INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

MODELO INTELIGENTE PARA EL ALINEAMIENTO DE BIOMOLÉCULAS

TESIS

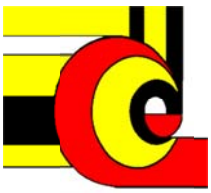
QUE PARA OBTENER EL GRADO DE
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ISRAEL ROMÁN GODÍNEZ

DIRECTORES DE TESIS:

DR. CORNELIO YÁÑEZ MÁRQUEZ
DR. CLAUDIO GARIBAY ORIJEL



MÉXICO, D.F. JUNIO DE 2011

RESUMEN

En este trabajo de tesis se propone un algoritmo para el alineamiento de biomoléculas. El modelo de memorias asociativas sobre el que está basado dicho algoritmo es el Alfa-Beta.

Usando un modelo actual de memoria asociativa no era posible llevar acabo reconocimiento de patrones en secuencias de ADN o Aminoacidos de distintos tamaños, debido a las características intrínsecas de dichas secuencias. Una de las principales limitantes es la longitud de las secuencias y sus tipos de alteraciones: inserciones, borrados y mutaciones. En el presente trabajo se desarrolla un modelo de memoria heteroasociativa Alfa-Beta robusta a este tipo de alteraciones. Dicho modelo es capaz de llevar a cabo reconocimiento de patrones y además es posible entregar un alineamiento de pares de secuencias basado en k-tuplas. El presente modelo cuenta con el soporte matemático que le brinda los modelos en los que está basado.

La segunda aportación en el presente trabajo es un algoritmo para el alineamiento de genomas. El alineamiento de genomas es una de las betas de investigación más importantes en los últimos tiempos, debido a la facilidad con la que es posible obtener información genética de los organismos vivos. Dicho algoritmo toma como base, para llevar a cabo el reconocimiento de patrones, el modelo presentado anteriormente.

Para las pruebas se utilizó el genoma de la bacteria Variovorax paradoxus. Creando a partir de dichas secuencias, un conjunto de datos de entrenamiento y de prueba que nos servirán para evaluar el desempeño del algoritmo y modelo propuesto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este trabajo de tesis se propone un algoritmo para el alineamiento de biomoléculas. El modelo de memorias asociativas sobre el que está basado dicho algoritmo es el Alfa-Beta.

Usando un modelo actual de memoria asociativa no era posible llevar acabo reconocimiento de patrones en secuencias de ADN o Aminoacidos de distintos tamaños, debido a las características intrínsecas de dichas secuencias. Una de las principales limitantes es la longitud de las secuencias y sus tipos de alteraciones: inserciones, borrados y mutaciones. En el presente trabajo se desarrolla un modelo de memoria heteroasociativa Alfa-Beta robusta a este tipo de alteraciones. Dicho modelo es capaz de llevar a cabo reconocimiento de patrones y además es posible entregar un alineamiento de pares de secuencias basado en k-tuplas. El presente modelo cuenta con el soporte matemático que le brinda los modelos en los que está basado.

La segunda aportación en el presente trabajo es un algoritmo para el alineamiento de genomas. El alineamiento de genomas es una de las betas de investigación más importantes en los últimos tiempos, debido a la facilidad con la que es posible obtener información genética de los organismos vivos. Dicho algoritmo toma como base, para llevar a cabo el reconocimiento de patrones, el modelo presentado anteriormente.

Para las pruebas se utilizó el genoma de la bacteria Variovorax paradoxus. Creando a partir de dichas secuencias, un conjunto de datos de entrenamiento y de prueba que nos servirán para evaluar el desempeño del algoritmo y modelo propuesto.

ABSTRACT

In this dissertation a model for biomolecules alignment is proposed. This model is based on Alpha-Beta associative memories.

Nowadays, there are no models of associative memories that do pattern recognition on DNA or Aminoacid sequences when the data source contains sequences of different size. Moreover, there are some other alterations like mutation, insertion and deletion, which make this problem more complex. In this work it is proposed a new model of Alpha-Beta heteroasociative memory capable of support that kind of alterations. Such a model is capable of do pattern recognition and deliver a *K*-tuple pair-wise sequence alignment of the sequence (pattern) presented to de memory and the input sequence (pattern) associate with the output pattern. The present work is based on the mathematical support of the mathematical model in which our proposal is built.

The second contribution is an algorithm for genome sequence alignment. This is a new important branch in bioinformatics due to the easy access to genetic information of organisms. The pattern recognition step of this algorithm is based on the Alpha-Beta associative memory proposed here.

The tests were made using the Variovorax paradoxus genome. From it, it was created a set of training and test data sources which help us to proof some of the characteristics of the proposed model.

1 Introducción

1.1 Antecedentes

La biología molecular ha tenido avances significativos en las últimas décadas; avances tan importantes que, para su estudio, ha sido necesario echar mano no sólo de herramientas propias, sino de otras disciplinas, como es el caso de las ciencias de la computación.

Particularmente, el desarrollo de la genómica ha sido uno de los principales generadores de información biológica, la cual había sido analizada prácticamente a mano por los expertos en el área. Pero la necesidad de acelerar dichos procesos, aunado con la velocidad en la generación de nueva información, ha conducido a un requerimiento cada vez mayor de recursos computacionales para la realización de tareas tales como: almacenamiento, organización, recuperación y principalmente visualización y análisis [1].

La Bioinformática o biología computacional (como se le conoce a la ciencia multidisciplinaria que surge de la fusión de la biología con la computación) puede ser definida en palabras simples como: "El uso de métodos computacionales para el análisis de información biológica" [2], concepto que, visto desde otra perspectiva, es el campo interdisciplinario que engloba la biología, ciencias de la computación, matemáticas y estadística para analizar secuencias biológicas, contenido genético, y predicción de funciones y estructuras de macromoléculas. De acuerdo con [3], la meta propuesta más recientemente de la bioinformática es: *permitir el descubrimiento de nuevos elementos biológicos que apoyen la creación de una perspectiva global, de la cual se puedan derivar principios biológicos uniformes.*

Dentro de esta, relativamente nueva, rama de la ciencia, los principales problemas abordados pueden ser clasificados dentro de dos categorías: *tareas genómicas* y *tareas proteómicas*. El término *tareas genómicas* es usado para denotar el estudio de secuencias conocidas como Ácido Desoxirribonucleico (ADN) o Ácido Ribonucleico (ARN), mientras que el término *tareas proteómicas*, hace referencia al estudio de todas las proteínas que surgen a partir de un genoma [4].

Dentro de las tareas más importantes y demandantes están: la construcción de árboles filogenéticos, elaboración de matrices representativas de familias de proteínas conocidas como *profiles*, la predicción de estructuras y secuencias consenso. Para llevar a cabo dichas tareas es necesario el análisis de secuencias genéticas en cuyo caso el alineamiento de secuencias, pares o múltiples, juega un papel fundamental [5][61]. Los principios básicos del alineamiento de secuencias son la similitud y homología [5]. Entendemos como similitud a los elementos que comparten ambas secuencias en las mismas posiciones y homología como el concepto que relaciona dos secuencias a través de un ancestro en común [6][7]. Además de las tareas antes mencionadas podemos encontrar, como tareas genómicas, a la *búsqueda de patrones*, la cual consiste en la localización de patrones nucleicos en una secuencia de ácido nucleico tomados de una base de datos [8], la *localización de genes* que busca la identificación automática de

genes a partir de una larga secuencia de ADN y la *identificación de promotores* que consiste en encontrar el comienzo de un gen conocido como promotor [9].

Por otro lado, dentro de las tareas proteómicas se encuentran; la *alineación múltiple de secuencias*, en la que se manejan secuencias de aminoácidos en lugar de trabajar con cadenas de ADN [2]; *búsqueda de motivos*, que tiene como principal objetivo encontrar patrones específicos en secuencias de proteínas [2] [10], y *Genómica estructural*, la cual se encarga de la predicción de estructuras tridimensionales de proteínas a partir de una secuencia primaria de aminoácidos [11]. Otra tarea, no menos importante, es la *expresión genética* que es conocida como el proceso por el cual la información que codifica un gen es traducida a una proteína [12].

Las tareas antes mencionadas son sólo algunas de las tantas que se pueden considerar dentro de la bioinformática, ya que cuanto más crece esta rama de la ciencia, más se incrementan las tareas que la componen. De los antes mencionados, uno de los primeros problemas que se abordó en el área de la genómica es el de medir la *similitud entre las secuencias* de ADN o de proteínas, ya sea dentro de una misma secuencia genómica o entre secuencias de diferentes organismos. El ADN y las proteínas pueden ser similares en términos de *función*, *de su estructura* o *de la secuencia de nucleótidos o aminoácidos*. La hipótesis fundamental respecto a las secuencias de ADN es que: dos secuencias que son similares probablemente compartan la misma función, aunque se encuentren en distintas partes del genoma o en genomas de especies diferentes. Por otro lado, la hipótesis fundamental, en el caso de las proteínas, es: dada una secuencia lineal es posible determinar la *forma* y dada ésta, la *función* [13].

Para poder medir la similitud entre dos o más secuencias genómicas o proteómicas, es necesario llevar a cabo un procedimiento que se conoce como *alineamiento de secuencias*. Dicho procedimiento se define como la comparación de dos (alineamiento de pares de secuencias) o más (alineamiento múltiple) secuencias, mediante la búsqueda de una serie de caracteres individuales o patrones de caracteres que estén en el mismo orden en las secuencias [14]. Existen dos tipos de alineamiento: *global*, con el cual se intenta alinear la secuencia completa usando tantos caracteres como sea posible con respecto a ambos extremos de la secuencia; y *local*, en el cual los tramos de una secuencia con mayor densidad de apareamientos son alineados, generando una o más islas de alineamientos o subalineamientos en las secuencias [1].

El alineamiento de secuencias es muy útil para el descubrimiento de información funcional, estructural y evolutiva en ciertas secuencias. Es muy importante obtener el *óptimo alineamiento* para que la información sea relevante. Secuencias que son muy similares, en lo que se refiere a los elementos de las secuencias, tienen alta probabilidad de tener funciones similares para el caso del ADN, y funciones bioquímicas y estructuras tridimensionales para las proteínas. Además, el hecho de que dos secuencias de organismos diferentes son similares, puede ser un indicativo de la existencia de un ancestro en común; es decir, son homólogas [1][14].

El problema del alineamiento de secuencias, en especial el de múltiples secuencias, ha sido visto como un problema de optimización. En dichos problemas se busca la optimización de una función $f(x)$ donde x es un conjunto de variables. Estos problemas pueden ser divididos en dos categorías: aquellos cuyas variables están en un dominio *continuo* y aquellos que están en un dominio *discreto,* también llamado *combinatorio*. Los problemas en esta última categoría se conocen como *problemas de*

optimización combinatoria (OC) y generalmente se distinguen entre problemas *completos* y *de aproximación.* En los problemas completos se busca obtener una solución óptima para cada instancia en un problema OC en un tiempo finito. Por otro lado, en los algoritmos de aproximación, frecuentemente llamados *algoritmos heurísticos*, se garantiza encontrar una solución *cercana a la óptima* en un menor tiempo [2][4].

Algunos de los métodos más utilizados para el alineamiento de secuencias son: *La matriz de puntos, programación dinámica y el método k-tupla o palabra.* La matriz de puntos describe un método para la comparación de aminoácidos y secuencias de nucleótidos, en la cual un grafo es dibujado colocando una de las secuencias en el eje de las **x** y la otra en el eje de las **y**. Dondequiera que haya intersección de dos palabras iguales, se coloca un punto. En la matriz resultante se buscan series de puntos que formen diagonales, los cuales revelarán patrones similares. La programación dinámica es el segundo de estos métodos y utiliza el alineamiento de secuencias tanto global como local, creados por Needleman y Wunsch [6] , y por Smith y Waterman [7], respectivamente; este procedimiento genera una matriz de números que representa todas las posibles alineaciones entre las secuencias, y el conjunto de puntuaciones más altas corresponde al alineamiento óptimo. El método k-tupla es uno de los más rápidos, y funciona de la siguiente manera: primero se buscan tramos de secuencias idénticas (conocidas como palabras) y, una vez hecho esto, los tramos son unidos mediante el método de programación dinámica [14].

Los enfoques más conocidos para resolver el problema del alineamiento de secuencias son básicamente 4: los **algoritmos exactos**, como la programación dinámica, que encuentran el *óptimo alineamiento* desde el punto de vista matemático, pero que sufre de serios problemas con la complejidad computacional. **Los iterativos** que no aseguran, matemáticamente, el óptimo alineamiento pero que estadísticamente entregan resultados fiables y en un mucho menor tiempo; entre éstos podemos encontrar el algoritmo k-tupla, los basados en *algoritmos genéticos*, *cadenas de Markov* y *búsqueda de Tabu*, *sistemas inmunes.* **Los métodos progresivos,** los cuales agregan secuencias, una a una, al alineamiento de acuerdo con una secuencia obtenida previamente. Además, están basados en los algoritmos de optimización complementados con algoritmos de agrupamientos [14][61]

Algunos de los trabajos más relevantes propuestos para el alineamiento múltiple de secuencias son los siguientes: ClustalW [62], MUSCLE [63] y Kalign [64] los cuales usan matrices de substitución para determinar el costo del alineamiento de secuencias. Existen otros que en lugar de utilizar matrices de substitución utilizan esquemas basados en consistencias (*consistency-based schemes*) como T-COFFEE [65] y los derivado de éste como M-COFFEE [66] , PCMA [67], ProbCons [68], MUMMALS[69] y MAFFT[70].

En últimas fechas se han realizado investigaciones fuertes sobre mejoras en algoritmos ya existentes, y un ejemplo de ellos se presenta en el artículo presentado por Yenatan Bilu *et. al* [15], en la cual se propone un alineamiento basado en programación dinámica, que reduce el espacio de búsqueda mediante técnicas de *segmentación de secuencias*. Otro intento de la reducción del tiempo de cómputo en la programación dinámica lo presentan Nur'Aini A. Rashid *et. al* [16], quienes proponen una reducción de aminoácidos en la matriz de substitución, de 20 a 10. Algunas otras mejoras sobre algoritmos como A*, clasificados dentro de los algoritmos de optimización, han sido

presentadas en [17]. Algunos otros investigadores, en lugar de hacer búsquedas sobre heurísticas computacionales, prefieren incursionar en la óptima implementación de dichos algoritmos en dispositivos de hardware [18].

Por último, dado que estas no se ha podido dar una solución definitiva, las propuestas en lo que respecta al alineamiento de secuencias se sigue incrementando, y el ejemplo más claro hasta ahora es Expresso [71], un nuevo algoritmo que incorpora dentro del conjunto de datos información estructural y de homología, obtenida de búsquedas hechas en bases de datos grandes mediante BLAST y PSI-BLAST. Esta nuevo enfoque es conocido como métodos de alineamiento basados en plantillas (**Template-based alignment methods**) [72][73][74][75].

La aplicación de algoritmos computacionales basados en sistemas bio-inspirados, tales como algoritmos genéticos, templado simulado (*simulated annealing*), sistemas inmunes artificiales y colonias de hormigas (*Ant-Colony*), han tomado fuerza en estos días, debido a las siguientes características que comparten: *capacidad de aprendizaje*, *tolerancia al ruido*, y su *capacidad de adaptación*. Entre éstos se encuentran aportaciones como la presentada en [19] que utiliza a los *sistemas inmunes artificiales*, [76][77] aplicando *algoritmos genéticos*, el presentado en [78][79] que utilizan el templado simulado y por último las colonias de hormigas [80][81]. Otros trabajos aprovechan el conocimiento de los expertos para poder limitar el espacio de búsqueda en el alineamiento. Dichos trabajos están clasificados en un área de computación conocida como *constrain programming* [20][21].

Cabe mencionar que algunos de los más recientes trabajos presentados tanto en [72] como [73][74][75] parecen ser el nuevo enfoque para el alineamiento de secuencias; dichos algoritmos utilizan BLAST para la búsqueda de plantillas, y BLAST utiliza principalmente algoritmos como k-tupla, el cual identifica patrones en secuencias dentro de las cadenas a alinear.

Dados los avances tecnológicos que se han presentado en las últimas fechas en los que es posible secuenciar el DNA completo de cualquier organismo, es decir su genoma, ha surgido la necesidad, no sólo de alinear posibles genes secuenciados, sino la de hacer un alineamiento que nos permita identificar dichos genes sin tener que llevar a cabo todo el proceso de experimentación biológico para poder detectarlos. La necesidad de identificar los bloques de secuencias que representan los genes, y su relación con los organismos ya conocidos se incrementa drásticamente. Por esta razón ha sido necesario crear algoritmos que puedan alinear genomas completos [37][38][39]. Estos, toman como base los algoritmos de alineamiento mencionados anteriormente, lo cual presenta un problema ya que la organización del genoma varía con lo que respecta a los genes individuales; por ejemplo, los algoritmos que se presentan en la actualidad no toman en cuenta, el fenómeno de **re-ordenamiento** que se presenta en los genomas, dicha situación perjudica en gran medida el rendimiento de dichos elementos. En el presente trabajo se presenta un modelo que ataca el problema del alineamiento de genomas, tomando en cuenta el fenómeno de reordenamiento.

Además, el presente trabajo de tesis se identifica con estos avances, tanto con la alineación de genomas como con los algoritmos basados en k-tuplas. Como una de sus principales aportaciones se utilizarán las memorias asociativas para darle flexibilidad al momento de comparar tuplas. Dicha flexibilidad se debe a la capacidad de las memorias asociativas de soportar alteraciones en los patrones [45][48][58][59]. Dado que

podemos considerar a las k-tuplas como patrones y a las mutaciones como alteraciones, podemos sugerir como una posible aportación la unión de ambas metodologías para crear un nuevo método de alineamiento. Cabe mencionar que en la literatura no se ha encontrado, hasta la fecha, nada que una estos dos campos de investigación.

1.2 Justificación

El alineamiento de secuencias y en particular de genomas es una de las tareas más importantes en la biología molecular, ya que permite identificar similitudes entre secuencias o genomas desconocidos con algunos conocidos, con el objeto de conocer sus características y de ese modo poder clasificarlas.

Actualmente, muchos de los métodos hasta ahora conocidos, como la programación dinámica, presentan una complejidad exponencial lo cual se traduce en un incremento del tiempo de procesamiento [14]. Por otro lado, existen como alternativas modelos basados en reglas heurísticas, los cuales no aseguran, por completo, la obtención del alineamiento óptimo pero sí disminuyen el tiempo de procesamiento[38][39][40][41]. La reducción del tiempo en los algoritmos de alineamiento es importante para los investigadores de las ciencias biológicas ya que les permite hacer, más rápido, inferencias basados en resultados *in silico* (análisis por computadora) en lugar de hacer experimentación *in vivo* (experimentación en vivo) la cual, por sí misma, requiere de mucho tiempo, que no es posible reducir fácilmente.

Debido a lo anterior, es importante encontrar algoritmos que cumplan con estos dos puntos importantes: la velocidad de respuesta y el soporte matemático que asegure la obtención del óptimo alineamiento. El modelo propuesto en este trabajo de tesis cumple con estos dos puntos.

1.3 Objetivo General

Proponer un modelo, basado en memorias asociativas, que permita el alineamiento de biomoléculas y de genomas. Investigar y desarrollar el soporte teórico que permita que con este modelo se asegure el alineamiento óptimo, en un tiempo considerable, disminuyendo el espacio de búsqueda mediante la identificación de patrones.

1.4 Objetivos Específicos

- Diseño de un modelo de alineamiento de genomas.
- Análisis de rendimiento de los algoritmos incluidos en la creación del modelo.
- Desarrollar el soporte teórico del modelo.
- Investigar respecto de las condiciones suficientes y necesarias para encontrar el alineamiento óptimo.

1.5 Organización del documento

En este capítulo se presentaron: los antecedentes, justificación, objetivos generales y específicos relacionados con esta tesis. La estructura general del documento se especifica a continuación:

En el capítulo 2 se presenta el estado del arte referente al alineamiento de biomoléculas. Se presentan los principales algoritmos de alineamiento de secuencias y de genomas así como los métodos que permiten evaluar los alineamientos, tanto por pares, como múltiples secuencias.

En el capítulo 3 se muestra una de las herramientas que serán utilizadas para el alineamiento: las memorias asociativas así como bases sobre el análisis algorítmico.

En el capítulo 4 se desarrollan los modelos presentados en este trabajo de tesis

En el capítulo 5 podemos ver los resultados obtenidos de las pruebas y comparaciones que se han llevado a cabo durante el presente trabajo.

El capítulo 6 nos muestra las conclusiones que emergen del presente trabajo, así como trabajos posteriores a realizar.

En el anexo A se describe uno de los principales algoritmos utilizados en el alineamiento de biomoléculas, T-COFFEE.

Además se anexa un glosario con los principales términos manejados en el ámbito de la bioinformática

Finalmente, se incluyen las referencias bibliográficas.

2 Estado del Arte

En este capítulo se abordan los principales algoritmos de alineamiento de secuencias utilizados en el ámbito de la biología molecular.

El alineamiento de secuencias es el procedimiento de comparar dos (*alineamiento de pares de secuencias*) o más (*alineamiento múltiple de secuencias*) secuencias mediante la búsqueda de caracteres individuales o patrones de caracteres que estén en el mismo orden en las secuencias [1][14]. Cada uno de estos alineamientos tiene ciertas características y cierto objetivo. El alineamiento global es útil cuando se busca saber si dos o más secuencias determinadas comparten características generales; por otro lado el alineamiento local permite buscar patrones dentro de una secuencia determinada.

2.1 Alineamiento Local vs. Alineamiento Global

Existen dos tipos de alineamiento, global y local, ilustrados en la Figura 1. En el alineamiento global se intenta alinear la secuencia por completo, usando tantos caracteres como sea posible, utilizando como tope los extremos de las secuencias. Por otro lado en el alineamiento local, se busca alinear bloques de secuencias con respecto de otra, cada uno de los bloques con límites individuales. El alineamiento local posibilita alinear secuencias que son muy similares en algunos puntos y disímiles en otros o muy diferentes en longitud; por ejemplo, para encontrar dominios y regiones conservadas compartidas. El alineamiento global es usado principalmente para alinear secuencias muy parecidas en cuanto a tamaño y contenido [14].

```
Global FTFTALILLAVAV
       F--TAL-LLA-AV

Local  FTFTALILL-AVAV
       --FTAL-LLAAV--
```

Figura 1. Alineamiento de secuencias global y local

2.2 Alineamiento: conceptos básicos

La importancia del alineamiento de secuencias surge ante la necesidad de clasificar secuencias *desconocidas* con base en grupos ya conocidos. Sus principales objetivos son obtener información estructural, funcional y evolutiva de secuencias biológicas. Es importante determinar el ***"alineamiento óptimo"*** ya que de esto depende la cantidad y calidad de la información[1][2].

La hipótesis fundamental establece que, mientras más similares sean las secuencias, sus funciones y estructura también lo serán; es por eso que si las secuencias que se comparan provienen de organismos distintos, es posible inferir que ambos organismos provengan del mismo ancestro, o no, dependiendo de la similitud en sus secuencias. A las secuencias que comparten un mismo ancestro se les conocen como *homólogas* [22], y dicha similitud está dada en función del cálculo del alineamiento óptimo.

En las secuencias homólogas el número de pasos requeridos para cambiar de una secuencia a otra se conoce como distancia evolutiva; dicho de otra manera, es la suma de pasos para cambiar de una secuencia al ancestro común y de éste a la segunda secuencia [4]. La Figura 2 es la representación gráfica, en forma de árbol, de la similitud que existe entre dos secuencias dadas, donde la distancia que existe entre un punta del árbol y el nodo en común es proporcional a la que existe entre una de las secuencias y el ancestro en común.

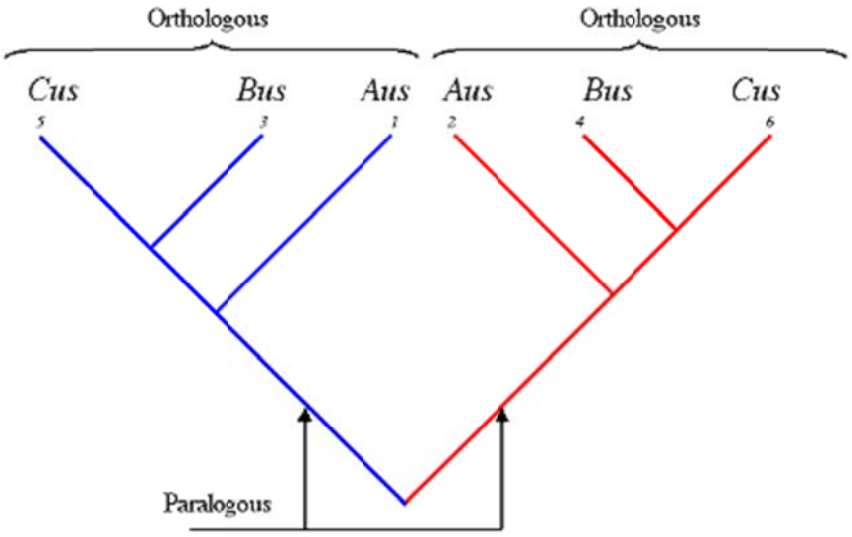


Figura 2. Relación de similitud entre secuencias

Para poder entender a qué nos referimos con "alineamiento óptimo" es necesario establecer, formalmente, el alineamiento múltiple de secuencias y sus conceptos principales [23] :

Definición 1: (Alfabeto y secuencia) *Sea* $\sum$ *un alfabeto con* $\sum \neq \phi$. *Una secuencia* s *sobre el alfabeto* $\sum$ *es una cadena finita de símbolos pertenecientes a* $\sum$. *La longitud de la secuencia* s, *denotada por* $|s|$ *es el número de los símbolos en la secuencia. Para una secuencia,* $s[i]$ *define el i-ésimo elemento de la secuencia.*

En biología molecular el alfabeto $\sum$ está dado por cuatro elementos Adenina, Timina, Guanina y Citosina, para el caso de ADN y ARN, y de 23 aminoácidos para el caso de las proteínas. Además de éstos se agrega un elemento adicional conocido como *gap* (denotado por $\{-\}$) que representa las inserciones o borrados de elementos en una secuencia.

Definición 2: (Seudo Alineamiento Múltiple) *Sea* $\sum$ *un alfabeto y* S *un conjunto de* k *secuencias,* $\{s_1 ..... s_k\}$, *sobre el alfabeto* $\sum$. *Un Seudo alineamiento múltiple es un conjunto de secuencias* $S' = \{s_1 ' ..... s_k '\}$ *sobre el alfabeto* $\sum' = \sum \cup \{-\}$ *donde:*

- $k \geq 2$

- *Todas las secuencias en* S' *tienen la misma longitud:* $\forall s_i', s_i' \in S' \left| s_i' \right| = \left| s_i' \right|$

- $\forall i \le k, S_i' \in S'$ *pueden ser reducidas a su correspondiente secuencia* $s_i \in S$ *removiendo todos los símbolos de gap* $\{-\}$ *de* s_i'.

- *El orden de los símbolos en cada secuencia* $s_i' \in S'$ *es la misma que la correspondiente secuencia* s_i

El número de la secuencia de en el alineamiento es $\left| S' \right|$*; y el número de columnas es definido como la longitud de la secuencia* S'.

$PMSA(S)$ *es el conjunto de todos los posibles seudo alineamientos de la secuencia* S.

El conjunto de seudo alineamientos múltiples de secuencias es un conjunto infinito dado que el número de columnas consistentes a sólo a *gaps* no esta restringida. Lo que significa que no hay una cota para la longitud de la secuencia en un seudo alineamiento.

Definición 2.3 (Alineamiento Múltiple de Secuencia) *Sea* n *el número de secuencias en el alineamiento y* k *el número de columnas. Un conjunto de secuencias* S *es un alineamiento múltiple si y sólo si es un seudo alineamiento y cumple con las siguientes condiciones.*

$$\forall i \le k, \exists j \le n, s_j[i] \ne -$$

$MSA(S)$ *denota el conjunto de todos los posibles alineamientos múltiples de la secuencia* S.

Esta definición limita el conjunto de las secuencias del seudo alineamiento óptimo a aquellos alineamientos que no permiten que una columna consista sólo de *gaps.*

Definición 2.4 (Función Objetivo) *Sea* $\sum'$ *definido en 2.2. Para un número de secuencias n, una función*

$$f : (\sum'^{*})^n \to \Re$$

Es llamada la función objetivo (FO), y $\sum^*$ *se define como el conjunto de todas las palabras obtenidas por la concatenación de símbolos del alfabeto* $\sum$.

Dicha función permite asignar un valor que represente la calidad del alineamiento múltiple hecho con un n número de secuencias.

Definición 2.5 (Alineamiento Óptimo) *Para un conjunto de secuencias* $S = s_i ... s_k$ *del alfabeto* $\sum$ *y una FO* f, $S = s_i' ... s_k'$ *es un alineamiento óptimo si y sólo si:*

- $S' \in MSA(S)$

- $\forall S^* \in MSA(S) : S^* \ne S' \to f(S^*) \le f(S')$

El segundo punto establece que $f(S')$ es el máximo valor de la función f en el conjunto de todos los alineamientos en S. En esta definición se incluye la posibilidad de más de un alineamiento óptimo.

Esta última definición establece el concepto de ***alineamiento óptimo***.

2.3 Alineamiento de pares de secuencias

Existen varios métodos para el alineamiento de secuencias; entre los más usados están: análisis de matriz de puntos, programación dinámica y método k-tupla o palabras.

2.3.1 Análisis de matriz de puntos

El análisis de matriz de puntos es un método preliminar en el alineamiento de pares de secuencias, y fue descrito por primera vez en 1970 por Gibas y McIntyre [24]. Este método es usado para encontrar repeticiones de caracteres, de izquierda a derecha o viceversa, en secuencias de proteínas o de ADN, o para la predicción de regiones en ARN que sean autocomplementarias. La principal ventaja de este método es que encuentra todas las posibles relaciones entre los elementos de dos secuencias, y el alineamiento de estas dos secuencias se deja a otro tipo de algoritmos; por ejemplo, programación dinámica.

Este método se considera dentro del alineamiento de pares de secuencias, ya que sólo es posible comparar dos secuencias. El procedimiento de este método es el siguiente: se tienen dos secuencias, la secuencia (A) es colocada en la parte superior de una matriz y la segunda, denominada (B), se coloca de arriba a abajo al lado izquierdo de la misma, de tal forma que se forme una especie de tabla de verdad o matriz (ver Figura 3). Comenzando con el primer carácter en la secuencia (B) se recorre la secuencia (A) colocando un punto en la fila del primer carácter donde los elementos en ambas secuencias sean iguales. Posteriormente, se selecciona el segundo elemento de la secuencia (B) y se recorre nuevamente la secuencia (A), un punto en la segunda fila es colocada donde los elementos de la secuencias sean iguales. Este proceso es llevado a cabo hasta que se hayan recorrido todos los elementos de las dos secuencias. Las secuencias similares se muestran por secuencias de puntos en diagonal. Puntos aislados que no conforman alguna diagonal, representan concordancias aleatorias que probablemente estén relacionadas con secuencias no significativas. Las gráficas que resultan de este proceso pueden ser más representativas si se aplica sobre ellas un filtro que permita identificar dichas regiones. La Figura 4. muestra cómo se colocan las secuencias en el plano y el resultado de este proceso [24][14].

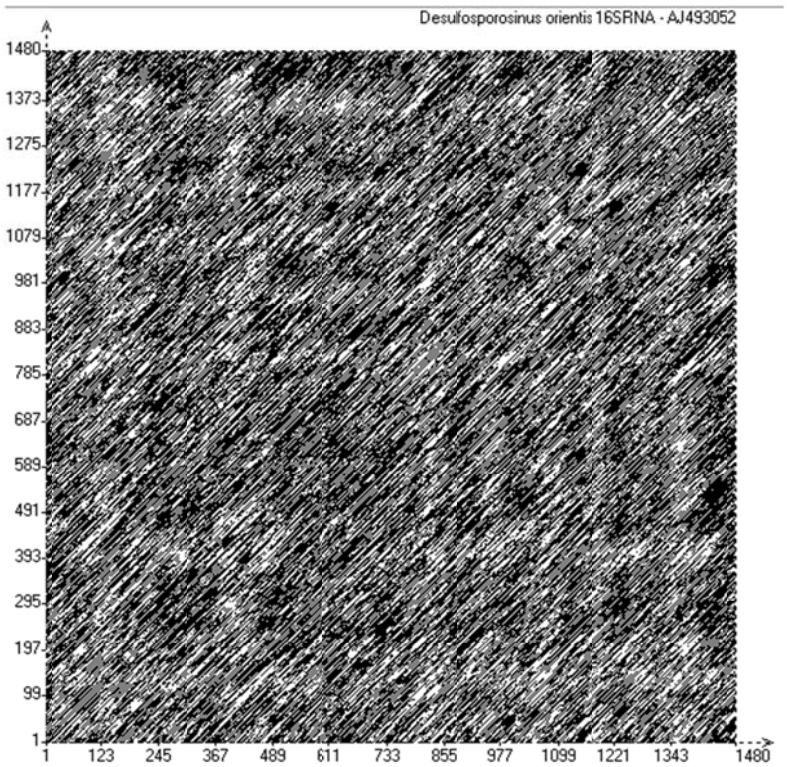


Figura 3. Matriz de puntos.

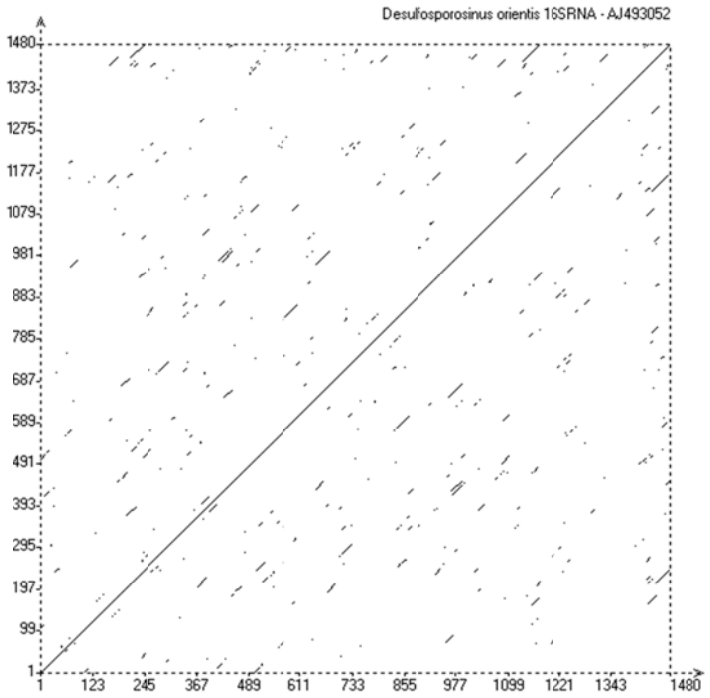


Figura 4. Matriz de puntos filtrada.

Ya que es muy probable que en la matriz de puntos se muestre mucha información irrelevante, es necesario seleccionarla, por lo que la aplicación de un filtro es de mucha ayuda. Los filtros analizan la vecindad con respecto a un punto y deciden su estado respecto a ello, es decir, un punto es colocado en cierta posición si un número determinado de puntos está en la vecindad determinada. El típico tamaño de la ventana (vecindad) para una secuencia de ADN es de 15 y el requerimiento mínimo de concordancias entre los pares de secuencias que caen dentro de la ventana es 10. Las proteínas regularmente no son filtradas, pero en caso de ser necesario, la ventana regularmente usada es de 2 o 3 y el requerimiento de concordancias es de 2.

Otra de las utilidades de la matriz de puntos es la búsqueda de patrones dentro de una secuencia, ya que si esto sucede se verá reflejado en la gráfica como bloques de diagonales repetidos intermitentemente a través de la diagonal principal.

2.3.2 Programación Dinámica

La programación dinámica es un método computacional usado para alinear dos proteínas o ácidos nucleicos. Este método es muy importante ya que encuentra el alineamiento óptimo entre secuencias; sin embargo, el método requiere del uso inteligente de varias variables en el programa.

Este método compara cada par de secuencias y genera un alineamiento. Este alineamiento incluye concordancias y desajustes de caracteres y de gaps (un espacio en blanco simbolizado por un guión (-) representa un carácter cualquiera) en las secuencias comparadas de tal forma que el número de concordancias entre caracteres idénticos o relacionados sea el máximo.

Tanto el alineamiento global como el local pueden ser llevados a cabo utilizando la programación dinámica, haciendo pequeños cambios en el algoritmo. Los programas de alineamiento global utilizan el algoritmo Needleman-Wunsch, y los programas de alineamiento local utilizan el algoritmo Smith-Waterman [6][7].

2.3.3 Alineamiento Global: Algoritmo Needleman-Wunsch

1. Las secuencias son colocadas en una matriz cuadrada de manera similar a como lo hace la matriz de puntos, la secuencia A, con $A = a_1 a_2 a_3 ... a_i$, es colocada de izquierda a derecha en la parte superior de la matriz y la secuencia B, con $B = b_1 b_2 b_3 ... b_j$, de arriba abajo en la parte izquierda de la misma. Una fila y columna extras son agregadas en la primera posición de la matriz para poder contabilizar los gaps. La fila de los gaps es llenada con una penalización, la primera contiene un 0, y a partir de ésta las posteriores posiciones se van incrementando de acuerdo con el valor establecido para el gap.

2. El valor máximo para cada una de las componentes es calculado a partir de la siguiente ecuación:

$$S_{ij} = \max\{S_{i-1,j-1} + s(a_i b_j), (S_{i-1,j} - w_x), (S_{i,j-1} - w_y)\}$$

donde S_{ij} es la puntuación en la posición *i* de la secuencia A y de la posición *j* en la secuencias B, $s(a_i b_j)$ es el valor tomado de la *matriz de sustitución* en la posición *i*

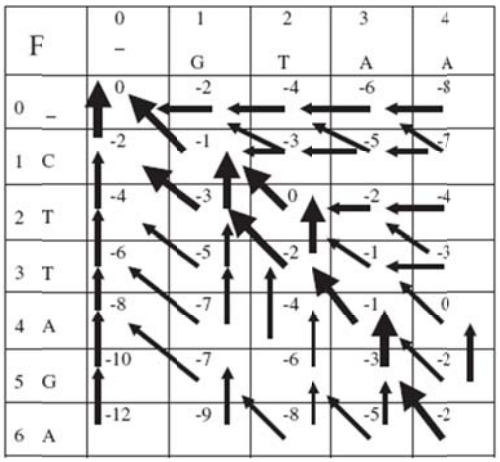
y j, w_x es la penalización por un gap de longitud x en la secuencia A, y w_y es la penalización por un gap de longitud y en la secuencia B.

3. Se recorre la matriz y se almacenan los caminos que produzcan la máxima puntuación para alcanzar cada una de las posiciones de la matriz. Dichos caminos son almacenados en una matriz conocida como trace-back.

4. Por último, los caminos en la matriz trace-back son unidos para producir el alineamiento.

Cada elemento de la *matriz de sustitución* muestra la razón entre la probabilidad de que un par de aminoácidos sea encontrado en un alineamiento de proteínas relacionadas y la probabilidad de que el mismo par de aminoácidos esté alineado por casualidad en la secuencia, dado que algún aminoácido sea abundante en proteínas; lo mismo sucede para el ADN y ARN.

Ejemplo 1: Supongamos que $x = CTTAGA$ y $y = GTA$ son dos secuencias a ser alineadas. Supongamos que usamos el siguiente esquema de puntuación: $s(a,a) = 1, s(a,b) = -1$ si $a \neq b$ y $s(-,a) = s(a,-) = -2$. La matriz resultante se muestra en la siguiente figura.



Siguiendo las líneas que marcan el alineamiento óptimo en la matriz, los alineamientos resultantes son los siguientes:

x:	C	T	T	A	G	A
y:	G	-	T	A	-	A

x:	C	T	T	A	G	A
y:	G	T	-	A	-	A

x:	C	T	T	A	G	A
y:	-	G	T	A	-	A

2.3.4 Alineamiento Local: Algoritmo Smith-Waterman

Las reglas para calcular la matriz de resultados es muy similar excepto por lo siguiente: 1) la matriz de sustitución debe incluir valores negativos para los desajustes y 2) cuando en la matriz de puntuación los valores se conviertan en negativos, el valor es puesto a 0, lo que le da un efecto de terminación de cadena.

El alineamiento es producido comenzando desde la posición con los valores más altos en la matriz de puntuación, continuando por la ruta de aquellas posiciones que lo lleven a una posición que contenga 0.

La siguiente ecuación es una modificación a la propuesta por Needleman y Wusch, con los cambios correspondientes.

$$H_{ij} = \max\left\{H_{i-1,j-1} + s(a_i b_j), (H_{i-1,j} - w_x), (H_{i,j-1} - w_y), 0\right\}$$

donde H_{ij} es la puntuación en la posición *i* de la secuencia A y de la posición *j* en la secuencias B, $s(a_i b_j)$ es el valor tomado de la *matriz de sustitución* en la posición *i* y *j*, w_x es la penalización por un gap de longitud *x* en la secuencia A, y w_y es la penalización por un gap de longitud *y* en la secuencia B.

Ejemplo 2: Para las secuencias del ejemplo 1, el mejor alineamiento local es:

x:	T	A
y:	T	A

F	0 –	1 G	2 T	3 A	4 A
0 –	0	0	0	0	0
1 C	0	0	0	0	0
2 T	0	0	1	0	0
3 T	0	0	1	0	0
4 A	0	0	0	2	1
5 G	0	1	0	0	1
6 A	0	0	0	1	1

Existe un método estadístico que calcula si el alineamiento obtenido es significativo o no, el cual consiste en que una de las secuencias es cortada y mezclada varias veces de manera aleatoria; posteriormente es alineada con la segunda para demostrar que el alineamiento original es único. Para secuencias que tienen aproximadamente 95% de similitud, el alineamiento de secuencia es obvio y la computadora tardaría más de lo que

una persona lo haría; por otro lado, si las secuencias son cada vez más disímiles el alineamiento se vuelve cada vez más complejo y, por lo tanto, existen más dudas sobre su exactitud. En el caso de las secuencias proteínicas, la similitud puede ser medida a niveles del 25% de similitud, pero a este nivel el número relativo de alineamientos desiguales y colocación de gaps en el alineamiento tiene que ser empírico.

2.3.5 Algoritmo K-tupla o Palabras

Estos algoritmos son heurísticas que no garantizan la obtención del óptimo alineamiento de la secuencia pero que son significativamente más eficientes. Se basan principalmente en la identificación de secuencias pequeñas (conocidas como palabras) dentro de otras más grandes, sin superposición de secuencias. Estos métodos son principalmente utilizados cuando se intenta alinear una secuencia con respecto a una gran cantidad de secuencias. La posición relativa de una palabra con respecto a dos secuencias que están siendo comparadas es obtenida restando cada una de estas posiciones y obteniendo un valor conocido como offset. Dicho valor ayudará en la determinación del alineamiento si múltiples palabras producen el mismo offset. FAST y BLAST son dos importantes métodos que utilizan este tipo de heurísticas para el alineamiento de secuencias [25][26].

Uno de los primeros y más utilizados algoritmos es el desarrollado por W. J. Wilbur y David J. Lipman [25] quienes propusieron un método de búsqueda rápida de secuencias en bases de datos grandes. El método consiste de los siguientes pasos:

Sean S1 la primera secuencia y S2 la segunda, dada una *k* y una *p*, siendo *k* el tamaño de las tuplas y *p* la dimensión del alfabeto de las secuencias.

1. Se localizan las secuencias que son idénticas entre dos secuencias.

2. Para no considerar todas las posibles secuencias iguales se discriminan aquellas secuencias que no cumplan con cierta condición. En el caso de esta propuesta se plantea crear un conjunto de secuencias que se encuentren a una distancia *m* una de la otra, donde $m = i - j$, con *i* y *j* las posiciones de los comienzos de las *k*-tuplas de S1 y S2.

2.4 Alineamiento múltiple de secuencias

Los métodos para el alineamiento de pares de secuencias se limitan a la comparación de dos secuencias, por ello cuando se tienen más de dos secuencias y se quiere alinear todas, es necesario utilizar métodos de alineamiento múltiple de secuencias.

En el alineamiento múltiple de secuencias se busca obtener el máximo número de caracteres similares en una columna del alineamiento parecido al que se obtiene en el alineamiento de pares de secuencias. Los retos principales dentro de este método son: encontrar el alineamiento óptimo de más de dos secuencias que incluya concordancias, gaps y elementos disímiles, además de tomar en cuenta el grado de variación de todas las secuencias al mismo tiempo.

En el alineamiento múltiple se intenta encontrar la relación evolutiva entre las secuencias, ya que en un alineamiento, si las secuencias tienen una alta similitud,

entonces se podrá decir que han derivado recientemente del mismo ancestro; por otro lado, si no existe un buen alineamiento, las secuencias tienen una relación más compleja y distante.

Un método utilizado para esta tarea es la programación dinámica, la cual presenta un grave problema, ya que al aumentar el número de secuencias aumenta considerablemente la complejidad. Debido a esto, ha sido necesario desarrollar otro tipo de métodos como *los progresivos*, los cuales parten de alinear las secuencias más similares y van agregando secuencias, una por una, hasta completar el análisis; otro de los métodos se basa en encontrar patrones conservados localmente en el mismo orden de la secuencia. También la *estadística* ha jugado un papel importante, proporcionando las Cadenas Escondidas de Markov como principal herramienta [1][14].

Algunos otros retos dentro de la secuenciación múltiple son: el de identificar un método razonable para obtener la puntuación acumulativa en las columnas del alineamiento múltiple, además de la colocación y asignación de puntos de los gaps en varias secuencias.

2.4.1 Programación Dinámica

El método de programación dinámica también se utiliza para realizar alineamientos múltiples de secuencias [7]. El algoritmo es el mismo: se llena completamente la matriz de puntuación y se realiza una búsqueda por todo el espacio hasta encontrar el camino que contenga las máximas posiciones. Su gran desventaja es que el número de pasos empleados para llevar a cabo un alineamiento crece exponencialmente conforme el número de secuencias que van a ser analizadas, además de la cantidad de memoria. Por lo tanto, dicho método está limitado a un número pequeño de secuencias.

Si para el caso de la alineación de pares de secuencias el espacio de búsqueda es de $m \times n$ donde n y m son las longitudes de las secuencia A y B, respectivamente, para el alineamiento de tres secuencias, con k la longitud de la nueva secuencia, el espacio de búsqueda sería de $m \times n \times k$ y se iría incrementando de esta manera cada vez que se agreguen secuencias al alineamiento. Un caso particular es aquel con las secuencias del mismo tamaño; en este caso se podría hablar de un crecimiento exponencial.

Para poder reducir el espacio de búsqueda se crean atajos como el siguiente: la idea básica para el alineamiento múltiple de secuencias es realizar alineamientos en pares de secuencias. Primero se crean todos los posibles pares y se llevan a cabo sus respectivos alineamientos, posteriormente se crea un árbol filogenético (representación, en forma de árbol, de las distancias evolutivas entre los organismos) usando los alineamientos producidos por los pares y, a partir de éste, se produce un nuevo alineamiento tomando siempre los elementos más cercanos, de tal manera que dos secuencias queden representadas por una, y así sucesivamente hasta alcanzar un único alineamiento. Una vez que se tienen ambos alineamientos, por programación dinámica y por heurística, se realiza una proyección del alineamiento heurístico hacia todas las caras del cubo y con ello se limita el espacio de búsqueda de todo el cubo. Se realiza una proyección de las áreas encontradas en las caras, y la intersección de esta proyección es el área en donde se busca el mejor alineamiento. Al final se busca el mejor alineamiento en el volumen dado.

2.4.2 Métodos progresivos

El método progresivo de alineamiento múltiple de secuencias es una alternativa de solución para el problema del espacio de búsqueda exponencial que muestran los algoritmos anteriores. Este algoritmo busca, dentro de un grupo, las secuencias más relacionadas y utiliza el método de programación dinámica de alineamiento múltiple de secuencias para generar un alineamiento óptimo [27][28]. Progresivamente se van agregando secuencias menos relacionadas hasta completar el alineamiento [14][1]. La relación evolutiva entre las secuencias es creada mediante métodos de construcción de árboles filogenéticos, por ejemplo *neighbor-joining* [29].

2.4.3 CLUSTAL

Clustal nació hace más de 20 años [30][31], y ha sido utilizado y mejorado por muchas personas en este tiempo. Dos de las más recientes mejoras de dicho algoritmo son CLUSTALW y CLUSTALX; el primero tiene la posibilidad de asignar pesos a las secuencias y parámetros del programa, y el segundo es una versión gráfica del algoritmo.

Este método realiza alineamiento global múltiple de acuerdo con los siguientes pasos:

1. Crea todos los posibles pares de secuencias y realiza, para cada par, su correspondiente alineamiento.

2. Utiliza el alineamiento para producir un árbol filogenético.

3. Alinea las secuencias, guiándose con la relación evolutiva marcada por el árbol filogenético. Es decir, se lleva a cabo un alineamiento múltiple con las secuencias más relacionadas, y una vez que se tiene el alineamiento, se agrega una secuencia o un grupo de secuencias y se alinean con respecto al alineamiento previo para producir un alineamiento múltiple que muestre en las columnas la variación de la secuencia en las secuencias.

Para el primer alineamiento se pude usar alineamiento múltiple utilizando programación dinámica o el método k-tupla [32][33].

La contribución de las secuencias al alineamiento múltiple es medida de acuerdo con la relación evolutiva mostrada en el árbol filogenético. El peso de cada secuencia, es entonces la distancia que hay entre la secuencia y la raíz. Por lo tanto, el puntaje del alineamiento es considerado un factor del peso.

La puntuación de los gaps en este método debe de ser distinta a la que se lleva a cabo en el alineamiento de pares de secuencias, ya que cada secuencia que se agrega al alineamiento múltiple tiene una influencia en los alineamientos futuros. Para ello, Pascarella y Argos [32] prepararon una tabla de frecuencias de los gaps adjuntos a ciertos aminoácidos, y se dieron cuenta de que en el alineamiento de un conjunto de proteínas relacionadas, los gaps se encontraban en ciertas zonas, llamados elementos estructurales secundarios. CLUSTALW utiliza esta tabla e intenta determinar dichas zonas para poder hacer inserciones de gaps.

Como cualquier otro programa CLUSTALW utiliza ciertas penalizaciones, como cuando es introducido un gap en la secuencia o cuando un gap incrementa su longitud.

Dicha penalización se ajusta dependiendo del promedio de concordancias que haya entre la nueva secuencia y el alineamiento existente, con el objetivo de afectar lo menos posible las secuencias que contengan mayor número de concordancias y secuencias que contengan longitudes menores a las del alineamiento existente [34].

Por último, cada vez que el algoritmo agrega una nueva secuencia al alineamiento, el árbol filogenético es recalculado.

2.4.4 PILEUP

Este programa es parte de un paquete del Grupo de Computación Genética para el análisis de secuencias, de la universidad de Oxford. El método es muy similar el presentado por el CLUSTALW.

1. Se crean todos los posibles pares de secuencias y se alinean utilizando el algoritmo de Needleman-Wunsch.

2. El resultado de todos los alineamientos es utilizado para crear un árbol filogenético mediante el método UPGMA (*unweighted pair-group method*) [33].

3. Utilizando dicho árbol como guía, se identifican las secuencias más relacionadas y se lleva a cabo el alineamiento de las secuencias, agregando una a una las secuencias menos relacionadas.

El resultado es un alineamiento global, y las penalizaciones para gaps y disimilitudes de la matriz de puntuación.

Los principales problemas que presenta este tipo de métodos es la dependencia al alineamiento de pares de secuencias producido en primera instancia. Si estas secuencias alinean muy bien, es decir, están muy relacionadas, entonces habrá pocos errores, de otra forma las secuencias serán muy distantes y el error será mayor.

2.4.5 Métodos iterativos

Los métodos iterativos nacen como un intento por remediar el problema de dependencia al primer alineamiento que presentan los algoritmos progresivos. La solución que presenta es alinear repetidamente subgrupos de las secuencias y posteriormente dicho subgrupo alinearlo de manera global. El objetivo es mejorar la puntuación de alineamiento. La selección de los grupos puede ser hecha de acuerdo con el orden del árbol, separación de secuencias del resto o selección aleatoria [13][14].

2.4.6 Algoritmos Genéticos

Los algoritmos genéticos son un método de machine-learning diseñado para optimización de funciones inspirado en el proceso biológico de la herencia. Este método define dos algoritmos similares a la *mutación* y la *recombinación* bautizándolos con el mismo nombre. El primero de ellos es la mutación, en la cual se modifican los elementos de la secuencia, cambiando alguno de los existentes por otro de los elementos posibles. El segundo es la recombinación, en la cual se toman trozos de dos distintas secuencias y se intercambian colocándose en posiciones seleccionadas de manera aleatoria [2][14][1].

Los algoritmos genéticos fueron adaptados para el alineamiento de secuencias [34][35] y es de considerable interés ya que puede encontrar el alineamiento óptimo en un menor tiempo que los algoritmos basados en optimización exhaustiva.

La idea principal del método es tratar de generar muchos alineamientos múltiples mediante el reacomodo de secuencias que simule la inserción y borrado de gaps, con el objetivo de alcanzar la puntuación máxima.

Los pasos en el algoritmo son:

1. Se generan varios conjuntos de secuencias para generar varios alineamientos múltiples. Para cada uno, se alínea un número de elementos, generado aleatoriamente, en todas las secuencias de todos los alineamientos. Los extremos son llenados con gaps para hacer que todas las secuencias midan lo mismo.

2. Se contabiliza la puntuación de los alineamientos mediante el método de suma de pares para obtener los mejores alineamientos. Se aplican penalizaciones para gaps y fallas en el alineamiento.

3. Del total de alineamientos múltiples, se selecciona la mejor mitad y se pasa intacta a la siguiente generación. De la segunda mitad, las alineaciones múltiples son seleccionadas por sorteo.

4. El conjunto de alineamientos múltiples que se seleccionó por sorteo, se mutan. La mutación consiste en inserción y reacomodo de gaps.

5. Se recombinan los alineamientos múltiples, tanto los escogidos primeramente como los mejores, así como los que se mutaron.

6. Se buscan nuevas intersecciones entre las nuevas secuencias, se evalúan y se repiten los pasos del 2 al 5, el número de repeticiones puede ser de 100 a 1000; y se obtiene el mejor puntaje.

2.4.7 Cadenas Escondidas de Markov (HMM)

Son modelos estadísticos que consideran todas las posibles combinaciones de coincidencias, no-coincidencias y gaps para generar un alineamiento de secuencias. Inserciones y borrados pueden ser modelados por las HMM [2][14]. Una de sus cualidades, es que describe una serie de observaciones a través de un proceso estocástico. A partir de un alineamiento existente, cada una de las columnas se considera un estado, y después se cuenta el número de bases por columna y se asigna probabilidad a cada una de las bases y se prueba [36].

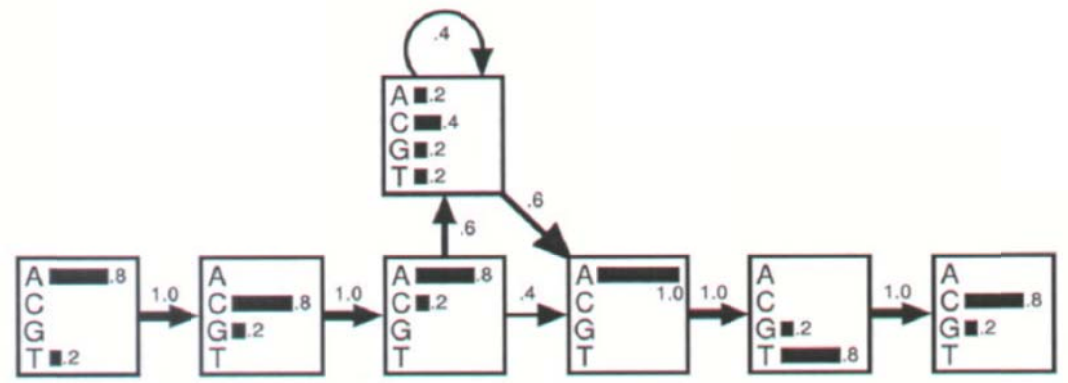


Figura 5. Cadena de Markov

2.5 Métodos de evaluación

Los métodos para medir la relevancia biológica de un alineamiento, básicamente pueden ser de dos formas: A) análisis estadístico, b) comparación con alineamientos hechos manualmente [14].

2.5.1 Análisis estadístico

El análisis estadístico para determinar si un alineamiento es significativo o no lo es, se basa principalmente el cálculo de la probabilidad de que dos secuencias no relacionadas se alineen y con el cálculo de su correspondiente puntuación.

Existen varias propuestas, pero dos son las más importantes: la basada en que las secuencias alineadas cumplen con una distribución estadística y la que asume que sigue una distribución conocida como *gumbel extrema distribution* [14][4][2].

Para ambos casos el principio es el mismo: una vez que se ha realizado el alineamiento de dos secuencias A y B y se ha obtenido una puntuación para dicho alineamiento, las secuencias son modificadas de manera aleatoria cortándolas en pequeños segmentos y posteriormente uniéndolos para formar dos nuevas cadenas a y b. Una vez hecho eso se calcula la puntuación para dicho alineamiento. Este proceso de creación y alineamiento de las secuencias se lleva a cabo un número representativo de veces, de tal forma que sea posible calcular la distribución de dicha puntuación[14].

Para el caso de que se calcule la distribución normal de dichos datos se verifica cuál es la desviación estándar que tiene el alineamiento de las secuencias originales. Según la estimación presentada en algunos artículos, la desviación estándar de un alineamiento que tenga sentido debe estar a por lo menos 3-5 desviaciones estándar; lo que nos dice que la probabilidad de que un alineamiento aleatorio de la puntuación del original, es muy baja [14] [4].

Para el caso del segundo método de validación, la única diferencia está en que se utiliza la distribución gamble extrema. Esta surge ya que se han dado cuenta de que al alinear secuencias que están muy alejadas evolutivamente hablando, la distribución se asemeja más a ésta que a una distribución normal. Esto tiene sentido desde el punto de vista que,

cuando las secuencias se van alejando cada vez más, existen ciertos patrones en las secuencias que tienden a prevalecer.

2.5.2 Comparación con alineamientos hechos manualmente

La forma con la que se evalúa el alineamiento de forma estadística tiene sentido, matemáticamente hablando; pero para el caso del alineamiento de secuencias, es más importante que tenga un sentido biológico. Citando al artículo donde se presenta clustal [30] se menciona lo siguiente: *"Si es imposible mejorar el alineamiento a simple vista (de forma manual), entonces se habrá obtenido un buen resultado"*. Esto nos dice que la principal medida para evaluar un buen alineamiento es el que es llevado a cabo por un experto.

Es por eso que grupos de personas se han dedicado a hacer alineamientos de secuencias, combinando alineamientos automáticos con ajustes hechos manualmente, creando bases de datos de prueba (*benchmarks)* que permitan probar los alineamientos que se hacen de manera automática [2].

Algunas de las principales bases de datos de prueba son:

1. BAliBASE
2. OXBENCH
3. PREFAB
4. SABmark
5. IRMBASE

La dinámica al realizar pruebas en estas bases de datos radica en que se cuenta con un conjunto de secuencias que están alineadas manualmente, y se brinda al usuario el mismo conjunto de secuencias pero sin alinear. La idea es comparar el alineamiento automático (el del usuario) con el de la base de datos y entregar una puntuación.

2.5.3 Alineamiento de genomas

Los algoritmos de alineamientos de genomas están clasificados en dos tipos: alineamiento global y alineamiento local [37]. El objetivo de alineamiento global es alinear genomas completos para mostrar que existe un orden, entre los genomas comparados, en sus características biológicas, por lo que su uso es más conveniente cuando se desea alinear genomas de organismos cercanos o cuando se quiere ayudar en la secuenciación de un genoma.

Por otro lado, cuando comparamos dos o más genomas es posible que los genes que conforman a un genoma, estén en un orden diferente comparado con el otro este fenómeno es conocido como *re-ordenamiento*. Este re-ordenamiento de los genes ocasiona que el alineamiento global de los genomas dé como resultado una puntuación baja ya que producirá pocas coincidencias entre las secuencias. Es por esta razón que alineamientos locales serían más convenientes, ya que podrían localizar similitud en genes que se encuentren en un orden distinto en las secuencias.

Algunos de los principales algoritmos desarrollados para el alineamiento de genomas son GRIMM-SYNTENTY[38], CHAINNET [39], CP [40], MAUVE [41], BLASTZ [42].

La estrategia tomada, en general, por estos algoritmos es la siguiente:

1. Se detectan regiones potencialmente homólogas conocidas como anclas.
2. Se filtran las anclas, eliminando falsos positivos y se elige alguno de los elementos duplicados.
3. Se alinean las secuencias potencialmente homólogas.

El primer paso, la búsqueda de anclas, generalmente es similar en todos los algoritmos, mientras que los últimos dos difieren dependiendo del objetivo que busquen, sólo alinear, o estudiar la dinámica del genoma.

Las técnicas de búsquedas de anclas son, básicamente, las utilizadas en BLAST, también conocida como siembra-y-extiende (*seed-and-extend*), donde se encuentra una k-tupla que sea coincidente entre las secuencias que se comparan y posteriormente se va extendiendo siempre y cuando mantenga una puntuación sobre un umbral. Otro método de localización de anclas es el utilizado por Pattern Hunter llamado sembrado-espaciado (*spaced-seeds*) en el que una k-tupla de tamaño l debe de coincidir en al menos un número x de posiciones, donde éstas no tienen que ser contiguas. Dichas posiciones son determinadas por el usuario [37]. Por último MAUVE es un algoritmo que utiliza un método diferente de anclaje, el cual requiere de una coincidencia total del elemento buscado y además éste debe de ser único [37].

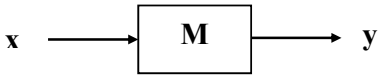
2.6 Memorias Asociativas

En esta sección se describen los conceptos básicos de memorias asociativas y se muestran, además, los modelos más representativos anteriores a las memorias asociativas Alfa-Beta.

2.6.1 Conceptos básicos

Los conceptos presentados en esta sección se han tomado de las referencias que, a nuestro juicio, son las más representativas [43][44][45][46][47].

Una *memoria asociativa* **M** puede visualizarse como un sistema de entrada y salida, donde los patrones de entrada y salida están representados por vectores columna denotados **x** y **y** respectivamente. A continuación se muestra un esquema de la idea anterior.



Cada uno de los patrones de entrada forma una asociación con el correspondiente patrón de salida. Tal asociación es similar a la una pareja ordenada; por ejemplo, los patrones **x** y **y** del esquema anterior forman la asociación (**x**,**y**).

No obstante que a lo largo de este capítulo se manejaran algunas de las notaciones originales establecidas en los distintos modelos, a continuación se presenta la nomenclatura que será manejada en lo sucesivo, para la descripción de los conceptos básicos sobre las memorias asociativas y el resto de esta tesis.

Los patrones de entrada y salida se denotarán con las letras negrillas, **x** y **y**, agregándoles números naturales como superíndices para efectos de discriminación simbólica. Por ejemplo, a un patrón de entrada $\mathbf{x}^1$ le corresponderá el patrón de salida $\mathbf{y}^1$, y ambos formarán la asociación $(\mathbf{x}^1,\mathbf{y}^1)$; del mismo modo, para un número entero positivo *k* específico, la asociación correspondiente será $(\mathbf{x}^k,\mathbf{y}^k)$.

La memoria asociativa **M** se representa mediante una matriz, la cual se genera a partir de un conjunto finito de asociaciones conocidas de antemano, a este nuevo conjunto se le conoce como **conjunto fundamental de aprendizaje**, o simplemente **conjunto fundamental**.

El conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\}$$

donde *p* es un número entero positivo que representa la cardinalidad del conjunto fundamental.

A los patrones que conforman las asociaciones del conjunto fundamental se les llama **patrones fundamentales**. La naturaleza del conjunto fundamental proporciona un importante criterio para clasificar las memorias asociativas:

Una memoria es **Autoasociativa** si se cumple que $\mathbf{x}^\mu = \mathbf{y}^\mu \ \forall \mu \in \{1, 2, ..., p\}$, por lo que uno de los requisitos que se debe de cumplir es que $n = m$. De otra manera si $\exists \mu \in \{1, 2, ..., p\}$ para el que se cumple que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$ se trata de una memoria **Heteroasociativa**. Nótese que puede haber memorias heteroasociativas con $n = m$.

Con el fin de especificar las componentes de los patrones, se requiere dejar en claro cual será la notación para dos conjuntos a los que se llamarán arbitrariamente A y B. Las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, serán elementos del conjunto A, y las entradas de la matriz **M** serán elementos del conjunto B.

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Esto significa que el número de posibilidades para escoger A y B es infinito.

Como se aclaro previamente, cada uno de los modelos de memorias asociativas que se presentan en este capítulo tiene sus propias especificaciones para dichos conjuntos, de acuerdo a las necesidades del creador.

Por convención, cada vector columna que representa a un patrón de entrada tendrá *n* componentes cuyos valores pertenecen al conjunto *A*, y cada vector columna que representa a un patrón de salida tendrá *m* componentes cuyos valores pertenecen también al conjunto *A*. Es decir:

$$\mathbf{x}^{\mu} \in A^{n} \text{ y } \mathbf{y}^{\mu} \in A^{m} \; \forall \mu \in \{1, 2, ..., p\}$$

La *j*-ésima componente de un vector columna se indicará con la misma letra del vector, pero sin negrilla, colocando a *j* como subíndice ($j \in \{1, 2, ..., n\}$ o $j \in \{1, 2, ..., m\}$ según corresponda). La *j*-ésima componente del vector columna $\mathbf{x}^{\mu}$ se representa por: x_j^{μ}

En los problemas donde intervienen las memorias asociativas, se consideran dos fases importantes: La fase de aprendizaje, que es donde se genera la memoria asociativa a partir de las *p* asociaciones del conjunto fundamental, y la fase de recuperación que es donde la memoria asociativas opera sobre un patrón de entrada, a la manera del esquema que aparece al inicio de este capítulo.

1. **Fase de Aprendizaje** (Generación de la memoria asociativa). Encontrar los operadores adecuados y una manera de generar una matriz **M** que almacene las *p* asociaciones del conjunto fundamental $\{(\mathbf{x}^1,\mathbf{y}^1), (\mathbf{x}^2,\mathbf{y}^2), ..., (\mathbf{x}^p,\mathbf{y}^p)\}$, donde $\mathbf{x}^{\mu} \in A^{n}$ y $\mathbf{y}^{\mu} \in A^{m} \; \forall \mu \in \{1, 2, ..., p\}$. Si $\exists \mu \in \{1, 2, ..., p\}$ tal que $\mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$, la memoria será heteroasociativa; si $m = n$ y $\mathbf{x}^{\mu} = \mathbf{y}^{\mu} \; \forall \mu \in \{1, 2, ..., p\}$, la memoria será autoasociativa.

2. **Fase de Recuperación** (Operación de la memoria asociativa). Hallar los operadores adecuados y las condiciones suficientes para obtener el patrón fundamental de salida $\mathbf{y}^{\mu}$, cuando se opera la memoria **M** con el patrón fundamental de entrada $\mathbf{x}^{\mu}$; lo anterior para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo.

Se dice que una memoria asociativa **M** exhibe **recuperación correcta** si al presentarle como entrada, en la fase de recuperación, un patrón $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$, ésta responde con el correspondiente patrón fundamental de salida $\mathbf{y}^{\omega}$.

Este último punto es muy importante ya que el hecho de que la memoria asociativa tenga recuperación perfecta aumenta, ampliamente, la gama de posibilidades en donde es posible aplicar a las memorias asociativas.

A continuación se presenta una breve reseña de los modelo de memorias asociativas, con el objeto de establecer el marco de referencia sobre el que se basan la memorias asociativas alfa-beta.

Los modelos principales sobre los que se abordará son: *Lernmatrix*, *Correlograph*, *Linear Associator*, *Memoria de Hopfield*, *Memorias Asociativas Morfológicas*, *Memorias Asociativas Alfa-Beta* y *Memorias Asociativas Mediana*, siendo los cuatro primeros modelos basados en el anillo de los números racionales con las operaciones de multiplicación y adición, el antepenúltimo modelo esta basado sobre las operaciones morfológicas y el paradigma de suma de productos y el penúltimo utiliza máximos y mínimos de dos operaciones nuevas expresadas en [45] conocidas como operación Alfa y operación Beta.

2.6.2 Lernmatrix de Steinbuch

Karl Steinbuch fue uno de los primeros investigadores en desarrollar un método para codificar información en arreglos cuadriculados conocidos como *crossbar* [49]. La importancia de la *Lernmatrix* [49][50] se evidencia en una afirmación que hace

Kohonen [51] en su artículo de 1972, donde apunta que las matrices de correlación, base fundamental de su innovador trabajo, vinieron a sustituir a la *Lernmatrix* de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $\mathbf{x}^{\mu} \in A^n, \quad A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^{\mu} \in A^p$ que le corresponde (de entre p clases diferentes), codificada ésta con un método que en la literatura se le ha llamado *one-hot* [50].

La codificación *one-hot* funciona así: para representar la clase $k \in \{1, 2, ..., p\}$, se asignan a las componentes del vector de salida $\mathbf{y}^{\mu}$ los siguientes valores: $y_k^{\mu} = 1$, y $y_j^{\mu} = 0$ para $j = 1, 2, ..., k\text{-}1, k\text{+}1, ..., p$.

Fase de Aprendizaje
Se genera el esquema (*crossbar*) al incorporar la pareja de patrones de entrenamiento $\left(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}\right) \in A^n \times A^p$. Cada uno de los componentes m_{ij} de **M**, la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } x_j^{\mu} = 1 = y_i^{\mu} \\ -\varepsilon & \text{si } x_j^{\mu} = 0 \text{ y } y_i^{\mu} = 1 \\ 0 & \text{en otro caso} \end{cases}$$

donde ε una constante positiva escogida previamente: es usual que ε es igual a 1.

Fase de Recuperación
La i-ésima coordenada y_i^{ω} del vector de clase $\mathbf{y}^{\omega} \in A^p$ se obtiene como lo indica la siguiente expresión, donde $\vee$ es el operador *máximo*:

$$y_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij}.x_j^{\omega} = \bigvee_{h=1}^{p} \left[\sum_{j=1}^{n} m_{hj}.x_j^{\omega} \right] \\ 0 & \text{en otro caso} \end{cases}$$

2.6.3 Correlograph de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental capaz de funcionar como una memoria asociativa [52]. En palabras de los autores "el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental".

Fase de Aprendizaje
La *red asociativa* se genera al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \in A^n \text{ x } A^m$. Cada uno de los componentes m_{ij} de la *red asociativa* **M** tiene valor cero al inicio, y se actualiza de acuerdo con la regla:

$$m_{ij} = \begin{cases} 1 & \text{si } y_i^{\mu} = 1 = x_j^{\mu} \\ \textit{valor anterior} & \text{en otro caso} \end{cases}$$

Fase de Recuperación

Se le presenta a la *red asociativa* **M** un vector de entrada $\mathbf{x}^{\omega} \in A^n$. Se realiza el producto de la matriz **M** por el vector $\mathbf{x}^{\omega}$ y se ejecuta una operación de umbralizado, de acuerdo con la siguiente expresión:

$$y_i^{\omega} = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij}.x_j^{\omega} \geq u \\ 0 & \text{en otro caso} \end{cases}$$

donde *u* es el valor de umbral. Una estimación aproximada del valor de umbral *u* se puede lograr con la ayuda de un número indicador mencionado en el artículo [51] de Willshaw *et al.* de 1969: $log_2 n$.

2.6.4 Linear Associator de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972 publicados por Anderson y Kohonen [53][85][54].

Para presentar el *Linear Associator* se considera de nuevo el conjunto fundamental:

$$\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\} \text{ con } A = \{0, 1\}, \quad \mathbf{x}^{\mu} \in A^n \text{ y } \mathbf{y}^{\mu} \in A^m$$

Fase de Aprendizaje

1) Para cada una de las *p* asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se encuentra la matriz $\mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t$ de dimensiones *m* x *n*
2) Se suman la *p* matrices para obtener la memoria

$$\mathbf{M} = \sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t = \left[m_{ij}\right]_{m \times n}$$

de manera que la *ij*-ésima componente de la memoria M se expresa así:

$$m_{ij} = \sum_{\mu=1}^{p} y_i^{\mu} x_j^{\mu}$$

Fase de Recuperación

Esta fase consiste en presentarle a la memoria un patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$ y realizar la operación

$$\mathbf{M} \cdot \mathbf{x}^{\omega} = \left[\sum_{\mu=1}^{p} \mathbf{y}^{\mu} \cdot \left(\mathbf{x}^{\mu}\right)^t\right] \cdot \mathbf{x}^{\omega}$$

2.6.5 La memoria asociativa de Hopfield

El artículo de John J. Hopfield de 1982, publicado por la prestigiosa y respetada *National Academy of Sciences* (en sus *Proceedings*), impactó positivamente y trajo a la palestra internacional su famosa memoria asociativa [55].

En el modelo que originalmente propuso Hopfield, cada neurona x_i tiene dos posibles estados, a la manera de las neuronas de McCulloch-Pitts: $x_i = 0$ y $x_i = 1$; sin embargo, Hopfield observa que, para un nivel dado de exactitud en la recuperación de patrones, la capacidad de almacenamiento de información de la memoria se puede incrementar por

un factor de 2, si se escogen como posibles estados de las neuronas los valores $x_i = -1$ y $x_i = 1$ en lugar de los valores originales $x_i = 0$ y $x_i = 1$.

Al utilizar el conjunto {-1, 1} y el valor de umbral cero, la fase de aprendizaje para la Memoria de Hopfield será similar, en cierta forma, a la fase de aprendizaje del *Linear Associator*. La intensidad de la fuerza de conexión de la neurona x_i a la neurona x_j se representa por el valor de m_{ij}, y se considera que hay simetría, es decir, $m_{ij} = m_{ji}$. Si x_i no está conectada con x_j entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona a sí misma, lo cual significa que $m_{ij} = 0$. El estado instantáneo del sistema está completamente especificado por el vector columna de dimensión n cuyas coordenadas son los valores de las n neuronas.

La Memoria de Hopfield es autoasociativa, simétrica, con ceros en la diagonal principal. En virtud de que la memoria es autoasociativa, el conjunto fundamental para la Memoria de Hopfield es $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, ..., p\}$ con $\mathbf{x}^\mu \in A^n$ y $A = \{-1, 1\}$

Fase de Aprendizaje

La fase de aprendizaje para la Memoria de Hopfield es similar a la fase de aprendizaje del *Linear Associator*, con una ligera diferencia relacionada con la diagonal principal en ceros, como se muestra en la siguiente regla para obtener la ij-ésima componente de la Memoria de Hopfield **M**:

$$m_{ij} = \begin{cases} \sum_{\mu=1}^{p} x_i^\mu x_j^\mu \text{ si } i \neq j \\ 0 \text{ si } i = j \end{cases}$$

Fase de Recuperación

Si se le presenta un patrón de entrada $\tilde{\mathbf{x}}$ a la Memoria de Hopfield, ésta cambiará su estado con el tiempo, de modo que cada neurona x_i ajuste su valor de acuerdo con el resultado que arroje la comparación de la cantidad $\sum_{j=1}^{n} m_{ij} x_j$ con un valor de umbral, el cual normalmente se coloca en cero.

Se representa el estado de la Memoria de Hopfield en el tiempo t por $\mathbf{x}(t)$; entonces $x_i(t)$ representa el valor de la neurona x_i en el tiempo t y $x_i(t+1)$ el valor de x_i en el tiempo siguiente $(t+1)$.

Dado un vector columna de entrada $\tilde{\mathbf{x}}$, la fase de recuperación consta de tres pasos:

1) Para $t = 0$, se hace $\mathbf{x}(t) = \tilde{\mathbf{x}}$; es decir, $x_i(0) = \tilde{x}_i$, $\forall i \in \{1,2,3,...,n\}$

2) $\forall i \in \{1,2,3,...,n\}$ se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) > 0 \\ x_i(t) \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) = 0 \\ -1 \text{ si } \sum_{j=1}^{n} m_{ij} x_j(t) < 0 \end{cases}$$

3) Se compara $x_i(t+1)$ con $x_i(t)$ $\forall i \in \{1, 2, 3,...,n\}$. Si $\mathbf{x}(t+1) = \mathbf{x}(t)$ el proceso termina y el vector recuperado es $\mathbf{x}(0) = \tilde{\mathbf{x}}$. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario hasta llegar a un valor $t = \tau$ para el cual $x_i(\tau+1) = x_i(\tau)$ $\forall i \in \{1, 2, 3,...,n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

En el artículo original de 1982, Hopfield había estimado empíricamente que su memoria tenía una capacidad de recuperar $0.15n$ patrones, y en el trabajo de Abu-Mostafa & St. Jacques [56] se estableció formalmente que una cota superior para el número de vectores de estado arbitrarios estables en una Memoria de Hopfield es n.

2.6.6 Memoria Asociativa Bidireccional (BAM) de Kosko.

Bart Kosko, investigador de la *University of Southern California*, propuso en 1988 la *Bidireccional Associative Memory* (BAM) para subsanar la clara desventaja de la autoasociatividad de la Memoria de Hopfield. La BAM maneja pares de vectores (A_1,B_1), ... (A_m,B_m), donde $A \in \{0, 1\}^n$ y $B \in \{0, 1\}^p$.

Al igual que Austin ensambló dos redes asociativas de Willshaw para diseñar su ADAM [57], Kosko ideó un arreglo de dos memorias Hopfield, y demostró que este diseño es capaz de asociar patrones de manera heteroasociativa.

La matriz $\mathbf{M}$ es una Memoria de Hopfield con la única diferencia que la diagonal principal es diferente de cero. $\mathbf{M}^T$ es la matriz transpuesta de $\mathbf{M}$ que, ahora como entrada, recibe a B y la salida será A. El proceso bidireccional anteriormente ilustrado continúa hasta que A y B convergen a una pareja estable (A_i, B_i).

$$\begin{aligned} A &\rightarrow \mathbf{M} \rightarrow B \\ A' &\leftarrow \mathbf{M}^T \leftarrow B \\ A' &\rightarrow \mathbf{M} \rightarrow B' \\ A'' &\leftarrow \mathbf{M}^T \leftarrow B' \\ &\cdots \\ A_i &\rightarrow \mathbf{M} \rightarrow B_i \\ A_i &\leftarrow \mathbf{M}^T \leftarrow B_i \\ &\cdots \end{aligned}$$

Kosko descubrió que su memoria funcionaba mejor con patrones bipolares que con patrones binarios (a la manera de Hopfield), por tanto: $A \in \{-1, 1\}^n$ y $B \in \{-1, 1\}^p$
Para la codificación de la BAM se superponen las m asociaciones sumándolas para formar la matriz de correlación:

$$\mathbf{M} = \sum_i A_i^T B_i$$

y la memoria dual $\mathbf{M}^T$ que está dada por:

$$\mathbf{M}^T = \sum_i \left(A_i^T B_i\right)^t = \sum_i B_i^T A_i$$

En el proceso de decodificación, cada neurona a_i que se encuentra en el campo A y cada neurona b_i localizada en el campo B, de forma independiente y asíncrona, examina la suma de entrada de las neuronas del otro campo, entonces puede o no cambiar su estado si la suma de entrada es mayor, igual o menor que un umbral dado. Si la suma de entrada es igual al umbral, entonces la neurona no cambia su estado. La suma de entrada para b_j es el producto interno columna:

$$A\mathbf{M}^j = \sum_i a_i m_{ij}$$

donde $\mathbf{M}^j$ es la j-ésima columna de $\mathbf{M}$. La suma de entrada para a_i es, de manera similar,

$$B\mathbf{M}_i^T = \sum_j b_j m_{ij}$$

donde $\mathbf{M}_i$ es la i-ésima fila de $\mathbf{M}$. Se toma el 0 como el umbral para todas las neuronas. Las funciones de umbral para a_i y b_j son:

$$a_i = \begin{cases} 1, & \text{si} \quad B\mathbf{M}_i^T > 0 \\ -1, & \text{si} \quad B\mathbf{M}_i^T < 0 \end{cases}$$

$$b_j = \begin{cases} 1, & \text{si} \quad A\mathbf{M}^j > 0 \\ -1, & \text{si} \quad A\mathbf{M}^j < 0 \end{cases}$$

Cuando se le presenta un patrón (A, B) a la BAM, las neuronas en los campos A y B se prenden o se apagan de acuerdo a la ocurrencia de 1's y 0's en los vectores de estado A y B. Las neuronas continúan sus cambios de estado hasta que se alcance un estado estable bidireccional (A_f, B_f).

2.6.7 Memorias Asociativas Morfológicas

La diferencia fundamental entre las memorias asociativas clásicas (*Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Asociativa de Hopfield) y las memorias asociativas morfológicas radica en los fundamentos operacionales de éstas últimas, que son las operaciones morfológicas de *dilatación* y *erosión*; el nombre de las memorias asociativas morfológicas está inspirado precisamente en estas dos operaciones básicas.

Estas memorias rompieron con el esquema utilizado a través de los años en los modelos de memorias asociativas clásicas, que utilizan operaciones convencionales entre vectores y matrices para la fase de aprendizaje y suma de productos para la recuperación de patrones. Las memorias asociativas morfológicas cambian los productos por sumas y las sumas por máximos o mínimos en ambas fases, tanto de aprendizaje como de recuperación de patrones [58][59][60].

Hay dos tipos de memorias asociativas morfológicas: las memorias *max*, simbolizadas con $\mathbf{M}$, y las memorias *min*, cuyo símbolo es $\mathbf{W}$; en cada uno de los dos tipos, las memorias pueden funcionar en ambos modos: heteroasociativo y autoasociativo.

Se definen dos nuevos productos matriciales:

El *producto máximo* entre **D** y **H**, denotado por $\mathbf{C} = \mathbf{D} \nabla \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya *ij*-ésima componente c_{ij} es

$$c_{ij} = \bigvee_{k=1}^{r} \left(d_{ik} + h_{kj} \right)$$

El *producto mínimo* de **D** y **H** denotado por $\mathbf{C} = \mathbf{D} \Delta \mathbf{H}$, es una matriz $[c_{ij}]_{mxn}$ cuya *ij*-ésima componente c_{ij} es

$$c_{ij} = \bigwedge_{k=1}^{r} \left(d_{ik} + h_{kj} \right)$$

Los productos máximo y mínimo contienen a los operadores máximo y mínimo, los cuales están íntimamente ligados con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

Fase de Aprendizaje

1. Para cada una de las *p* asociaciones $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se usa el producto mínimo para crear la matriz $\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^{t}$ de dimensiones *m* x *n*, donde el negado transpuesto del patrón de entrada $\mathbf{x}^{\mu}$ se define como $(-\mathbf{x}^{\mu})^{t} = \left(-x_1^{\mu}, -x_2^{\mu}, ..., x_n^{\mu} \right)$.
2. Se aplica el operador máximo $\vee$ a las *p* matrices para obtener la memoria **M**.

$$\mathbf{M} = \bigvee_{\mu=1}^{p} \left[\mathbf{y}^{\mu} \Delta (-\mathbf{x}^{\mu})^{t} \right]$$

Fase de Recuperación

Esta fase consiste en realizar el producto mínimo Δ de la memoria **M** con el patrón de entrada $\mathbf{x}^{\omega}$, donde $\omega \in \{1, 2, ..., p\}$, para obtener un vector columna **y** de dimensión *m*:

$$\mathbf{y} = \mathbf{M} \Delta \mathbf{x}^{\omega}$$

Las fases de aprendizaje y de recuperación de las **memorias morfológicas *min*** se obtienen por dualidad.

Para las memorias autoasociativas se utilizan los mismos algoritmos descritos anteriormente y que son aplicados a las memorias heteroasociativas; lo único que cambia es el conjunto fundamental. Para este caso, se considera el siguiente conjunto fundamental:

$$\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mathbf{x}^{\mu} \in A^{n}, \text{ donde } \mu = 1, 2, ..., p\}$$

2.6.8 Memorias Asociativas Alfa-Beta

Las memorias asociativas Alfa-Beta [45] utilizan máximos y mínimos, y dos operaciones binarias originales α y β de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β se deben especificar los conjuntos *A* y *B*, los cuales son:

$$A = \{0, 1\} \quad \text{y} \quad B = \{0, 1, 2\}$$

La operación binaria α: A x $A \rightarrow B$ se define como:

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

La operación binaria β: B x $A \rightarrow A$ se define como:

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

El fundamento teórico de las memorias asociativas Alfa-Beta se presenta en [45]

3 Materiales y Métodos

3.1 Memorias Asociativas Alfa-Beta

En la presente sección se expone el modelo fundamental sobre el que se basa este trabajo de tesis, las *Memorias Asociativas Alfa-Beta*. Se presentan las definiciones de las operaciones α y β, las operaciones matriciales utilizando estas operaciones originales, y se describe la fase de aprendizaje y recuperación de las memorias heteroasociativas y autoasociativas Alfa-Beta, tanto $\vee$ (*max*) como $\wedge$ (*min*)[45].

La numeración de los Lemas y Teoremas que aparecen aquí, es respetada tal y como se presentan en [45].

3.1.1 Operaciones binarias α y β: definiciones y propiedades

Las memorias Alfa-Beta utilizan máximos y mínimos, y dos operaciones binarias originales α y β de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β se deben especificar los conjuntos A y B, los cuales son:

$$A = \{0, 1\} \quad \text{y} \quad B = \{0, 1, 2\}$$

La operación binaria α: $A \times A \rightarrow B$ se define como se muestra en la Tabla 1.

Tabla 1. Operación binaria α: $A \times A \rightarrow B$

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

La operación binaria β: $B \times A \rightarrow A$ se define como se muestra en la Tabla 2

Tabla 2. Operación binaria β: $B \times A \rightarrow A$

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Los conjuntos A y B, las operaciones binarias α y β junto con los operadores $\wedge$ (mínimo) y $\vee$ (máximo) usuales conforman el sistema algebraico (A, B, α, β, $\wedge$, $\vee$) en el que están inmersas las memorias asociativas Alfa-Beta [45].

Se requiere la definición de cuatro operaciones matriciales, de las cuales se usarán sólo 4 casos particulares [45]:

Operación ***αmax***: $P_{mxr}\nabla_{\alpha}Q_{rxn}=\left[f_{ij}^{\alpha}\right]_{mxn}$, donde $f_{ij}^{\alpha}=\bigvee_{k=1}^{r}\alpha(p_{ik},q_{kj})$

Operación ***βmax***: $P_{mxr}\nabla_{\beta}Q_{rxn}=\left[f_{ij}^{\beta}\right]_{mxn}$, donde $f_{ij}^{\beta}=\bigvee_{k=1}^{r}\beta(p_{ik},q_{kj})$

Operación ***αmin***: $P_{mxr}\Delta_{\alpha}Q_{rxn}=\left[h_{ij}^{\alpha}\right]_{mxn}$, donde $h_{ij}^{\alpha}=\bigwedge_{k=1}^{r}\alpha(p_{ik},q_{kj})$

Operación ***βmin***: $P_{mxr}\Delta_{\beta}Q_{rxn}=\left[h_{ij}^{\beta}\right]_{mxn}$, donde $h_{ij}^{\beta}=\bigwedge_{k=1}^{r}\beta(p_{ik},q_{kj})$

El siguiente lema muestra los resultados obtenidos al utilizar las operaciones que involucran al operador binario α con las componentes de un vector columna y un vector fila dados[45].

Lema 3.9. (*Numeración tal como aparece en* [45]). *Sean* $\mathbf{x}\in A^n$ *y* $\mathbf{y}\in A^m$*; entonces* $\mathbf{y}\,\nabla_{\alpha}\,\mathbf{x}^t$ *es una matriz de dimensiones m x n, y además se cumple que:* $\mathbf{y}\,\nabla_{\alpha}\,\mathbf{x}^t=\mathbf{y}\,\Delta_{\alpha}\,\mathbf{x}^t$.

Demostración.

$$\mathbf{y}\nabla_{\alpha}\mathbf{x}^t=\begin{pmatrix}y_1\\y_2\\\vdots\\y_m\end{pmatrix}\nabla_{\alpha}\left(x_1,x_2,\ldots,\mathrm{x_n}\right)$$

$$=\begin{pmatrix}\bigvee_{k=1}^{1}\alpha(y_1,x_1) & \bigvee_{k=1}^{1}\alpha(y_1,x_2) & \cdots & \bigvee_{k=1}^{1}\alpha(y_1,x_n)\\ \bigvee_{k=1}^{1}\alpha(y_2,x_1) & \bigvee_{k=1}^{1}\alpha(y_2,x_2) & \cdots & \bigvee_{k=1}^{1}\alpha(y_2,x_n)\\ \vdots & \vdots & \cdots & \vdots\\ \bigvee_{k=1}^{1}\alpha(y_m,x_1) & \bigvee_{k=1}^{1}\alpha(y_m,x_2) & \cdots & \bigvee_{k=1}^{1}\alpha(y_m,x_n)\end{pmatrix}_{mxn}$$

$$=\begin{pmatrix}\alpha(y_1,x_1) & \alpha(y_1,x_2) & \cdots & \alpha(y_1,x_n)\\ \alpha(y_2,x_1) & \alpha(y_2,x_2) & \cdots & \alpha(y_2,x_n)\\ \vdots & \vdots & \cdots & \vdots\\ \alpha(y_m,x_1) & \alpha(y_m,x_2) & \cdots & \alpha(y_m,x_n)\end{pmatrix}_{mxn}$$

$$= \begin{pmatrix} \bigwedge_{k=1}^{1} \alpha(y_1, x_1) & \bigwedge_{k=1}^{1} \alpha(y_1, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_1, x_n) \\ \bigwedge_{k=1}^{1} \alpha(y_2, x_1) & \bigwedge_{k=1}^{1} \alpha(y_2, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_2, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ \bigwedge_{k=1}^{1} \alpha(y_m, x_1) & \bigwedge_{k=1}^{1} \alpha(y_m, x_2) & \cdots & \bigwedge_{k=1}^{1} \alpha(y_m, x_n) \end{pmatrix}_{mxn}$$

$$= \mathbf{y} \Delta_{\alpha} \mathbf{x}^t$$

En efecto, resulta que $\mathbf{y} \nabla_{\alpha} \mathbf{x}^t$ es una matriz de dimensiones *m* x *n*, y que $\mathbf{y} \nabla_{\alpha} \mathbf{x}^t = \mathbf{y} \Delta_{\alpha} \mathbf{x}^t$.

Dado el resultado del lema anterior, es conveniente escoger un símbolo único, digamos el símbolo $\boxtimes$, que represente a las dos operaciones ∇_{α} y Δ_{α} cuando se opera un vector columna de dimensión *m* con un vector fila de dimensión *n*:

$$\mathbf{y} \nabla_{\alpha} \mathbf{x}^t = \mathbf{y} \boxtimes \mathbf{x}^t = \mathbf{y} \Delta_{\alpha} \mathbf{x}^t$$

La *ij*-ésima componente de la matriz está $\mathbf{y} \boxtimes \mathbf{x}^t$ dada por:

$$[\mathbf{y} \boxtimes \mathbf{x}^t]_{ij} = \alpha(y_i, x_j)$$

Dado un índice de asociación μ, la expresión anterior indica que la *ij*-ésima componente de la matriz $\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^t$ se expresa de la siguiente manera:

$$[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^t]_{ij} = \alpha\left(y_i^{\mu}, x_j^{\mu}\right)$$

Ahora se analizará el caso en el que se opera una matriz de dimensiones *m* x *n* con un vector columna de dimensión *n* usando las operaciones∇_{β} y Δ_{β}. En los lemas 3.11 y 3.12 se obtiene la forma que exhibirán las *i*-ésimas componentes de los vectores columna resultantes de dimensión *m*, a partir de ambas operaciones ∇_{β} y Δ_{β}. [45]

Lema 3.11. (*Numeración tal como aparece en* [45]). *Sean* $\mathbf{x} \in A^n$ y $\mathbf{P}$ *una matriz de dimensiones m x n. La operación* $\mathbf{P}_{mxn} \nabla_{\beta} \mathbf{x}$ *da como resultado un vector columna de dimensión m, cuya i-ésima componente tiene la siguiente forma:*

$$\left(\mathbf{P}_{mxn} \nabla_{\beta} \mathbf{x}\right)_i = \bigvee_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$$

Demostración.

$$\mathbf{P}_{mxn} \nabla_{\beta} \mathbf{x} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \nabla_{\beta} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{P}_{mxn}\nabla_{\beta}\mathbf{x} = \begin{pmatrix} \beta(p_{11},x_1)\vee\beta(p_{12},x_2)\vee\ldots\vee\beta(p_{1n},x_n) \\ \beta(p_{21},x_1)\vee\beta(p_{22},x_2)\vee\ldots\vee\beta(p_{2n},x_n) \\ \vdots \\ \beta(p_{m1},x_1)\vee\beta(p_{m2},x_2)\vee\ldots\vee\beta(p_{mn},x_n) \end{pmatrix} = \begin{pmatrix} \bigvee_{j=1}^{n}\beta(p_{1j},x_j) \\ \bigvee_{j=1}^{n}\beta(p_{2j},x_j) \\ \vdots \\ \bigvee_{j=1}^{n}\beta(p_{mj},x_j) \end{pmatrix}$$

Se obtiene un vector columna de dimensión *m* cuya *i*-ésima componente es

$$\left(\mathbf{P}_{mxn}\nabla_{\beta}\mathbf{x}\right)_i = \bigvee_{j=1}^{n}\beta\left(p_{ij},x_j\right) \qquad \textbf{3.1}$$

Lema 3.12. (*Numeración tal como aparece en* [45]). *Sean* $\mathbf{x} \in A^n$ y $\mathbf{P}$ *una matriz de dimensiones m x n. La operación* $\mathbf{P}_{mxn}\ \Delta_{\beta}\ \mathbf{x}$ *da como resultado un vector columna de dimensión m, cuya i-ésima componente tiene la siguiente forma:* $\left(\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x}\right)_i = \bigwedge_{j=1}^{n}\beta\left(p_{ij},x_j\right)$

Demostración.

$$\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \Delta_{\beta} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x} = \begin{pmatrix} \beta(p_{11},x_1)\wedge\beta(p_{12},x_2)\wedge\ldots\wedge\beta(p_{1n},x_n) \\ \beta(p_{21},x_1)\wedge\beta(p_{22},x_2)\wedge\ldots\wedge\beta(p_{2n},x_n) \\ \vdots \\ \beta(p_{m1},x_1)\wedge\beta(p_{m2},x_2)\wedge\ldots\wedge\beta(p_{mn},x_n) \end{pmatrix} = \begin{pmatrix} \bigwedge_{j=1}^{n}\beta(p_{1j},x_j) \\ \bigwedge_{j=1}^{n}\beta(p_{2j},x_j) \\ \vdots \\ \bigwedge_{j=1}^{n}\beta(p_{mj},x_j) \end{pmatrix}$$

Se obtiene un vector columna de dimensión *m* cuya *i*-ésima componente es

$$\left(\mathbf{P}_{mxn}\Delta_{\beta}\mathbf{x}\right)_i = \bigwedge_{j=1}^{n}\beta\left(p_{ij},x_j\right)$$

3.1.2 Memorias Heteroasociativas Alfa-Beta

Se tienen dos tipos de memorias heteroasociativas Alfa-Beta: tipo **V** y tipo **Λ**. En la generación de ambos tipos de memorias se usará el operador $\boxtimes$ el cual tiene la siguiente forma:

$$[\mathbf{y}^{\mu}\boxtimes(\mathbf{x}^{\mu})^t]_{ij} = \alpha(y_i^{\mu},x_j^{\mu});\mu\in\{1,2,\ldots p\},i\in\{1,2,\ldots,p\},j\in\{1,2,\ldots n\}$$

Algoritmo Memorias Alfa-Beta tipo V

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se construye la matriz

$$\left[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}\right]_{mxn}$$

Paso 2. Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso 1:

$$\mathbf{V} = \bigvee_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$v_{ij} = \bigvee_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu})$$

Fase de Recuperación

Se presenta un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$, a la memoria heteroasociativa $\alpha\beta$ tipo $\mathbf{V}$ y se realiza la operación Δ_{β}: $\mathbf{V}\ \Delta_{\beta}\ \mathbf{x}^{\omega}$.

Dado que las dimensiones de la matriz $\mathbf{V}$ son de *m* x *n* y $\mathbf{x}^{\omega}$ es un vector columna de dimensión *n*, el resultado de la operación anterior debe ser un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$(V\Delta_{\beta} x^{\varpi})_i = \bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^{\varpi})$$

Teorema 4.7. (*Numeración tal como aparece en* [45]). *Sea* $\left\{\left(x^{\mu}, y^{\mu}\right) \mid \mu = 1,2,....p\right\}$ *el conjunto fundamental de una memoria heteroasociativa Alfa-Beta representada por* $\mathbf{V}$ *. Si* ω *es un valor de índice arbitrario tal que* $\omega \in \{1,2,3,4,.....p\}$ *, y si además para cada* $i \in \{1,2,3,4,.....n\}$ *se cumple que* $\exists j \neq j_0 \in \{1,.....,n\}$ *, el cual depende de* ω *y de* i *, tal que* $v_{ij_0} = \alpha(y_i^{\omega}, x_{j_0}^{\omega})$ *, entonces la recuperación* $V\Delta_{\beta} x^{\omega}$ *es correcta; es decir* $V\Delta_{\beta} x^{\omega} = y^{\omega}$ *.*

Demostración. Ver detalles de la demostración en [45].

Algoritmo Memorias Alfa-Beta tipo Λ

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se construye la matriz

$$\left[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}\right]_{mxn}$$

Paso 2. Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas en el paso 1:

$$\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]$$

La entrada *ij*-ésima está dada por la siguiente expresión:

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu})$$

Fase de Recuperación

Se presenta un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$, a la memoria heteroasociativa *Alfa-Beta* tipo $\mathbf{\Lambda}$ y se realiza la operación ∇_{β}: $\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega}$.

Dado que las dimensiones de la matriz $\mathbf{\Lambda}$ son de *m* x *n* y $\mathbf{x}^{\omega}$ es un vector columna de dimensión *n*, el resultado de la operación anterior debe ser un vector columna de dimensión *m*, cuya *i*-ésima componente es:

$$\left(\mathbf{\Lambda} \nabla_{\beta} \mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left(\lambda_{ij}, x_j^{\mu}\right)$$

Teorema 4.20. (*Numeración tal como aparece en* [45]). *Sea* $\left\{\left(x^{\mu}, y^{\mu}\right) | \mu = 1,2,....p\right\}$ *el conjunto fundamental de una memoria heteroasociativa Alfa-Beta representada por* Λ *. Si* ω *es un valor arbitrario fijo tal que* $\omega \in \{1,2,3,4,.....p\}$ *, y si además para cada* $i \in \{1,2,3,4,.....n\}$ *se cumple que* $\exists j \neq j_0 \in \{1,.....,n\}$ *, el cual depende de* ω *y de* i *, tal que* $\lambda_{ij_0} = \alpha(y_i^{\omega}, x_{j_0}^{\omega})$ *, entonces la recuperación* $V \nabla_{\beta} x^{\omega}$ *es correcta; es decir* $V \nabla_{\beta} x^{\omega} = y^{\omega}$.

Demostración. Ver detalles de la demostración en [45].

3.1.3 Memorias Autoasociativas Alfa-Beta

Si a una memoria heteroasociativa se le impone la condición de que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}$ $\forall \mu \in \{1, 2, ..., p\}$ entonces, deja de ser heteroasociativa y ahora se le denomina autoasociativa.

A continuación se enlistan algunas de las características de las memorias autoasociativas Alfa-Beta :

1. El conjunto fundamental toma la forma $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$
2. Los patrones fundamentales de entrada y salida son de la misma dimensión; denotémosla por *n*.
3. La memoria es una matriz cuadrada, para ambos tipos, $\mathbf{V}$ y $\mathbf{\Lambda}$. Si $\mathbf{x}^{\mu} \in A^n$ entonces

$$\mathbf{V} = \left[v_{ij}\right]_{nxn} \text{ y } \Lambda = \left[\lambda_{ij}\right]_{nxn}$$

Algoritmo Memorias Autoasociativas Alfa-Beta tipo V

Las fases de aprendizaje y recuperación son similares a las memorias heteroasociativas Alfa-Beta.

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ se construye la matriz

$$\left[\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}\right]_{mxn}$$

Paso 2. Se aplica el operador binario máximo $\mathbf{V}$ a las matrices obtenidas en el paso 1:

$$\mathbf{V} = \bigvee_{\mu=1}^{p} [\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]$$

La entrada ij-ésima de la memoria está dada así:

$$v_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$$

y de acuerdo con que α: A x $A \rightarrow B$, se tiene que $v_{ij} \in B$, $\forall i \in \{1, 2, ..., n\}$. $\forall j \in \{1, 2, ..., n\}$.

Fase de Recuperación. La fase de recuperación de las memorias autoasociativas Alfa-Beta tipo V tiene dos casos posibles. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino la versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\tilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, ..., p\}$, que corresponde al patrón fundamental respecto del cual $\tilde{\mathbf{x}}$ es una versión alterada.

CASO 1: Patrón fundamental. Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa Alfa-Beta tipo V y se realiza la operación Δ_{β}:

$$\mathbf{V}\ \Delta_{\beta}\ \mathbf{x}^{\omega}$$

El resultado de la operación anterior será el vector columna de dimensión n.

$$\left(\mathbf{V}\Delta_{\beta}\mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta(v_{ij}, x_j^{\omega})$$

$$\left(\mathbf{V}\Delta_{\beta}\mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], x_j^{\omega}\right\}$$

CASO 2: Patrón alterado. Se presenta el patrón binario $\tilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^{\omega}$) que es un vector columna de dimensión n, a la memoria autoasociativa Alfa-Beta tipo V y se realiza la operación

$$\mathbf{V}\Delta_\beta \tilde{\mathbf{x}}$$

Al igual que en el caso 1, el resultado de la operación anterior es un vector columna de dimensión *n*, cuya *i*-ésima componente se expresa de la siguiente manera:

$$\left(\mathbf{V}\Delta_\beta \tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta(v_{ij}, \tilde{x}_j)$$

$$\left(\mathbf{V}\Delta_\beta \tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left\{\left[\bigvee_{\mu=1}^{p} \alpha(x_i^\mu, x_j^\mu)\right], \tilde{x}_j\right\}$$

Lema 4.27. (*Numeración tal como aparece en* [45]). *Una memoria autoasociativa Alfa-Beta tipo* **V** *tiene únicamente unos en la diagonal principal*.

Demostración. La *ij*-ésima entrada de una memoria autoasociativa Alfa-Beta tipo **V** está dada por $v_{ij} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_j^\mu\right)$. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$:

$$v_{ii} = \bigvee_{\mu=1}^{p} \alpha\left(x_i^\mu, x_i^\mu\right), \quad \forall i \in \{1,2,...n\}$$

Por la propiedad de la tabla se tiene que $\alpha(x_i^\mu, x_i^\mu) = 1$, por lo que la expresión anterior se transforma en:

$$v_{ii} = \bigvee_{\mu=1}^{p} (1) = 1, \quad \forall i \in \{1,2,...n\}$$

Teorema 4.28. (*Numeración tal como aparece en* [45]). *Una memoria autoasociativa Alfa-Beta tipo* **V** *recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje.*

Demostración. Sea $\omega = \{1, 2, ..., p\}$ arbitrario. De acuerdo con el lema 4.27, para cada $i \in \{1, ..., n\}$ escogida arbitrariamente

$$v_{ii} = 1 = \alpha(x_i^\omega, x_i^\omega)$$

Es decir, para $i \in \{1, ..., n\}$ escogida arbitrariamente, $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$v_{ij_0} = \alpha(x_i^\omega, x_{j_0}^\omega)$$

Por lo tanto, de acuerdo con el Teorema 4.7

$$\mathbf{V}\ \Delta_\beta\, \mathbf{x}^\omega = \mathbf{x}^\omega,\ \forall \omega \in \{1, 2, ..., p\}$$

Esto significa que la memoria autoasociativa Alfa-Beta tipo V recupera de manera correcta el conjunto fundamental completo.

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre *p,* que es la cardinalidad del conjunto fundamental; y esto quiere decir que el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo V, con recuperación correcta, es máximo.

El Teorema 4.28 se puede enunciar desde un enfoque matricial de la siguiente manera: dado que para cada asociación $(\mathbf{x}^{\omega},\mathbf{x}^{\omega})$ del conjunto fundamental de una memoria autoasociativa Alfa-Beta V, se cumple que cada fila de la matriz $V - x^{\mu} \boxtimes (x^{\mu})^{t}$ contiene una entrada cero, entonces de la memoria V recupera el conjunto completo de patrones fundamentales en forma correcta.

Teorema 4.30. (*Numeración tal como aparece en* [45]). *Sea* $\{(\mathbf{x}^{\mu},\mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta representada por* **V**, *y sea* $\tilde{\mathbf{x}} \in A^{n}$ *un patrón alterado positivamente respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* **V** *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* ω *y de i tal* que $v_{ij_0} \leq \alpha(x^{\omega}, \tilde{x}_{j_0})$, *entonces la recuperación* $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}}$ *es correcta, es decir,* $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

Demostración. Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}$ $\forall \mu \in \{1, 2, ..., p\}$ y, por consiguiente, $m = n$. Al establecer estas dos condiciones en el Teorema 4.13 (Ver referencia), se obtiene el resultado: $\mathbf{V}\Delta_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

El Teorema 4.30 nos dice que las memorias autoasociativas Alfa-Beta tipo V son inmunes a cierta cantidad de ruido aditivo.

Algoritmo Memorias Autoasociativas Alfa-Beta tipo Λ

Fase de Aprendizaje

Paso 1. Para cada $\mu = 1, 2, ..., p$, a partir de la pareja $(\mathbf{x}^{\mu},\mathbf{x}^{\mu})$ se construye la matriz

$$\left[\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}\right]_{mxn}$$

Paso 2. Se aplica el operador binario máximo Λ a las matrices obtenidas en el paso 1:

$$\Lambda = \bigvee_{\mu=1}^{p} [\mathbf{x}^{\mu} \boxtimes (\mathbf{x}^{\mu})^{t}]$$

La entrada *ij*-ésima de la memoria está dada así:

$$\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$$

y de acuerdo con que α: A x $A \rightarrow B$, se tiene que $\lambda_{ij} \in B$, $\forall i \in \{1, 2, ..., n\}$. $\forall j \in \{1, 2, ..., n\}$.

Fase de Recuperación. La fase de recuperación de las memorias autoasociativas $\alpha\beta$ tipo Λ tiene dos casos posibles. En el primer caso el patrón de entrada es un patrón fundamental; es decir, la entrada es un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$. En el segundo caso, el patrón de entrada NO es un patrón fundamental, sino la versión distorsionada de por lo menos uno de los patrones fundamentales; lo anterior significa que si el patrón de entrada es $\tilde{\mathbf{x}}$, debe existir al menos un valor de índice $\omega \in \{1, 2, ..., p\}$, que corresponde al patrón fundamental respecto del cual $\tilde{\mathbf{x}}$ es una versión alterada de alguno de los tres tipos: aditivo, sustractivo o mezclado.

CASO 1: Patrón fundamental. Se presenta a un patrón $\mathbf{x}^{\omega}$, con $\omega \in \{1, 2, ..., p\}$ a la memoria autoasociativa *Alfa-Beta* tipo V y se realiza la operación ∇_{β}:

$$\mathbf{\Lambda}\, \Delta_{\beta}\, \mathbf{x}^{\omega}$$

El resultado de la operación anterior será el vector columna de dimensión *n*.

$$\left(\mathbf{\Lambda}\nabla_{\beta}\mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta(\lambda_{ij}, x_j^{\omega})$$

$$\left(\Lambda\nabla_{\beta}\mathbf{x}^{\omega}\right)_i = \bigvee_{j=1}^{n} \beta\left\{\left[\bigwedge_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], x_j^{\omega}\right\}$$

CASO 2: Patrón alterado. Se presenta el patrón binario $\tilde{\mathbf{x}}$ (patrón alterado de algún patrón fundamental $\mathbf{x}^{\omega}$) que es un vector columna de dimensión *n*, a la memoria autoasociativa $\alpha\beta$ tipo Λ y se realiza la operación:

$$\Lambda\nabla_{\beta}\tilde{\mathbf{x}}$$

Al igual que en el caso 1, el resultado de la operación anterior es un vector columna de dimensión *n*, cuya *i*-ésima componente se expresa de la siguiente manera:

$$\left(\Lambda\nabla_{\beta}\tilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta(\lambda_{ij}, \tilde{x}_j)$$

$$\left(\Lambda\nabla_{\beta}\tilde{\mathbf{x}}\right)_i = \bigvee_{j=1}^{n} \beta\left\{\left[\bigwedge_{\mu=1}^{p} \alpha(x_i^{\mu}, x_j^{\mu})\right], \tilde{x}_j\right\}$$

Lema 4.31. (*Numeración tal como aparece en* [45]). *Una memoria autoasociativa Alfa-Beta tipo Λ tiene únicamente unos en la diagonal principal*.

Demostración. La *ij*-ésima entrada de una memoria autoasociativa Alfa-Beta tipo $\mathbf{\Lambda}$ está dada por $\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_j^{\mu}\right)$. Las entradas de la diagonal principal se obtienen de la expresión anterior haciendo $i = j$:

$$\lambda_{ii} = \bigwedge_{\mu=1}^{p} \alpha\left(x_i^{\mu}, x_i^{\mu}\right), \quad \forall i \in \{1,2,...n\}$$

Por la propiedad de la tabla se tiene que $\alpha(x_i^{\mu}, x_i^{\mu}) = 1$, por lo que la expresión anterior se transforma en:

$$\lambda_{ii} = \bigwedge_{\mu=1}^{p} (1) = 1, \quad \forall i \in \{1,2,...n\}$$

Teorema 4.32. (*Numeración tal como aparece en* [45]). *Una memoria autoasociativa Alfa-Beta tipo* $\boldsymbol{\Lambda}$ *recupera de manera correcta el conjunto fundamental completo; además, tiene máxima capacidad de aprendizaje.*

Demostración. Sea $\omega = \{1, 2, ..., p\}$ arbitrario. De acuerdo con el lema 4.31, para cada $i \in \{1, ..., n\}$ escogida arbitrariamente $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$\lambda_{ii} = 1 = \alpha(x_i^{\omega}, x_i^{\omega})$$

Es decir, para $i \in \{1, ..., n\}$ escogida arbitrariamente, $\exists\, j_0 = i \in \{1, ..., n\}$ que cumple con:

$$\lambda_{ij_0} = \alpha(x_i^{\omega}, x_{j_0}^{\omega})$$

Por lo tanto, de acuerdo con el Teorema 4.20

$$\boldsymbol{\Lambda}\nabla_{\beta}\, \mathbf{x}^{\omega} = \mathbf{x}^{\omega},\ \forall\, \omega \in \{1, 2, ..., p\}$$

Esto significa que la memoria autoasociativa Alfa-Beta tipo Λ recupera de manera correcta el conjunto fundamental completo.

Además, en la demostración de este Teorema, en ningún momento aparece restricción alguna sobre p que es la cardinalidad del conjunto fundamental; y esto quiere decir que el conjunto fundamental puede crecer tanto como se quiera. La consecuencia directa es que el número de patrones que puede aprender una memoria autoasociativa Alfa-Beta tipo Λ, con recuperación correcta, es máximo.

El Teorema 4.32 se puede enunciar desde un enfoque matricial de la siguiente manera: dado que para cada asociación $(\mathbf{x}^{\omega}, \mathbf{x}^{\omega})$ del conjunto fundamental de una memoria autoasociativa Alfa-Beta Λ, se cumple que cada fila de la matriz $\mathbf{x}^{\omega} \boxtimes (\mathbf{x}^{\omega})^t - \Lambda$ contiene una entrada cero, entonces de la memoria Λ recupera el conjunto completo de patrones fundamentales en forma correcta.

Teorema 4.33. (*Numeración tal como aparece en* [45]). *Sea* $\{(\mathbf{x}^{\mu}, \mathbf{x}^{\mu}) \mid \mu = 1, 2, ..., p\}$ *el conjunto fundamental de una memoria autoasociativa Alfa-Beta representada por* $\boldsymbol{\Lambda}$, *y sea* $\tilde{\mathbf{x}} \in A^n$ *un patrón alterado con ruido sustractivo respecto a algún patrón fundamental* $\mathbf{x}^{\omega}$ con $\omega \in \{1, 2, ..., p\}$. *Si se presenta* $\tilde{\mathbf{x}}$ *a la memoria* $\boldsymbol{\Lambda}$ *como entrada, y si además para cada* $i \in \{1, ..., n\}$ *se cumple la condición de que* $\exists j = j_0 \in \{1, ..., n\}$, *el cual depende de* ω *y de i tal que* $\lambda_{ij_0} \leq \alpha(x^{\omega}, \tilde{x}_{j_0})$, *entonces la recuperación* $\Lambda\nabla_{\beta}\tilde{\mathbf{x}}$ *es correcta, es decir,* $\Lambda\nabla_{\beta}\tilde{\mathbf{x}} = \mathbf{x}^{\omega}$

Demostración. Por hipótesis se tiene que $\mathbf{y}^{\mu} = \mathbf{x}^{\mu}\ \forall \mu \in \{1, 2, ..., p\}$ y, por consiguiente, $m = n$. Al establecer estas dos condiciones en el Teorema 4.23 (Ver referencia), se obtiene el resultado:

$$\Lambda \nabla_{\beta} \tilde{\mathbf{x}} = \mathbf{x}^{\omega}$$

El Teorema 4.33 nos dice que las memorias autoasociativas Alfa-Beta tipo V son inmunes a cierta cantidad de ruido sustractivo

3.2 Multimemorias Asociativas Alfa-Beta

En memorias asociativas, particularmente las memorias asociativas Alfa-Beta, es común encontrar una correspondencia de uno a uno entre el conjunto fundamental y el número de memorias creadas a partir de éste; es decir, para cada conjunto fundamental distinto existe una y sólo una matriz de aprendizaje [48][43].

La creación de varias memorias asociativas, las cuales se comportarán como una sola en el sentido de que en la fase de recuperación se obtenga un solo vector de salida puede ser útil para el tratamiento del ruido mezclado.

Para el desarrollo de este algoritmo es necesario establecer las siguientes definiciones [48].

Definición 1: Sean $A = \{0,1\},\ n, q \in Z^{+}$ y $q \neq 0$ y sea $\mathbf{x}^{\varpi} \in A^{n}$, $\varpi \in \{1,2,..., p\}$. Es posible dividir a $\mathbf{x}^{\varpi}$ en q particiones iguales si $\frac{n}{q} \in Z^{+}$ [48].

Definición 2: Sea $A = \{0,1\}, \mathbf{x}^{\alpha} \in A^{n}$, $\alpha \in \{1,2,..., p\}$ un vector columna y q el número de particiones que se desean de dicho vector con $n, q, \in Z^{+}$. La operación de particionamiento vectorial ρ de $\mathbf{x}^{\alpha}$ se define como el conjunto de vectores columna binario $\frac{n}{q}$ – dimensionales. Dicho particionamiento se denota por [48]:

$$\rho(\mathbf{x}^{\alpha}, q) = \left\{\mathbf{x}^{\alpha 1}, \mathbf{x}^{\alpha 2}, ..., \mathbf{x}^{\alpha q}\right\}$$

tal que $\mathbf{x}^{\alpha l} \in A^{\frac{n}{q}}$ con $l \in \{1,2,..., q\}$

3.2.1 Multimemoria Heteroasociativa Max

Sean $A = \{0,1\}, n, p \in Z^{+}$, $\mu \in \{1, 2, ...p\}, i \in \{1, 2, ..., p\}$ y $j \in \{1, 2, ..n\}$, y sean $\mathbf{x} \in A^{n}$ y $\mathbf{y} \in A^{p}$ los vectores de entrada y salida, respectivamente. A partir de estos vectores es posible construir el conjunto fundamental expresado por $\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$ de acuerdo con el nuevo algoritmo de memoria heteroasociativa Alfa-Beta tipo *Max* [48].

Fase de Aprendizaje

Una vez que se tiene el conjunto fundamental, para construir a partir de un solo conjunto fundamental varias matrices de aprendizaje, en lugar de sólo una, es necesario

dividir cada patrón de entrada $\mathbf{x}^{\mu}$ en q particiones, generando así q vectores de entrada $\frac{n}{q}$ – dimensionales y por lo tanto q memorias asociativas [48].

Para cada $\mu \in \{1,2,...p\}$ a partir de la pareja ordenada $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se aplica la *operación de particionamiento vectorial* a cada uno de los vectores $\mathbf{x}^{\mu}$ de las p parejas ordenadas. Por lo que el nuevo conjunto fundamental puede ser expresado como sigue:

$$\{(\rho(\mathbf{x}^{\mu}, q), \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$$

aplicando el operador de particionamiento el conjunto fundamental queda como sigue:

$$\{(\{\mathbf{x}^{\mu 1}, \mathbf{x}^{\mu 2}, ..., \mathbf{x}^{\mu q}\}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$$

y por distributividad

$$\{(\mathbf{x}^{\mu 1}, \mathbf{y}^{\mu}), (\mathbf{x}^{\mu 2}, \mathbf{y}^{\mu}), ..., (\mathbf{x}^{\mu q}, \mathbf{y}^{\mu}) \mid \mu = 1, 2, ..., p\}$$

Ahora, para $l \in \{1,2,...,q\}$ a partir de las parejas $(\mathbf{x}^{\mu l}, \mathbf{y}^{\mu})$ se construyen las q matrices según la fase de aprendizaje del nuevo algoritmo de memorias heteroasociativas presentado en esta tesis.

$$[\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu l})^{t}]_{mx(\frac{n}{q})}$$

Se aplica el operador binario máximo $\vee$ a las matrices obtenidas

$$\mathbf{V}^{l} = \bigvee_{\mu=1}^{p} [\mathbf{y}^{\mu} \boxtimes (\mathbf{x}^{\mu l})^{t}]_{mx(\frac{n}{q})}$$

del tal modo que la entrada $ij-ésima$ de cada una de las q matrices esté dada por

$$v_{ij}^{l} = \bigvee_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu l})$$

Fase de Recuperación

Al igual que en la construcción de las matrices de aprendizaje, para la fase de recuperación, es necesario dividir en q particiones el patrón que se presenta a las q memorias [48].

PASO 1

Al presentarse un patrón $\mathbf{x}^{\varpi}$, con $\varpi \in \{1,2,...p\}$ a las $\mathbf{V}^{l}$ memorias, lo primero es aplicar el *operador de particionamiento* al vector $\mathbf{x}^{\varpi}$. De esta forma el vector quedará dividido en q nuevos vectores.

$$\rho(\mathbf{x}^{\varpi}, q) = \left\{\mathbf{x}^{\varpi 1}, \mathbf{x}^{\varpi 2}, ..., \mathbf{x}^{\varpi q}\right\}$$

Ahora, para cada una de las q matrices y q particiones del vector, se realiza la fase de recuperación del nuevo algoritmo de memorias heteroasociativas Alfa-Beta tipo *Max* propuesto en esta tesis y, se realiza para cada uno de ellos la operación Δ_{β}, asignando el resultado al vector $z^{\varpi l}$ [48]

$$z^{\varpi l} = V^{l} \Delta_{\beta} \mathbf{x}^{\varpi l}$$

Por lo que la $i-ésima$ componente de cada uno de los vectores columna resultantes se expresa como:

$$z_i^{\varpi l} = \bigwedge_{j=1}^{n} \beta(v_{ij}^{l}, x_j^{\varpi l})$$

posteriormente se continúa con el paso 2 de la fase de recuperación del nuevo algoritmo de memoria heteroasociativa tipo *Max* propuesto en esta tesis, para cada uno de los vectores $z^{\varpi l}$ [48].

Una vez que se lleve a cabo este procedimiento se obtendrán q vectores resultantes, uno por cada una de las matrices de aprendizaje.

PASO 2

Una vez que se tienen los q vectores $z^{\varpi l}$. Para obtener un solo vector resultante se deberá llevar a cabo el siguiente algoritmo.

Se crea un vector intermedio I el cual contendrá la sumatoria de las $i-ésimas$ componentes de los vectores $z^{\varpi l}$ [48].

$$I_i^{\varpi} = \sum_{l=1}^{q} z_i^{\varpi l}$$

Una vez que se obtiene el vector I^{ϖ}, el vector y^{ϖ} se obtendrá mediante la siguiente expresión

$$y^{\varpi} = \begin{cases} 1 & I_i^{\varpi} = \bigvee_{k=1}^{p} I_k^{\varpi} \\ 0 & otro\ caso \end{cases}$$

3.2.2 Multimemoria Heteroasociativa Min

Sean $A = \{0,1\}$, $n, p \in Z^{+}$, $\mu \in \{1,2,...p\}, i \in \{1,2,...,p\}$ y $j \in \{1,2,...n\}$, y sean $\mathbf{x} \in A^{n}$ y

$\mathbf{y} \in A^p$ vectores de entrada y salida, respectivamente. A partir de estos vectores es posible construir el conjunto fundamental expresado por $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,...,p\}$ de acuerdo con el nuevo algoritmo de memoria heteroasociativa Alfa-Beta tipo *Min* [48].

Fase de Aprendizaje

Para construir a partir de un solo conjunto fundamental varias matrices de aprendizaje, en lugar de solo una, es necesario dividir cada patrón de entrada $\mathbf{x}^\mu$ en q particiones, generando así q vectores de entrada $\frac{n}{q}-$dimensionales y por lo tanto q memorias asociativas [48].

Para cada $\mu \in \{1,2,...p\}$ a partir de la pareja ordenada $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se aplica la *operación de particionamiento vectorial* a cada uno de los vectores $\mathbf{x}^\mu$ de las p parejas ordenadas. Por lo que el nuevo conjunto fundamental puede ser expresado como sigue [48]:

$$\{(\rho(\mathbf{x}^\mu, q), \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

Aplicando el operador de particionamiento el conjunto fundamental queda como sigue:

$$\{(\{\mathbf{x}^{\mu 1}, \mathbf{x}^{\mu 2},..., \mathbf{x}^{\mu q}\}, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

y por distributividad

$$\{(\mathbf{x}^{\mu 1}, \mathbf{y}^\mu), (\mathbf{x}^{\mu 2}, \mathbf{y}^\mu),..., (\mathbf{x}^{\mu q}, \mathbf{y}^\mu) \mid \mu = 1,2,..., p\}$$

Ahora, a partir de las parejas $(\mathbf{x}^{\mu l}, \mathbf{y}^\mu)$ se construyen las q matrices de acuerdo con la fase de aprendizaje del nuevo algoritmo presentado en [48].

$$[\mathbf{y}^\mu \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})}$$

Se aplica el operador binario mínimo $\wedge$ a las matrices obtenidas

$$\mathbf{\Lambda}^l = \bigwedge_{\mu=1}^{p} [\mathbf{y}^\mu \boxtimes (\mathbf{x}^{\mu l})^t]_{mx(\frac{n}{q})}$$

del tal modo que la entrada $ij-ésima$ de cada una de las q matrices esté dada por

$$\lambda_{ij}^l = \bigwedge_{\mu=1}^{p} \alpha(y_i^\mu, x_j^{\mu l})$$

Fase de Recuperación

Al igual que en la construcción de las matrices de aprendizaje, para la fase de recuperación es necesario dividir en q particiones el patrón que se presentará a las q

memorias [48].

PASO 1

Al presentarse un patrón $\mathbf{x}^{\varpi}$, con $\varpi \in \{1,2,...p\}$ a la multimemoria heteroasociativa Alfa-Beta tipo *Min,* lo primero es aplicar el *operador de particionamiento* al vector $\mathbf{x}^{\varpi}$ De esta forma el vector quedará dividido en q nuevos vectores [48].

$$\rho(\mathbf{x}^{\varpi}, q) = \left\{\mathbf{x}^{\varpi 1}, \mathbf{x}^{\varpi 2}, ..., \mathbf{x}^{\varpi q}\right\}$$

Ahora, para cada una de las q matrices y particiones del vector, se realiza la fase de recuperación del algoritmo de memorias heteroasociativas Alfa-Beta tipo *Min* propuesto en esta tesis, es decir, se realiza para cada uno de ellos la operación Δ_{β}, asignando el resultado a el vector $z^{\varpi l}$ [48].

$$z^{\varpi l} = \Lambda^{l} \nabla_{\beta} \mathbf{x}^{\varpi l}$$

Por lo que la $i-ésima$ componente de cada uno de los vectores columna resultantes, se expresa como:

$$z_i^{\varpi l} = \bigvee_{j=1}^{n} \beta(\lambda_{ij}^{l}, x_j^{\varpi l})$$

posteriormente se continúa con el paso 2 de la fase de recuperación del algoritmo de memoria heteroasociativa tipo *Min* propuesto en esta tesis, para cada $z^{\varpi l}$.

Una vez que se lleve a cabo este procedimiento se obtendrán q vectores resultantes, uno por cada una de las matrices de aprendizaje.

PASO 2

Una vez que se tienen los q vectores resultantes, a los que denominaremos $z^{\varpi l}$, para obtener un solo vector resultante se deberá llevar a cabo el siguiente algoritmo [48].

Se crea un vector intermedio I el cual contendrá la sumatoria de las $i-ésimas$ componentes de los vectores $z^{\varpi l}$.

$$I_i^{\varpi} = \sum_{l=1}^{q} z_i^{\varpi l}$$

Una vez que se obtiene el vector I^{ϖ}, el vector y^{ϖ} se obtendrá mediante la siguiente función:

$$y_i^{\varpi} = \begin{cases} 0 & I_i^{\varpi} = \bigwedge_{k=1}^{p} I_k^{\varpi} \\ 1 & en\ otro\ caso \end{cases}$$

3.3 Análisis algorítmico

Un algoritmo es un método computacional para resolver una clase de problema dado. La complejidad de un algoritmo es el costo, en tiempo de ejecución o de almacenamiento, de usar un algoritmo para resolver un problema en específico [82]. Por lo tanto, el estudio de la cantidad de esfuerzo computacional que se necesita para llevar a cabo un número determinado de cálculos, es lo que se conoce como complejidad computacional.

Un algoritmo será más eficiente comparado con otro, siempre y cuando consuma menos recursos, como el tiempo y el espacio necesarios para ejecutarlo. La medida del tiempo se conoce como complejidad temporal y la del espacio, complejidad espacial.

El esfuerzo computacional que se mide al llevar a cabo una tarea, es medido por el número de pasos que se requiere para resolver un determinado problema. Dado que llevar a cabo el conteo sería una tarea extenuante, lo que se obtiene son los límites asintóticos de dicho conteo [82][83].

Existen varias notaciones que permiten representar dichos límites, algunas de ellas son:

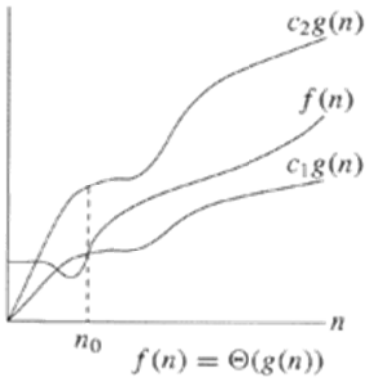
1. Notación Big O
2. Notación Big Theta
3. Notación Big Omega

Siendo la primera una de las más utilizadas.

En las tres notaciones la manera de representar el desempeño del algoritmo es por medio de una función. Dicha función es dependiente del tamaño de los valores de entrada.

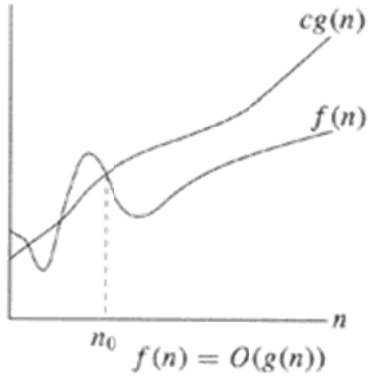
3.3.1 Notación Big Theta o mismo orden

Esta notación delimita a una función dentro de dos factores constantes. Dicho de otra manera, $f(n) = \Theta(g(n))$ si existen constantes positivas n_0, c_1, c_2 de tal forma que a la derecha de n_0 el valor de de $f(n)$ siempre está entre $c_1 g(n)$ y $c_2 g(n)$ inclusive.



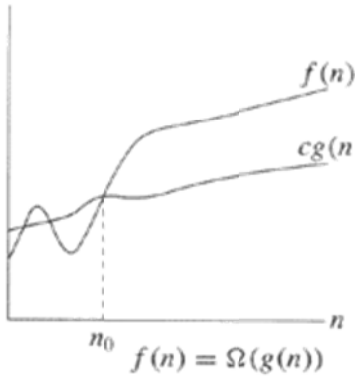
3.3.2 Notación Big O (cota superior)

Esta notación da una cota superior para una función dentro un factor constante. Se puede definir como $f(n) = O(g(n))$ si existe una constante positiva n_0 y c tal que a la derecha de n_0 el valor para $f(n)$ siempre está debajo de $cg(n)$.



3.3.3 Notación Ω (cota inferior)

Esta notación da una cota inferior para una función dentro de un factor constante. Lo podemos escribir como $f(n) = \Omega(g(n))$ si existe n_0 constante positiva y c tal que a la derecha de n_0 el valor para $f(n)$ siempre está sobre $cg(n)$.



El análisis de un algoritmo es una función $g(n)$ que da la cota superior de un número de operaciones llevadas a cabo por un algoritmo cuando el tamaño de la entrada de datos es *n*. Para este caso existen dos posibles interpretaciones.

1. Peor de los casos: el tiempo de ejecución de una entrada de datos de tamaño *n* será menor que la cota superior excepto para algunos valores de la entrada de datos para los cuales se alcanzara el máximo.
2. Caso promedio: el tiempo de ejecución será el promedio para cualquier tamaño dado para *n*.

3.3.4 Ordenes de complejidad

Se conoce como orden de complejidad a una familia de funciones que comparten un mismo comportamiento asintótico. Para cada uno de estos conjuntos se suele identificar a un miembro $f(n)$ que se utiliza como representante de la familia. Frecuentemente no es necesario conocer el comportamiento exacto, basta con conocer una cota superior, es decir, alguna función que se comporte aun peor[82][83].

Las funciones de complejidad algorítmica más habituales en las cuales el único factor del que depende es el tamaño de la muestra de entrada *n*, ordenadas de mayor a menor frecuencia son:

$O(1)$	Orden constante
$O(\log n)$	Orden logarítmico
$O(n)$	Orden lineal
$O(n \log n)$	Orden cuasi-lineal
$O(n^2)$	Orden cuadrático
$O(n^3)$	Orden cúbico
$O(n^a)$	Orden polinómico
$O(2^n)$	Orden exponencial
$O(n!)$	Orden factorial

4 Modelo propuesto

En este capítulo se presenta la propuesta del presente trabajo de tesis. En la sección 4.1 se presentará la hipótesis sobre la cual está basado el modelo propuesto. La sección 4.2 versará sobre una nueva propuesta de búsqueda de k-tuplas en secuencias, basada en la localización de palíndromos, mientras que en la sección 4.3 con base en la propuesta de la sección 4.2, se propone un método para llevar a cabo el alineamiento de pares de secuencias; en la sección 4.4 se presenta el análisis de complejidad de dicho algoritmo. En la sección 4.5 se describe el contexto y la justificación relacionados con un importante ajuste que hemos aplicado al modelo original de este trabajo de tesis, que consiste en una extensión relevante a la propuesta original de la presente investigación; además de realizar el alineamiento de secuencias, extendemos los alcances de la propuesta a fin de enfrentar un problema de actualidad en la bioinformática, cuya solución es tema de investigación en los principales laboratorios del mundo: el alineamiento de genomas. Para ello, en la sección 4.6 se describe un nuevo modelo de memoria heteroasociativa Alfa-Beta robustas a alteraciones en patrones del tipo inserciones, mutaciones y borrados. Por último, en la sección 4.7 se presenta en detalle la propuesta del modelo asociativo para el alineamiento de genomas.

4.1 Hipótesis

Como se mencionó anteriormente, una de las principales aportaciones en las heurísticas para el alineamiento de secuencias está basada en algoritmos que extraen bloques de secuencias de longitud *k* y los comparan con otros segmentos obtenidos de secuencias distintas, con el fin de identificar cuáles de dichas secuencias son las que comparten un mayor número de segmentos iguales.

Para poder llevar a cabo estos alineamientos, y no hacer búsquedas exhaustivas, es necesario identificar características importantes dentro de las secuencias, que nos permitan caracterizarlas. Es por ello que el modelo propuesto en esta tesis sugiere una nueva manera de extraer segmentos y discriminarlos a partir de un conjunto de secuencias que se deseen alinear. *"La hipótesis principal es que existen características intrínsecas en las secuencias de ácidos nucleicos y aminoácidos que son propias de cada familia de proteínas, por lo que si están relacionadas, dichas secuencias contendrán un conjunto de caracteres muy similares"*.

4.2 Búsqueda de similitudes en ADN

Tomando como base lo mencionado en la sección anterior, para el algoritmo propuesto en esta tesis se buscaron ciertas características que se pudieran localizar en una secuencia de ácidos nucleicos. Las primeras que se tomaron son las siguientes:

1. Secuencias palindrómicas.

2. Repeticiones de nucleótidos

El sentido de palíndromo en las secuencia de ADN es distinto del concepto que se aplica en las cadenas de caracteres. Una cadena de caracteres es considerada palíndromo si al leerlo de izquierda a derecha y viceversa se lee de la misma forma, por ejemplo ATA, ABCBA,….

Las secuencias palindrómicas en secuencias de ADN, son aquellas que al leer la cadena guía de izquierda a derecha y la cadena complementaria de derecha a izquierda son lo mismo; son características de sitios de restricción en secuencias de ADN. Estas son secuencias marcadoras que indican a las enzimas dónde pegarse para llevar a cabo un corte en una secuencia determinada. Un ejemplo de una secuencia palindrómica es: CTGCAG, AAATTT ya que sus cadenas complementarias serían las GACGTC y TTTAAA respectivamente

Por otro lado, las repeticiones de nucleótidos son bloques de secuencias de un solo nucleótido, de una longitud dada, por ejemplo: AAAA, GGGG

La razón por la que se decidió tomar dichos parámetros es porque en la literatura en la que se hace referencia a las características que se presentan en secuencias que codifican para proteínas, aparecen estas dos situaciones, que hacen posible encontrar grupos de secuencias que formen un palíndromo [84][85]. Por otro lado, en lo que se refiere a las repeticiones de nucleótidos, conocidas también como regiones de baja complejidad, en los algoritmos de alineamiento se menciona que son las que generan más problemas al evaluar [62][65].

Con base en la hipótesis y las dos características señaladas anteriormente, se propone un algoritmo para el alineamiento de pares de secuencias, que permita buscar similitudes en secuencias de ADN y realizar un primer alineamiento de las secuencias.

1. Es necesario seleccionar dos parámetros:
    a. La longitud de los bloques de la secuencia *k*
    b. El número *r* de nucleótidos repetidos consecutivamente que deseamos que se permita en los bloques de longitud *k*.
2. Para las dos secuencias que se pretende alinear se obtiene el *conjunto característico*:
    a. Se recorre la secuencia, uno a uno, hasta encontrar un bloque de longitud *k* que satisfaga las siguientes condiciones:
        i. Que sea una secuencia palíndromo.
        ii. Que la secuencia tenga a lo más *r* nucleótidos repetidos consecutivamente
    b. Una vez que se encuentra un bloque en la secuencia que cumpla las características, se almacena y se avanza *k* posiciones a partir del primer elemento de la secuencia encontrada.
    c. El proceso se repite hasta llegar al final de la cadena.

3. Una vez que se tienen los conjuntos característicos para cada secuencia, éstos son comparados utilizando el modelo de multimemorias heteroasociativas Alfa-Beta [48], y se eliminan de cada uno de los conjuntos los elementos que no pertenezcan a ambas cadenas. El proceso de comparación utilizando las multimemorias heteroasociativas es el siguiente:

    a. Una vez que se tienen los conjuntos características de las secuencias, se toma el conjunto de la primera secuencia y se crea la memoria asociativa de acuerdo con el modelo de multimemoria heteroasociativa Alfa-Beta [48]. Para ello, es necesario contar con una codificación para las secuencias de ADN que las convierta en código binario, como se muestra en la tabla 3.

    b. Se le presentan a la memoria los elementos del conjunto característico de la segunda cadena, codificándolos de acuerdo con la tabla 3. El objetivo es identificar similitudes entre ambos conjuntos.

4. Posteriormente, para cada uno de los conjuntos característicos:

    a. Se ordenan los arreglos de tal forma que en la misma posición de los arreglos se encuentren secuencias iguales, eliminando del conjunto aquellos que no sean iguales.

    b. Se mide la distancia que hay entre uno de los elementos característicos y el siguiente, comenzando del primero. A esta medida la conoceremos como desplazamiento (*Offset)*. Por cada conjunto característico existirá un *conjunto desplazamiento*.

5. Se comparan dichos conjuntos desplazamiento:

    a. Si para todos los elementos el desplazamiento es idéntico, entonces la secuencia no es muy lejana, evolutivamente hablando, y se lleva a cabo el desplazamiento pertinente para alinear la secuencia.

    b. Si existe(n) algún(os) desplazamiento que no concuerde(n) con el de alguno otro de los demás conjuntos desplazamiento, entonces podremos inferir que en ese bloque de secuencia ha habido un cambio evolutivo más severo, como es el caso de una inserción o borrado de algún o algunos nucleótidos. Por ello, será necesario hacer un alineamiento distinto, y se propone el algoritmo de Smith-Waterman de programación dinámica para aplicarse en los bloques que no concuerden.

Tabla 3. Codificación de nucleótidos a código binario.

Nucleótido	Secuencia Binaria
a	1000
c	0100
t	0010
g	0001

El inciso (b) del punto 5 no está terminado en su totalidad, así que sólo se alinean elementos que no contengan inserciones o borrados.

Algunas de las principales características que presenta este modelo son las siguientes:

1. El rango de longitudes de los bloques, es decir *k*, que son analizados va de entre 5-15, ya que con longitudes menores es más probable de encontrarla más de una vez en la misma secuencia.

2. Se identifican elementos característicos de cada una de las secuencias, que nos permite identificar secciones en una secuencia y llevar a cabo el alineamiento con respecto a ellos.

3. Dependiendo de las secciones que se localicen en las secuencias, será posible determinar si ha habido un cambio evolutivo más significativo como lo son las inserciones y los borrados.

4. La lectura de la secuencia es menor ya que en los algoritmos del estado del arte se realizan $N-k+1$ recorridos en la secuencia para localizar los bloques o tuplas, y en el caso de nuestra propuesta el recorrido se va **decrementando** a $N-k-s(k-1)+1$ donde s es el número de tuplas que cumplen con las características deseadas.

Es importante hacer notar que con el algoritmo propuesto para la nueva búsqueda de k-tuplas **reducimos el tiempo** en el que se recorren las secuencias. Esta afirmación queda demostrada por lo siguiente:

Supongamos el peor de los casos, cuando $s=0$, es decir, que el conjunto característico es el vacío. Entonces, el término $s(k-1)$ es eliminado de la ecuación, luego entonces ambas ecuaciones son idénticas.

El uso del modelo de multimemorias heteroasociativas nos permite identificar, no sólo aquellos elementos de los conjuntos características que son idénticos, sino aquellos que muestran un cierto grado de mutación, esto gracias a las capacidades de soportar alteraciones aditivas, sustractivas y mezcladas de dicho modelo [48].

El algoritmo anterior está diseñado sólo para cadenas de ADN, no para cadenas de aminoácidos. Las características que se utilizan son sólo una primera aproximación, por lo que se seguirá investigando más características que sean más significativas.

Por otro lado, para el caso de las secuencias de aminoácidos, se está investigando sobre características que permitan agrupar ciertos bloques en las secuencias de aminoácidos que, aunque no sean muy similares aminoácido a aminoácido, como conjunto tengan la funcionalidad que caracteriza a la proteína. Entre los elementos característicos en que nos vamos a enfocar, están ciertos patrones de las proteínas como son los $\alpha-helix$, $\beta-sheet$, loop, coil, las cuales son conformaciones estructurales que toman los aminoácidos.

4.3 Algoritmo propuesto para el alineamiento basado en palíndromos

Con base en el algoritmo que se presentó en la sección anterior, se plantea un bosquejo del algoritmo para el alineamiento de las secuencias.

En la figura 6 se muestra el funcionamiento del algoritmo, el cual consta de tres etapas:

- Extractor de características: En esta etapa se identifican los rasgos característicos de las secuencias. Para el caso del ADN se consideran las que están en la sección 4.2; por otro lado, para el caso de las proteínas aún no se tienen seleccionadas las características, las cuales serán buscadas posteriormente.
- Alineamiento Primario: Consiste en alinear los rasgos característicos que se obtuvieron en la etapa anterior permaneciendo sólo aquellos que son coincidentes en ambas secuencias. Una vez alineado, de acuerdo con el conjunto de offset, se lleva a cabo el respectivo desplazamiento.
    - Si para la misma posición en ambos conjuntos de offset los valores son iguales, entonces se lleva a cabo el alineamiento con un desplazamiento.
    - De lo contrario, se pasará a la fase del alineamiento secundario.
- Alineamiento Secundario: Consiste en identificar los lugares donde será necesario utilizar el método de programación dinámica para llevar a cabo el alineamiento. La identificación se llevará a cabo mediante deducciones a partir de los conjunto de offset.

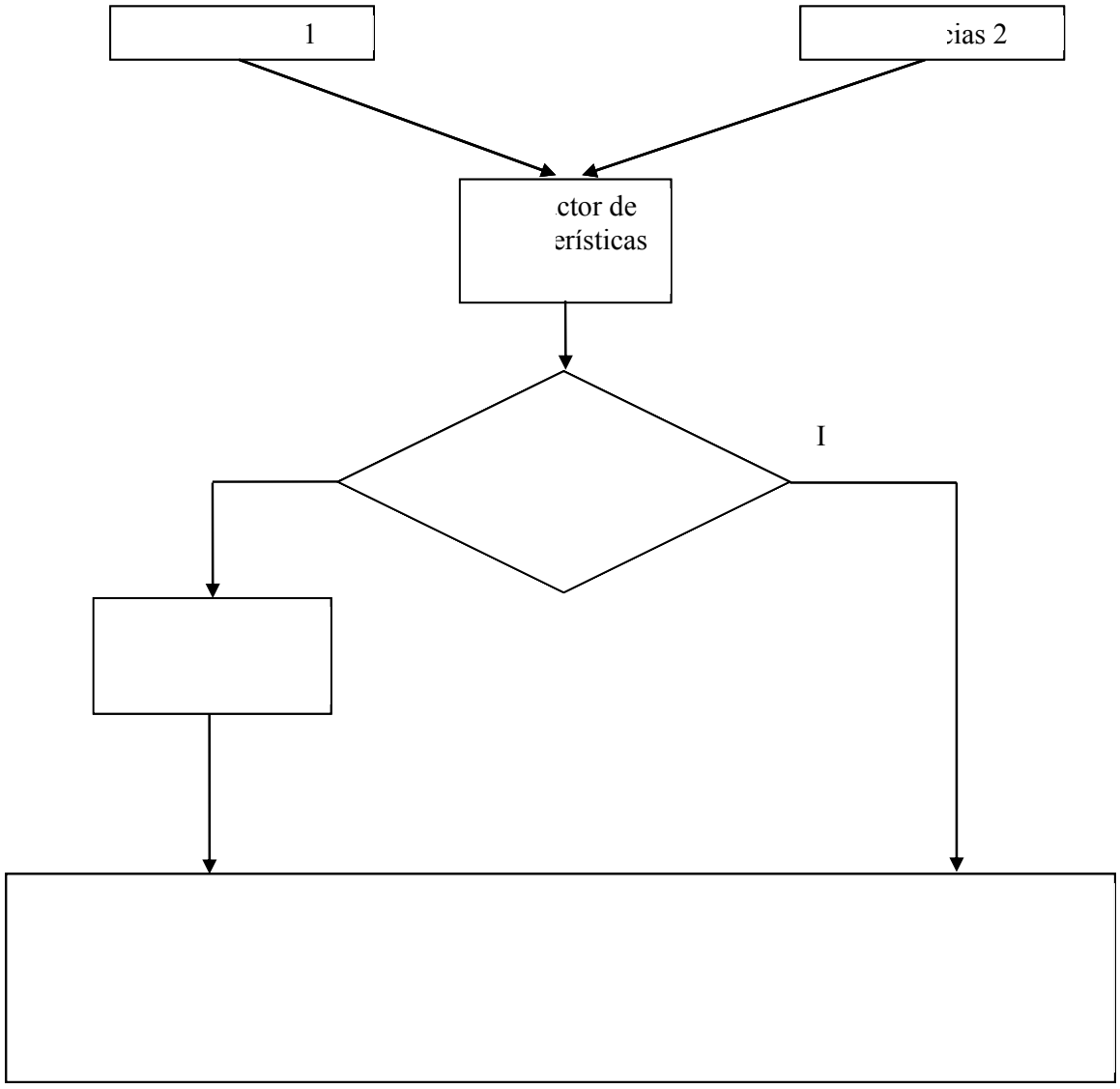


Figura 6. Alineamiento de pares de secuencias

En la figura 7 se muestra el caso del alineamiento múltiple de secuencias, donde se propone un bosquejo similar al del alineamiento por pares de secuencias. La principal diferencia estará en la fase 3 nombrada alineamiento secundario progresivo.

- Extractor de características: En esta etapa se identifican los rasgos característicos de las secuencias. Para el caso del ADN se consideran las que están en la sección 4.1; por otro lado, para el caso de las proteínas aún no se tienen seleccionadas las características.
- Alineamiento Primario: Consiste en alinear los rasgos característicos que se obtuvieron en la etapa anterior permaneciendo sólo aquellos que son

coincidentes en ambas secuencias. Una vez alineado, de acuerdo con el conjunto de offset, se lleva a cabo el respectivo desplazamiento

  - Si para la misma posición en ambos conjuntos de offset los valores son iguales, entonces se lleva a cabo el alineamiento con un desplazamiento.
  - De lo contrario, se pasará a la fase del alineamiento secundario.

- Alineamiento Secundario progresivo: Consiste en identificar los lugares donde será necesario utilizar el método de programación dinámica para llevar a cabo el alineamiento. La identificación se llevará a cabo mediante deducciones a partir de los conjunto de offset. Los alineamiento se harán progresivamente, es decir:
  - Se calculará una matriz de distancia entre los bloques que se pretende alinear.
  - El alineamiento se llevará a cabo con programación dinámica, alineando por pares de secuencias; es decir, primero las dos más cercanas, al alineamiento resultante se le agregará la tercera y así sucesivamente, hasta completar el alineamiento.

Cabe mencionar que los rasgos característicos obtenidos en cada secuencia, pueden variar tanto en cantidad como en contenido, pero sólo se muestran aquellas que son coincidentes entre las secuencias que se están alineando. Por otro lado, para acelerar el proceso de aprendizaje, los rasgos característicos pueden ser almacenados en una biblioteca, y de esta forma ahorrarnos el tiempo que se lleva calcularlo.

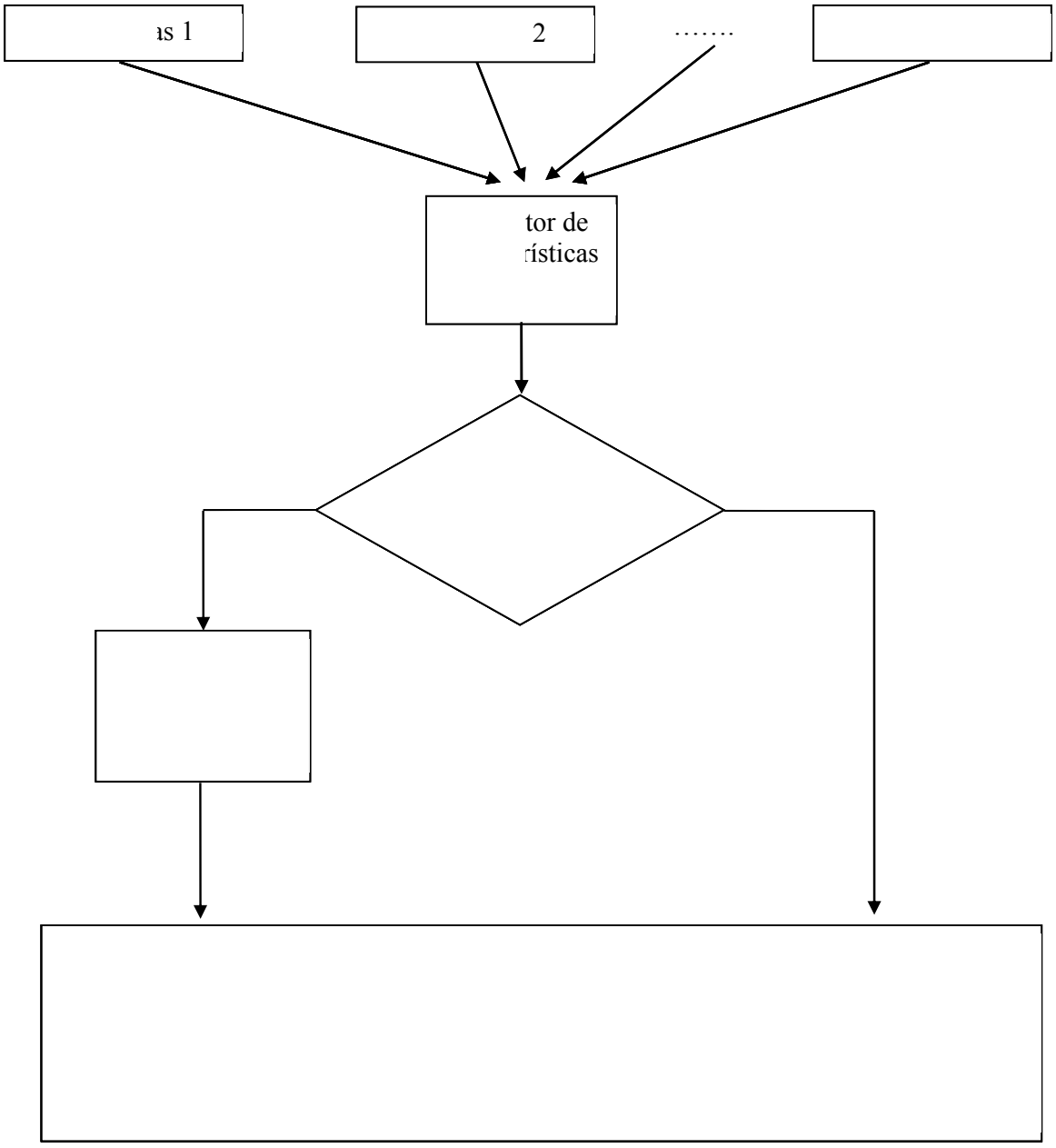


Figura 7. Alineamiento múltiple de secuencias

4.4 Análisis de complejidad

En esta sección presentaremos el análisis de complejidad hecho para el algoritmo presentando en este capítulo, y para ello se utilizará la notación Big O la cual dice que se puede definir como $f(n) = O(g(n))$ si existe una constante positiva n_0 y c tal que a la derecha de n_0 el valor para $f(n)$ siempre está debajo de $cg(n)$.

El análisis del algoritmo se divide en las siguientes tres partes:

1. Obtención del conjunto característico
2. Fase de aprendizaje de la memoria asociativa
3. Fase de recuperación de la memoria asociativa

Por otro lado, es necesario tener en cuenta las siguientes definiciones:

- L : longitud de la secuencia.
- N: número de secuencias.
- K_tupla: longitud de la k-tupla.
- S: número de k-tuplas que cumplen con la característica deseada.
- R: número de recorridos, donde $R = L - K\_tupla - S(K\_tupla - 1) + 1$.

4.4.1 Obtención del conjunto característico

En esta parte se recorre la secuencia para seleccionar el conjunto característico. A continuación se muestra las partes principales del código:

```
for(i=0:N){
      for(j=0:R){
            for(k=0:K_tupla/2){

            }
      }
}
```

El número total de instrucciones es : $N * R * k\_tupla / 2$

$$f(N, R, K\_tupla) = N * R * K\_tupla / 2 .$$

si considerando a N como el parámetro más importante podemos fijar R y K_tupla como valores constantes, la función $f(N, R, K\_tupla)$ la podemos definir como:

$$f(N) = C * N$$

Es claro que $f(N)$ es una ecuación lineal por lo que si definimos $g(N) = c * x + b$ con , $c = C, n_0 = 1$ y $b > 0$ entonces:

$$|f(N)| \leq C|g(k)| \text{ cuando } n_0 > 1$$

por lo que la complejidad es

$$O(N)$$

4.4.2 Fase de aprendizaje de la memoria asociativa

La memoria asociativa que se utilizó en este proceso es la memoria heteroasociativa Alfa-Beta [91] de la cual, por su nueva forma de aprendizaje podemos calcular la siguiente complejidad. El pseudocódigo que representa esta parte se muestra a continuación:

```
for(i=0:N){
      for(k=0:K_tupla){

      }
}
```

El número total de instrucciones es : $N * K\_tupla$

$$f(N, K\_tupla) = N * K\_tupla\ .$$

si considerando a *N* como el parámetro más importante podemos fijar *K_tupla* como valor constante, la función $f(N, K\_tupla)$ la podemos definir como:

$$f(N) = C * N$$

Es claro que $f(N)$ es una ecuación lineal por lo que si definimos $g(N) = c * x + b$ con , $c = C, n_0 = 1$ y $b > 0$ entonces:

$$|f(N)| \leq C|g(k)| \text{ cuando } n_0 > 1$$

por lo que la complejidad es

$$O(N)$$

4.4.3 Fase de recuperación de la memoria asociativa

La memoria asociativa que se utilizó en este proceso es la memoria heteroasociativa Alfa-Beta [45][91] de la cual, por su nueva forma de recuperación podemos calcular la siguiente complejidad. El pseudocódigo que representa esta parte se muestra a continuación:

```
for(i=0:N){
      for(k=0:K_tupla){

      }
}
```

El número total de instrucciones es : $N * K\_tupla$

$$f(N, k\_tupla) = N * K\_tupla\ .$$

si considerando a *N* como el parámetro más importante podemos fijar *K_tupla* como valor constante, la función $f(N, K\_tupla)$ la podemos definir como:

$$f(N) = C * N$$

Es claro que $f(N)$ es una ecuación lineal por lo que si definimos $g(N) = c * x + b$ con , $c = C$, $n_0 = 1$ y $b > 0$ entonces:

$$|f(N)| \leq C|g(k)| \text{ cuando } n_0 > 1$$

por lo que la complejidad es

$$O(N)$$

Como se puede ver, ambos procesos tienen la misma complejidad, debido a que se utilizan las propuestas hechas en [91].

4.5 Ajustes al modelo

En esta sección se describe el contexto y la justificación relacionados con un importante ajuste que hemos aplicado al modelo original de este trabajo de tesis, que consiste en una extensión relevante a la propuesta original de la presente investigación; además de realizar el alineamiento de secuencias extendemos los alcances de la propuesta a fin de enfrentar un problema de actualidad en la bioinformática, cuya solución es tema de investigación en los principales laboratorios del mundo: el alineamiento de genomas. En la sección 4.6 se describirá en detalle la propuesta del modelo asociativo para el alineamiento de genomas.

En la actualidad, los modelos de alineamiento de secuencias más importantes [30][63][65] basan su algoritmo, principalmente, en alineamientos progresivos. Dichos alineamientos progresivos utilizan métricas de distancia entre secuencias y cálculos de árboles filogenéticos, para determinar aquellas que son más similares. Las métricas de distancia, como primer paso, sirven para determinar la cercanía entre las secuencias que se pretenden alinear; posteriormente, dicha matriz es presentada a algún algoritmo de creación de árboles filogenéticos para determinar el orden del alineamiento del conjunto de secuencias [30][63][65]. Dicha situación encierra en un círculo vicioso a estos métodos, ya que el alineamiento depende de un árbol filogenético, el cual a su vez depende de algún tipo de alineamiento [63].

En el modelo presentado en este trabajo se intenta seguir una alternativa diferente, basándose en la hipótesis de que "*existen características intrínsecas en las secuencias de ácidos nucleicos y aminoácidos que son propias de cada familia de proteínas, por lo que si están relacionadas, dichas secuencias contendrán un conjunto de caracteres muy similares*".

En el presente trabajo de tesis se propuso considerar como una característica, dentro de las secuencias de ADN, segmentos que fueran palíndromos. Dichos elementos fueron considerados ya que constituían una característica importante dentro de las cadenas de

ADN [84][85]. No obstante que los resultados presentados fueron prometedores, nos dimos cuenta que dicha característica podría resultar en un menor desempeño por dos razones principales:

1. Los palíndromos que son importantes, y que por lo tanto prevalecen, en las secuencias de ADN se encuentran principalmente en segmentos que NO son codificantes [84], es decir, que NO codifican para proteínas.
2. Debido a la alta mutabilidad de las secuencia, mientras más lejanas éstas fuesen, más complicado sería encontrar palíndromos que fueran compartidos por las secuencia que se pretende alinear [86].

Al reflexionar sobre esto, y siguiendo con nuestra hipótesis, nos hemos percatado de que existen características en el genoma (el cual es una secuencia completa de ADN o aminoácidos) de un organismo que sí pueden ser identificables y que permanecen en la mayoría de los organismos [87]. Dichas características nos permitirán reconocer puntos de anclaje sobre los cuales realizar nuestro alineamiento. A esta práctica se le conoce como alineamiento de genomas [87].

Considerando las aportaciones que tiene el alineamiento de genomas, como: ayuda en la secuenciación de genomas, identificación de regiones conservadas entre genomas, identificación de puntos de ruptura, en las que se incluyen no sólo regiones codificantes sino también No-Codificantes y regiones reguladoras [87], creemos que dicha rama de investigación sería un aporte de gran impacto.

Con base en lo anterior, hemos decidido incursionar en el alineamiento de genomas, y a continuación se presenta un modelo para realizar alineamiento de secuencias en genomas, el cual utiliza memorias asociativas.

4.6 Memoras heteroasociativas Alfa-Beta robustas.

Para poder llevar a cabo el reconocimiento de secuencias de aminoácidos, en el presente trabajo se desarrolla un nuevo modelo de memorias heteroasociativas Alfa-Beta robustas a 3 tipos de alteraciones comunes en las secuencias de aminoácidos: mutaciones, inserciones y borrados. El presente modelo toma como base las memorias asociativas Alfa-Beta con recuperación completa [91] y el algoritmo desarrollado por Needleman–Wunsch [6] para el alineamiento de secuencias.

El presente modelo, al igual que sus antecesores, presenta dos tipos de memorias: *Max* y *Min.*

Primero que nada es necesario expresar la siguiente definición: Sea $x^{\alpha} \in A^{n}$ con $A=\{0,1\}$, $n \in Z^{+}$ y $\alpha \in \{1,2,...,p\}$ un vector fila. Por otro lado, sea $q \in Z^{+}$ la dimensión del vector más pequeño extraido de x^{α}. Entonces la *operación de particionamiento vectorial* $\rho(x^{\alpha},q)$ se define como el conjunto de vectores fila q-dimensionales y se denota por:

$$\rho(x^{\alpha},q)=\left\{x^{\alpha 0},x^{\alpha 1},...,x^{\alpha\frac{n}{q}-1}\right\}$$

tal que $x_r^{\alpha l} = x_{l+r}^{\alpha}$ donde $x^{\alpha l} \in A^q$ con $l \in \left\{0,1,2,...,\frac{n}{q}-1\right\}$ y $r \in \{1,2,...,q\}$.

4.6.1 Memoria heteroasociativa robusta Alfa-Beta tipo Max

4.6.1.1 Fase de aprendizaje

Sea $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^p$ con $n, p \in Z^+$, vectores de entrada y de salida respectivamente. El conjunto fundamental derivado de dichos vectores, será $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1,2,...,p\}$ de tal forma que $\exists \delta, \sigma \in \{1,2,...,p\}$ donde $\mathbf{x}^\delta \in A^b$, $\mathbf{x}^\sigma \in A^c$ con $b, c \in Z^+$ y $b \neq c$. Por otro lado, los vectores $\mathbf{y}$ son construidos con la codificación *one-hot*: se asigna para $\mathbf{y}^\mu$ los siguientes valores: $y_k^\mu = 1$, y $y_j^\mu = 0$ para $j = 1,2,\ldots,k-1,k+1,\ldots,p$ donde $k \in \{1,2,\ldots,p\}$. Y para cada vector $\mathbf{y}^\mu$ le corresponde *un y solo un* vector $\mathbf{x}^\mu$.

Para cada $\mu \in \{1,2,...p\}$, de la pareja $(\mathbf{x}^\mu, \mathbf{y}^\mu)$ se construye la matriz: $[y^\mu \boxtimes (x^\mu)^t]$
una vez hecho esto, el operador binario max $\vee$ se aplica a las matrices. Posteriormente, la matriz $\mathbf{V}$ se obtiene como sigue: $V = \bigvee_{\mu=1}^{p} [(y^\mu) \boxtimes (x^\mu)^t]$ donde el componente ij-th esta dado por: $v_{ij} = \bigvee_{\mu=1}^{p} \alpha(y_i^\mu, x_j^\mu)$.

4.6.1.2 Fase de recuperación

El patron $\mathbf{x}^\varpi$, el cual puede ser o no de la misma dimensión de los que forman el conjunto fundamental, se presenta a la matriz $\mathbf{V}$, obteniendo como resultado el vector $\mathbf{y} \in A^p$ el cual es contruido como se presenta a continuación: Primero, el operador de particionamiento es aplicado a $(v_i)^t \in B^a$ y $\mathbf{x}^\varpi \in A^b$ donde a y b pueden ser o no de la misma dimensión.

$$\rho(x^\varpi, q) = \left\{x^{\varpi 0}, x^{\varpi 1}, ..., x^{\varpi \frac{b}{q}-1}\right\}; \rho(v_i, q) = \left\{v_i^0, v_i^1, ..., v_i^{\frac{a}{q}-1}\right\}$$

Para toda $i \in \{1,2,...,p\}$, $c \in \left\{0,1,2,...\frac{a}{q}-1\right\}$, $h \in \left\{0,1,2,...,\frac{b}{q}-1\right\}$ with $C = \frac{a}{q} - 1$ and $H = \frac{b}{q} - 1$ el i-esimo componente del vector de salida Z^ω se calcula como sigue:

$$z_i^\omega = \begin{cases} 1, & F_{CH}^i = \bigvee_{k=1}^{p} F_{CH}^k \quad y \quad F_{CH}^i > 0 \\ 0, & en\ otro\ caso \end{cases}$$

a partir de este se obtiene la matriz $\mathbf{F}^i$ de la siguiente manera:

$$F_{ch}^i = \bigvee \left(F_{c-1,h-1} + S(v_i^{h-1}, x^{\omega c-1}) * \eta, F_{c-1,h} + d, F_{c,h-1} + d\right)$$

donde:

$$S(v_i^{h-1}, x^{\omega c-1}) = \begin{cases} -1, & \bigwedge_{j=1}^{q} \beta(v_{ij}^{h-1}, x_j^{\omega c-1}) \neq 1 \\ 1, & en\ otro\ caso \end{cases}$$

donde $F_{c,0} = c * d, F_{0,h} = h * d$ con $d \in \mathbf{Z}^-$ siendo una penalización llamada gap, y donde η un factor que multiplica el resultado del reconocimiento de la memoria asociativa Alfa-Beta.

Una vez que se tiene el vector columna $\mathbf{z}^{\varpi}$, es necesario construir el *vector suma max* **s**, el cual contiene en su $i-ésima$ componente la suma de las componentes de la $i-ésima$ fila de la matriz **V**:

$$s_i = \sum_{j=1}^{n} T_j$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_i = \begin{cases} 1 & \leftrightarrow \quad v_{ij} = 1 \\ 0 & \leftrightarrow \quad v_{ij} \neq 1 \end{cases}$$

$\forall j \in \{1, 2, ..., n\}$.

por lo tanto la correspondiente $\mathbf{y}^{\varpi}$ está dada por:

$$\mathbf{y}_i^{\varpi} = \begin{cases} 1 & \text{si} \quad s_i = \bigvee_{k \in \theta} s_k \quad \wedge \quad z_i^{\varpi} = 1 \\ 0 & \textit{en otro caso} \end{cases}$$

donde $\theta = \{i \mid Z_i^{\varpi} = 1\}$.

Dado que no sólo es importante obtener la puntuación optima sino obtener el alineamiento optimo, es necesario tener un puntero que indique el siguiente movimiento en la matriz de alineamiento *F*, estos punteros son: (I) *izquierda,(D)derecha, (A)arriba.* Cada vez que se calcula el máximo de cada posición de la matriz se almacena un puntero que indica la componente de donde se obtubo el máximo. En caso de que dos componentes distintos den un máxmimo identico, ambos valores son almacenados. Una vez que la matriz ha sido calculada, el valor F_{pn} se toma como el punto de partida para construir el alineamiento optimo y se va siguiendo los punteros que fueron asignados a cada posición de la matriz.

4.6.2 Memoria heteroasociativa robusta Alfa-Beta tipo Min

4.6.2.1 Fase de aprendizaje

Sean $\mathbf{x} \in A^n$ y $\mathbf{y} \in A^p$ con $n, p \in Z^+$, vectores de entrada y de salida respectivamente. Su correspondiente conjunto fundamental está dado por $\{(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}) \mid \mu = 1,2,...,p\}$ tal que $\exists \delta, \sigma \in \{1,2,...,p\}$ donde $\mathbf{x}^{\delta} \in A^b$, $\mathbf{x}^{\sigma} \in A^c$ con $b, c \in Z^+$ y $b \neq c$. Además, los vectores **y** son construidos con la codificación *cero-hot*: asignando para $\mathbf{y}^{\mu}$ los siguientes valores: $y_k^{\mu} = 0$, y $y_j^{\mu} = 1$ para $j = 1,2,\ldots,k-1,k+1,\ldots,p$ donde $k \in \{1,2,\ldots,p\}$. Y para cada vector $\mathbf{y}^{\mu}$ corresponde *un y sólo un* vector $\mathbf{x}^{\mu}$.

Para cada $\mu \in \{1,2,...p\}$, a partir de la pareja $(\mathbf{x}^{\mu}, \mathbf{y}^{\mu})$ se construye la matriz: $\left[\mathbf{y}^{\mu})(\mathbf{x}^{\mu})^{t}\right]_{p\times n}$ luego entonces, el operador binario $\wedge$ se aplica a las matrices. La matriz $\mathbf{\Lambda}$ se esta dada por: $\mathbf{\Lambda} = \bigwedge_{\mu=1}^{p}\left[\mathbf{y}^{\mu})(\mathbf{x}^{\mu})^{t}\right]$ donde la ij -esima componente esta dada por: $\lambda_{ij} = \bigwedge_{\mu=1}^{p} \alpha(y_i^{\mu}, x_j^{\mu})$.

4.6.2.2 Fase de recuperación

El patrón fundamental $\mathbf{x}^{\varpi}$, que puede o no ser del mismo tamaño que otro patrón fundamental, se presenta a $\mathbf{\Lambda}$, luego entonces el vector $\mathbf{y} \in A^{p}$ se construye como sigue: Primero, se aplica la operación de particionamiento vectorial a $\lambda_i \in B^{a}$ y $\mathbf{x}^{\varpi} \in A^{b}$ donde a y b pertenecen al conjunto fundamental.

$$\rho\left(x^{\varpi}, q\right) = \left\{x^{\varpi 0}, x^{\varpi 1}, ..., x^{\varpi \frac{b}{q}-1}\right\}; \rho\left(v_i, q\right) = \left\{v_i^{0}, v_i^{1}, ..., v_i^{\frac{a}{q}-1}\right\}$$

Dado $i \in \{1,2,...,p\}$, $c \in \left\{0,1,2,...\frac{a}{q}-1\right\}$, $h \in \left\{0,1,2,...,\frac{b}{q}-1\right\}$, la i-esima componente del vector de salida z^{ω} se expresa como:

$$z_i^{\omega} = \begin{cases} 1, & F_{CH}^{i} = \bigvee_{k=1}^{p} F_{CH}^{k} \quad y \quad F_{CH}^{i} > 0 \\ 0, & en\ otro\ caso \end{cases}$$

luego entonces la matriz $\mathbf{F}^{i}$ se construye como sigue:

$$F_{ch}^{i} = \bigvee\left(F_{c-1,h-1} + (S(\lambda_i^{h-1}, x^{\omega c-1}) * \eta), F_{c-1,h} + d, F_{c,h-1} + d\right)$$

donde:

$$S(\lambda_{i}^{h-1}, x^{\omega c-1}) = \begin{cases} -1, & \bigvee_{j=1}^{q} \beta(\lambda_{ij}^{h-1}, x_j^{\omega c-1}) \neq 1 \\ 1, & en\ otro\ caso \end{cases}$$

con $F_{c,0} = c * d, F_{0,h} = h * d$ y $d \in \mathbf{Z}^{-}$ una penalización conocida como gap y η un factor que incrementa el resultado

Una vez que se tiene el vector columna $\mathbf{z}^{\varpi}$, es necesario construir el *vector suma min* $\mathbf{r} \in Z^{p}$ el cual contiene en su $i-ésima$ componente el número de ceros de la $i-ésima$ fila de la matriz $\mathbf{\Lambda}$.

$$r_i = \sum_{j=1}^{n} \sim T_j$$

donde $T \in B^n$ y las componentes de T se definen así:

$$T_i = \begin{cases} 0 & \leftrightarrow \quad \lambda_{ij} = 0 \\ 1 & \leftrightarrow \quad \lambda_{ij} \neq 0 \end{cases}$$

$\forall j \in \{1, 2, ..., n\}$.

por la tanto la correspondiente $\mathbf{y}^{\varpi}$ esta dada por:

$$\mathbf{y}_i^{\varpi} = \begin{cases} 0 & \text{si} \quad r_i = \bigwedge_{k \in \theta} r_k \quad \wedge \quad z_k^{\varpi} = 0 \\ 1 & \text{de } \textit{otra caso} \end{cases}$$

donde $\theta = \{i \mid Z_i^{\varpi} = 0\}$.

Dado que no sólo es importante obtener la puntuación optima sino obtener el alineamiento optimo, es necesario tener un puntero que indique el siguiente movimiento en la matriz de alineamiento *F*, estos punteros son: (I) *izquierda,(D)derecha, (A)arriba.* Una vez que la matriz ha sido calculada, el valor F_{pn} se toma como el punto de partida para construir el alineamiento optimo y se va siguiendo los punteros que fueron asignados a cada posición de la matriz.

4.6.3 Ejemplo

4.6.3.1 Tipo Max

El siguiente ejemplo nos muestra la fase de aprendizaje y recuperación del nuevo algoritmo de memoria heteroasociativa Alfa-Beta robusta tipo *Max*:

Sean los patrones de entrada:

$$x^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Y sus correspondientes clases

$$y^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, y^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, y^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, y^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Las clases están construidas de acuerdo a la codificación one-hot y a cada clase le corresponde uno y sólo uno de los patrones de entrada, por lo que el conjunto fundamental queda de la siguiente manera:

$$\left\{\left(x^1, y^1\right), \left(x^2, y^2\right), \left(x^3, y^3\right), \left(x^4, y^4\right)\right\}$$

Fase de aprendizaje

A partir del conjunto fundamental se construye la memoria asociativa como se muestra a continuación:

$$\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \boxtimes \begin{pmatrix}1 & 0 & 1 & 1 & 0\end{pmatrix} = \begin{pmatrix}1 & 2 & 1 & 1 & 2\\0 & 1 & 0 & 0 & 1\\0 & 1 & 0 & 0 & 1\\0 & 1 & 0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}0\\1\\0\\0\end{pmatrix} \boxtimes \begin{pmatrix}0 & 1 & 0 & 0 & 0\end{pmatrix} = \begin{pmatrix}1 & 0 & 1 & 1 & 1\\2 & 1 & 2 & 2 & 2\\1 & 0 & 1 & 1 & 1\\1 & 0 & 1 & 1 & 1\end{pmatrix}$$

$$\begin{pmatrix}0\\0\\1\\0\end{pmatrix} \boxtimes \begin{pmatrix}1 & 0 & 0 & 1 & 0\end{pmatrix} = \begin{pmatrix}0 & 1 & 1 & 0 & 1\\0 & 1 & 1 & 0 & 1\\1 & 2 & 2 & 1 & 2\\0 & 1 & 1 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}0\\0\\0\\1\end{pmatrix} \boxtimes \begin{pmatrix}1 & 1 & 0 & 1 & 1\end{pmatrix} = \begin{pmatrix}0 & 0 & 1 & 0 & 0\\0 & 0 & 1 & 0 & 0\\0 & 0 & 1 & 0 & 0\\1 & 1 & 2 & 1 & 1\end{pmatrix}$$

Se aplica el operador binario máximo $\vee$ a las matrices obtenidas en el paso anterior.

$$\mathbf{V} = \begin{pmatrix}1 & 2 & 1 & 1 & 2\\2 & 1 & 2 & 2 & 2\\1 & 2 & 2 & 1 & 2\\1 & 1 & 2 & 1 & 1\end{pmatrix}$$

Fase de recuperación

Al presentarle un patrón x^{ϖ} con $\varpi \in \{1, 2, ..., p\}$, particularmente x^4, a la memoria $\mathbf{V}$ obtenida en la fase de aprendizaje, de acuerdo con la fase de recuperación del algoritmo de memorias heteroasociativas robustas tipo *Max* propuesto en esta tesis, tenemos que:

Se aplica el operador de particionamiento para los vectores x^4 $y\ para\ v_i$ con una $q = 3$, obteniendo los siguientes vectores:

$$\rho(x^4, 3) = \{(1,1,0), (1,0,1), (0,1,1)\}$$
$$\rho(v_1, 3) = \{(1,2,1), (2,1,1), (1,1,2)\}$$
$$\rho(v_2, 3) = \{(2,1,2), (1,2,2), (2,2,2)\}$$
$$\rho(v_3, 3) = \{(1,2,2), (2,2,1), (2,1,2)\}$$
$$\rho(v_4, 3) = \{(1,1,2), (1,2,1), (2,1,1)\}$$

Una vez que se cuenta con los vectores particionados lo siguiente es obtener, para cada indice $i \in \{1,2,\dots,p\}$ las matrices F^i. Para ello es necesario definir los valores para $d = -1$ y $\eta = 3$. Las matrices se calculan de la siguiente forma:

$$\rho(x^4, 3) = \{(1,1,0), (1,0,1), (0,1,1)\}$$
$$\rho(v_1, 3) = \{(1,2,1), (2,1,1), (1,1,2)\}$$

$$F_{11}^1 = \vee\left(F_{0,0} + S(v_1^0, x^{4(1-1)}) * 3, F_{0,1} + (-1), F_{1,0} + (-1)\right) = \vee(0+0*3, -1-1, -1-1) = \vee(0, -2, -2) = 0$$
$$F_{12}^1 = \vee\left(F_{0,1} + S(v_1^1, x^{4(1-1)}) * 3, F_{0,2} + (-1), F_{1,1} + (-1)\right) = \vee(-1+0*3, -2-1, 0-1) = \vee(-1, -3, -1) = -1$$
$$F_{13}^1 = \vee\left(F_{0,2} + S(v_1^2, x^{4(1-1)}) * 3, F_{0,3} + (-1), F_{1,2} + (-1)\right) = \vee(-2+1*3, -3+(-1), -1+(-1)) = \vee(-1, -4, -2) = -1$$
$$F_{21}^1 = \vee\left(F_{1,0} + S(v_1^0, x^{4(2-1)}) * 3, F_{1,1} + (-1), F_{2,0} + (-1)\right) = \vee(-1+1*3, 0-1, -2-1) = \vee(0, -1, -3) = 0$$
$$F_{22}^1 = \vee\left(F_{1,1} + S(v_1^1, x^{4(2-1)}) * 3, F_{1,2} + (-1), F_{2,1} + (-1)\right) = \vee(0+0*3, -1+(-1), 0+(-1)) = \vee(0, -2, -1) = 0$$
$$F_{23}^1 = \vee\left(F_{1,2} + S(v_1^2, x^{4(2-1)}) * 3, F_{1,3} + d, F_{2,2} + d\right) = \vee(-1+0*3, -1+(-1), 0+(-1)) = \vee(-1, -2, -1) = -1$$
$$F_{31}^1 = \vee\left(F_{2,0} + S(v_1^0, x^{4(3-1)}) * 3, F_{2,1} + (-1), F_{3,0} + (-1)\right) = \vee(-2+0*3, 0+(-1), -3+(-1)) = \vee(-2, -1, -4) = -1$$
$$F_{32}^1 = \vee\left(F_{2,1} + S(v_1^1, x^{4(3-1)}) * 3, F_{2,2} + (-1), F_{3,1} + (-1)\right) = \vee(0+1*3, 0-1, -1-1) = \vee(3, -1, -2) = 3$$
$$F_{33}^1 = \vee\left(F_{2,2} + S(v_1^2, x^{4(3-1)}) * 3, F_{2,3} + (-1), F_{3,2} + (-1)\right) = \vee(0+0*3, 1-1, 3-1) = \vee(0, 0, 2) = 2$$

$$F^1 = \begin{bmatrix} & & v_1^0 & v_1^1 & v_1^2 \\ & 0 & -1 & -2 & -3 \\ x^{40} & -1 & 0 & -1 & -1 \\ x^{41} & -2 & 0 & 0 & 1 \\ x^{42} & -3 & -1 & 3 & 2 \end{bmatrix}$$

Una vez calculada la matriz se pude determinar cual es la puntuación optima para la matriz F^1 siendo la componente F_{CH}^1=2.

$$\rho(x^4, 3) = \{(1,1,0), (1,0,1), (0,1,1)\}$$
$$\rho(v_2, 3) = \{(2,1,2), (1,2,2), (2,2,2)\}$$

$$F_{11}^2 = \vee\left(F_{0,0} + S(v_2^0, x^{4(1-1)}) * 3, F_{0,1} + (-1), F_{1,0} + (-1)\right) = \vee(0+1*3, -1-1, -1-1) = \vee(3, -2, -2) = 3$$
$$F_{12}^2 = \vee\left(F_{0,1} + S(v_2^1, x^{4(1-1)}) * 3, F_{0,2} + (-1), F_{1,1} + (-1)\right) = \vee(-1+1*3, -2-1, 3-1) = \vee(2, -3, 2) = 2$$
$$F_{13}^2 = \vee\left(F_{0,2} + S(v_2^2, x^{4(1-1)}) * 3, F_{0,3} + (-1), F_{1,2} + (-1)\right) = \vee(-2+1*3, -3+(-1), 2+(-1)) = \vee(1, -4, 1) = 1$$
$$F_{21}^2 = \vee\left(F_{1,0} + S(v_2^0, x^{4(2-1)}) * 3, F_{1,1} + (-1), F_{2,0} + (-1)\right) = \vee(-1+0*3, 3-1, -2-1) = \vee(-1, 2, -3) = 2$$

$$F_{22}^{2} = \bigvee\left(F_{1,1} + S(v_2^1, x^{4(2-1)}) * 3, F_{1,2} + (-1), F_{2,1} + (-1)\right) = \vee(3+1*3,2+(-1),2+(-1)) = \vee(6,1,1) = 6$$
$$F_{23}^{2} = \bigvee\left(F_{1,2} + S(v_2^2, x^{4(2-1)}) * 3, F_{1,3} + d, F_{2,2} + d\right) = \vee(2+1*3,1+(-1),6+(-1)) = \vee(5,0,5) = 5$$
$$F_{31}^{2} = \bigvee\left(F_{2,0} + S(v_2^0, x^{4(3-1)}) * 3, F_{2,1} + (-1), F_{3,0} + (-1)\right) = \vee(-2+1*3,2+(-1),-3+(-1)) = \vee(1,1,-4) = 1$$
$$F_{32}^{2} = \bigvee\left(F_{2,1} + S(v_2^1, x^{4(3-1)}) * 3, F_{2,2} + (-1), F_{3,1} + (-1)\right) = \vee(2+0*3,6-1,1-1) = \vee(2,5,0) = 5$$
$$F_{33}^{2} = \bigvee\left(F_{2,2} + S(v_2^2, x^{4(3-1)}) * 3, F_{2,3} + (-1), F_{3,2} + (-1)\right) = \vee(6+1*3,5-1,5-1) = \vee(9,4,4) = 9$$

$$F^1 = \begin{bmatrix} & v_2^0 & v_2^1 & v_2^2 \\ & 0 & -1 & -2 & -3 \\ x^{40} & -1 & 3 & 2 & 1 \\ x^{41} & -2 & 2 & 6 & 5 \\ x^{42} & -3 & 1 & 5 & 6 \end{bmatrix}$$

$$\rho(x^4, 3) = \{(1,1,0), (1,0,1), (0,1,1)\}$$
$$\rho(v_3, 3) = \{(1,2,2), (2,2,1), (2,1,2)\}$$

$$F_{11}^{3} = \bigvee\left(F_{0,0} + S(v_3^0, x^{4(1-1)}) * 3, F_{0,1} + (-1), F_{1,0} + (-1)\right) = \vee(0+1*3,-1-1,-1-1) = \vee(3,-2,-2) = 3$$
$$F_{12}^{3} = \bigvee\left(F_{0,1} + S(v_3^1, x^{4(1-1)}) * 3, F_{0,2} + (-1), F_{1,1} + (-1)\right) = \vee(-1+0*3,-2-1,3-1) = \vee(-1,-3,2) = 2$$
$$F_{13}^{3} = \bigvee\left(F_{0,2} + S(v_3^2, x^{4(1-1)}) * 3, F_{0,3} + (-1), F_{1,2} + (-1)\right) = \vee(-2+1*3,-3+(-1),2+(-1)) = \vee(1,-4,1) = 1$$
$$F_{21}^{3} = \bigvee\left(F_{1,0} + S(v_3^0, x^{4(2-1)}) * 3, F_{1,1} + (-1), F_{2,0} + (-1)\right) = \vee(-1+1*3,3-1,-2-1) = \vee(2,2,-3) = 2$$
$$F_{22}^{3} = \bigvee\left(F_{1,1} + S(v_3^1, x^{4(2-1)}) * 3, F_{1,2} + (-1), F_{2,1} + (-1)\right) = \vee(3+1*3,2+(-1),2+(-1)) = \vee(6,1,1) = 6$$
$$F_{23}^{3} = \bigvee\left(F_{1,2} + S(v_3^2, x^{4(2-1)}) * 3, F_{1,3} + d, F_{2,2} + d\right) = \vee(2+0*3,1+(-1),6+(-1)) = \vee(2,0,5) = 5$$
$$F_{31}^{3} = \bigvee\left(F_{2,0} + S(v_3^0, x^{4(3-1)}) * 3, F_{2,1} + (-1), F_{3,0} + (-1)\right) = \vee(-2+0*3,2+(-1),-3+(-1)) = \vee(-2,1,-4) = 1$$
$$F_{32}^{3} = \bigvee\left(F_{2,1} + S(v_3^1, x^{4(3-1)}) * 3, F_{2,2} + (-1), F_{3,1} + (-1)\right) = \vee(2+1*3,6-1,1-1) = \vee(-1,5,0) = 5$$
$$F_{33}^{3} = \bigvee\left(F_{2,2} + S(v_3^2, x^{4(3-1)}) * 3, F_{2,3} + (-1), F_{3,2} + (-1)\right) = \vee(6+1*3,5-1,5-1) = \vee(9,4,4) = 9$$

$$F^1 = \begin{bmatrix} & v_2^0 & v_2^1 & v_2^2 \\ & 0 & -1 & -2 & -3 \\ x^{40} & -1 & 3 & 2 & 1 \\ x^{41} & -2 & 2 & 6 & 5 \\ x^{42} & -3 & 1 & 5 & 9 \end{bmatrix}$$

$$\rho(x^4, 3) = \{(1,1,0), (1,0,1), (0,1,1)\}$$
$$\rho(v_4, 3) = \{(1,1,2), (1,2,1), (2,1,1)\}$$

$$F_{11}^{4} = \bigvee\left(F_{0,0} + S(v_4^0, x^{4(1-1)}) * 3, F_{0,1} + (-1), F_{1,0} + (-1)\right) = \vee(0+1*3,-1-1,-1-1) = \vee(3,-2,-2) = 3$$
$$F_{12}^{4} = \bigvee\left(F_{0,1} + S(v_4^1, x^{4(1-1)}) * 3, F_{0,2} + (-1), F_{1,1} + (-1)\right) = \vee(-1+0*3,-2-1,3-1) = \vee(-1,-3,2) = 2$$
$$F_{13}^{4} = \bigvee\left(F_{0,2} + S(v_4^2, x^{4(1-1)}) * 3, F_{0,3} + (-1), F_{1,2} + (-1)\right) = \vee(-2+0*3,-3+(-1),2+(-1)) = \vee(-2,-4,1) = 1$$
$$F_{21}^{4} = \bigvee\left(F_{1,0} + S(v_4^0, x^{4(2-1)}) * 3, F_{1,1} + (-1), F_{2,0} + (-1)\right) = \vee(-1+0*3,3-1,-2-1) = \vee(-1,2,-3) = 2$$
$$F_{22}^{4} = \bigvee\left(F_{1,1} + S(v_4^1, x^{4(2-1)}) * 3, F_{1,2} + (-1), F_{2,1} + (-1)\right) = \vee(3+1*3,2+(-1),2+(-1)) = \vee(6,1,1) = 6$$

$$F_{23}^4 = \bigvee\left(F_{1,2} + S(v_4^2, x^{4(2-1)}) * 3, F_{1,3} + d, F_{2,2} + d\right) = \vee(2+0*3{,}1+(-1){,}6+(-1)) = \vee(2{,}0{,}5) = 5$$
$$F_{31}^4 = \bigvee\left(F_{2,0} + S(v_4^0, x^{4(3-1)}) * 3, F_{2,1} + (-1), F_{3,0} + (-1)\right) = \vee(-2+0*3{,}2+(-1){,}-3+(-1)) = \vee(-2{,}1{,}-4) = 1$$
$$F_{32}^4 = \bigvee\left(F_{2,1} + S(v_4^1, x^{4(3-1)}) * 3, F_{2,2} + (-1), F_{3,1} + (-1)\right) = \vee(2+1*3{,}6-1{,}1-1) = \vee(-1{,}5{,}0) = 5$$
$$F_{33}^4 = \bigvee\left(F_{2,2} + S(v_4^2, x^{4(3-1)}) * 3, F_{2,3} + (-1), F_{3,2} + (-1)\right) = \vee(6+1*3{,}5-1{,}5-1) = \vee(6{,}5{,}1) = 9$$

$$F^4 = \begin{bmatrix} & v_4^0 & v_4^1 & v_4^2 \\ & 0 & -1 & -2 & -3 \\ x^{40} & -1 & 3 & 2 & 1 \\ x^{41} & -2 & 2 & 6 & 5 \\ x^{42} & -3 & 1 & 5 & 9 \end{bmatrix}$$

Finalmente se calcula el vector z^ω y porsteriormente y^ω:

$$z^4 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{s} = \begin{pmatrix} 3 \\ 1 \\ 2 \\ 4 \end{pmatrix}$$

Para finalizar se calcula el vector de salida y nuevo algoritmo de memorias heteroasociativas robustas tipo *Max*:

$$y^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

ya que el máximo valor de $\mathbf{s}_j$ donde $z_j^4 = 1 \forall j \in \{1,2,3,4\}$ es 4.

Dada, la forma en que está definida la creación de las memorias heteroasociativas Alfa-Beta presentadas en esta sección y de acuerdo con [91] el índice *i* del vector resultante *y* corresponde al *i*-esimo patrón de aprendizaje, por lo que es posible alinear el patrón x^i de los patrones de aprendizaje con el correspondiente patrón desconocido x^ω que se pretende identificar.

El alineamiento se lleva a cabo de acuerdo con la matriz de punteros que se almacena para cada F^i. Para el caso de x^4 la matriz de punteros sería la siguiente:

$$F^4 = \begin{bmatrix} & v_4^0 & v_4^1 & v_4^2 \\ & 0 & I & I & I \\ x^{40} & A & D & I & I \\ x^{41} & A & A & D & D \\ x^{42} & A & A & D & D \end{bmatrix}$$

Para construir el alineamiento, tomamos la componente inferior derecha y partimos de ahí para seguir los punteros:

$$\begin{aligned} \rho(x^4, 3) &\rightarrow 110 \quad 101 \quad 011 \\ \rho(x^\omega, 3) &\rightarrow 110 \quad 101 \quad 011 \end{aligned}$$

Por último se regresan las secuencias a su estado original, tomando el primer triplete la secuencia completa y concatenando el par de caracteres de la extrema derecha, como se muestra a continuación:

$$\begin{matrix} 110 & 1 & 1 \\ 110 & 1 & 1 \end{matrix}$$

4.6.3.2 Tipo Min

En el caso de la memoria heteroasociativa Alfa-Beta robusta tipo *Min*. Para este caso no se coloca el ejemplo ya que el procedimiento es muy similar al de la memoria heteroasociativa robusta tipo *Max* con la salvedad de que la las asociaciones con las que se crean el conjunto fundamental se asocian con vectores zero-hot y que la operación:

$$\bigwedge_{j=1}^{q} \beta\left(v_{ij}^{h-1}, x_j^{\omega c-1}\right)$$

Cambia por:

$$\bigvee_{j=1}^{q} \beta(\lambda_{ij}^{h-1}, x_j^{\omega c-1})$$

en la función:

$$S(\lambda_i^{h-1}, x^{\omega c-1}) = \begin{cases} -1, & \bigwedge_{j=1}^{q} \beta(\lambda_{ij}^{h-1}, x_j^{\omega c-1}) \neq 1 \\ 1, & en\ otro\ caso \end{cases}$$

4.7 Modelo Asociativo para el alineamiento de genomas

En esta sección se propone el modelo de alineamiento de genomas, basado en características intrínsecas de las secuencias de ADN.

4.7.1 Características de los genomas

Primero que nada, y tomando como base la hipótesis planteada en esta tesis, es importante identificar características intrínsecas en las secuencias, las cuales nos permitan determinar ciertas anclas para el alineamiento.

El proceso de reconocimiento de dichas características o patrones en los seres vivos es como sigue; cuando el organismo necesita codificar una proteína, existen elementos que se encargan de localizar los genes utilizando ciertos patrones de secuencias, el principal es conocido como promotor [87], dicho promotor es el que determina el comienzo de un gen y su frecuencia de traducción en proteína. Dentro del promotor, las secuencias que se han identificado son la caja **Pribnow** y la secuencia consenso **-35** [87]. Además existen secuencias consenso, particularmente en bacterias, conocida como **Shine-Dalgarno** de 4-8 bases antes del codón de inicio. En el caso de los eucariontes existe la secuencia **Kozak** la cual funge la misma función que la secuencia Shine-Dalgarno. Por otro lado, también se conoce el codón que indica el inicio de la secuencia de un gen, ATG, y los codones que indican la terminación del gen, TAG, TAA, TGA. Aunque esto

puede variar en algunos casos muy particulares, por lo que el algoritmo deberá de sufrir ligeras modificaciones cuando se analicen las excepciones [87].

Dado que tenemos el genoma de un organismo, es posible encontrar ciertos patrones que no es posible encontrar en secuencias que no sean genomas. En la tabla 4 se muestran algunos de esos patrones:

Tabla 4. Patrones notables en genomas.

Secuencia	Función
Codón de inicio Metionina	ATG
Codones de paro	TAG, TAA, TGA
Caja Pribnow	TATA
Secuencia consenso -35	TTGACAT
Shine-Dalgarno	AGGAGGU
Kozak	ACCAUGG

Por otro lado, existen características físico-químicas en las secuencias que también pueden ser analizadas; entre otras, está la débil interacción electrostática de los enlaces de hidrógeno creados entre las cadenas que forman el DNA. La fuerza del enlace de esta unión es medida por la cantidad de energía liberada en la formación del enlace. El cambio de energía es expresado de la siguiente manera [87]:

$$\Delta G = -RT \ln K_{eq}$$

donde R una constante natural (8.314 JK^{-1} mol^{-1}), T es la temperatura y $\ln K_{eq}$ es el logaritmo natural del equilibrio constante entre la molécula con y sin enlace.

Otra característica que puede ser muy útil y que está relacionada con dicha energía, es el hecho de que cuando se localiza el sitio promotor del gen, la estructura de DNA es separada en sus dos hebras [90]; esta separación se lleva a cabo en una zona de poca energía (ΔG). Por lo tanto, identificar dichas zonas es importante para encontrar el promotor e inicio del gen [91].

Para hacer más evidentes dichas zonas, en esta tesis se creó la siguiente codificación:

Tabla 5.Codificación JK

Nucleótido	Código JK
A	J
C	K
T	J
G	K

en el capítulo de experimentación quedará más claro esto con un ejemplo.

4.7.2 Algoritmo de alineamiento

Como se ha visto con anterioridad, el alineamiento de genomas tiene tres pasos principalmente:

1. Se detectan regiones potencialmente homólogas (anclas).
2. Se filtran las anclas.
3. Se alinean las secuencias potencialmente homólogas.

El modelo que proponemos para el alineamiento de genomas sigue estos tres pasos.

4.7.2.1 Primera Fase

En esta parte se identifican las anclas que serán utilizadas en el alineamiento. Para ello es necesario definir lo que serán nuestras anclas, los genes virtuales. Un gen virtual es un bloque de secuencia que cumple con dos características, codón de inicio y codón de terminación en el mismo marco de lectura.

Entonces para cada genoma que se analiza se realiza lo siguiente:

1. Se recorre la secuencia identificando las posiciones de los codones de inicio y paro para cada uno de los marcos de lectura.

2. Para cada marco de lectura, se identifican los genes virtuales que se encuentren en la secuencia, partiendo de un codón de inicio y terminando en el siguiente codón de terminación que encuentre. El siguiente codón de inicio deberá empezar en una posición posterior al codón de terminación del gen virtual encontrado anteriormente.

3. Para localizar el gen virtual se recorre la secuencia en tres marcos de lectura distintos. El marco de lectura está determinado por la posición en la que se comienza e leer la secuencia. La lectura a partir de la posición uno es el marco de lectura uno, posición dos corresponde el marco de lectura dos y posición tres al marco de lectura tres.

Ejemplo:
Supongamos el segmento de secuencia: ATGGCATAAATGGCATAA.

Marco de lectura 1: ATG-GCA-TAA-ATG-GCA-TAA
Marco de lectura 2: TGG-CAT-AAA-TGG-CAT-AA
Marco de lectura 3: GGC-ATA-AAT-GGC-ATA-A
Marco de lectura 4: GCA-TAA-ATG-GCA-TAA

Como se ve en el ejemplo, el marco de lectura uno y cuatro se consideran los mismos ya que el segmento es dividido en los mismos elementos.

4.7.2.2 Segunda Fase

En esta parte se discrimina cuál de los genes virtuales serán considerados dentro del alineamiento y cuáles serán eliminados.

Para determinar si un gen permanece o si es eliminado del conjunto de genes virtuales, se propone lo siguiente para cada uno de los genomas que se analice:

1. Se eliminan genes virtuales de longitud menor a *tamanio_secuencia*.

2. De cada gen virtual que quede después de la primera fase, se obtienen los últimos sc_longitud nucleótidos. A dicho segmento lo llamaremos secuencia característica (SC).

En esta tesis se propone por primera vez el final de cada gen virtual como secuencia característica ya que sabemos que es una zona de poca variación en los organismos [91].

Se codifica el segmento extraído de la secuencia de nucleótidos en su correspondiente secuencia de aminoácidos y se construye una memoria heteroasociativa Alfa-Beta robusta tipo *Min* para cada genoma tomando como conjunto fundamental los segmentos de secuencias obtenidos en 2. La tabla 6 muestra el código genético que relaciona un tiplete o codón a un aminoácido.

Tabla 6.Código genético

		2ª Base			
		T	C	A	G
1ª BASE	T	TTT (Phe/F) Fenilalanina	TCT (Ser/S) Serina	TAT (Tyr/Y) Tirosina	TGT (Cys/C) Cisteína
		TTC (Phe/F) Fenilalanina	TCC (Ser/S) Serina	TAC (Tyr/Y) Tirosina	TGC (Cys/C) Cisteína
		TTA (LeT/L) Leucina	TCA (Ser/S) Serina	TAA Parada (Ocre)	TGA Parada (Ópalo)
		TTG (LeT/L) Leucina	TCG (Ser/S) Serina	TAG Parada (Ámbar)	TGG (Trp/W) Triptófano
	C	CTT (LeT/L) Leucina	CCT (Pro/P) Prolina	CAT (His/H) Histidina	CGT (Arg/R) Arginina
		CTC (LeT/L) Leucina	CCC (Pro/P) Prolina	CAC (His/H) Histidina	CGC (Arg/R) Arginina
		CTA (LeT/L) Leucina	CCA (Pro/P) Prolina	CAA (Gln/Q) Glutamina	CGA (Arg/R) Arginina
		CTG (LeT/L) Leucina	CCG (Pro/P) Prolina	CAG (Gln/Q) Glutamina	CGG (Arg/R) Arginina
	A	ATT (Ile/I) Isoleucina	ACT (Thr/T) Treonina	AAT (Asn/N) Asparagina	AGT (Ser/S) Serina
		ATC (Ile/I) Isoleucina	ACC (Thr/T) Treonina	AAC (Asn/N) Asparagina	AGC (Ser/S) Serina
		ATA (Ile/I) Isoleucina	ACA (Thr/T) Treonina	AAA (Lys/K) Lisina	AGA (Arg/R) Arginina
		ATG (Met/M) Metionina, Comienzo	ACG (Thr/T) Treonina	AAG (Lys/K) Lisina	AGG (Arg/R) Arginina
	G	GTT (Val/V) Valina	GCT (Ala/A) Alanina	GAT (Asp/D) Ácido aspártico	GGT (Gly/G) Glicina
		GTC (Val/V) Valina	GCC (Ala/A) Alanina	GAC (Asp/D) Ácido aspártico	GGC (Gly/G) Glicina
		GTA (Val/V) Valina	GCA (Ala/A) Alanina	GAA (GlT/E) Ácido glutámico	GGA (Gly/G) Glicina
		GTG (Val/V) Valina	GCG (Ala/A) Alanina	GAG (GlT/E) Ácido glutámico	GGG (Gly/G) Glicina

4.7.2.3 Tercera Fase

En esta fase se utiliza el modelo de memoria heteroasociativa Alfa-Beta robusta tipo *Min* para llevar a cabo la identificación de las secuencias en el genoma y poder alinearlo.

1. Para cada genoma, se presentan los segmentos utilizados para construir el conjunto fundamental del primer genoma a la memoria asociativa construida a partir del segundo genoma, con el fin de identificar cuáles son los genes virtuales que más se parecen.
2. Se selecciona la combinación de pares de genes virtuales que entregue la mayor puntuación.
3. Se analiza si es posible alinearlos en el orden en el que fueron encontrados en la secuencias con alguno de los algoritmos de alineamiento. Si no es así, se propone un nuevo alineamiento considerando re-ordenamientos.

La medida de similitud entre un gen virtual y otro será dado por la memoria heteroasociativa Alfa-Beta robusta al momento de presentarle los patrones del conjunto fundamental.

Para poder probar el modelo con secuencias de aminoacidos, es necesario establecer una relación entre los caracteres de los aminoacidos y secuencias binarias, ya que el modelo trabaja en este último dominio. La Tabla 7 muestra dicha relación. La representación binarya fue contruida usando la conocida matriz de substitución blosum62 [93]. La razón para ello es poder dar mayor información al modelo para llevar a cabo la recuperación de patrones en secuencias de aminoacido con una distancia evolutiva mayor.

Tabla 7 Tabla relacional

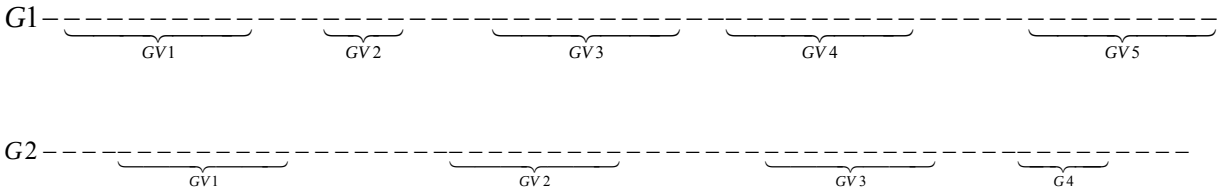
Carácter del aminoácido	Secuencia binaria de aprendizaje	Secuencia binaria de recuperación
F	100000010000000100000000	100000000000000000000000
S	011100000000001000000000	010000000000000000000000
T	011000000000000000000000	001000000000000000000000
N	010100000000000000111000	000100000000000000000000
K	000010100110000000000100	000010000000000000000000
*	000000000000000000000000	111111111111111111111111
E	000010100110000000101000	000000100000000000000000
Y	100000010000000100010000	000000010000000000000000
V	000000001001010000000010	000000001000000000000000
Z	000010100110000000101000	000000100000000000000000
Q	000010100110000000010100	000000000010000000000000
M	000000001001010000000010	000000000001000000000000
C	000000000000100000000000	000000000000100000000000
L	000000001001010000000010	000000000000010000000000
A	010000000000001000000000	000000000000001000000000
W	100000010000000100000000	000000000000000100000000
X	000000000000000000000000	011000000000001000000000
P	000000000000000001000000	000000000000000001000000
B	000100100100000000101000	000000000000000000101000

H	000100010010000000010000	00000000000000000010000
D	000100100100000000101000	00000000000000000001000
R	000010000010000000000100	00000000000000000000100
I	000000001001010000000010	00000000000000000000010
G	000000000000000000000001	00000000000000000000001

A continuación se muestran un diagrama que resume el algoritmo propuesto.

4.7.2.4 Diagrama - Primera Fase

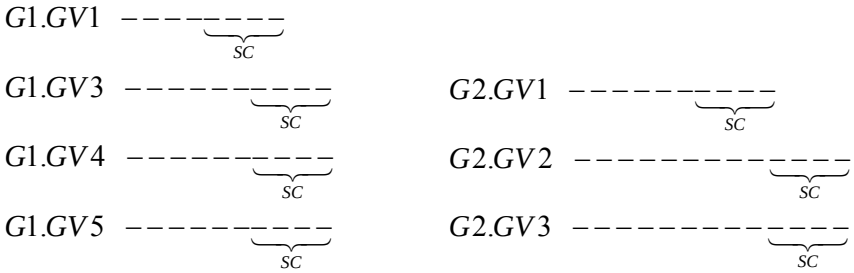
Recorrido y obtención de los genes virtuales



4.7.2.5 Diagrama - Segunda Fase

Se eliminan aquellos genes virtuales que tengan un número de secuencias menor al umbral *tamano_secuencia*.

Se extrae el segmento final de longitud sc_longitud de cada gen virtual.



Se construye una memoria heteroasociativa Alfa-Beta tipo *Min* para cada uno de los genomas utilizando los siguientes conjuntos fundamentales.

$$CFG1 = \left\{\left(G1.GV1.SC, y^1\right), \left(G1.GV3.SC, y^2\right), \left(G1.GV3.SC, y^3\right), \left(G1.GV3.SC, y^4\right)\right\}$$

$$CFG2 = \{(G2.GV1.SC, y^1), (G2.GV2.SC, y^2), (G2.GV3.SC, y^3)\}$$

4.7.2.6 Diagrama - Tercera Fase

Se le presenta a la memoria del genoma uno (G1) los patrones con los que se construyó la memoria del genoma dos (G2).

De acuerdo con el proceso de aprendizaje de la memoria heteroasociativa Alfa-Beta robusta tipo *Min*, cada patrón x^{ϖ} quedará aprendido en la fila ϖ de la memoria asociativa. Por lo que en la fase de recuperación, el vector y^{ϖ} que se obtiene al presentar un patrón x^{ϖ} a la memoria tendrá 1's en los índices en donde el valor tenga un máximo [92]. Debido a lo anterior, podemos decir que el gen virtual que ocupe la fila ϖ de la memoria asociativa será aquel que tenga una mayor similitud con x^{ϖ}.

De esta forma es como asociaremos los genes virtuales de un genoma con los genes virtuales del otro genoma. El alineamiento, a diferencia de lo que sucede en el alineamiento múltiple o de pares de secuencias, no deberá estar estrictamente en el orden en el que aparecen los genes en el genoma, debido al fenómeno de re-ordenamiento que se presenta en los genomas de los organismos. Es por eso que antes de aplicar algún algoritmo de alineamiento, de los que se han venido utilizando, la propuesta es organizar aquellos genes que tengan mayor similitud, para posteriormente alinearlos utilizando algún método de alineamiento ya conocido, como MUSLE, BLAST, FASTA, T-COFFEE, entre otros.

5 Resultados

5.1 Modelo basado en palíndromos

Dado que el algoritmo propuesto no ocupa, por el momento, ningún método matemático para el alineamiento de biomoléculas y por lo tanto no usa matrices de substitución como PAM o BLOSUM, no es posible determinar una puntuación para el alineamiento generado que sea comparable con algún otro método. Por otro lado, el algoritmo tampoco está preparado para lidiar con inserciones y borrados por lo que no es posible probar con las bases de datos de pruebas que se mencionaron en secciones anteriores.

Es por eso que, para tener una idea del desempeño que se está teniendo en el alineamiento, en esta ocasión se han comparado lo alineamientos hechos por el algoritmo presentado en esta tesis, con los mostrados por uno de los principales algoritmos de alineamiento de secuencia clustalW.

Las secuencias que se utilizaron como objetos de prueba son seis pares de secuencias recopiladas por nosotros. Los primeros tres correspondiente al virus H1N1 tipo A Estacionario y los tres restantes al virus H1N1/09. La tabla 7 muestra la información de cada archivo. El nombre de los archivos identifica las dos secuencias que contiene.

Tabla 8. Archivos de datos

Nombre del Archivo	Tipo	Número de secuencias
AB434099-AB434103E	Estacionario	2
AF117241-AB434099E	Estacionario	2
AF117241-AB434103E	Estacionario	2
CY039527-CY039893S	H1N1/09	2
CY039527-CY039901S	H1N1/09	2
CY039893-CY039901S	H1N1/09	2

Para poder comparar el alineamiento realizado con el algoritmo presentado en esta tesis y el realizado por clustalW utilizamos la medida tomada del artículo original [62], el número de identidades en la secuencia sobre el número de residuos comparados. La tabla 8 muestra el porcentaje de identidad asignado a cada alineamiento de pares de secuencias.

Tabla 9. Comparación entre clustalW vs Propuesta

Nombre del Archivo	ClustalW %	Propuesta %
AB434099-AB434103E	99	99
AF117241-AB434099E	85	41
AF117241-AB434103E	85	41
CY039527-CY039893S	99	98
CY039527-CY039901S	99	98
CY039893-CY039901S	99	99

Como se puede ver en la tabla 8 la secuencia AB434099-AB434103E tiene la misma puntuación. Las secuencias AF117241-AB434099E, AF117241-AB434103E están a la mitad por debajo del alineamiento presentado por clustalW. Por último, CY039527-CY039893S, CY039527-CY039901S, CY039893-CY039901S; aquí, la puntuación de nuestro algoritmo está por debajo en sólo uno por ciento del presentado clustalW.

Como se puede ver, cuatro de los seis resultados mostrados por nuestra propuesta, están cerca de los alineamientos presentados por uno de los mejores métodos de alineamiento.

De los alineamientos que tienen una puntuación lejana a la mostrada por clustalW, suponemos que es porque son secuencias que presentan, además de mutaciones, inserciones o borrados de nucleótidos. Y dado que el método propuesto todavía no lidia con inserciones y borrados, el alineamiento no se acerca al de clustalW.

Para solucionar este problema, se llevará a cabo un alineamiento secundario progresivo tal y como se indica en el algoritmo propuesto.

Por otro lado, para hacer una comparación en cuanto a tiempo de respuesta de los sistemas, en la tabla 9 se muestra una comparación con otro de los principales algoritmos, T-Coffe. El tiempo tomado en el caso del algoritmo propuesto, se considera desde la extracción del conjunto característico hasta que termina el alineamiento, aún no se ha considerado el caso en el que la biblioteca haya sido construida.

Tabla 10. Comparación entre T-Coffe vs Propuesta

Nombre del Archivo	T-Coffe %	T-Coffe secs	Propuesta %	Propuesta Secs
AB434099-AB434103E	100	2.10	99	0.46
AF117241-AB434099E	99	2.12	41	0.61
AF117241-AB434103E	99	2.21	41	0.46
CY039527-CY039893S	99	3.59	98	0.46
CY039527-CY039901S	99	2.10	98	0.62
CY039893-CY039901S	99	2.72	99	0.62

En la tabla 9, es evidente que el tiempo de respuesta del algoritmo propuesto, al momento de calcular el alineamiento, es menor con respecto al de T-Coffe. Por otro lado, el porcentaje de identidad que presenta el algoritmo propuesto permanece cercano al de T-Coffe.

5.2 Resultados del modelo asociativo para el alineamiento de biomoléculas.

En la presente sección mostramos los resultados del modelo asociativo para el alineamiento de biomoléculas.

Ya que no existe una metodología para la evaluación del alineamiento de genomas, se creará una base de datos con genomas conocidos, los cuales ya han sido etiquetados con nombre y función por expertos en el área de biología molecular.

La tabla 10 presenta los resultados mostrados por el archivo de salida.

El modelo necesita de dos parámetros, el tamaño mínimo de una proteína, con lo cual eliminamos aquellas proteínas de longitud menor a éste, y la longitud de la secuencia característica.

De los resultados presentados aquí hemos podido inferir los siguientes hechos:

1. Es posible eliminar ciertos codones de inicio que se traslapen con otros genes virtuales, reduciendo así la longitud del gen.
2. Es posible determinar las posiciones tentativas de los promotores dentro de un genoma.
3. Es posible identificar genes policistrónicos, los cuales son genes que comparten un promotor, y por ello varios genes son convertidos en proteínas en un mismo tiempo.

Tabla 11. Identificación de genes virtuales.

>gi|239813019|ref|NC_012791.1| Variovorax paradoxus S110 chromosome 1, complete genom					
Inicio(s)	Final	Longitud más larga	Metionina	Inicio del Promotor	ORF
457,475,490,556,808,913,9(	1414	957	9	160	1
1457,	2732	1275	1	1160	2
1506,1752,1947,2118,2157,	2610	1104	10	1209	3
2747,2912,3089,3236,3347,	5369	2622	20	2450	2
5659,5674,5758,5818,5851,	6643	984	11	5362	1
7335,7344,7362,7407,7446,	7845	510	15	7038	3
7977,8022,8094,8115,8151,	8463	486	7	7680	3
8257,8329,8989,9358,	10036	1779	4	7960	1

La columna A muestra las posiciones en donde se encontró el codón que indica el inicio de la codificación (AUG), la columna B indica la posición en la que se encuentra alguno de los codones de paro (TAG, TAA, TGA), la columna C muestra la longitud desde el primero codón de inicio hasta el codón de paro, y la última columna, D, nos indica el marco de lectura en el que se encuentra el gen virtual.
Por otro lado, se desarrolló un algoritmo para llevar a cabo un análisis del contenido de la secuencia, a fin de encontrar ciertas características que nos ayuden a determinar el comienzo y el final de un gen. Lo pasos para el análisis son los siguientes:

1. Se indica la ventana del segmento a ser analizado.
2. Se lee el archivo que contiene el genoma del organismo.

3. Se recorre todo el genoma segmentándolo de acuerdo al tamaño dado en el paso 1.
4. Se calcula ΔG
5. Se obtiene la codificación JK.
6. Se va calculando la frecuencia de cada segmento en el genoma.
7. Se escribe un reporte final en un archivo de salida.

El valor ΔG indica la fuerza que liberará un bloque de secuencia de DNA al ser separado de su cadena complementaria por una fuerza externa. La codificación JK, que se muestra en la tabla 5, es un código propuesto en esta tesis para poder hacer evidente la relación que hay entre la composición del segmento y su ΔG .

Por último se calcula la frecuencia de un determinado segmento de DNA con el fin de identificar si hay una relación entre la aparición de ciertos segmentos de DNA y la localización de los promotores e inicio del gen.

La tabla 11 nos muestra un ejemplo de la salida del algoritmo con un tamaño de segmento de 6.

Tabla 12. Análisis por hexanucleotido.

Hexanucleotido	CodigoJK	Frecuencia	DeltaG	Posición
AACCTT	JJKKJJ	301	9.8	7337,30957,32629,61532,76298,99258,152640,161616,
GCTATC	KKJJJK	353	8.7	8535,57924,58014,170241,171114,177011,178827,197
AAGGCT	JJKKKJ	851	11.3	2858,10058,15461,16288,23453,26591,27724,27747,28
GCCTCT	KKKJKJ	864	11	61,6716,6771,8005,24713,32788,52111,54992,60720,7
AAGATC	JJKJJK	1846	8.2	228,1035,1173,1338,3265,4021,4036,4261,6159,7238,1
TCCACC	JKKJKK	1870	11	12918,18106,21896,22868,27815,29194,29723,32477,3
AAAAGC	JJJJKK	813	10.4	27825,43665,44570,49250,49644,50219,52761,54323,5
AGGGAA	JKKKJJ	395	11.3	5622,48152,53988,76283,82942,87058,182013,198037,
ACACCC	JKJKKK	692	10.7	427,9847,13084,16137,32337,54920,55473,74278,8075

Una vez que se tienen separados los genes virtuales, en ambos genomas, se construye la memoria heteroasociativa Alfa-Beta robusta tipo Min propuesta en este trabajo utilizando para la codificación de las secuencias de aminoácido la Tabla 7. Para las pruebas se construye el conjunto fundamental de acuerdo con lo establecido en el modelo, utilizando como patrones de aprendizaje las secuencias características obtenidas de uno de los genomas. Para llevar a cabo las pruebas se toman los elementos del segundo genoma como patrones de prueba y se codifican utilizando, también, la Tabla 7. Además, el modelo requiere que η y q sean dadas. Se asignan los siguientes valores para $\eta = 5$, $q = 1$. Dado que no se cuenta con dos genomas para hacer las pruebas, los archivos que contienen las genes virtuales fueron procesados para generar un conjunto de prueba que permitiera evidenciar las caracteristicas del modelo de memoria asociativa propuesto en este trabajo: la recuperación completa del conjunto fundamental y la nueva caracteristica para soportar inserciones, mutaciones y borrados en los patrones.

Se construyeron dos conjuntos de bases de datos de pruebas: el primer conjunto se utilizará ára mostrar que el nuevo modelo mantiene su capacidad de recuperación completa del conjunto findamental. Para ello en cada archivo se colocaron un número distinto de genes virtuales y se entrenó y probó con el mismo conjunto de información. La Tabla 13 muestra los resultados de las pruebas.

Tabla 13 Recuperación completa del conjunto fundamental

Nombre de archivos	*p*	% Recuperación
p50.txt	50	100
p100.txt	100	100
p150.txt	150	100
p200.txt	200	100

También es importante conocer como se conporta el algoritmo cuando se le presentan patronos desconocidos (diferentes de los del conjunto fundamental). Tomando como base el archivo p50.txt, se construyeron seis versiones del archivo con secuencias cambiadas aleatoriamente. Los resultados se muestran en la Tabla 14.

Tabla 14 Recuperación de patrones alterados

Archivo de datos	*p*	% Recuperación
P70B50M50.txt	50	2
P70M50I50.txt	50	0
P70M100.txt	50	100
P90M100.txt	50	100
P90B50M50.txt	50	94
P90M50I50.txt	50	94

Las alteraciones que se hicieron en los archivos fueron de los siguientes tipos:

1. Alteración : Indica el porcentaje de cambio de una secuencia. Los cambios pueden ser mutación, inserción y borrado
2. Mutación: porcentaje de substitución de un aminoacido en otro
3. Inserción: porcentaje de inserción de un aminoacido en una secuencia
4. Borrado: porcentaje de borrado de aminoacidos en una secuencias

La Tabla 15 muestran los archivos generados y sus alteraciones.

Tabla 15 Porcentaje de alteraciones por archivo

Nombre de archivo	% de alteración	% Mutación	% Borrado	% Inserción
P70B50M50.txt	30	50	50	0
P70M50I50.txt	30	50	0	50
P90B50M50.txt	10	50	50	0
P90M50I50.txt	10	50	0	50

6 Conclusiones y Trabajo a futuro

En este capítulo se presentan las conclusiones que se desprenden de este tema de tesis; además, se hacen propuestas sobre posibles trabajos a futuro, basados en las aportaciones de este trabajo.

6.1 Conclusiones

1. Se desarrollaron dos propuestas para el alineamiento de biomoléculas, la primera alinea genes codificantes y la segunda genomas.
2. Se llevó a cabo el primer bosquejo del alineamiento de pares de secuencias, logrando alinear secuencias evolutivamente cercanas.
3. Es posible ubicar segmentos de secuencias donde se lleva a cabo inserciones y borrados.
4. Nos dimos cuenta de que existen k-tuplas pertenecientes a los conjuntos característicos de cada secuencia que difieren en pocos nucleótidos entre ellos.
5. Se propone una nueva forma de extraer bloques de una secuencia en lo que se refiere a los métodos k-tuplas
6. El algoritmo para encontrar las k-tuplas reduce el tiempo, en comparación de como se hacía normalmente.
7. Al tener menor tiempo de procesamiento en el alineamiento, es posible aplicar dicho algoritmo en problemas de cómo predicción de estructuras proteínicas, construcción de árboles filogenéticos, alineamiento de genomas entre otros.
8. Es posible identificar genes policistrónicos.
9. Es posible determinar las posiciones tentativas de los promotores dentro de un genoma.
10. Se propone el final de cada gen virtual como secuencias características.
11. Se propone un nuevo modelo de memoria heteroasociativa que es robusta a **inserciones**, **mutaciones** y **borrados** en los patrones.
12. Por lo anterior ahora es posible construir memorias asociativas con patrones de distintas dimensiones.
13. El nuevo modelo de memoria heteroasociativa mantiene la capacidad de recuperación completa del conjunto fundamental.
14. Se propuso un nuevo mapeo de secuencias binarias a aminoácidos basado en la matriz de blosum62 la cual mantiene la probabilidad de cambio de un aminoácido a otro.

6.2 Trabajo a futuro

1. Extender el algoritmo para trabajar con alineamientos múltiples de secuencias [34][63][65].
2. Incluir una experimentación más extensa, utilizando las bases de datos de pruebas más importantes [88][89].

3. Ampliar el algoritmo no sólo para alinear secuencias codificantes, sino para el alineamiento de genomas completos. Como primera aproximación en organismos procariontes [37][38].
4. Una propuesta para reducir el tiempo del algoritmo sería crear una biblioteca que contenga cada secuencia asociada con su respectivo rasgo característico, con lo que podríamos disminuir tiempo de alineamiento [63].
5. Se completará el modelo asociativo para el alineamiento de genomas, con el alineamiento de secuencias [48].
6. Se creará una base de datos de genomas para la realización de los experimentos.
7. Se analizara la complejidad del algoritmo [82][83].
8. Se harán comparaciones con el estado del arte en el alineamiento de genomas [39][40][41][42].

Referencias

[1] Lesk, Arthur M.: *Introduction to Bioinformatics*, Oxford University Press, Second Edition, 2005.

[2] Baldi, P., and Brunak, S.: *"Bioinformatics: the machine learning approach"* MIT Press, Cambridge, MA 1998.

[3] Altman, R.B., Valencia, A., Miyano, S. and Ranganathan, S.: *Challenges for intelligent system in biology*, IEEE Intell. Syst, 2001, 16,(6), pp 14-20.

[4] Jacques Cohen. *Bioinformatics—An Introduction for Computer Scientists*. ACM Computing Surveys, Vol. 36, No. 2, pp. 122–158, 2004

[5] Nash, H., Blair, D., and Grefenstette, J.: *Comparing algorithms for large scale sequence analysis*. Proc. 2nd IEEE Int. Symp. on Bioinformatics and Bioengineering. 2001.

[6] Needleman, S.B., and Wunsch, C.D: *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, J. Mol. Biol. 48, pp. 443-453, 1970.

[7] Smith, T.F., and Waterman, M.S.*: Identification of common molecular sequence*, J. Mol. Biol., 147, pp. 195-197, 1981

[8] Gautheret, D., Major, F. and Cedergren, R.: *Pattern searching/ alignment with RNA primary and secondary structure: an effective descriptor for tRNA*, Comp. Appl. Biosci. 6, pp. 325-331, 1990.

[9] Fickett, J.W.: *'Finding genes by computer: the state of the art'*, Trends Genet., 12, (8), pp. 316–320, 1996.

[10] Mitra, S. Hayashi, Y. *Bioinformatics with soft computing*, Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, pp. 616-635, 2006.

[11] Chou, P., and Fasmann, G.: *'Prediction of the secondary structure of proteins from their amino acid sequence'*, Adv. Enzym., 47, pp. 145–148, 1978.

[12] Luscombe, N.M., Greenbaum, D., and Gerstein, M.: *'What is bioinformatics?* A *proposed definition and overview of the field'*, Methods Informat. Med., 40, (4), pp. 346–358, 2001.

[13] Edward Keedwell and Ajit Narayanan.: *"Intelligent Bioinformatics"*, John Wiley & Sons, 2005.

[14] David W. Mount .:*"Bioinformatics: sequence and Genome Analysis"*, Cold Spring Harbor Laboratory Press. Second Edition 2004.

[15] Yonatan Bilu, Pankaj K. Agarwal, and Rachel Kolodny: "*Faster Algorithms for Optimal Multiple Sequence Alignment Based on Pairwise Comparisons"*. IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, pp. 408-422. Dicember, 2006

[16] Nur'Aini Abdul Rashid, Rosni Abdullah, Abdullah Zawawi Haji Talib, Zalila Ali.: *"Fast Dynamic Programming Based Sequence Alignment Algorithm"*. The 2nd International Conference on Distributed Frameworks for Multimedia Applications, IEEE, 2006.

[17] Triztan Cazenave.: *"Overestimation for Multiple Sequence Alignment"*, Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology CIBCB 2007, IEEE, 2007.

[18] Weiguo Liu, Bertil Schmidt, Gerrit Voss, and Wolfgang Muller-Wittig.: *"Streaming Algorithms for Biological Sequence Alignment on GPUs"*. IEEE Transactions On Parallel And distributed Systems. Vol. 8 No.9 pp. 1270-1281. September 2007

[19] Abdesslem Layeb, Abdel Hakim Deneche.: *"Multiple Sequence Alignment by Immune Artificial System*". International Conference on Computer Systems and Applications. AICCSA '07. IEEE/ACS, 2007.

[20] Yun-Sheng Chung, Chin Lung Lu, ChuanYi Tanga.: *"Constrained sequence alignment: A general model and the hardness results"*. Discrete Applied Mathematics. pp. 2471-2486. 2007.

[21] Abdullah N. Arslan and Dan He.: *"An improved algorithm for the regular expression constrained multiple sequence alignment problems"*. Sixth IEEE Symposium on BionInformatics and BioEngineering BIBE'06, 2006.

[22] Fitch W.M and Smith T.F.: "*Optimal Sequence alignments"*. Proc. Natl. Acad. Sci. 80: 1382-1386, 1983.

[23] Auer, J.: *"Multiple sequence alignment by itereted local search"*. Institutionen för kommunikation och information, 2004

[24] Gibbs A.J. and McIntyre G.A.: "*The diagram, a method for comparing sequences"*. It is used with amino acid and nucleic sequences. Eur. J. Biochem. 16: 1-11, 1970.

[25] Wilbur W.J. and Limpan D.J.: "*Rapid similarity search of nucleic acid and protein data banks"*. Proc. Natl. Acad. Sci. 80: 726-730. 1983.

[26] Wilbur W.J.: *"On the PAM model protein evolution"*. Mol. Biol. Evol. 2: 434-447. 1985.

[27] Waterman M. and Perlwitz M.D.: "*Line geometries for sequence comparison"*. Bull. Math. Biol. 46: 567-577. 1984.

[28] Feng D.F. and Doolitle R.F.: "*Progressive sequence alignment as a prerequisite to correct phylogenetic trees"*. J. Mol. Evol. 25: 351-360. 1987.

[29] Saitou N and Nei M.: "*The neighbor-joining method: a new method for reconstructing phylogenetic trees"*. Mol. Biol. Evol. 4: 406-435. 1987.

[30] Higgins D.G. and Sharp P.M.: "*CLUSTAL: A package for performing multiple sequence alignment on a microcomputer"*. Gene 73: 237-244. 1988.

[31] Higgins D.G., Thomson J.D; and Gibson T.J.: "*Using CLUSTAL for multiple sequence alignment"*. Methods Enzymol. 266: 383-402. 1996.

[32] Pascarella S. and Argos P.: "*Analysis of insertions/deletions in protein sequences"*. J. Mol. Biol. 224: 461-471. 1992.

[33] Sneath P.H.A. and Sokal R.R.: "*Numerical Taxonimy"*. W.H.Freeman, San Francisco, California. 1973.

[34] Notredame C. and Higgins D.G.: "*SAGA: Sequence alignment by genetic algorithm"*. Nucleic Acid Res. 24: 1515-1524. 1996.

[35] Notredame C., O´Brien E.A. and Higgins D.G.: "*RAGA: RNA sequence alignment by genetic algorithms"*. Nucleic Acid Res. 25:4570-4580. 1997.

[36] Baldi P., Chauvin Y., Hunkapillar T. and McClure M.A.: "*Hidden Markov models of biological primary sequence information"*. Proc. Natl. Acad. Sci. 91: 1059-1063. 1994.

[37] Lemaitre, C. and Sagot, M.: "*A small trip in the untranquil world of gnomes, a survey on the detection and analysis of genome rearrangment breakpoints"*. Theoretical computer science, 395, pp 171-192, 2008.

[38] P. Pevzner, G. Tesler.: "*Genome rearragment in mamalian evolution: Lesson from human and mouse genomes"*, Genome Res. 13(1), pp. 37-45 2003.

[39] W.J. Kent, R, Baertsch, A. Jinrichs, W. Miller, D. Haussler.: "*Evolution's cauldron, deletion, and rearrangement in the mouse and human genomes"*, Proc. Natl. Acad. Sci. USA 100 (20), 11484-11489. 2003.

[40] O. Couronne, A. Poliakov, N. Bray, T. Ishkhanov, D. Ryaboy, E. Rubin, L. Patcher, I Dubchak.: "*Strategies and tools for whole-genomes alignments"*, Genome Res. 13 (1), pp. 73-80. 2003

[41] A.C.E. Darling, B.Mau, F.R. Blattner, N.T. Perna: "*Mauve:Multiple alignment of conserved genomic sequence with rearrangement"*, Genome Res. 14(7) 1394-1403. 2004.

[42] S. Schwartz, E. Zlotorynsky, W.J. Kent, A. Smit, Z. Zhang, R Baertsch, R.C. Haussler, W. Miller.: "*Human-mouse alignment with Blastz"*. Genome Res 13(1), pp. 103-107. 2003.

[43] Hassoun, M. H.: "*Associative Neural Memories"*, Oxford University Press, New York. 1993.

[44] Kohonen, T.: "*Self-Organization and Associative Memory"*, Springer-Verlag, Berlin. 1989.

[45] Yánez, C.: "*Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios"*, Tesis de Doctorado, Centro de Investigación en Computación, México. 2002.

[46] Acevedo-Mosqueda, M.E.: "*Memorias Asociativas Bidireccionales Alfa-Beta"*, Tesis de Doctorado en Ciencias de la Computación, Centro de Investigación en Computación, México. 2006.

[47] Simpson, P. K.: "*Artificial Neural Systems"*, Pergamon Press , New York. 1990.

[48] Román-Godínez, I., Yáñez-Márquez, C., López-Yáñez, I.: *"Classifying Patterns in Bioinformatics Databases by Using Alpha-Beta Associative Memories".* Biomedical Data and Application. Studies in Computational Intelligence , Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-02192-3. 2009.

[49] Steinbuch, K.: *"Die Lernmatrix"*, *Kybernetik*, vol. 1, no. 1, pp. 36-45. 1961.

[50] Steinbuch, K. & Frank, H.: *"Nichtdigitale Lernmatrizen als Perzeptoren"*, *Kybernetik*, vol. 1, no.3, pp. 117-124. 1961.

[51] Kohonen, T.: *"Correlation matrix memories"*, *IEEE Transactions on Computers*, *C-21*, vol. 4, pp. 353-359. 1972.

[52] Willshaw, D., Buneman, O. and Longuet-Higgins, H.: *"Non-holographic associative memory"*, *Nature*, no. 222, pp. 960-962. 1969.

[53] Anderson, J. A.: *"A simple neural network generating an interactive memory"*, *Mathematical Biosciences*, vol. 14, pp. 197-220. 1972.

[54] Anderson, J. A. and Rosenfeld, E.: "*Neurocomputing: Fundations of Research"*, MIT Press ,Cambridge. 1990.

[55] Hopfield, J.J.: *"Neural networks and physical systems with emergent collective computational abilities"*, *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554-2558. 1982.

[56] Abu-Mostafa, Y. and St. Jacques, J.: *"Information capacity of the Hopfield model"*, *IEEE Transactions on Information Theory*, IT-31, no. 4, pp. 461-464. 1985.

[57] Austin, J.: *"ADAM: A Distributed Associative Memory for Scene Analysis"*, In *Proceedings of First International Conference on Neural Networks*, pp. 285-295. 1987.

[58] Ritter, G. X., Sussner, P. and Diaz-de-Leon, J. L.: *"Morphological associative memories"*, *IEEE Transactions on Neural Networks*, vol. 9, pp. 281-293. 1998.

[59] Yánez-Márquez, C. and Díaz de León Santiago, J.L.: "*Memorias Morfológicas Autoasociativas"*, IT-58, Serie Verde, ISBN 970-18-6698-3, CIC-IPN, México. 2001.

[60] Díaz de León Santiago, J.L. and Yáñez Márquez, C.: *Memorias Morfológicas Heteroasociativas*, IT-57, Serie Verde, ISBN 970-18-6697-5, CIC-IPN, México. 2001.

[61] Notredame, C.: "*Recent progress in multiple sequence alignment: a survey"*. Pharmacogenomics, 3, 131-144. 2002.

[62] Thompson, J. D., Higgins, D. G. and Gibson, T. J.: "*CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting position-specific gap penalties and weight matrix choice"*. Nucleic Acids Research, 22 rentice-Hall, (22), 4673--80. 1994.

[63] Edgar, R.C.: "*MUSCLE: A multiple sequence alignment method with reduced time and space complexity"*. BMC Bioinformatics 5: 113. 2004.

[64] Lassmann T, Sonnhammer EL: "*Kalign, Kalignvu and Mumsa: Web servers for multiple sequence alignment"*. Nucleic Acids Res 34: W596– W599. 2006.

[65] Notredame C., Higgins D.G., Heringa, J: "*T-Coffee: A novel method for fast and accurate multiple sequence alignment"*. J Mol Biol 302: 205–217. 2000.

[66] Wallace IM, O'Sullivan O, Higgins DG, Notredame C.: "*M-Coffee: Combining multiple sequence alignment methods with T-Coffee"*. Nucleic Acids Res 34: 1692–1699. 2006.

[67] Pei J, Sadreyev R, Grishin N.V.: "*PCMA: Fast and accurate multiple sequence alignment based on profile consistency"*. Bioinformatics 19: 427– 428. 2003

[68] Do C.B., Mahabhashyam M.S., Brudno M, Batzoglou S.: "*ProbCons: Probabilistic consistency-based multiple sequence alignment"*. Genome Res 15: 330–340. 2005.

[69] Pei J, Grishin N.V.: "*MUMMALS: Multiple sequence alignment improved by using hidden Markov models with local structural information"*. Nucleic Acids Res 34: 4364–4374. 2006.

[70] Katoh K, Kuma K, Toh H, Miyata T.: "*MAFFT version 5: Improvement in accuracy of multiple sequence alignment"*. Nucleic Acids Res 33: 511–518. 2005.

[71] Armougom F, Moretti S, Poirot O, Audic S, Dumas P, et al.: "*Expresso: Automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee"*. Nucleic Acids Res 34: W604–W608. 2006.

[72] Notredame, C.: "*Recent Evolution of multiple sequence alignment algorithms"*. LoS Comput Biol 3(8): e123.doi:10.1371/journal.pcbi.0030123. 2007.

[73] Pei J, Grishin N.V.: "*MUMMALS: Multiple sequence alignment improved by using hidden Markov models with local structural information"*. Nucleic Acids Res 34: 4364–4374. 2006.

[74] Simossis VA, Heringa J.: "*PRALINE: A multiple sequence alignment toolbox that integrates homology-extended and secondary structure information"*. Nucleic Acids Res 33: W289–W294. 2005.

[75] Zhou H, Zhou Y.: "*SPEM: Improving multiple sequence alignment with sequence profiles and predicted secondary structures"*. Bioinformatics 21: 3615–3621. 2005.

[76] Chen, Y, et al.: "*Multiple sequence alignment based on genetic algorithms with reserve selection"*. Proceedings of 2008 IEEE International Conference on Networking, sensing and Control, vols 1 and 2 , 1511-1516 . 2008.

[77] Brizuela, CA., Luhrs-Olmos, E.: "*A team of genetic algorithms for the multiple sequence alignment problem*: *Preliminary results*". IEE International Conference on Systems, Man and Cybernetics, vols 1-8, 2767-2772, 2007.

[78] Uren, PJ., Cameron-Jones, RM., Sale, A.H.J.: "*MAUSA: Using simulated annealing for guide tree construction in multiple sequence alignmen"t*. AI 2007: Advances in Artificial Intelligence, proceedings. v. 4830 . 599-608. 2007.

[79] Huo, HW., Stojkovic, V.: "*A simulated annealing algorithm for multiple sequence alignment with guaranteed accuracy"*. ICNC 2007: Third International Conference on Natural Computation, vol 2, proceedings. 270-274 . 2007.

[80] Guinand, F., Pigne, Y.: "*An ant-based model for multiple sequence alignment"*. Large-Scale Scientific Computing. Lecture Notes in Computer Science. 4818. 553-560. 2007.

[81] Chen, L., Liu, W., Chen, J.: "*Ant colony optimization method for multiple sequence alignment"*. Proceedings of 2007 International Conference on Machine Learning and Cybernetics, vols 1-7. 914-919. 2007.

[82] Ingo Wegener.: "*Complexity theory: exploring the limits of efficienta algorithms"*. Springer-Verlag Berlin Heidelberg. 2005.

[83] Herber S. Wilf.: "*Algorithms and Complexity"*. Internet edition, summer, 1994.

[84] Ohno S.: *"Intrinsic evolution of proteins. The role of peptidic palindromes". Riv. Biol.* 83 (2-3): 287–91, 405–10. PMID 2128128. 1990.

[85] Giel-Pietraszuk M, Hoffmann M, Dolecka S, Rychlewski J, Barciszewski J.: "*Palindromes in proteins"*. J Protein Chem. 22(2):109-13. Entrez Pubmed 12760415. 2003.

[86] Gareth J. Warren and Robert L. Green.: *"Comparison of Physical and Genetic Properties of Palindromic DNA Sequences"*. Journal of Bacteriology, pp. 1103-1111. 1985.

[87] Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, Peter Walter. *Molecular Biology of the Cell*. Fifth Edition. 2008.

[88] National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov/. 2010

[89] UniProt Knowledgebase. http://expasy.org/sprot/. 2010.

[90] deHaseth, L, P. and Helmann, J.: "*Open Complex Formation by Escherichia Coli RNA polymerasa: the mechanism of polymerase-induce strand separation of double helical" DNA*. Molecular Microbiology. pp. 817-824. 1995.

[91] Garibay-Orijel. Comunicación personal. 2010.

[92] Román-Godínez, I., Yáñez-Márquez, C.: "*Complete Recall on Alpha-Beta Heteroassociative Memory"*. In: Gelbukh, A., Kuri Morales, A.F. (eds.) MICAI 2007.LNCS, vol. 4827, pp. 193–202. Springer, Heidelberg. 2007.

[93] S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89(22):10915–-10919,, 1992.

ANEXO A

En este anexo se describe el algoritmo llamado T-COFFEE, el cual ha sido utilizado en el presente trabajo para realizar comparaciones con la propuesta presentada.

A.1 Función Objetivo Coherente Basada en Árboles para la Evaluación de Alineamiento (T-COFFEE).

El algoritmo T-COFFE (*Tree-Based Consistency Objective Function for alignment evaluation*, por sus siglas en inglés) [65] tiene dos características importantes, da la posibilidad de generar, de manera flexible y simple, alineamientos múltiples de secuencias, a través de una fuente de datos heterogenia conocida como **librería de alineamiento de pares de secuencias**; dicha librería fue generada a través de una mezcla de alineamientos globales y locales de pares de secuencias [65].

La segunda característica importante es el método de optimización utilizado para encontrar el alineamiento múltiple que mejor represente los alineamientos de pares de secuencias en la librería. Para ello utiliza una estrategia progresiva similar a la utilizada en clustalW, en donde las secuencias que van a ser alineadas son separadas en pares, juntando aquellas que sean más idénticas de acuerdo a un métrica de distancia de secuencias [65].

Librería de alineamiento de pares de secuencias.

Los algoritmos basados en programación dinámica, ocupan ciertos parámetros que definen el resultado del alineamiento, dichos parámetros son: matriz de substitución y penalización por gap. El primero es obtenido a partir de un conjunto de secuencias alineadas manualmente y calculando la probabilidad de que un nucleótido o proteína *x* cambie por otro nucleótido o proteína *y*; el segundo es una variable seleccionada por la experiencia de la persona que solicita el alineamiento [65].

El cálculo de la librería de alineamiento es, precisamente, un intento por obtener automáticamente, y para un conjunto dado de secuencias, dicha matriz de substitución.

La librería consta de dos fases:

1. Librería primaria:
    a. Se crea los N(N-1)/2 posibles alineamientos globales de las N secuencias a alinear, usando clustalW con parámetros por default.
    b. Se crea los N(N-1)/2 posibles alineamientos locales de las N secuencias a alinear, usando clustalW con parámetros por default.
    c. Para cada par de secuencias en cada conjunto de alineamientos, tanto global como local, se asigna un peso, el cual representa que tan similares son las secuencias.
    d. Se combinan las librerías, local y global, en un solo conjunto. Si un par es duplicado en ambas librerías, entonces se mezclan en uno solo, siendo el peso de este último la suma de los pesos de ambos pares de alineamientos.

2. Extensión de librería: La idea de esta fase es combinar la información de tal forma que los pesos finales para cada par de residuos, refleje la información contenida en toda la librería.
    a. Se toma cada par de residuos alineados de la librería y se checa con cada uno de los residuos de las secuencias restantes para ver si concuerdan.
    b. Si se encuentran una secuencia que pueda ser una un punto intermedio entre cualquier par de secuencias, entonces se agrega en el alineamiento. Este nuevo alineamiento constara de tres secuencias y su peso será la puntuación minima entre el alineamiento de la primera con la intermedia y la intermedia con la segunda.

De esta forma se tiene ahora una librería extendida. Ambas librerías están representadas como una lista de alineamientos de residuos que concuerdan, por ejemplo, el residuo *x* de la secuencia A esta alineado con el residuo *y* de la secuencia B. Siendo esta lista la utilizada como matriz de substitución [65]. El peso que se le asigna a los tripletes que se obtuvieron de extender la librería primaria se calcula con la suma del peso del alineamiento de la primera secuencia con la secuencia intermedia, con el alineamiento de la secuencia intermedia con la segunda secuencia[65].

El siguiente paso es el alineamiento progresivo de las secuencias. Para ello se utilizan los pesos asignados a cada alineamiento almacenado en la librería para generar un árbol filogenético, utilizando la metodología neighbor-joining, que dicte el orden en el que serán alineadas las secuencias. Las secuencias más cercanas en el árbol serán alineadas usando un algoritmo basado en programación dinámica, donde la matriz de substitución es remplazada por los valores obtenidos en la librería extendida. Una vez que se terminan de alinear las secuencias más cercanas, se alinean los resultado se dichos alineamientos, sucesivamente, hasta tener el alineamiento completo[65].

Cabe mencionar que este último proceso no requiere del parámetro conocido como penalización por gap, ya que los pesos de las librerías fueron calculados con alineamientos donde dicha penalización ya había sido usada.

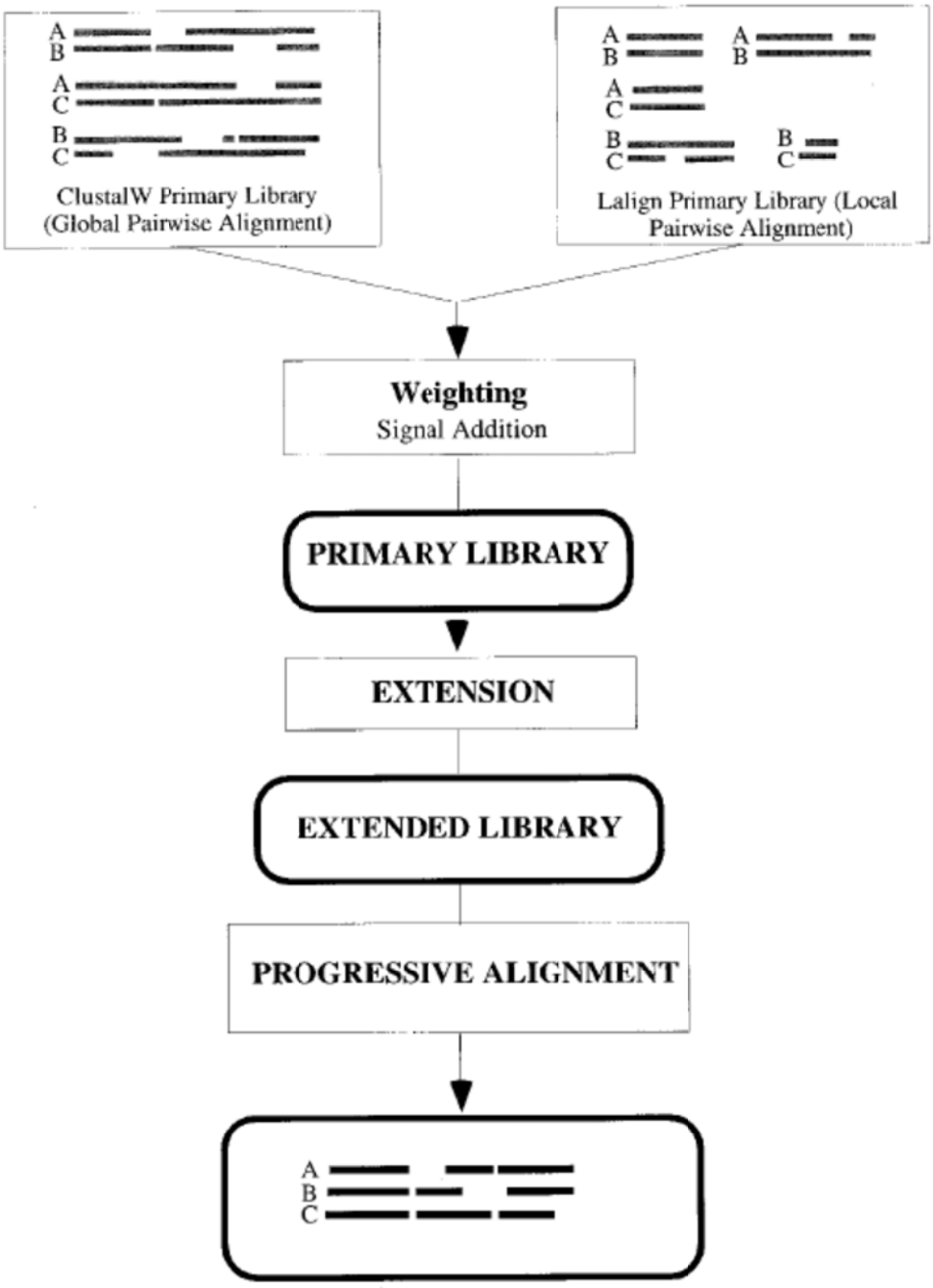


Figura 10. Diagrama de algoritmo T-COFFEE

En el artículo original se indica que la complejidad para calcular la extensión de la librería es, en el peor de los casos, de $O(N^3L^2)$ con L como el promedio de las longitudes de las secuencias, pero indican que esto solo podría ocurrir si los alineamiento de los pares de secuencias son totalmente inconsistentes [65].

GLOSARIO

Ácido Desoxirribonucleico (ADN)	Constituye el material genético de los organismos. Es el componente químico primario de los cromosomas y el material del que los genes están formados.
Ácido Nucleico	Una gran molécula compuesta de subunidades de nucleótidos.
Ácido Ribonucleico de Transferencia (tRNA)	Es un tipo de RNA que tiene estructura con secuencias de tripletes de nucleótidos que son complementarias a las secuencias de los tripletes de nucleotidos del mRNA. La función del tRNA en la síntesis de proteínas es enlazar los aminoácidos y transportarlos a los ribosomas, donde las proteínas son ensambladas de acuerdo al código genético portado por el mRNA.
Ácido Ribonucleico Mensajero (mRNA)	El RNA que sirve como plantilla para la síntesis de proteínas.
Ácido Ribonucleico Polimerasa (RNA polimerasa)	Enzimas que catalizan la síntesis de ácidos nucleicos sobre plantillas de ácidos nucleicos preexistentes, ensamblando el RNA a partir de los ribonucleótidos o el ADN a partir de desoxiribonucleotidos.
Ácido Ribonucleico(RNA)	Un compuesto químico que se encuentran en el núcleo y citoplasma de las células; desempeña una importante función en la síntesis de proteínas y otras actividades químicas de la células. La estructura del RNA es muy similar a la del ADN. Existen varias clases de moléculas RNA, como mensajero, de transferencia y ribosomal así como otros muy pequeños, pero cada uno tiene propósitos diferentes.
Adenina	Una base nitrogenada, es un miembro del par de bases A-T (Adenina-Timina). ADN (ácido desoxirribonucleico). La molécula que contiene codificada la información genética. El ADN es una molécula

enrolada en forma de hélice, que se mantiene unida entre sí por medio de enlaces dobles entre los pares de bases o nucleotidos. Las cuatro bases que contiene los nucleotidos en el ADN son: adenina (A), guanina (G), citocina (C) y Timina (T). En la naturaleza, las pares de bases se forman solamente entre A y T y entre G y C, por lo tanto la secuencia de bases de cada una de las cadenas puede deducirse a partir de uno de sus pares.

Alineamiento de Secuencias Es una forma de mostrar DNA, RNA, o estructuras primarias proteicas para resaltar las zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas consultados. Las secuencias alineadas se escriben con las letras (representando aminoácidos o nucleótidos) en columnas en las que se insertan espacios para que las zonas con idéntica o similar estructura se alineen.

Aminoácidos Una cadena de compuestos proteicos de al menos 20 bases nitrogenadas, que se combinan para formar una proteína. La secuencia de aminoácidos en una proteína están determinados por la acción del código genético.

Backpropagation Es un algoritmo de aprendizaje supervisado que se usa para entrenar redes neuronales artificiales. El algoritmo consiste en minimizar un error (comúnmente cuadrático) por medio de gradiente descendiente, por lo que la parte escencial del algoritmo es cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

Bioinformática Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

Biología Molecular Es el estudio de la vida a un nivel molecular. Esta área se solapa con otros campos de la Biología y la Química,

particularmente Genética y Bioquímica. La biología molecular concierne principalmente al entendimiento de las interacciones de los diferentes sistemas de la célula, lo que incluye muchísimas relaciones, entre ellas las del ADN con el RNA, la síntesis de proteínas, el metabolismo, y el cómo todas esas interacciones son reguladas para conseguir un afinado funcionamiento de la célula.

BLAST (*Basic Local Alignment Search Tool*) es un programa informático de alineamiento de secuencias de tipo local ya sea de ADN o de proteínas. El programa es capaz de comparar una secuencia problema (comunmente llamada query) contra una gran cantidad de secuencias que se encuentren en una base de datos. El algoritmo encuentra las secuencias de la base de datos que tienen mayor parecido a la secuencia query

Cadena polipeptídica Los péptidos están formados por la unión de aminoácidos mediante un enlace peptídico.

Carbono El carbono es un elemento químico de número atómico 6 y símbolo C. Es sólido a temperatura ambiente. Dependiendo de las condiciones de formación, puede encontrarse en la naturaleza en distintas formas alotrópicas, carbono amorfo y cristalino en forma de grafito o diamante. Es el pilar básico de la química orgánica; se conocen cerca de 10 millones de compuestos de carbono, y forma parte de todos los seres vivos conocidos.

Célula En biología, la célula es la unidad más esencial que tiene todo ser vivo. Es además la estructura funcional fundamental de la materia viva según niveles de organización biológica, capaz de vivir independientemente como entidades unicelular, o bien, formar parte de una organización mayor, como un organismo pluricelular. La célula presenta dos modelos

básicos: la procarionte y eucarionte. Su organización general comprende: membrana plasmática, citoplasma y ADN.

Citocina Una base nitrogenada, un miembro del par de bases G-C (guanina y citocina).

Clase Son los grupos o conjuntos de patrones que representan un mismo tipo de concepto

Clasificador El objetivo de un clasificador es agrupar patrones con base en un conocimiento a priori o información estadística extraída de los patrones. Los patrones a clasificar suelen ser grupos de medidas u observaciones, definiendo puntos en un espacio multidimensional apropiado.

Cloroplasto Los cloroplastos son los orgánulos en donde se realiza la fotosíntesis en las células vegetales y de los otros organismos fotosintetizadores. Están formados por un sistema de membranas interno en donde se encuentran ubicados los sitios en que se realiza cada una de las partes del proceso fotosintético.

Codificación Proporcionar la secuencia de nucleótidos adecuada para la síntesis de una determinada proteína.

Codificación one-hot La codificación one-hot funciona de la siguiente manera: para representar la clase k $\in$ {1, 2, ..., p}, se asignan a las componentes del vector de salida yμ los siguientes valores: $y_k^\mu = 1,$ y $y_j^\mu = 0$ para j = 1, 2, ..., k-1, k+1, ..., p.

Código Genético El código genético es la regla de correspondencia entre la serie de nucleótidos en que se basan los ácidos nucleicos y las series de aminoácidos (polipéptidos) en que se basan las proteínas. Es como el diccionario que permite traducir la información genética a estructura de proteína. A, T, G, y C son las "letras" del código genético y representan las bases

nitrogenadas adenina, timina, guanina y citosina, respectivamente. Cada una de estas bases forma, junto con un glúcido (pentosa) y un grupo fosfato, un nucleótido; el ADN y el RNA son polímeros formados por nucleótidos encadenados.

Cada tres nucleótidos de la cadena (cada triplete) forman una unidad funcional llamada codón.

Codones

La información genética, contenida en el NRNA, se escribe a partir de cuatro letras, que corresponden a las bases nitrogenadas del RNA (A, C, G y U), las cuales van agrupadas de tres en tres. Cada grupo de tres se llama codón y lo que hace es codificar un aminoácido o un símbolo de puntuación (Comienzo, parada).

Conjunto fundamental

Es el conjunto finito de patrones $\{(x\omega,y\omega)|\omega \in \{1,2,\ldots p\}\}$ a partir del cual se diseña una memoria asociativa M, donde $x\omega \in A$, $y\omega \in Am$, $p \in N$, y n y m son números naturales que representan las dimensiones de los patrones de entrada y salida, respectivamente. El conjunto A es de donde toman valores las componentes de los patrones de entrada $x\omega$, y es escogido arbitrariamente por el diseñador de la memoria M. El número p indica la cardinalidad del conjunto fundamental.

Contenido Genético

La secuencia de nucleótidos, codificada en tripletes (codones) a lo largo del mRNA, que determina la secuencia de aminoácidos en la síntesis de proteínas. La secuencia de ADN de un gen puede usarse para predecir la secuencia de mRNA, el código genético entonces puede usarse para predecir la secuencia de aminoácidos.

Cromosomas

La estructura genética con capacidad de autoreplicación de células conteniendo el ADN celular que porta en su nucleótido un arreglo de genes en forma de secuencia.

	En procariotes, el ADN cromosomal es de tipo circular y todo el genoma es portado por un cromosoma. En eucariotes, los genomas consisten de un número de cromosomas cuyo ADN está asociado con diferentes tipos de proteínas.
Desoxirribosa	Su fórmula es C5H10O4 . Ésta contiene toda la información genética que será transferida así de generación en generación. Por todo esto la desoxirribosa tiene una gran importancia en todo ser vivo existente. La información genética no se transfiere en la desoxirribosa pero sí es una parte fundamental de todo proceso de información genética ya que de éste se derivará la Ribosa
E. coli	Bacteria común que se ha estudiado intensamente por genetistas debido al tamaño pequeño de su genoma, baja patogenicidad y fácil crecimiento en el laboratorio.
Enlace Peptídico	Véase Cadena Polipeptídica.
Enlaces de Hidrógeno	Se produce un enlace de hidrógeno o puente de hidrógeno (correctamente llamado enlace por puente de hidrógeno) cuando un átomo de hidrógeno se encuentra entre dos átomos más electronegativos, estableciendo un vínculo entre ellos
Estructura Primaria de Proteínas	Se puede decir que la estructura primaria de las proteínas no es más que el orden de aminoácidos que la conforman.
Estructura Secundaria de Proteínas	Es el plegamiento que la cadena polipeptídica adopta gracias a la formación de enlaces de hidrógeno entre los átomos que forman el enlace peptídico. Los puentes de hidrógeno se establecen entre los estables.
Estructura Terciaria de Proteínas	Es el modo en que esa cadena polipeptídica se pliega en el espacio, es decir, a cómo se arrolla una determinada proteína globular. Es la disposición de los dominios en el espacio. La estructura terciaria se realiza de manera que los

aminoácidos apolares se sitúan hacia el interior y los polares hacia el exterior.

Eucariontes Célula u organismo con membrana limitada, estructuralmente núcleo discreto y otros compartimentos subcelulares bien definidos. Los eucariotes incluyen todos los organismos excepto virus, bacterias y algas azul-verdosas. Compárese con procariotes.

Exones Las secuencias de ADN de una proteína codificante de un gen, o de otra manera son las secuencias que están representadas en el mRNA en el momento que llega a los ribosomas, porque son las regiones del gen que se expresan. Comparar con intrones. Exonucleasa. Una enzima que penetra o rompe nucleótidos secuencialmente a partir de terminaciones libres de una línea de ácido nucleico sustraído.

Expresión Genética El proceso por el cual la información codificada de un gen es convertida dentro de la estructura presente y operando en la célula. Los genes expresados incluyen aquellos que se transcriben dentro del mRNA y luego trasladados dentro de proteínas y aquellos que se transcriben dentro de RNA pero que no son trasladados como proteínas (por ej. los RNAs de transferencia y ribosomal).

Factor de Transcripción Un factor de transcripción es una proteína que participa en la iniciación de la transcripción del ADN, pero que no forma parte de la RNA polimerasa. Los factores de transcripción actúan reconociendo sitios en el ADN o a otro factor, o a la RNA polimerasa.

Los factores de transcripción son proteínas que, tras ser estimuladas por señales citoplasmáticas, tienen la capacidad de regular la expresión génica en el núcleo celular.

Los factores de transcripción pueden ser activados o desactivados selectivamente por otras proteínas, a menudo

como paso final de la cadena de transmisión de señales intracelulares.

Filogenéticos Filogenia (del griego: phylon = tribu, raza y genetikos = relativo al nacimiento, de génesis = nacimiento) es la disciplina que estudia las relaciones evolutivas entre las distintas especies, reconstruyendo la historia de su diversificación (filogénesis) desde el origen de la vida en la Tierra hasta la actualidad. La filogenia proporciona el fundamento para la clasificación de los organismos.

Fosfatos El fosfato forma parte de los nucleótidos, los monómeros en que se basa la composición del ADN y demás ácidos nucleicos. También hay fosfato en la composición de algunos lípidos formadores de membranas, como los fosfoglicéridos, donde su elevada constante de ionización constribuye a la carga eléctrica de la «cabeza hidrófila».

Genes La unidad física y fundamental de la herencia. Es una secuencia ordenada de nucleótidos localizados en una posición particular, sobre un cromosoma particular, que codifica un producto funcional específico (p.e. una proteína o una molécula de RNA).

Genoma Todo el material genético en los cromosomas de un organismo particular, el tamaño de un genoma se da generalmente por el número total de pares de bases.

Genómica La genómica es la rama de la biología que se encarga del estudio de los genomas. Se considera a un genoma como el conjunto de información genética (ADN) de un organismo.

Guanina Una base nitrogenada, un miembro del par de bases G-C (guanina-citocina).

Homología Similitudes en ADN o secuencias de proteínas entre

individuos de la misma especie o entre diferentes especies. In vitro. Afuera de un organismo viviente.

Intrones La secuencia de bases de ADN que interrumpen la secuencias de proteína codificante de un gen, estas secuencias se transcriben dentro del RNA, pero se cortan fuera del mensaje, antes de que se trasladen como proteínas, razón por la que también se les conoce como secuencias interpuestas.

Macromoléculas Son moléculas que tienen una masa molecular elevada, formadas por un gran número de átomos. Generalmente podemos describirlas como la repetición de una o unas pocas unidades mínimas o monómeros, formando los polímeros. A menudo el término macromolécula se refiere a las moléculas que contienen más de 100 átomos. Pueden ser tanto orgánicas como inorgánicas, y se encuentran algunas de gran relevancia en el campo de la bioquímica, al estudiar las biomoléculas. Dentro de las moléculas orgánicas sintéticas se encuentran los plásticos.

Material Genético Ver *Genoma.*

Matriz Una matriz es un conjunto de elementos de cualquier naturaleza aunque, en general, suelen ser números ordenados en filas y columnas.
Se llama matriz de orden "m × n" a un conjunto rectangular de elementos aij, dispuestos en filas "m" y en columnas "n". El orden de una matriz también se denomina dimensión o tamaño, siendo m y n números naturales.

Membrana Celular La membrana que universalmente envuelve al citoplasma de las células.

Memoria Asociativa Una memoria asociativa tiene por objetivo: recuperar de manera perfecta patrones, a partir de patrones de entrada, que quizá estén alterados con algún tipo de ruido.

Memoria Asociativa Bidireccional

Consta de dos capas de elementos que están completamente interconectados entre capas. Las unidades pueden o no tener conexiones de retroalimentación consigo mismas.

Al igual que en otras redes neuronales, en la arquitectura de la memoria asociativa existen pesos que se asocian a las conexiones entre elementos de un proceso. A diferencia de otras arquitecturas, estos pesos se determinan por anticipado, así es posible identificar a todos los vectores de entrenamiento.

Memoria Autoasociativa

Si en cada asociación sucede que el patrón de entrada es igual al de salida, la memoria es autoasociativa; en caso contrario, la memoria es heteroasociativa; esto significa que una memoria autoasociativa puede considerarse como un caso particular de una memoria heteroasociativa

Una memoria es autoasociativa si se cumple que: $x\mu = y\mu \quad \forall \mu \in \{1,2,\ldots..,p\}$

Memoria de Hopfield

Se considera a la memoria de Hopfield como una derivación de las bidireccionales, aunque existe la duda que sea ésta la forma en que se originó la memoria de Hopfield. Las versiones son la memoria discreta de Hopfield y la memoria continua de Hopfield, dependiendo de si las unidades de salida son una función discreta o continua de las entradas, respectivamente.

Memorias Heteroasociativas

Si en cada asociación sucede que el patrón de entrada es diferente al de salida, la memoria es heteroasociativa; en caso contrario, es autoasociativa.

Una memoria es heteroasociativa si se cumple lo siguiente: $\exists \mu \in \{1,2,\ldots..,p\}$ para el que $x\mu \neq y\mu$

Microarreglos

Un microarreglo de ADN (del inglés DNA microarrays) es una superficie sólida a la cual se unen una serie de

fragmentos de ADN. Las superficies empleadas para fijar el ADN son muy variables y pueden ser vidrio, plástico e incluso chips de silicio. Los arreglos de ADN son utilizadas para averiguar la expresión de genes, monitorizandose los niveles de miles de ellos de forma simultanea.

Mitocondria

Las mitocondrias son los orgánulos que se encuentran en prácticamente todas las células eucariotas (también hay en células gaméticas), encargados de suministrar la mayor parte de la energía necesaria para la actividad celular; actúan por tanto, como centrales energéticas de la célula y sintetizan ATP por medio de la fosforilación oxidativa. La mitocondria presenta una membrana exterior permeable a iones, metabolitos y muchos polipéptidos.

Molécula

Una molécula es una partícula formada por un conjunto de átomos ligados por enlaces covalentes, metálicos, o iónicos de forma que permanecen unidos el tiempo suficiente como para completar un número considerable de vibraciones moleculares

Morfología Matemática

La palabra morfología significa forma y estructura de un objeto. Para imágenes binarias se definen operaciones morfológicas. y con estas se constituye una herramienta de extracción de componentes de imagen útiles en la representación y descripción de la forma de las regiones.

Las operaciones básicas de la morfología matemática son: dilatación se puede simplificar a decir que es agregar pixeles a un objeto, hacerlo mas grande, y luego erosión es hacerlo mas chico. La erosión saca los "outlayers del objeto".

Luego la combinación de estas dan origen a los operadores Apertura y Clausura. El primero consiste en aplicar una erosión seguida de una dilatación aplicando la misma forma estructurante, como resultado esta tiende a "abrir pequeños huecos". La clausura es el la aplicación de las operaciones

básicas en el sentido inverso, y resulta en "cerrar los huecos".

Motifs En genética, una secuencia de motifs es una secuencia de patrones de nucleótidos o aminoácidos, que es extensa y es, o se cree que es, biológicamente significativa. Para las proteínas, una secuencia de motifs se distingue por una estructura de motifs, un motif formado por un arreglo tridimensional de aminoácidos, que pueden no ser adyacentes.

Mutación Cualquier cambio heredable en la secuencia de ADN. Comparar con polimorfismo.

Núcleo En biología celular y citología: al núcleo celular, parte central de la célula rodeada de una membrana propia, llamada membrana nuclear, que contiene el ácido desoxirribonucleico (ADN o en inglés DNA) celular, donde se encuentran codificados los genes.

Nucleótidos Molécula formada por una base nitrogenada (A, G, P, C) con azúcar de 5 carbonos y un grupo fosfato. El ácido nucléico es un polímero de muchos nucleótidos.

Pares Bases Dos bases nitrogenadas (Adenina y Timina o Guanina y Citocina) que se mantienen unidas por medio de doble enlaces o puentes. Las dos cadenas de ADN se mantienen unidas en forma de doble hélice por enlaces entre los pares de bases.

Patrones Son representaciones abstractas de un concepto en el mundo físico. Los cuales exhiben cierta regularidad en una colección de observaciones conectadas en el tiempo, en el espacio, o en ambas, y que pueden servir como modelo.

Péptidos Los péptidos son un tipo de moléculas formadas por la unión de varios aminoácidos mediante enlaces peptídicos.

Perceptron

Un perceptrón se refiere a una neurona artificial y también como a la unidad básica de inferencia en forma de discriminador lineal, que suele formar parte de una red neuronal artificial.
Un perceptrón puede clasificar datos que sean linealmente separables. En el caso de un perceptrón con dos entradas deberá poder trazarse una única línea que separe las dos clases que permite identificar el perceptrón.

Polipéptidos

Un polipéptido es un péptido formado por una cadena simple de más de 10 aminoácidos y de menos de 50 aminoácidos. Varias cadenas de polipéptidos se pueden asociar para formar las proteínas.

Procariontes

Célula u organismo carente de una membrana definida, núcleo estructuralmente discreto así como de otros compartimentos celulares. Las bacterias son procariotes. Compárese con eucariotes.

Promotores

Un sitio sobre el ADN en el cual la RNA polimerasa se enlaza para iniciar la transcripción.

Proteínas

Una gran molécula compuesta de una o más cadenas de aminoácidos en un orden específico; el orden está determinado por la secuencia de bases de nucleotidos en el gen que codifica la proteína. Las proteínas se requieren para la estructura, función y regulación de las células corporales, tejidos y órganos y cada proteína posee funciones únicas. Ejemplos son las hormonas, enzimas y anticuerpos.

Proteómicas

La proteómica puede definirse como la genómica funcional a nivel de proteínas. Es la ciencia que correlaciona las proteínas con sus genes, estudia el conjunto completo de proteínas que se pueden obtener de un genoma. La proteómica intenta resolver preguntas como: ¿Qué función tienen las proteínas?,

¿Qué tipo de modificaciones postransduccionales sufren las proteínas y cuál es su función?, ¿Cómo varían las proteínas de una célula enfrentada a distintas condiciones ambientales?.

Purina Una base nitrogenada, contiene un sólo anillo púrico y la contiene los ácidos nucleicos. La purina en el ADN es la adenina y en el RNA es la guanina.

Reconocimiento de Patrones El reconocimiento de patrones, también llamado lectura de patrones, identificación de figuras y reconocimiento de formas1 es el reconocimiento de patrones en señales. No sólo es un campo de la informática sino un proceso fundamental que se encuentra en casi todos las acciones humanas.

El punto esencial del reconocimiento de patrones es la clasificación: se quiere clasificar un señal dependiendo de sus características. Señales, características y clases pueden ser de cualquiera forma.

Ribosa Es el componente del ácido ribonucleico y otras sustancias como nucleótidos

Secuencia Forma en que suceden o encadenan los nucleótidos a lo largo de las cadenas de ADN o RNA.

Similitud Se presenta cuando dos objetos tienen atributos cuyos que les son comunes y cuyas medidas o cualidades relativas se ajustan a un grado de variación.

Sitios de Empalme Las zonas de cambio entre un exon-intron (EI) o intron-exon (IE) son conocidos como sitios de empalme (splice-sites)

Técnicas Heurísticas Una heurística es un algoritmo que ofrece uno o ambos objetivos; por ejemplo, normalmente encuentran buenas soluciones, aunque en ocasiones no hay pruebas de que la solución no pueda ser arbitrariamente errónea; o se ejecuta razonablemente rápido, aunque no existe tampoco prueba de

que deba ser así.

Timina Una base nitrogenada, un miembro del par de bases A-T (adenina-timina).

Transcripción La síntesis de una copia de RNA a partir de una secuencia de ADN (un gen) el primer paso en la expresión del gen. Proceso por el que la enzima RNA polimerasa cataliza la síntesis de a partir de una de las cadenas (la cadena molde) de la doble hélice de ADN.

Upstream La técnica consiste en la activación de un gen reportero (s) por la acción de un factor de transcripción que se enlaza a la secuencia regulatoria "UAS" (en inglés, Upstream activating sequence) localizada en el promotor mas arriba del sitio de inicio de la transcripción.

El principio de la técnica es el siguiente, el factor de transcripción es separado en dos fragmentos, uno que reconoce UAS y el otro que promueve la activación de la maquinaria de transcripción. Cada fragmento es introducido a las proteínas cuya interacción se desea analizar, usando técnicas de ingeniería genética. Si las proteínas forman un complejo entre sí, los dos fragmentos del factor de trancripción se encontrarán y el gen reportero será tanscrito.

Vector Matemáticamente un vector puede ser un conjunto de elementos ordenados entre sí pero a diferencia de un conjunto normal como el de los números naturales, éste está ordenado. Así, se llama vector de dimension n a una tupla de n números reales (que se llaman componentes del vector). El conjunto de todos los vectores de dimensión n se representa como $\mathbb{R}^n$ (formado mediante el producto cartesiano).