zin	0.565	0.541	0.569	0.5583333 3
54	Referencia 1	Medianas, >35% de identidad	43	5ptp	0.325	0.368	0.295	0.3293333 3
55	Referencia 1	Largas, <25% de identidad	15	1ajsA	0.101	0.112	0.113	0.1086666 7
56	Referencia 1	Largas, <25% de identidad	20	1cpt	0.118	0.078	0.095	0.097

57	Referencia 1	Largas, <25% de identidad	19	1lvl	0.067	0.071	0.093	0.077
58	Referencia 1	Largas, <25% de identidad	18	1pamA	0.012	0.016	0.008	0.012
59	Referencia 1	Largas, <25% de identidad	22	1ped	0.048	0.138	0.095	0.0936666 7
60	Referencia 1	Largas, <25% de identidad	16	2myr	0.02	0.033	0.05	0.0343333 3
61	Referencia 1	Largas, <25% de identidad	19	4enl	0.057	0.067	0.067	0.0636666 7
62	Referencia 1	Largas, <25% de identidad	14	gal4	0.069	0.083	0.082	0.078
63	Referencia 1	Largas, 20-40% de identidad	29	1ac5	0.061	0.056	0.084	0.067
64	Referencia 1	Largas, 20-40% de identidad	35	1adj	0.513	0.414	0.417	0.448
65	Referencia 1	Largas, 20-40% de identidad	31	1bgl	0.181	0.121	0.153	0.1516666 7
66	Referencia 1	Largas, 20-40% de identidad	33	1dlc	0.188	0.253	0.164	0.2016666 7
67	Referencia 1	Largas, 20-40% de identidad	30	1eft	0.199	0.27	0.247	0.2386666 7
68	Referencia 1	Largas, 20-40% de identidad	34	1fieA	0.4	0.386	0.465	0.417
69	Referencia 1	Largas, 20-40% de identidad	31	1gowA	0.179	0.278	0.314	0.257
70	Referencia 1	Largas, 20-40% de identidad	34	1pkm	0.371	0.324	0.292	0.329
71	Referencia 1	Largas, 20-40% de identidad	33	1sesA	0.151	0.252	0.183	0.1953333 3
72	Referencia 1	Largas, 20-40% de identidad	28	2ack	0.132	0.198	0.112	0.1473333 3
73	Referencia 1	Largas, 20-40% de identidad	29	arp	0.136	0.107	0.087	0.11
74	Referencia 1	Largas, 20-40% de identidad	31	glg	0.282	0.251	0.277	0.27
75	Referencia 1	Largas,>35% de identidad	47	1ad3	0.201	0.236	0.276	0.2376666 7
76	Referencia 1	Largas,>35% de identidad	47	1gpb	0.257	0.267	0.268	0.264
77	Referencia 1	Largas,>35% de identidad	42	1gtr	0.32	0.279	0.303	0.3006666 7
78	Referencia 1	Largas,>35% de identidad	49	1lcf	0.195	0.187	0.176	0.186
79	Referencia 1	Largas,>35% de identidad	42	1rthA	0.416	0.383	0.475	0.4246666 7
80	Referencia 1	Largas,>35% de identidad	40	1taq	0.119	0.124	0.095	0.1126666 7
81	Referencia 1	Largas,>35% de identidad	51	3pmg	0.311	0.293	0.315	0.3063333 3
82	Referencia 1	Largas,>35% de identidad	45	actin	0.25	0.249	0.26	0.253
83	Referencia 2	Corta	28	1aboA	0.089	0.069	0.089	0.0823333 3
84	Referencia 2	Corta	28	1cys	0.111	0.084	0.078	0.091
85	Referencia 2	Corta	28	1idy	0.532	0.548	0.576	0.552
86	Referencia 2	Corta	26	1r69	0.081	0.09	0.09	0.087
87	Referencia 2	Corta	20	1tgxA	0.237	0.228	0.24	0.235
88	Referencia 2	Corta	19	1tvxA	0.088	0.085	0.091	0.088

89	Referencia 2	Corta	17	1ubi	0.418	0.353	0.344	0.37166667
90	Referencia 2	Corta	22	1wit	0.147	0.167	0.213	0.17566667
91	Referencia 2	Corta	36	2trx	0.367	0.368	0.384	0.373
92	Referencia 2	medianas	24	1sbp	0.075	0.086	0.084	0.08166667
93	Referencia 2	medianas	31	1havA	0.138	0.151	0.145	0.14466667
94	Referencia 2	medianas	31	1uky	0.086	0.076	0.093	0.085
95	Referencia 2	medianas	28	2hsdA	0.143	0.169	0.152	0.15466667
96	Referencia 2	medianas	31	2pia	0.099	0.12	0.119	0.11266667
97	Referencia 2	medianas	28	3grs	0.251	0.294	0.206	0.25033333
98	Referencia 2	medianas	32	kinase	0.189	0.164	0.215	0.18933333
99	Referencia 2	largas	35	1ajsA	0.174	0.153	0.18	0.169
100	Referencia 2	largas	29	1cpt	0.071	0.105	0.104	0.09333333
101	Referencia 2	largas	30	1lvl	0.096	0.109	0.137	0.114
102	Referencia 2	largas	35	1pamA	0.121	0.11	0.123	0.118
103	Referencia 2	largas	44	1ped	0.269	0.282	0.235	0.262
104	Referencia 2	largas	32	2myr	0.089	0.11	0.112	0.10366667
105	Referencia 2	largas	48	4enl	0.22	0.222	0.252	0.23133333
106	Referencia 3	cortas	19	1idy	0.22	0.248	0.259	0.24233333
107	Referencia 3	cortas	18	1r69	0.075	0.081	0.092	0.08266667
108	Referencia 3	cortas	20	1ubi	0.09	0.199	0.125	0.138
109	Referencia 3	cortas	22	1wit	0.23	0.2	0.23	0.22
110	Referencia 3	medianas	21	1uky	0.081	0.067	0.063	0.07033333
111	Referencia 3	medianas	26	2pia	0.083	0.087	0.086	0.08533333
112	Referencia 3	medianas	29	kinase	0.071	0.071	0.078	0.07333333
113	Referencia 3	largas	20	1ajsA	0.067	0.056	0.058	0.06033333
114	Referencia 3	largas	32	1pamA	0.115	0.108	0.092	0.105
115	Referencia 3	largas	32	1ped	0.116	0.124	0.091	0.11033333
116	Referencia 3	largas	24	2myr	0.047	0.047	0.056	0.05
117	Referencia 3	largas	41	4enl	0.222	0.201	0.225	0.216
121	Referencia 4		26	1ckaA	0.002	0.004	0.001	0.00233333
122	Referencia 4		32	1csp	0.002	0	0.001	0.001
123	Referencia 4		22	1dynA	0.002	0.002	0.004	0.00266667

124	Referencia 4	29	1lkl	0.003	0.002	0.002	0.0023333
125	Referencia 4	17	1mfa	0.004	0.001	0.006	0.0036667
126	Referencia 4	19	1pfc	0.004	0.008	0.006	0.006
127	Referencia 4	29	1pysA	0.005	0.013	0.013	0.0103333
128	Referencia 4	43	1vln	0.086	0.077	0.082	0.0816667
129	Referencia 4	36	1ycc	0.043	0.013	0.048	0.0346667
130	Referencia 4	26	2abk	0.003	0.001	0.002	0.002
131	Referencia 4	28	kinase 1	0.016	0.01	0.008	0.0113333
132	Referencia 4	23	kinase 2	0.003	0.003	0.003	0.003
133	Referencia 5	19	1eft	0.012	0.012	0.014	0.0126667
134	Referencia 5	36	1ivy	0.178	0.209	0.209	0.1986667
135	Referencia 5	25	1pysA	0.069	0.067	0.07	0.0686667
136	Referencia 5	35	1qpg	0.058	0.034	0.065	0.0523333
137	Referencia 5	32	1thm1	0.075	0.071	0.081	0.0756667
138	Referencia 5	38	1thm2	0.117	0.065	0.076	0.086
139	Referencia 5	29	2cba	0.043	0.039	0.04	0.0406667
140	Referencia 5	21	s51	0.062	0.064	0.073	0.0663333
141	Referencia 5	29	s52	0.041	0.121	0.042	0.068
142	Referencia 5	26	kinase 1	0.058	0.048	0.064	0.0566667
143	Referencia 5	29	kinase 2	0.046	0.044	0.054	0.048
144	Referencia 5	30	kinase 3	0.043	0.198	0.043	0.0946667

B.2. Iteración en la cual se alcanza el máximo valor

	Referencia	Tipo	Porcentaje medio de identidad	Archivo	Iteración en la que se encuntra el mejor valor			
					muestra 1	muestra 2	muestra 3	Promedio
1	Referencia 1	Cortas de <25% de identidad	15	1aboA	5526	1997	1277	2933
2	Referencia 1	Cortas de <25% de identidad	14	1idy	6441	4456	7574	6157
3	Referencia 1	Cortas de <25% de identidad	13	1r69	6992	7669	7058	7240
4	Referencia 1	Cortas de <25% de identidad	18	1tvxA	4748	6104	544	3799
5	Referencia 1	Cortas de <25% de identidad	18	1ubi	6482	7951	4205	6213
6	Referencia 1	Cortas de <25% de identidad	17	1wit	7510	6394	5303	6402
7	Referencia 1	Cortas de <25% de identidad	17	2trx	7646	4981	3167	5265
8	Referencia 1	Cortas de 20-40% de identidad	30	1aab	4945	7293	7892	6710
9	Referencia 1	Cortas de 20-40% de identidad	28	1fjlA	6676	7181	6563	6807
10	Referencia 1	Cortas de 20-40% de identidad	31	1hfh	6264	5809	6264	6112
11	Referencia 1	Cortas de 20-40% de identidad	33	1hpi	5371	6799	7245	6472
12	Referencia 1	Cortas de 20-40% de identidad	30	1cys	7397	7185	6509	7030
13	Referencia 1	Cortas de 20-40% de identidad	28	1pfc	5423	7812	7903	7046
14	Referencia 1	Cortas de 20-40% de identidad	31	1tgxA	1022	6164	3278	3488
15	Referencia 1	Cortas de 20-40% de identidad	29	1ycc	6096	3801	7711	5869
16	Referencia 1	Cortas de 20-40% de identidad	31	3cyr	6092	6362	7885	6780
17	Referencia 1	Cortas de 20-40% de identidad	27	451c	2972	3999	7083	4685
18	Referencia 1	Cortas de >35% de identidad	44	1aho	76671	5939	4832	29147
19	Referencia 1	Cortas de >35% de identidad	51	1csp	7999	7208	4902	6703
20	Referencia 1	Cortas de >35% de identidad	46	1dox	3839	5460	6857	5385
21	Referencia 1	Cortas de >35% de identidad	44	1fkj	5030	7761	1661	4817
22	Referencia 1	Cortas de >35% de identidad	49	1fmb	7564	7587	7559	7570
23	Referencia 1	Cortas de >35% de identidad	45	1krn	7663	1420	7192	5425
24	Referencia 1	Cortas de >35% de identidad	46	1plc	5771	6361	7689	6607
25	Referencia 1	Cortas de >35% de identidad	51	2fxb	3043	3145	4131	3440
26	Referencia 1	Cortas de >35% de identidad	45	2mhr	7157	4908	2737	4934
27	Referencia 1	Cortas de >35% de identidad	57	9rnt	6946	7786	7759	7497
28	Referencia 1	Medianas, <25% de idenidad	14	1bbt3	7635	7692	7121	7483

29	Referencia 1	Medianas, <25% de idenidad	19	1sbp	3421	7869	5778	5689
30	Referencia 1	Medianas, <25% de idenidad	15	1havA	7217	4680	7594	6497
31	Referencia 1	Medianas, <25% de idenidad	17	1uky	7450	7573	6290	7104
32	Referencia 1	Medianas, <25% de idenidad	18	2hsdA	7842	7324	7851	7672
33	Referencia 1	Medianas, <25% de idenidad	18	2pia	4238	7842	7731	6604
34	Referencia 1	Medianas, <25% de idenidad	17	3grs	4135	5167	4042	4448
35	Referencia 1	Medianas, <25% de idenidad	20	kinase	7728	7254	6362	7115
36	Referencia 1	Medianas, 20-40% identidad	31	1ad2	7738	5712	7021	6824
37	Referencia 1	Medianas, 20-40% identidad	32	1aym3	5959	7993	5395	6449
38	Referencia 1	Medianas, 20-40% identidad	30	1gdoA	7664	6973	6743	7127
39	Referencia 1	Medianas, 20-40% identidad	27	1ldg	7207	5969	7757	6978
40	Referencia 1	Medianas, 20-40% identidad	33	1mrj	7727	6332	7765	7275
41	Referencia 1	Medianas, 20-40% identidad	26	1pgtA	6765	5779	7195	6580
42	Referencia 1	Medianas, 20-40% identidad	33	1pii	6925	6986	3641	5851
43	Referencia 1	Medianas, 20-40% identidad	30	1ton	6232	7586	6906	6908
44	Referencia 1	Medianas, 20-40% identidad	26	2cba	7558	7889	5843	7097
45	Referencia 1	Medianas, >35% de identidad	49	1amk	4325	2892	7917	5045
46	Referencia 1	Medianas, >35% de identidad	43	1ar5A	2186	7245	7799	5743
47	Referencia 1	Medianas, >35% de identidad	60	1ezm	7514	7470	7264	7416
48	Referencia 1	Medianas, >35% de identidad	43	1led	6966	5367	3509	5281
49	Referencia 1	Medianas, >35% de identidad	46	1ppn	4359	7766	6567	6231
50	Referencia 1	Medianas, >35% de identidad	44	1pysA	6693	7691	6277	6887
51	Referencia 1	Medianas, >35% de identidad	49	1thm	6809	6893	3818	5840
52	Referencia 1	Medianas, >35% de identidad	50	1tis	7014	5349	7805	6723
53	Referencia 1	Medianas, >35% de identidad	42	1zin	6662	7236	7753	7217
54	Referencia 1	Medianas, >35% de identidad	43	5ptp	6552	4517	7495	6188
55	Referencia 1	Largas, <25% de identidad	15	1ajsA	7987	7460	6541	7329
56	Referencia 1	Largas, <25% de identidad	20	1cpt	6864	7906	5934	6901
57	Referencia 1	Largas, <25% de identidad	19	1lvl	1356	7785	7236	5459
58	Referencia 1	Largas, <25% de identidad	18	1pamA	7429	3087	5812	5443
59	Referencia 1	Largas, <25% de identidad	22	1ped	5141	7876	3018	5345
60	Referencia 1	Largas, <25% de identidad	16	2myr	2281	1203	7177	3554

61	Referencia 1	Largas, <25% de identidad	19	4enl	2918	6376	7322	5539
62	Referencia 1	Largas, <25% de identidad	14	gal4	5611	7844	7035	6830
63	Referencia 1	Largas, 20-40% de identidad	29	1ac5	7040	5989	6895	6641
64	Referencia 1	Largas, 20-40% de identidad	35	1adj	5195	7645	5577	6139
65	Referencia 1	Largas, 20-40% de identidad	31	1bgl	3836	7412	4762	5337
66	Referencia 1	Largas, 20-40% de identidad	33	1dlc	6360	7543	6500	6801
67	Referencia 1	Largas, 20-40% de identidad	30	1eft	7784	6078	7366	7076
68	Referencia 1	Largas, 20-40% de identidad	34	1fieA	6266	5902	2776	4981
69	Referencia 1	Largas, 20-40% de identidad	31	1gowA	7432	7309	5709	6817
70	Referencia 1	Largas, 20-40% de identidad	34	1pkm	4404	7654	7910	6656
71	Referencia 1	Largas, 20-40% de identidad	33	1sesA	5577	7880	7883	7113
72	Referencia 1	Largas, 20-40% de identidad	28	2ack	6628	7389	6033	6683
73	Referencia 1	Largas, 20-40% de identidad	29	arp	7447	6193	7275	6972
74	Referencia 1	Largas, 20-40% de identidad	31	glg	7877	6270	7425	7191
75	Referencia 1	Largas,>35% de identidad	47	1ad3	5347	5225	1251	3941
76	Referencia 1	Largas,>35% de identidad	47	1gpb	6242	6097	2327	4889
77	Referencia 1	Largas,>35% de identidad	42	1gtr	6363	6592	7872	6942
78	Referencia 1	Largas,>35% de identidad	49	1lcf	4791	7129	7212	6377
79	Referencia 1	Largas,>35% de identidad	42	1rthA	5331	7384	7888	6868
80	Referencia 1	Largas,>35% de identidad	40	1taq	7276	6602	5159	6346
81	Referencia 1	Largas,>35% de identidad	51	3pmg	7090	7385	7322	7266
82	Referencia 1	Largas,>35% de identidad	45	actin	4376	7803		6090
83	Referencia 2	Corta	28	1aboA	375	3237	2241	1951
84	Referencia 2	Corta	28	1cys	7590	7718	7524	7611
85	Referencia 2	Corta	28	1idy	7024	7790	7712	7509
86	Referencia 2	Corta	26	1r69	7606	7265	7497	7456
87	Referencia 2	Corta	20	1tgxA	8000	7625	7805	7810
88	Referencia 2	Corta	19	1tvxA	4658	5000	570	3409
89	Referencia 2	Corta	17	1ubi	7900	7727	7767	7798
90	Referencia 2	Corta	22	1wit	7596	7956	7946	7833
91	Referencia 2	Corta	36	2trx	7802	7574	7702	7693
92	Referencia 2	medianas	24	1sbp	7941	7852	7919	7904

93	Referencia 2	medianas	31	1havA	7968	7267	7456	7564
94	Referencia 2	medianas	31	1uky	7949	6332	5625	6635
95	Referencia 2	medianas	28	2hsdA	7350	7760	7754	7621
96	Referencia 2	medianas	31	2pia	7988	6087	5893	6656
97	Referencia 2	medianas	28	3grs	7785	7462	7924	7724
98	Referencia 2	medianas	32	kinase	7985	6385		7185
99	Referencia 2	largas	35	1ajsA	7861	7823	7825	7836
100	Referencia 2	largas	29	1cpt	7633	7144	7347	7375
101	Referencia 2	largas	30	1lvl	7943	7947	7383	7758
102	Referencia 2	largas	35	1pamA	7879	7647	7666	7731
103	Referencia 2	largas	44	1ped	7271	7903	7326	7500
104	Referencia 2	largas	32	2myr	7425	7726	7677	7609
105	Referencia 2	largas	48	4enl	7008	7470	7657	7378
106	Referencia 3	cortas	19	1idy	7845	7166	7951	7654
107	Referencia3	cortas	18	1r69	7916	7890	7090	7632
108	Referencia 3	cortas	20	1ubi	6630	6026	7811	6822
109	Referencia 3	cortas	22	1wit	7388	7657	7974	7673
110	Referencia 3	medianas	21	1uky	7964	7731	7859	7851
111	Referencia 3	medianas	26	2pia	7998	7981	7904	7961
112	Referencia 3	medianas	29	kinase	4330	7547	6863	6247
113	Referencia 3	largas	20	1ajsA	7878	7869	7749	7832
114	Referencia 3	largas	32	1pamA	3792	7882	8000	6558
115	Referencia 3	largas	32	1ped	7872	7410	7850	7711
116	Referencia 3	largas	24	2myr	7679	7842	7307	7609
117	Referencia 3	largas	41	4enl	7931	7993	7624	7849
121	Referencia 4		26	1ckaA	1	1	1	1
122	Referencia 4		32	1csp	1	1	1	1
123	Referencia 4		22	1dynA	1	1	1	1
124	Referencia 4		29	1lkl	1	1	1	1
125	Referencia 4		17	1mfa	1	1	1	1
126	Referencia 4		19	1pfc	7372	1	5076	4150
127	Referencia 4		29	1pysA	1	1	1	1

128	Referencia 4	43	1vln	7116	2519	1	3212
129	Referencia 4	36	1ycc	7505	1	6976	4827
130	Referencia 4	26	2abk	1	1	1	1
131	Referencia 4	28	kinase 1	4545	1431	7052	4343
132	Referencia 4	23	kinase 2	6844	7429	6663	6979
133	Referencia 5	19	1eft	3026	7727	7043	5932
134	Referencia 5	36	1ivy	7821	7798	7925	7848
135	Referencia 5	25	1pysA	7974	7438	5164	6859
136	Referencia 5	35	1qpg	6768	1289	3257	3771
137	Referencia 5	32	1thm1	7768	5977	3372	5706
138	Referencia 5	38	1thm2	6744	3909	4264	4972
139	Referencia 5	29	2cba	7256	3476	4053	4928
140	Referencia 5	21	s51	7703	7777	7529	7670
141	Referencia 5	29	s52	6530	2843	5476	4950
142	Referencia 5	26	kinase 1	5243	7618	79951	30937
143	Referencia 5	29	kinase 2	6692	7281	6565	6846
144	Referencia 5	30	kinase 3	7763	7214	7763	7580

Anexo C Comparación con otros métodos de alineamiento.

C.1. Referencia 1

	PRRP	CLUSTALX	SAGA	DIALIGN	SB_PIMA	ML_PIMA	MULTALIGN	PILEUP8	MULTAL	HMMT	Búsqueda armónica
1aboA	0.56	0.687	0.529	0.359	0.312	0.312	0.703	0.521	0.526	0.181	0.10433333
1idy	0.606	0.705	0.342	0.018	0.145	0.062	0.566	0.08	0.08	0.138	0.37033333
1r69	0.837	0.481	0.55	0.406	0.681	0.366	0.325	0.562	0.225	0.1	0.181
1tvxA	0.378	0.438	0.278	0.306	0.344	0.344	0.228	0.344	0.244	0.108	0.087
1ubi	0.498	0.415	0.452	0	0.37	0.493	0.488	0.428	0.428	0.14	0.17566667
1wit	0.991	0.982	0.899	0.851	0.963	0.444	0.842	0.773	0.763	0.549	0.312
2trx	0.494	0.754	0.801	0.728	0.451	0.496	0.5	0.453	0.235	0.292	0.14733333
1bbt3	0.907	0.706	0.652	0.45	0.26	0.26	0.582	0.43	0.16	0.128	0.16333333
1sbp	0.613	0.674	0.543	0.453	0.54	0.456	0.58	0.547	0.537	0.144	0.09766667
1havA	0.656	0.446	0.411	0.13	0.3	0.466	0.419	0.536	0.04	0.133	0.08
1uky	0.676	0.724	0.672	0.566	0.684	0.684	0.685	0.571	0.46	0.084	0.063
2hsdA	0.885	0.691	0.771	0.679	0.47	0.47	0.47	0.646	0.463	0.117	0.128
2pia	0.78	0.89	0.69	0.66	0.74	0.74	0.76	0.85	0.56	0.057	0.03666667
3grs	0.447	0.635	0.689	0.327	0.416	0.416	0.38	0.446	0.347	0.056	0.21733333
kinase	0.891	0.736	0.862	0.764	0.733	0.703	0.643	0.73	0.65	0.277	0.22333333
1ajsA	0.636	0.571	0.531	0.142	0.486	0.471	0.424	0.556	0.364	0.105	0.10866667
1cpt	0.862	0.827	0.78	0.829	0.792	0.792	0.835	0.865	0.777	0.156	0.097
1lvl	0.694	0.632	0.619	0.699	0.559	0.559	0.57	0.587	0.572	0.064	0.077
1pamA	0.724	0.582	0.596	0.694	0.513	0.549	0.596	0.641	0.204	0.034	0.012
1ped	0.846	0.812	0.748	0.743	0.756	0.756	0.727	0.723	0.73	0.032	0.09366667
2myr	0.506	0.381	0.285	0.284	0.67	0.613	0.422	0.621	0.547	0.05	0.03433333
4enl	0.677	0.706	0.414	0.529	0.701	0.701	0.754	0.669	0.787	0.044	0.06366667
gal4	0.742	0.407	0.5	0.581	0.445	0.445	0.368	0.543	0.406	0.055	0.078
1aab	1	1	0.823	1	1	1	1	1	1	0.214	0.35
1fjlA	1	1	0.993	1	1	1	1	1	1	0.436	0.50633333
1hfh	0.933	0.917	0.945	0.41	0.868	0.932	0.936	0.848	0.883	0.261	0.229
1hpi	0.918	0.861	0.916	0.785	0.909	0.909	0.89	0.859	0.852	0.962	0.41033333
1csy	0.949	1	0.969	0.98	0.976	0.968	0.981	0.932	0.935	0.779	0.49866667
1pfc	0.975	0.988	0.994	0.894	0.927	0.927	0.975	0.964	0.861	0.35	0.51066667
1tgxA	0.895	0.806	0.873	0.871	0.782	0.768	0.819	0.718	0.651	0.556	0.39433333
1ycc	0.856	0.935	0.837	0.749	0.815	0.815	0.932	0.939	0.94	0.217	0.15433333
3cyr	0.907	0.805	0.908	0.75	0.887	0.887	0.858	0.854	0.841	0.454	0.27766667
451c	0.681	0.719	0.662	0.729	0.541	0.552	0.683	0.668	0.713	0.346	0.40433333

1ad2	0.943	0.984	0.917	0.96	0.934	0.934	0.96	0.96	0.975	0.341	0.272
1aym3	0.976	0.969	0.955	0.962	0.976	0.958	0.969	0.952	0.969	0.567	0.39733333
1gdoA	0.926	0.929	0.907	0.81	0.911	0.911	0.935	0.882	0.84	0.604	0.194
1ldg	1	0.963	0.989	0.966	0.996	0.996	0.989	0.958	0.956	0.326	0.32766667
1mrj	1	1	0.949	0.977	1	1	0.994	1	1	0.561	0.331
1pgtA	0.969	1	0.982	0.996	0.993	1	0.937	0.975	1	0.545	0.274
1pii	0.883	0.864	0.896	0.89	0.832	0.788	0.884	0.854	0.859	0.193	0.22166667
1ton	0.965	0.935	0.945	0.867	0.904	0.844	0.899	0.826	0.749	0.512	0.281
2cba	0.946	0.926	0.946	0.941	0.928	0.935	0.855	0.846	0.859	0.176	0.18166667
1ac5	0.94	0.992	0.911	0.926	0.941	0.941	0.942	0.898	0.896	0.236	0.067
1bgl	0.979	0.959	0.972	0.959	0.934	0.929	0.959	0.958	0.977	0.365	0.15166667
1dlc	0.977	0.961	0.916	0.897	0.952	0.952	0.98	0.984	0.926	0.427	0.20166667
1eft	0.934	0.911	0.91	0.925	0.926	0.926	0.929	0.913	0.895	0.289	0.23866667
1fieA	0.965	0.944	0.947	0.979	0.905	0.926	0.958	0.958	0.939	0.637	0.417
1gowA	0.749	0.898	0.814	0.902	0.872	0.872	0.878	0.921	0.927	0.396	0.257
1pkm	0.942	0.921	0.955	0.927	0.907	0.911	0.946	0.931	0.851	0.748	0.329
1sesA	0.985	0.968	0.954	0.968	0.899	0.899	0.973	0.963	0.882	0.642	0.19533333
2ack	0.909	0.907	0.904	0.882	0.907	0.88	0.848	0.87	0.766	0.524	0.14733333
arp	0.956	0.945	0.933	0.93	0.914	0.927	0.938	0.925	0.927	0.293	0.11
glg	0.982	0.941	0.972	0.959	0.978	0.968	0.954	0.977	0.943	0.689	0.27
1aho	0.99	0.971	1	1	1	1	0.913	0.938	0.938	0.789	0.49066667
1csp	0.943	0.993	0.993	0.98	1	1	0.987	1	0.625	0.776	0.375
1dox	0.887	0.919	0.879	0.859	0.868	0.868	0.799	0.812	0.48	0.806	0.27033333
1fkj	0.982	0.981	0.981	0.958	0.944	0.987	0.951	0.913	0.609	0.879	0.48866667
1fmb	0.959	0.981	0.979	0.959	0.952	0.952	0.995	0.995	0.556	0.863	0.64733333
1krn	1	1	0.993	1	0.986	0.986	0.993	1	1	0.944	0.73333333
1plc	0.979	0.934	0.958	0.931	0.904	0.875	0.964	0.958	0.946	0.797	0.521
2fxb	0.945	0.945	0.951	0.945	0.945	0.945	0.945	0.945	0.945	0.93	0.507
2mhr	0.98	0.985	0.952	0.951	0.975	0.908	0.962	0.965	0.961	0.803	0.54333333
9rnt	0.965	0.974	0.965	0.864	0.97	0.961	0.965	0.97	0.974	0.832	0.57066667
1amk	0.986	0.989	0.997	0.993	0.987	0.987	0.991	0.993	0.992	0.941	0.39866667
1ar5A	0.971	0.994	0.971	0.92	0.994	0.963	0.99	0.98	0.98	0.903	0.42066667
1ezm	0.941	0.948	0.932	0.911	0.956	0.956	0.958	0.948	0.61	0.851	0.45566667
1led	0.969	0.946	0.923	0.516	0.987	0.93	0.94	0.933	0.882	0.761	0.40833333
1ppn	0.973	0.989	0.983	0.648	0.962	0.979	0.973	0.978	0.973	0.856	0.48366667
1pysA	0.931	0.922	0.906	0.936	0.935	0.935	0.926	0.931	0.917	0.616	0.28833333
1thm	0.956	0.961	0.956	0.946	0.971	0.971	0.966	0.961	0.936	0.697	0.49566667
1tis	0.971	0.977	0.953	0.971	0.951	0.965	0.942	0.985	0.971	0.846	0.273
1zin	0.988	0.977	0.977	0.966	0.916	0.916	0.954	0.961	0.927	0.894	0.55833333
5ptp	0.977	0.966	0.94	0.888	0.966	0.96	0.954	0.972	0.953	0.742	0.32933333
1ad3	0.972	0.968	0.967	0.963	0.962	0.962	0.979	0.97	0.971	0.738	0.23766667

1gpb	0.988	0.986	0.982	0.958	0.964	0.964	0.987	0.983	0.964	0.65	0.264
1gtr	0.992	0.986	0.995	0.994	0.961	0.961	0.976	0.995	0.99	0.796	0.30066667
1lcf	0.98	0.981	0.967	0.947	0.962	0.964	0.979	0.971	0.968	0.854	0.186
1rthA	0.961	0.977	0.96	0.958	0.962	0.952	0.966	0.982	0.956	0.856	0.42466667
1taq	0.963	0.963	0.931	0.889	0.961	0.934	0.944	0.942	0.88	0.691	0.11266667
3pmg	0.987	0.995	0.989	0.985	0.985	0.985	0.994	0.994	0.993	0.844	0.30633333
actin	0.979	0.965	0.965	0.968	0.968	0.98	0.975	0.969	0.962	0.788	0.253

C.2. Referencia 2

		PRRP	CLUSTALX	SAGA	DIALIGN	HMMT	SB_PIMA	ML_PIMA	MULTALIGN	PILEUP8	Búsqueda armónica
82	1aboA	0.256	0.65	0.489	0.384	0.724	0.391	0.22	0.528	0	0.24933333
83	1idy	0.37	0.515	0.548	0	0.353	0	0	0.401	0	0.13366667
84	1csy	0.35	0.154	0.154	0	0	0	0	0.154	0.114	0.08933333
85	1r69	0.675	0.675	0.475	0.675	0	0.675	0.675	0.675	0.45	0.6
86	1tvxA	0.207	0.552	0.448	0	0.276	0.241	0.241	0.138	0.345	0.24133333
87	1tgxA	0.695	0.727	0.773	0.63	0.622	0.678	0.543	0.696	0.318	0.519
88	1ubi	0.056	0.482	0.492	0	0.053	0.129	0.129	0	0	0.043
89	1wit	0.76	0.557	0.694	0.724	0.641	0.469	0.463	0.5	0.476	0.47966667
90	2trx	0.87	0.87	0.87	0.734	0.739	0.85	0.702	0.87	0.87	0.814
91	1sbp	0.231	0.217	0.374	0.043	0.214	0.043	0.054	0.186	0.177	0.139
92	1havA	0.52	0.48	0.448	0	0.194	0.259	0.238	0.5	0.493	0.41033333
93	1uky	0.351	0.656	0.476	0.216	0.395	0.256	0.306	0.585	0.562	0.48433333
94	2hsdA	0.404	0.484	0.498	0.262	0.423	0.39	0.561	0.593	0.278	0.47733333
95	2pia	0.767	0.752	0.763	0.612	0.647	0.73	0.695	0.765	0.766	0.742
96	3grs	0.363	0.192	0.282	0.35	0.141	0.183	0.211	0.192	0.159	0.18733333
97	kinase	0.896	0.848	0.867	0.692	0.749	0.755	0.651	0.83	0.799	0.76
98	1ajsA	0.227	0.324	0.311	0	0.242	0	0	0.311	0.227	0.17933333
99	1cpt	0.821	0.66	0.776	0.425	0.388	0.184	0.277	0.777	0.688	0.58066667
100	1lvl	0.772	0.746	0.726	0.783	0.539	0.62	0.688	0.614	0.678	0.66
101	1pamA	0.711	0.761	0.623	0.576	0.53	0.393	0.386	0.566	0.702	0.55133333
102	1ped	0.881	0.834	0.835	0.773	0.696	0.651	0.647	0.741	0.749	0.71233333
103	2myr	0.582	0.904	0.825	0.84	0.443	0.727	0.75	0.894	0.786	0.81
104	4enl	0.668	0.375	0.739	0.122	0.213	0.096	0.092	0.384	0.224	0.23333333

C.3. Referencia 3

	PRRP	CLUSTALX	SAGA	DIALIGN	HMMT	SB_PIMA	ML_PIMA	MULTALIGN	PILEUP8	Búsqueda armónica
1idy	0	0.273	0.364	0	0.227	0	0	0.045	0	0.242
1r69	0.905	0.524	0.524	0.524	0	0	0.905	0	0	0.083
1ubi	0.415	0.146	0.585	0	0.366	0	0	0	0.268	0.138
1wit	0.742	0.565	0.484	0.5	0.323	0.645	0.323	0.242	0.21	0.220
1uky	0.139	0.13	0.269	0.139	0.037	0.083	0.148	0.241	0.083	0.070
kinase	0.783	0.72	0.758	0.65	0.478	0.541	0.682	0.688	0.599	0.073
1ajsA	0.128	0.163	0.186	0	0.006	0	0	0	0.11	0.060
1 pamA	0.683	0.678	0.579	0.683	0.169	0.546	0.59	0.546	0.754	0.105
1ped	0.679	0.627	0.646	0.641	0.172	0.45	0.507	0.665	0.722	0.110
2myr	0.646	0.538	0.494	0.272	0.101	0.278	0.494	0.253	0.31	0.050
4enl	0.736	0.547	0.672	0.05	0.05	0.393	0.438	0.652	0.498	0.216

C.4. Referencia 4

	PRRP	CLUSTALX	SAGA	DIALIGN	SB_PIMA	ML_PIMA	MULTALIGN	PILEUP8	Búsqueda armónica
1dynA	0	0	0	0.6	0.6	0.6	0	0	0.00233333
1pysA	0	0	0.25	0.75	1	1	0	0.75	0.01033333
1ckaA	1	0	0.375	1	1	0	0	1	0.00233333
1csp	0	0	0	0.889	0	0	0	0	0.001
1lkl	1	1	0	1	1	1	0	1	0.00233333
1mfa	0.385	1	0.385	1	0.846	1	0.385	1	0.00366667
1pfc	1	0.969	1	0.719	1	1	1	1	0.006
1vln	0	0.879	0.606	0.545	0.636	0.576	0.636	0.848	0.08166667
1ycc	0.485	0.485	0.485	0.727	0.97	0.818	0.485	0.455	0.03466667
2abk	0	0	0	1	0.471	0.471	0	0.471	0.002
kinase1	0	0	0	1	1	1	0	1	0.01133333
kinase2	0	0	0.364	1	1	1	1	1	0.003

C.5. Referencia 5

	PRRP	CLUSTALX	SAGA	DIALIGN	SB_PIMA	ML_PIMA	MULTALIGN	PILEUP8	Búsqueda armónica
1pysA	0.429	0.429	0.429	0.762	0.19	0.762	0.429	0.19	0.01266667
1eft	0.211	0	0	0.579	0	0	0	0.211	0.19866667
1ivy	1	0.735	0.735	1	1	0.882	0.735	1	0.06866667

1qpg	1	1	0.521	1	1	1	1	1	0.05233333
1thm1	0.765	0.412	0.765	0.765	0.765	0.412	0.412	0.765	0.07566667
1thm2	0.645	0.774	0.774	1	0.194	0.194	0.774	0.645	0.086
2cba	0.867	0.717	0.767	1	0.533	0.767	0.55	0.6	0.04066667
S51	0.662	0.938	0.831	0.646	0.338	0.631	0.646	0.646	0.06633333
S52	1	1	1	1	0.515	0.515	1	0.515	0.068
kinase1	1	0.806	0.484	0.806	0.677	0.677	1	0.677	0.05666667
kinase2	0.489	1	0.667	0.667	0.556	0.444	0.333	0.689	0.048
kinase3	0.333	0.646	0.729	0.812	0.333	0.583	0.646	0.729	0.09466667

Anexo D Lista de símbolos.

$\mathcal{A}$	Alfabeto
$\{v_l^i, v_m^j\}$:	Alineamiento del caracter l de la i −ésima secuencia con el caracter m de la j −ésima secuencia.
T_N	Árbol ancestral.
$--$	Auto decremental del valor de una varible en una unidad
$++$	Auto incremental del valor de una varible en una unidad
s_{i}	Cadena i
$*$	Comodin en secuencia templete
N	Conjunto de N-secuencias
S	Conjunto de secuencias
S_l	Conjunto de secuencias de longitud l
X_i	Conjunto de valores discretos para la variable x.
β	Conjunto obtenido de $\mathcal{A} \cup \{-\}$
$cost$	Costo
$w_e\ w_a$	Costos asociados a los arcos
$s(a,b)$	Diferencia para el alineamiento del caracter a con b.
$d(s_a, s_b)$	Distancia entre la secuenca s_a y la s_b
$C_{ab} = d(s_a, s_b)$	Distancia entre la secuenca s_a y la s_b vista como costo.
$\widehat{s_{a_i}}$	Elemento i −ésimo de la secuencia $\widehat{s_a}$
$\mathcal{A}^*$	Espacio de secuencias
λ	Factor de escala.
w	Factor de penalización.
γ	Factor de ponderación entre los pares posibles.
$\mathrm{s_{ij}}$	Factor de relación entre los caracteres de dos secuencias, algoritmo de Needleman –Wunch variante por vinuesa,
$\mathrm{f_a}$	Frecuencia de aparición del aminoácido a.
f_j	Frecuencia de aparición del aminoácido b
f	Función
$f(s_{a_i}, \mathrm{s_{b_i}})$	función de comparación entre ambos caracteres de ambas secuencias en la posición $i - ésima$
$Optimizar\ \ f(x)$:	Función objetivo a optimizar
h	Homomorfismo
Input	Información requerida el inicio de un algoritmo.
Output:	Información resultado de aplicar un algoritmo a un conjunto de datos
$l_{s_{min}}$:	Longitud de la cadena más corta del conjunto N
$l_{s_{max}}$	Longitud de la cadena más larga del conjunto N
$l_a = \|s_a\|$	Longitud de la cadena s_a
t	Magnitud de los guiones incertados
P	Matriz auxiliar para la formación de la matriz de memoria armónica.
M	Matriz de alineamiento múltiple

$F_{l_a \ x \ l_b}$	Matriz de programación dinamica para el algoritmo de Needleman
m	Matriz de programación dinamica.
ls	Máxima longitud posible de la secuencia $\widehat{s_a}$.
M'	Mejor alineamiento encontrado.
li	Mínima longitud posible de la secuencia $\widehat{s_a}$.
v_l^i:	Nodo correspondiente en el lugar l de la i −ésima secuencia.
k	Número de elementos en el alfabeto
ns_a	Número de guiones insertados.
n_e	Número de nodos externos en el árbol filogenético
n_i	Número de nodos internos en el árbol filogenético
x	Número de secuencias del conjunto N
||	Operador logico o
&&	Operador logico y
H	Orden del esquema
PAR	Parámetro del algoritmo HS que indica el ajuste del ritmo.
b	Parámetro del algoritmo HS que indica el ancho de banda de ajuste del ritmo.
$MaxImp$	Parámetro del algoritmo HS que indica el máximo número de improvisaciones.
HMS	Parámetro del algoritmo HS que indica el tamaño de la memoria armónica.
$HMCR$	Parámetro del algoritmo HS que indica la consideración del ritmo de la memoria armónica.
w_2	Penalización por la extensión de un espacio vacío
w_1	Penalización por la introducción de un nuevo espacio vacío
$P(x_{a_i})$:	Probabilidad de asignar un espacio vacío en la i −ésima posición de la secuencia $\widehat{s_a}$
$P(s_a)$	Probabilidad de incluir a la secuencia s_a en la muestra.
$\mathrm{p_{a,b}}$	Probabilidiad de sustitución del aminoacido a por el aminoácido b de acuerdo aexperincia en observacíones
$\prod$	Producto cartesiano del conjunto
If (condición)	Proposición si
While (condición)	Proposición mientras.
For (condición)	Proposición para
$p_{a,b}(\gamma(s_1, s_2, \cdots, s_N))$	Proyección de la ruta de conección en el plano ab.
$\kappa(i, j, k)$	Relación de comparación simultanea para tres secuencias algoritmo de murata.
$x_i^L \leq x_i \leq x_i^U$	Restricción lógica para variables continuas que determinan el rango.
$x_i \in X_i$	Restricción lógica para variables discretas que indica el conjunto de valores posibles.
$g(x) \geq 0$	Restricciones de desigualdad del tipo mayor o igual.
$g''(x) \leq 0$	Restricciones de desigualdad del tipo menor o igual.
$g'(x) = 0$	Restricciones del tipo de igualdad
$\gamma(s_1, s_2, \cdots, s_N)$	Ruta de conceción de nodos algoritmo de Carrillo Lipman

$\widehat{s_a}$	Secuencia s_a con los espacios vacios incertados.
s_*	Secuencia esquema
s_m	Secuencia media (esquema arbitrario)
M^P	Submatriz de P que contiene el alineamiento propuesto.
G^P	Submatriz de P que contiene el vector de distancia y vector de fitness.
s_{sub-i}	Subsecuencia i de la secuencuencia s
$caracteres = \sum_{\substack{a=1 \\ s_a \in N}}^{x} l_{s_a}$	Suma se la longitud de todas las secuencias del conjunto.
T	Tamaño de muestra
$\mathcal{M}$:	Todos los ciclos en la gráfica G.
$V\ (M)$	Valor del alineamiento.
x_i^L	Valor inferior del rango posible en variables continuas.
x_i^U	Valor superior del rango posible en variables continuas
y_{ij}	Variable de decisión que sea 1 si alinean dos secuencias y 0 en otro caso
x_e:	Variable de alineamiento.
x_{a_i}	Variable de decisión del AMS para el método propuesto.
y_a:	Variable por la inserción de espacios vacíos

Anexo E Otros índices.

E.1. Figuras.

FIGURA 1. PROBLEMA DE PERMUTACIÓN DE OBJETOS 3

FIGURA 2. SUBSECUENCIA. 5

FIGURA 3. SUBSECUENCIA COMÚN. 6

FIGURA 4. ESQUEMA 30

FIGURA 5. MÉTODOS DE SOLUCIÓN DEL AMS ANALIZADOS. 36

FIGURA 6. CLASIFICACIÓN DE LOS ALGORITMOS UTILIZADOS PARA RESOLVER EL AMS. 38

FIGURA 7. MÉTODOS DE SOLUCIÓN AL AMS 40

FIGURA 8. MATRIZ TRIDIMENSIONAL PARA EL ALGORITMO DE MURATA. 53

FIGURA 9. RASTREO EN UNA MATRIZ DE PD TRIDIMENSIONAL. 53

FIGURA 10. COMPARATIVO DEL MÉTODO DE GOTOH VS. NEEDLEMAN 57

FIGURA 11. REPRESENTACIÓN GRÁFICA DEL ALINEAMIENTO DE DOS SECUENCIAS. 60

FIGURA 12. HIPERCUBO AL RESULTADO DEL MSA PARA TRES SECUENCIAS. . 62

FIGURA 13. ALINEAMIENTO LOCAL VS. GLOBAL. 63

FIGURA 14. GRÁFICA DE ALINEAMIENTO POR VACÍOS. 67

FIGURA 15. ÁRBOL EVOLUTIVO. 70

FIGURA 16. ALGORITMO DE FITCH. 72

FIGURA 17. EJEMPLO DEL ALGORITMO DESANKOF. 73

FIGURA 18. PROCEDIMIENTO PARA LOS ALGORITMOS DE LA FAMILIA CLUSTAL 79

FIGURA 19. ESQUEMA DE PUNTUACIÓN DE LA FUNCIÓN COFFEE. 82

FIGURA 20. MÉTODO COFFEE. 83

FIGURA 21. CARACTERÍSTICAS GENERALES DE LOS ALGORITMOS ITERATIVOS. 84

FIGURA 22. MODELO OCULTO DE MARKOV 84

FIGURA 23. ELEMENTOS DEL HMM 85

FIGURA 24. ESQUEMA DE RELACIÓN ENTRE PROGRAMAS Y ALGORITMOS 86

FIGURA 25. ANALOGÍA ENTRE EL PROCESO DE INNOVACIÓN MUSICAL Y LA OPTIMIZACIÓN. 91

FIGURA 26. OPERACIONES BÁSICAS DEL HS. 93

FIGURA 27. CONCEPTO DE NUEVA IMPROVISACIÓN. 95

FIGURA 28. ALGORITMO DE BÚSQUEDA DE ARMONÍA. 107

FIGURA 29. PROCESO DE RECOCIDO DE METALES. 109

FIGURA 30. PARTICULARIDADES DEL RS PARA ESCAPAR DE ÓPTIMOS LOCALES. 111

FIGURA 31. SUBGRUPOS DE LAS BASES NITROGENADAS. 131

FIGURA 32. COMPORTAMIENTO DE LA BÚSQUEDA ARMÓNICA CON RESPECTO A LA REFERENCIA 1. 152

FIGURA 33. COMPORTAMIENTO DE LA BÚSQUEDA ARMÓNICA CON RESPECTO A LA REFERENCIA 2. 153

FIGURA 34. COMPORTAMIENTO DE LA BÚSQUEDA ARMÓNICA CON RESPECTO A LA REFERENCIA 3 154

FIGURA 35. COMPORTAMIENTO DE LA BÚSQUEDA ARMÓNICA CON RESPECTO A LA REFERENCIA 4 155

FIGURA 36. COMPORTAMIENTO DE LA BÚSQUEDA ARMÓNICA CON RESPECTO A LA REFERENCIA 5. 156

FIGURA 37. COMPORTAMIENTO DE ALGORITMO DESARROLLADO PARA LAS REFERENCIAS DE BALIBASE. 157

FIGURA 38. ALGORITMO HÍBRIDO DE HS Y RS PARA SOLUCIONAR EL AMS... 160

FIGURA 39. PINZONES DE DARWIN 164

FIGURA 40. COTAS ASINTÓTICAS DE UNA FUNCIÓN. 167

FIGURA 41. CRECIMIENTO DE FUNCIONES DE COMPLEJIDAD COMPUTACIONAL CON RELACIÓN AL TAMAÑO DE LA ENTRADA. 168

FIGURA 42. CONJUNTO DE PROBLEMAS NP. 170

E.2. Tablas.

TABLA 1. DIFERENCIAS ENTRE ALGORITMOS. 64

TABLA 2. NÚMERO DE ALINEAMIENTOS POR CADA REFERENCIA. 89

TABLA 3. ANALOGÍA ENTRE OPTIMIZACIÓN Y INNOVACIÓN MUSICAL. 92

TABLA 4. VALOR DE LOS PARÁMETROS AL APLICAR DE BÚSQUEDA ARMÓNICA VALOR 108

TABLA 5. ANALOGÍA ENTRE LA OPTIMIZACIÓN Y EL RECOCIDO METALÚRGICO DE METALES. 110

TABLA 6. COMPARATIVO DE ALGUNOS DE LOS MÉTODOS DE SOLUCIÓN DEL AMS. 113

TABLA 7. RESUMEN DE LAS METAHEURÍSTICAS DE RS Y HS. 114

TABLA 8. VALOR DE LOS DATOS DE ENTRADA DEL ALGORITMO. 123

TABLA 9. COMPARACIÓN DE HS VS. ESTRATEGIA DESARROLLADA. 150

TABLA 10. VALOR DE LA SIMILITUD ENTRE LOS ALINEAMIENTOS ALCANZADOS Y REFERENCIALES CUANTIFICANDO LA MÉTRICA SSP. 151

TABLA 11. COMPARATIVO DE LA COMPLEJIDAD DE LOS ALGORITMOS PARA SOLUCIONAR EL AMS. 158

TABLA 12. ITERACIONES DEL ORDEN DE ALGORITMOS 168

TABLA 13. ALFABETO DE NUCLEÓTIDOS. 173

TABLA 14. ALFABETO DE AMINOÁCIDOS 173

E.3. Algoritmos.

ALGORITMO 1. MATRIZ DE PUNTOS. 9

ALGORITMO 2. DISTANCIA DE LEVENSHTEIN. 12

ALGORITMO 3. DISTANCIA LCS 15

ALGORITMO 4. DISTANCIA DE DAMERAU. 18

ALGORITMO 5. ALGORITMO NEEDLEMAN Y WUNSCH. 43

ALGORITMO 6. VARIANTE DEL ALGORITMO NEEDLEMAN Y WUNSCH VARIANTE PROPUESTA POR VINUESA. 49

ALGORITMO 7. ALGORITMO DE MURATA *ET AL* 54

ALGORITMO 8. ALGORITMO DE GOTOH 58

ALGORITMO 9. ALGORITMO DE SMITH Y WATERMAN. 65

ALGORITMO 10. MATRIZ DE DISTANCIA PAR A PAR. 69

ALGORITMO 11. ALGORITMO DE FENG DOOLITLE 73

ALGORITMO 12. ALGORITMO DE BARTON Y STERNBERG 74

ALGORITMO 13. 2-APROXIMACIÓN PARA AMS CON FUNCIÓN OBJETIVO DE SUMA POR PARES DE LA DIFERENCIA. 76

ALGORITMO 14. ALGORITMO GENERAL PARA LA FAMILIA CLUSTAL 78

ALGORITMO 15. ALGORITMO FASTA. 80

ALGORITMO 18. ACTUALIZACIÓN DE LA MEMORIA ARMÓNICA PARA EL TSP BINARIO. 105

ALGORITMO 19. CUMPLIMIENTO DEL CRITERIO DE PARO PARA EL TSP BINARIO. 106

ALGORITMO 20. RECOCIDO SIMULADO. 111

ALGORITMO 22. CONSTRUCCIÓN DE LA SUBMATRIZ MP. 125

ALGORITMO 23. CONSTRUCCIÓN DE LA MATRIZ D. 129

ALGORITMO 24. CONSTRUCCIÓN DE LA SUBMATRIZ GP. 134

ALGORITMO 25. CONSTRUCCIÓN DE LA MATRIZ P. 136

ALGORITMO 26. CONSTRUCCIÓN DE LA MEMORIA ARMÓNICA INICIAL. 137

ALGORITMO 27. GENERACIÓN DE UNA NUEVA IMPROVISACIÓN. 140

ALGORITMO 28. ACTUALIZACIÓN DE LA MEMORIA ARMÓNICA. 145

ALGORITMO 29. SATISFACER EL CRITERIO DE PARO. 147

ALGORITMO 30. ALINEAMIENTO DE MÚLTIPLES SECUENCIAS POR BÚSQUEDA ARMÓNICA. 149

Bibliografía.

- Abascal, F. (09 de julio de 2007). *Alineamiento de secuencias. Búsqueda de parecidos.Alineamientos múltiples.* Recuperado el 1 de octubre de 2008, de http://darwin.uvigo.es/people/fabascal/Teaching/Alineamiento_secuencias/teoria.html#por
- Agüero, F. (01 de junio de 2004). *genoma.unsam.edu.ar*. Recuperado el 01 de octubre de 2008, de Alineamiento de secuencias búsqueda de secuencias en bases de datos: http://brucella.unsam.edu.ar/bioinformatica2004/sss-msa.ppt.
- Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network Flows. Theory, Algorithms and Applications.* Estados Unidos de America: Prentice-Hall .
- Althaus, E., Caprara, A., & Lenho, H.-P. (1997). A Branch-and-Cut Algorithm for Multiple Sequence Alignment. *Mathematical Programming* .
- Altschul, S. F., & Erickson, B. W. (1986). Optimal sequence alignment using affine gap cost. *Bulletin of mathematical Biology* , 603-616.
- Bahr, A., Thompson, J., Thierry, J. C., & Poch, O. (1999). *BAliBASE (version 2.0): A benchmark alignment database, including enhancements for repeats, transmembrane sequences and circular permutations*. Recuperado el 2009, de Plate-Forme Bio-informatique de Strasbourg: http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE2/
- Barton, G. J., & Sternberg, M. J. (1987). A strategy for the rapid multiple alignment of protein sequences. *Journal molecular biology* , 327-337.
- Bezerra, B. R. (2006). *Comparación exacta paralela de secuenca biologicas largas con uso limitado de memoria.* (O. D. Melo, Ed.) Brasilia, Brasil: Universidad de Brasilia.
- Bioinformática, C. d. (2005). *EMBnet Colombia*. (I. d. Biotecnología, Editor, & U. N. Colombia, Productor) Recuperado el 15 de febrero de 2009, de http://bioinf.ibun.unal.edu.co/documentos/
- Carrillo, H., & Lipman, D. (1988). The multiple sequence alignment problem in biology. *Society for industrial and applied mathematics , 48* (5), 1078-1082.
- Castañeda Roldán, C. Y. (2000). *Estudio comparativo de diversos métodos de solución del problema del agente viajero (PAV).* Puebla, México: Universidad de las Américas Puebla.
- Chan, S. C., Wong, K., & Chiu, D. K. (1992). A survey of multiple sequence comparison methods. *Bulletin of Mathematical Biology Vol. 54 N°4* , 563-598.
- Coelho, L. d., & Mariani, V. C. (2009). An improved harmony search algorithm for power economic load dispatch. *Energy Conversion and Management* , 2522-2526.
- Cronquist, A. (1978). *Botanica Basica.* México: C.E.C.S.A.
- Dayhoff, M. O., Barker, W. C., & Hunt, L. T. (1983). Estabblishing Homologies in protein sequences. *Methods in enzymology* , 524-538.
- De Alba Romenus, K. (2004). *Un procedimiento heuristico para un problema de diseño de redes multiproducto con capacidad finita y cargos fijos.* San

Nicolas de los Garza, Nuevo León, México: Universidad Atónoma de Nuevo León.

- Desmond, G. H., & Thomson, J. D. (1996). Using Clustal for multiple sequence alignment. *methods in enzymology* , 383-402.
- Elias, I. (2003). Settling the Intractability of Multiple Alignment. *T. Ibaraki, N. Katoh, and H. Ono (Eds.): Proc. 14th Annual Int. Symp. on algorithms and computation (ISSAC), Springer-Verlag Berlin Heidelberg* , 352-363.
- Enderton, H. B. (1987). *Una introducción matemática a la lógica.* México D.F.: Universidad Nacional Autónoma de México.
- Flores de la Mota, I. (1980). *Apuntes de Programación Entera.* División de Estudios de Posgrado, Facultad de Ingeniería U.N.A.M .
- Gaarder, J. (2004). *El mundo de Sofia Novela sobre la historia de la filosofía.* México: Ediciones Patria.
- García, J. F. (julio de 2007). *Métricas de Similitud para Búsqueda Aproximada*. Recuperado el octubre de 2008, de http://sem.uno.googlepages.com/estado.1.1.pdf
- Gardner, E. J. (1980). *Principios de genetica.* Mexico: Editorial limusa.
- Gautham, N. (2006). *Bioinformatics database and Algorithms.* Alpha Science Int'l ltd.
- Geem, Z. W., & Choi, J.-Y. (2007). Music Composition Using Harmony Search Algorithm. *Springer-Verlag Berlin Heidelberg* , 593-600.
- Geem, Z. W., & Kim, J. H. (2001). A New heuristic Optimization Algoritm: Hamony Search. *Simulation* , 60-68.
- Geem, Z. W., & Lee, K. S. (2004). A new meta-heuristic algorithm for continuos engineering optimization: harmony search teory and practice. *Computer methods in applied mechanics and engineering* , 3902-3933.
- González, G. (2008). *Algoritmo Smith-Waterman*. Recuperado el 1 de 12 de 2008, de Bioinformaticos: http://www.bioinformaticos.com.ar/articulos/smith-waterman
- Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.* , 705-708.
- Grimaldi, R. P. (1998). *Matemáticas discretas y combinatorias. Una introducción con aplicaciones* (tercera edición ed.). México: Pearson Printice Hall.
- Gusfieeld, D. (1993). Efficient Methods for multiple secuence alignment whit guaranteed error bounds. *Bulletin of Mathematical Biology* , 141-154.
- Hall, & D., A. (1983). *Ingeniería de sistemas* (Decima ed.). México: C.E.C.S.A.
- Hernández Valdemar, E. J. (2003). Bioinformática práctica: procesamiento de cadenas y secuncias biológicas. *Congreso de sistemas e informática CONSI 2003.* San Luis Potosi: Fundación Arturo Rosenblueth.
- Huson, D. (2009). *Algorithms in bioinformatics*. (U. Tübingen, Productor) Recuperado el 15 de febreo de 2009, de http://www-ab.informatik.uni-tuebingen.de/teaching/ws04/phylo/script/02_11.pdf

- Ikeda, T., & Imai, H. (1999). Enhanced A* algortims for multiple alignments: optimal alignmet for several sequence an k-opt approximate alignments for large cases. *Theorical computer Science* , 341-374.
- Ingram, G., & Zhang, T. (2009). An Introduction to the Harmony Search Algorithm. En *Harmony search algorithms.* Spring Berlin.
- Jayanes Aguilar, L., & Zahonero Martínez, I. (2004). *Algoritmos y estructura de datos . Una perpectiva en C.* España: MacGraw-Hill.
- Johnsonbaugh, R. (2005). *Matemáticas discretas* (sexta ed.). (M. Gonzales Osuna, Trad.) México: Pearson Preice Hall.
- Kolman, B., Busy, R. C., & Ross, S. (1997). *Estructuras Mátematicas Discretas para computación.* México: Perarson Educación.
- Kuri Morales, Á., & Galaviz Casas, J. (2002). *Algoritmos genéticos.* México: IPN, UNAM, Fondo de cultura economica.
- Lee, R. C., Tseng, S. S., Chang, R. C., & Tsai, Y. (2007). *Introducción al diseño y análisis de algoritmos. Un enfoque estratégico.* México D. F.: Mc Graw-Hill.
- Leluk, J. (1998). A new algorithm for analysis of the homologya in Protein Primary Structure. *Computer Chem* , 123-131.
- Leluk, J. (2000). Regularities in mutational variability in selected protein families and the Markovian model of amino acid replacement . *Computers & Chemistry* , 659-672 .
- Lipman, D. J., Altschul, S. F., & Kececioglu, J. D. (1988). A tool for multiple sequence alignment. *Biochemistry* , 4412-4415.
- Ma, B., Wang, L., & Li, M. (2007). Near optimal multiple alignment within s bsnd in polynomisl time. *Journal of computer and system sciences* , 997-1011.
- Maciel, C., & Chiromatzo, A. (14 de mayo de 2004). *Avaliando o Significado de Seqüências Alinhadas por Algoritmos Globais.* (U. d. Catarina, Ed.) Recuperado el 15 de diciembre de 2008, de I workComo Sul: http://inf.unisul.br/~ines/workcomp/cd/pdfs/2381.pdf.
- Madrid, D. d.-U. (agosto de 2008). *Genetica.* Recuperado el 29 de agosto de 2008, de Genetica de Poblaciones: http://www.ucm.es/info/genetica/grupod/Genetica%20evolutiva/Seleccion%20natural/Seleccion%20Natural.htm
- Manthey, B. (2003). Non-approximability of weighted multiple sequence alignment. *Theoretical computer science* , 179-192.
- Martínez Castilla, L. (2008). *Reconstrucción de la historia de cambio de los caracteres*. (I. n. ecología, Editor, & Instituto nacional de ecología) Recuperado el diciembre de 2008, de http://www.ine.gob.mx/publicaciones/libros/530/cap4.pdf
- Martínez Gil, F. A., & Quetglás, G. M. (2003). *Introducción a la programación estructurada.* Valencia: Univeridad de Valéncia.
- McLachlan, A. D. (1971). Test for comparing Related Amino-acid sequences. Cytochrome C and Cytochrome c 551. *Journal moecular Biology* , 409-424.

- Melián, B., Moreno Peréz, J. A., & Moreno Vega, J. M. (2003). Metaherísticas: una visión global. *Revista Iberoamericana de Inteligencia Artificial* , 7-28.
- Moreno Pérez, J. A., & Melián Batista, B. (febrero de 2005). *Metaheurísticas para la planificación logística.* Recuperado el 15 de marzo de 2009, de webpages.ull.es/users/jamoreno/www/.../TRANSNOVA04M.pdf
- Morgenstern, B., Stoye, J., & Dress, A. (Marzo 1999). Consisten Equivalence Relations a set-Theorical Frameworks for Multiple Sequence Alignemet . *Bioinformatics, Volumen 15* , 211-218.
- Murata, M., Richardson, J. S., & Sussman, J. L. (1985). Simultaneous comparison of three protein sequences. *Biochemestry* , 3073-3077.
- Needleman, S. B., & Wunsh, C. D. (1970). A general method Applicable to the search for similarities in the Amino Acid Sequence of two proteins. *J. Mol. Biol.* , 443-453.
- Notredame, C. (2000). T-coffe: A novel method for fast and accurate multiple sequence alignment. *JMB* , 205-217.
- Omar, M. F., Salam, R. A., Abdullah, R., & Rashid, N. A. (2005). Multiple Sequence Alignment Using Optimization Algorithms. *International Journal of Computational Intelligence* , 81-89.
- Otto, J. H., & Towle, A. (1996). *Biologia Moderna.* México: McGraw-Hill / Interamericana de México.
- Perrodou, E., Chica, C., Poch, O., Gibson, T., & Thompson, J. (2000). *BALiBASE Reference Set 9 : benchmark alignments containing sequences with linear motifs (LMs)* . Recuperado el 2009, de http://www-bio3d-igbmc.u-strasbg.fr/balibase/BAliBASE_R9/index.html
- R., J. G. (22 de enero de 2008). *Los Pinzones de Darwin*. Recuperado el 28 de agosto de 2008, de http://dualjournal.wordpress.com/2008/01/22/los-pinzones-de-darwin/
- Rech, D. H., & Pilatti, R. (2004). *992Aling-UMA ferramenta para alinhamento múltiplo de seqüencias de DNA e proteínas.* Brasil: Universidade Federal de Santa Catarina.
- Restrepo-Montoya, D. (s.f.). *Aproximación al análisis de patrones en biosecuencias*. (Universidad Nacional de Colombia) Recuperado el Junio de 2008
- Sankoff, D., Cedergren, R. J., & McKay, W. (1982). A strategy for sequence phylogeny research. *Nucleic Acids Research* , 421-431.
- Sean, E. R. (agosto de 2004). Where did the BLOSUM62 alignment score matrix come from? *NATURE BIOTECHNOLOGY* .
- Smith, T. F., & Waterman, M. (1981). Identification of Common Molecular Subsequences. *J.Mol. Biol.* , 195-197.
- Solanilla, L. F., Gómez Teshima, C. A., & Múnera, L. E. (15 de diciembre de 2004). *Alineamiento de múltiples secuencias de aminoácidos usando algoritmos genéticos*. Recuperado el 01 de octubre de 2008, de http://dspace.icesi.edu.co/dspace/bitstream/item/828/1/s&t5-art3.pdf

- Sridhar, S., Lam, F., Blelloch, G. E., Ravi, R., & Schawartz, R. (2008). Mixed Integer Linear Programmig for Maximun Parsimony Phylogeny Inferenca. *Draft* , 1-20.
- Stoye, J. (1998). Multiple sequence alignment whit the divide-and-conquer method. *Gene-combis* , 45-59.
- Thompson, J. D., Plewniak, F., & Poch, O. (1999). A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research* , 2682-2690.
- Thompson, J. D., Plewnlak, F., & Poch, O. (18 de Noviembre de 1999). *BALiBASE: a benchmark aligment database for the evaluatios of multiple alignmet programs*. Recuperado el 15 de enero de 2009, de Plate- Forme Bio- Informatique the Strasbourgo: http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/
- Thompson, J. D., Plewnlak, F., & Poch, O. (1999). BALiBASE: a benchmark aligment database for the evaluatios of multiple alignmet programs. *Bioinformatics applications note , 15* (1).
- Times, T. N. (01 de agosto de 1993). *De Darwin, a la 'deriva genética'.* Recuperado el 28 de agosto de 2008, de El Pais ed. Impresa: http://www.elpais.com/articulo/sociedad/Darwin/deriva/genetica/elpepisoc/19930801elpepisoc_4/Tes
- Trelles, O. (1995). *Comparación de Secuencias Biológicas Algoritmia.* España.
- Valverde, J. R. (2006). *Bioinfor*. Recuperado el 01 de octubre de 2008, de http://imb.usal.es/bioinfor/html/Tema4_02.html
- Vélez, M. C., & Montoya, J. A. (diciembre de 2007). Metaherísticos: Una alternativa para la solución de problemas combinatorios en Administración de operaciones. *Revista EIA* , 99-115.
- Vinuesa, P. (2007). *Curso de Posgrado en Microbiología Agrícola de la Universidad Federal de Lavras, Material Suplementario 1: Algoritmos de Programación* . Recuperado el 2008 de noviembre de 22, de http://www.ccg.unam.mx/~vinuesa/curso_UFLA07/PDFs/Material_supl1_programacion_dinamica.pdf
- Zne-Jung, L., Shun-Feng, S., Chen-Chia, C., & Kuan-Hung, L. (2008). Genetic algoritm with ant colony optimizatión (GA-ACO) for multiples sequence aligment. *8*, 55-78.