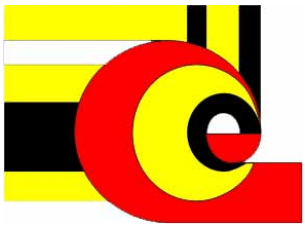

Instituto Politécnico Nacional

Centro de Investigación en Computación

Búsqueda de patrones en cadenas de ADN

T E S I S

Que para obtener el grado de:

Maestría en Ciencias de la Computación

Presenta:

Ing. Luis Alberto Ortíz Chan

Directores de Tesis:

Dr. Adolfo Guzmán Arenas
Dr. Gilberto Lorenzo Martínez Luna

México, D.F. Enero 2016

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 16:00 horas del día 14 del mes de diciembre de 2015 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Búsqueda de patrones en cadenas de ADN"

Presentada por el alumno:

ORTIZ	**CHAN**	**LUIS ALBERTO**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	3	0	1	2	2

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Adolfo Guzmán Arenas

Dr. Gilberto Lorenzo Martínez Luna

Dr. Grigori Sidorov

Dr. Ricardo Barrón Fernández

Dr. Guillermo Manuel Mallen Fullerton

Dr. Rolando Menchaca Méndez

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

ESTADOS UNIDOS MEXICANOS
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 16 del mes diciembre del año 2015, el (la) que suscribe Luis Alberto Ortíz Chan alumno (a) del Programa de Maestría en Ciencias de la Computación con número de registro B130122, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Adolfo Guzmán Arenas y Dr. Gilberto Lorenzo Martínez Luna y cede los derechos del trabajo intitulado "Búsqueda de patrones en cadenas de ADN", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección myluisortiz@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Luis Alberto Ortiz Chan

Nombre y firma

Resumen

Uno de los problemas clásicos en la bioinformática es la búsqueda de patrones frecuentes con una tarea bien identificada en las secuencias de ADN.

En este trabajo se desarrolla un algoritmo alternativo llamado KTreeMotif para la búsqueda de motifs en secuencias de ADN, con rendimiento similar a los algoritmos más usados hoy en día como son el Gibbs Sampler, el Motif Sampler, el MEME y el SP-Star. KTreeMotif busca aprovechar las metodologías y cualidades de un conjunto de estos algoritmos con la finalidad de reafirmar los resultados obtenidos por otros medios.

KTreeMotif además, implementa una nueva estructura de datos para almacenar y recorrer las subcadenas de una manera más sistemática y rápida que el recorrido secuencial que suele usarse; también se hace una simplificación de la función de distancia entre una PWM y una secuencia sin perder información logrando el mismo resultado, y finalmente una mejora en exactitud de la función de distancia entre dos secuencias para el caso específico de la búsqueda de patrones frecuentes. Todas estas novedades son integrables a los otros algoritmos que atacan este problema.

Para llevar a cabo las pruebas de evaluación de KTreeMotif fue necesario implementar los algoritmos contra los que se realizó su comparación; y utilizando la base de datos JASPAR que contiene los resultados correctos, estos fueron buscados por los algoritmos implementados y de igual manera por el algoritmo propuesto.

El algoritmo adolesce en el tiempo de respuesta pero el objetivo de la propuesta es ofrecer una alternativa más a la solución del problema, al corroborar que los resultados que fueron obtenidos iguales por distintos algoritmos y por esta propuesta son más fiables de ser los correctos.

Además del análisis y comparación de las metodologías que siguen los algoritmos de búsqueda de motifs, se encontró que el principal problema con la precisión está en la función objetivo que no representa de manera correcta un patrón motif y califica como mejores motifs a patrones diferentes al correcto. De esto se concluye que con una mejor función que modele al motif sería posible aumentar la precisión de cualquier algoritmo.

Abstract

One of the classic problems in bioinformatics is the search of useful frequent patterns with a well-defined task among DNA sequences.

In this work an alternative algorithm called KTreeMotif is developed for motif finding in DNA sequences, with similar performance to the most used algorithms nowadays such as Gibbs Sampler, Motif Sampler, MEME and SP-Star. KTreeMotif seeks to exploit the methodologies and advantages of a set of these algorithms in order to validate the outcome of other means.

KTreeMotif, additionally, implements a new data structure to store and look through the substrings in a more systematic and fast way than brute force simple iteration that is mostly used; also, a simplification of the distance function between a PWM and a sequence without losing information reaching the same result was added; and finally an improvement in accuracy of the distance function between two sequences for the specific case of frequent pattern searching was included. All these innovations are easily integrable into other algorithms that tackle this problem.

In order to make the performance tests on KTreeMotif it was necessary to implement the algorithms against it was compared; and using the JASPAR database that provides the correct motifs, the performance tests were run on the implemented algorithms and the proposed algorithm to get their motifs results to compare against the correct ones.

The proposed algorithm lacks a good response time but the aim of the proposal was to offer one more alternative to the solution of the problem, validating the results obtained by different algorithms with this proposal's, so that their results can be more reliable to be the correct ones.

Besides the analysis and comparison of the methodologies that lead the motif finding algorithms, it was found that the main challenge on the accuracy is in the objective function that does not represent in a right way a pattern motif and scores wrong patterns as better motifs. We conclude that with a function that describe better the motif it would be possible to increase the accuracy of any algorithm.

Agradecimientos

Agradezco a mi familia, a mis padres y mis hermanos que me han dado su apoyo incondicional antes y durante la maestría.

Agradezco a mis amigos por su apoyo moral y técnico durante el trabajo de esta tesis.

Agradezco a mis profesores de la maestría, en especial a mis directores de tesis, el Dr. Gilberto Lorenzo Martínez Luna y el Dr. Adolfo Guzmán Arenas, por sus valiosos consejos, su apoyo, las herramientas y conocimientos para llevar a cabo esta tesis de manera exitosa.

Agradezco al Dr. Guillermo Manuel Mallén Fullerton por su ayuda con el tema, sus ideas y su experiencia prestadas durante el desarrollo de la tesis desde que surgió la idea.

A los profesores András Benczúr, Istvan Miklos y al centro de investigación MTA SZTAKI, por permitirme hacer de la estancia en Budapest, Hungría una valiosa experiencia de aprendizaje que acrecentó este trabajo.

Agradezco al Instituto Politécnico Nacional y al Centro de Investigación en Computación por brindar un ambiente propicio para la investigación y el conocimiento, por la visión y habilidades que en este lugar se enseñan para aplicar en México y el mundo; y sobre todo por darme la oportunidad de aprovechar todo esto y la beca BEIFI de apoyo a los alumnos.

Agradezco al Consejo Nacional de Ciencia y Tecnología, ya que gracias a las becas que proporciona fomenta la investigación y el estudio de posgrados.

Índice

Índice de Imágenes

Índice de Tablas

Capítulo 1: Introducción

Desde el siglo pasado con el avance de la biología y la genética se han encontrado problemas difíciles de tratar mediante la sola experimentación, al llegar la computación se empezó a aprovechar su intervención en las ciencias para resolver estos problemas que se presentaban, así nació la bioinformática.

La bioinformática es la unión de la biología y la informática e involucra el uso de la computación para el almacenamiento, recuperación, manipulación y distribución de información relacionada a las macromoléculas biológicas como son el ADN, el ARN y las proteínas. Se enfatiza el uso de la computación porque la mayoría de las tareas en el análisis de datos genómicos son altamente repetitivas o matemáticamente complejas. (Luscombe et al., 2001).

La bioinformática difiere de un campo relacionado conocido como *biología computacional* en que la bioinformática se limita al análisis de los genes, genomas y sus componentes; análisis hecho sobre sus secuencias, sus estructuras y su funcionamiento, por lo que se le considera también *biología molecular computacional*. Mientras tanto la biología computacional abarca todas las áreas de la biología que involucran computación (Xiong, 2006).

Entre los muchos problemas que actualmente trata la bioinformática, destacaremos uno que surgió en la investigación del ADN en 1989 (Stormo and Hartzell, 1989).

1.1. Planteamiento del problema

Las cadenas de ADN, responsables de transmitir la información necesaria para el funcionamiento de cualquier ser vivo, están compuestas de secuencias de cuatro elementos distintos llamados bases a los que identificamos con las letras A, C, G y T. En cada secuencia de ADN se encuentran secciones que sirven para distintos objetivos, una de esas secciones son los llamados motifs que son rastreados por unas moléculas para activar el proceso de creación de una nueva proteína, cuya "plantilla" también está codificada en el ADN inmediatamente posterior a la aparición del motif.

Los motifs pueden reconocerse principalmente por ser combinaciones de bases muy frecuentes en porciones significatinas del ADN y además tener un tamaño relativamente pequeño en comparación con el tamaño de la secuencia y de sus demás elementos. Se han desarrollado técnicas para extraer estas porciones significativas de la secuencia donde se encuentran los motifs, aunque sin poder conocer el motif interior, que sigue siendo necesario para saber su función específica, las moléculas que lo reconocen y cuándo lo usan. Con ayuda de estas técnicas el problema disminuye a encontrar los motifs solo en algunas partes de la cadena de ADN (las porciones extraídas), en vez de en toda la secuencia pero el problema sigue siendo NP-completo: un patrón desconocido en posiciones desconocidas de varias secuencias de las que solo se conoce que el patrón es la subcadena más probable. En la figura 1 se puede ver un ejemplo del tipo de subcadenas que se busca.

Figura 1: Ejemplo de un motif encontrado en varias secuencias.
Las apariciones están marcadas con negritas y sus "errores" o mutaciones subrayadas.

El problema de la búsqueda de motifs en secuencias de ADN presenta características específicas que aumentan su grado de complejidad, aunque la mayoría ya ha sido tratada en los algoritmos actuales creados específicamente para resolver este problema:

- No se conoce el tamaño del motif y aunque éste varía entre 5 y 20 bases aprox. la mayoría de los algoritmos solo saben trabajar con un tamaño de motif exacto indicado por el usuario.
- Existen secuencias que tienen un mismo motif más de una vez, secuencias con varios motifs distintos y secuencias donde ningun motif aparece, secuencias ruido, en el conjunto de secuencias a trabajar.
- Es poco probable encontrar segmentos completamente iguales, aunque el motif es un patrón específico, sus apariciones suelen tener mutaciones en cualquier posición y por lo tanto casi ninguna aparición es igual a otra, aun así son muy similares pues las mutaciones son pocas para permitir que siga siendo el patrón más frecuente.
- En cada posición del motif no hay precisamente una sola base más probable, a veces 2 ó más comparten la máxima probabilidad en determinada posición.
- Se ha encontrado que la presencia de ciertas bases en ciertas posiciones del motif genera que otras bases en ciertas posiciones aparezcan, por lo que tienen menor probabilidad de mutar, la mayoría de los algoritmos no sabe encontrar ni utilizar esta característica.
- Existen motifs que no están completamente continuos, tienen una parte en medio que no pertenece al motif y varía como cualquier otro segmento exterior, esta parte genera ruido y hace al motif bajar su probabilidad si se considera parte de él.

1.2. Justificación

El ADN contiene la información que necesitamos para funcionar. Sin embargo, a pesar de ser una estructura sencilla, se desconoce cómo funciona su codificación y cómo convierte combinaciones de cuatro caracteres en instrucciones y datos que los demás elementos a su alrededor reconocen. Por tanto, encontrar una mejor manera de reconocer una parte de su contenido es útil para entender mejor cómo se codifica.

Una vez que se obtiene un motif computacionalmente se procede a la experimentación biológica, ésta consiste en suprimir las subsecuencias y comprabar que desencadena en el ADN funcionando como si el motif no existiera, esto es, que pese a la petición de llamado del motif para producir una proteína o elemento biológico éste no se produjera. Sin los algoritmos de búsqueda de motifs los biólogos experimentaban con las subsecuencias, suprimiendo al azar cualquier zona hasta obtener algun resultado diferente al funcionamiento normal, lo cual requería demasiados experimentos diferentes para un solo caso, un sólo estudio, donde la biología requiere la investigación de varios motifs para trabajos más grandes.

Resolver la búsqueda de motifs ayuda a encontrar las secciones completas que participan en la creación de una nueva proteína a partir del ADN; debido a que un motif, un promotor y el gen que es el esquema o "plantilla" para la proteína se encuentran en secciones muy cercanas en el ADN y encontrar y conocer al motif ayuda a encontrar los demás, conociendo de mejor manera en el siguiente capítulo la ubicación de cada uno de estos elementos en la secuencia de ADN, sin embargo, lo que es claro es que los promotores y genes son estructuras más grandes, más complejas de encontrar y delimitar, entonces encontrar una subsecuencia de cierto motif ayuda mucho al problema de encontrar promotores y genes.

Poniendo un caso de ejemplo, teniendo un motif y una de sus subsecuencia, se busca genes o promotores de genes de las bases de datos biológicas en la zona del ADN posterior cercana a la subsecuencia, si se encuentra algun gen se puede trabajar con suprimir la subsecuencia y verificar que el gen deje de hacer su trabajo, esto significa que efectivamente ese motif afecta a ese gen; si no se encuentra posiblemente se encuentre uno frente a un gen no documentado y se trabaja con la búsqueda de genes en la zona cercana posterior a la subsecuencia.

No solo se desconoce cómo codifica el ADN sus elementos, sino también cómo las moléculas a su alrededor saben cuándo usar su contenido y cuándo no, un mecanismo llamado **regulación genética**. Al estilo de una biblioteca, podría decirse que el ADN contiene libros de instrucciones y no se sabe cómo organiza la persona cuáles libros ir a buscar y cuáles no, de acuerdo a las necesidades de cada momento, aún así éstas siempre recurren a los motifs para solo ver los libros que necesitan y tomarlos. Conocer mejor a los motifs ayuda a conocer cómo la célula sabe las instrucciones que necesita ir a buscar al ADN ante cada situación. Finalmente entender la regulación de las proteínas ayuda a conocer cuándo las células trabajan bien y cuándo tienen un error de regulación, lo que hace que una célula enferme por no poder producir una proteína que necesita.

Otro problema donde la búsqueda de motifs ayuda mucho es la alineación de secuencias, cuando se tiene un conjunto de secuencias que se procuran alinear, es decir, indicar las posiciones equivalentes entre todas, para que la información que se conoce de una secuencia sea aprovechada para conocer la otra, básicamente este problema consiste en encontrar los fragmentos de las secuencias que son comunes, es decir, sus patrones comunes. Tratar esto como una búsqueda de patrones frecuentes es útil para encontrar una sección común a partir de la cual empezar a alinear lo demás de las secuencias.

Se podría decir que la búsqueda de estos patrones frecuentes en cadenas de texto puede ampliarse a encontrar patrones más complejos, como son los mutables en este caso. En fin, una búsqueda de patrones difusos.

Los métodos actuales suelen ser muy lentos y sobre todo muy inexactos en sus resultados a pesar de que el problema ha sido tratado por varios años por biólogos, matemáticos y científicos de la computación, lo que dificulta el aprovechamiento de sus resultados como base de conocimiento para la biología.

1.3. Objetivo general

Desarrollar un algoritmo alternativo para la búsqueda de motifs en secuencias de ADN que aproveche las metodologías y cualidades de los algoritmos actuales que tratan el problema y reafirme los resultados obtenidos .

1.4. Objetivos específicos

Para alcanzar el objetivo de la tesis es preciso cumplir los siguientes objetivos específicos:

- Encontrar bases de datos para probar y calificar el rendimiento de los algoritmos de búsqueda de motifs en distintos casos de secuencias de entrada.
- Conocer los algoritmos actuales de solución al problema y las ventajas que tienen para solucionar el problema.
- Programar algunos de los algoritmos actuales y observar sus resultados con la base de datos.
- Crear un método que combine las mejores características de los algoritmos y entregue resultados equiparables a los demás.
- Probar el nuevo algoritmo en los datos de prueba para conocer su rendimiento y compararlo con los otros métodos.

1.5. Aportaciones

Las aportaciones de este trabajo de tesis son:

1. Se implementó una nueva estructura de datos **k-mer tree** para almacenar las secuencias y recorrer sus subcadenas de una manera más sistemática y rápida que el recorrido secuencial que suele usarse, aprovechando la programación dinámica. Esta estructura se encuentra descrita en el apartado 3.2.2.

2. Se realizó una mejora en exactitud de la función de distancia entre dos cadenas del mismo tamaño para el caso específico de la búsqueda de patrones frecuentes. Esta función además de servir para comparación de cadenas también mantiene a los patrones comunes cerca y representa una mejora respecto a la distancia Hamming. Se explica en el apartado 3.2.3.

3. Se hizo una simplificación de la función de distancia entre una PWM y una cadena del mismo tamaño, sin perder información logrando el mismo resultado mediante menos cálculos. Se explica en el apartado 3.2.4.

4. Se diseñó el algoritmo **KTreeMotif** que es una combinación del algoritmo Weeder (Pavesi et al., 2001) y el algoritmo MEME (Bailey and Elkan, 1995) donde se usa el k-mer tree para crear candidatos a patrones ampliando las posibilidades de encontrar el motif y disminuyendo la probabilidad de caer en máximos locales y se entrega al algoritmo MEME mejores valores de entrada para que converga más rápido. Este algoritmo se encuentra explicado en el apartado 3.3.

Algunos detalles de las aportaciones se dan en la sección 5.2.

Capítulo 2: Marco teórico y estado del arte

2.1. Conceptos básicos

Todos los organismos vivos están compuestos de unidades básicas de vida llamadas células. Dentro de las células la vida depende de tres tipos de moléculas: el ADN, el ARN y las proteínas. En términos generales, el ADN de una célula contiene una vasta biblioteca que describe como funciona la célula. El ARN transfiere pequeñas piezas de esta biblioteca a las distintas partes de la célula donde se utilizarán para crear proteínas. Las proteínas forman enzimas que desencadenan reacciones bioquímicas, envían señales a otras células, forman los principales componentes del cuerpo, o incluso llevan a cabo el trabajo de la célula (Wong, 2004).

2.1.1. La estructura del ADN

El nombre ADN es un acrónimo para *ácido desoxirribonucleico*. Las moléculas de ADN de todos los organismos –animales, plantas, hongos, bacterias por igual– están hechas química y físicamente de igual forma. Cuando decimos que la información del ADN es usada como una biblioteca nos referimos en el sentido de que es leída y copiada, quizá millones de veces para que esas copias ejecuten las tareas de la célula dejando el ADN disponible para más lecturas (a modo de sólo consulta).

El ADN consiste de dos hebras que se entrelazan juntas para formar una doble hélice. Cada hebra es una cadena de pequeñas moléculas llamadas nucleótidos, bloques constructores del ADN y ARN. Cada uno de los nucleótidos consiste de tres partes: una molécula (pentosa) de azúcar desoxirribosa que sirve de esqueleto de la hebra; un grupo fosfato para conexión y transporte de energía; y una base nitrogenada, la cual determina el tipo de nucleótido. A lo largo de una hebra se van alternando un fosfato, una pentosa conectada a una base nitrogenada y a continuación el siguiente fosfato como se puede ver en la figura 2. Esta alternancia permite conocer el inicio de una hebra en una pentosa desconectada en su carbono 5' y el final de la hebra en una pentosa desconectada en su carbono 3' y así identificar el sentido y la orientación de la hebra completa.

Existen cuatro tipos de bases nitrogenadas: (A) Adenina, (C) Citosina, (G) Guanina y (T) Timina. De esta forma llamamos a los nucleótidos para mayor practicidad por la inicial de su base y representamos el contenido de una molécula de ADN como una secuencia de A's, C's, G's y T's. Estas bases se conectan desde una hebra a la hebra contraria mediantes enlaces de hidrógeno para mantener la doble hélice siguiendo una regla específica: la (A) Adenina solo se conecta con la (T) Timina y la (C) Citosina solo se conecta con la (G) Guanina. Cade mencionar que la molécula de ARN es similar a la de ADN reemplazando a la (T) Timina en toda ocasión por otra base nitrogenada llamada (U) Uracilo, así cuando una molécula de ARN se crea a partir de un segmento del ADN es similar excepto que contiene (U) en vez de (T). La fuerte regla de emparejamiento natural de las bases sirve a un simple proósito para la molécula de ADN, la protegen de enlaces con agentes externos indeseados.

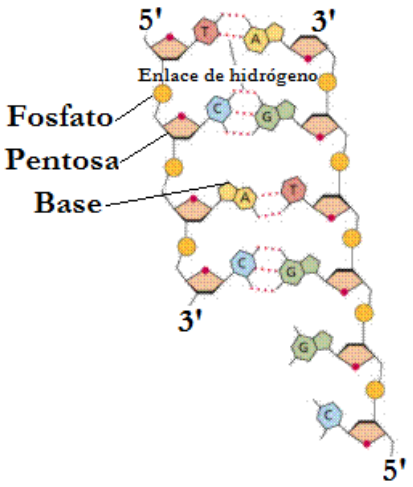

Figura 2: Molécula del ADN

El usual término *par de bases* (bp) es a menudo usado para referirse a los nucleótidos en una doble hélice como medida de longitud de secuencias de ADN o ARN pues por cada base en una longitud, también se cuenta su complemento enlazado desde la hebra opuesta. Como resultado de los emparejamientos de nucleótidos, conociendo la secuencia de una hebra de ADN se entiende la secuencia de la otra hebra. En otras palabras, cada hebra contiene toda la información necesaria para reconstruir la hebra complementaria. De hecho, ésta es la base para la replicación de ADN en las células.

No sólo es importante el contenido de la secuencia, sino la dirección también; con un segmento de ADN y dos etiquetas, extremo 5'- al inicio y extremo 3'- al final; una secuencia 5'-GATCATTGGC-3' tiene su correspondiente hebra complementaria 3'-CTAGTAACCG-5'.

Por lo general, el ADN en una célula está organizado en no una sino varias moléculas físicamente separadas llamadas **cromosomas**, y cada una contiene una doble hélice de ADN; al conjunto de cromosomas (todo el ADN de la célula) se le llama **genoma**, el cual es igual en cada célula del mismo organismo. Mientras que diferentes especies tienen diferente número de cromosomas, la organización específica en cada cromosoma entre los organismos de la misma especie es siempre igual. Cualquier aberración en la organización es a menudo letal o conlleva a desórdenes genéticos serios (Sung, 2010).

No está de más mencionar que también se puede encontrar ADN circular, es decir, cuyos extremos se enlazaron, en los cromosomas de muchas bacterias y en otros organismos en regiones distintas de sus células y con una utilidad diferente.

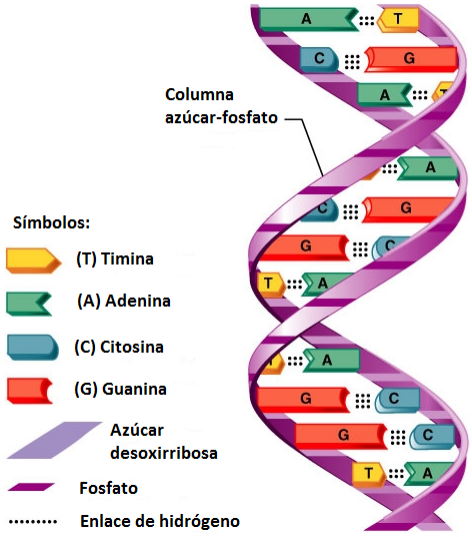

Figura 3: Partes del ADN

2.1.2. Del ADN a las proteínas

Un gen es una secuencia porción de una hebra de ADN que se copia para servir en alguna función de la célula una vez que se traduce en ARN y/o una proteína.

A continuación se describe el proceso por el que la información se transfiere del ADN al ARN y luego a una proteína mediante los genes en dos pasos:

1. Transcripción: El ADN se transcribe en ARN mensajero (mARN). Durante el proceso un gen del ADN se sintetiza en mARN.

2. Traducción: El mARN se traduce a una proteína. En este proceso el mARN se convierte a una secuencia de aminoácidos enlazándose uno por uno.

Es de notar que el proceso de transcripción es diferente entre dos tipos de células. Las células procariotas, las más simples, carecen de núcleo y constituyen los organismos unicelulares. Las células eucariotas, tienen un núcleo donde se almacena el genoma y forman a los organismos pluricelulares.

2.1.3. Transcripción en procariotas

De cierta manera la transcripción en las células procariotas es simple y con pocas variaciones, lo que hace que sea más fácil de trabajar que con las células eucariotas.

1. El *ARN polimerasa* (pARN) temporalmente separa las dos hebras del ADN.

2. El pARN localiza el punto de inicio de la transcripción, un marcador que denota el inicio de un gen. Este paso se detallará más adelante.

3. Luego, el pARN sintetiza a partir del punto de inicio un *ARN mensajero* (mARN) siguiendo dos reglas: mediante emparejamiento por cada C, G o T en el ADN se añade un G, C o A correspondiente y por cada A en el ADN se añade el nuevo nucleótido con una base (U) Uracilo que el ARN contiene en lugar de la T.

4. Cuando el pARN llega al punto de terminación de la transcripción, un marcador que denota el fin del gen, el proceso se acaba y se tiene un mARN nuevo completo.

2.1.4. Transcripción en eucariotas

Para una célula eucariota la transcripción ocupa un paso intermedio en que no todo el contenido de un gen se convierte en mARN, las porciones del gen que se convierten en mARN se llaman exones y las que no se transcriben se llaman intrones, ambos tipos son de tamaño variable.

1. El *ARN polimerasa* (pARN) temporalmente separa la doble hélice de ADN.

2. El pARN localiza el punto de inicio de la transcripción, un marcador que denota el inicio de un gen. Este paso se detallará más adelante.

3. El pARN produce un *pre-ARN mensajero* (pre-mARN) que contiene intrones y exones siguiendo las mismas reglas que el paso 3 de la transcripción en procariotas.

4. Luego añade al extremo 5' (el inicio) del pre-mARN un casquete, que es un nucleótido modificado de G y al extremo 3' (el final) una cola poli-A, es decir, un tramo de ARN cuyas bases son todas A. El casquete sirve para el ribosoma en el siguiente paso reconozca el pre-mARN y la cola para resistencia a la degradación y así poder sintetizar más proteína.

5. A continuación con la ayuda de espliceosomas, se remueven los intrones que normalmente empiezan con GU (GT en el gen) y terminan con AG. Los espliceosomas son sistemas de moléculas ("máquinas moleculares") formados por unas diez proteínas y una pequeña molécula de ARN, que pueden reconocen las señales del inicio y fin de un intrón y extraerlo.

6. Una vez que los intrones fueron extraídos se produce el mARN solo con exones, que se transporta fuera del núcleo e inicia el proceso de traducción.

Finalmente es de notar que en diferentes condiciones se suelen extraer diferentes intrones, produciendo diferentes mARN, esto para que un mismo gen pueda producir diferentes proteínas.

En la figura 4 se puede ver una representación gráfica de la transcripción en una eucariota.

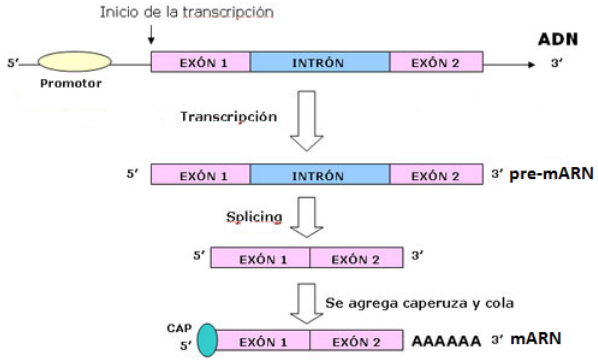

Figura 4: Transcripción en una eucariota

2.1.5. Traducción

La traducción, también llamada síntesis de proteínas, consiste en crear una proteína a partir de un mARN. Este proceso es manejado por un organelo de la célula llamado ribosoma a donde se dirige el mARN e involucra traducir el código genético en el mARN en su correspondiente cadena de aminoácidos que se pliega en una proteína, cambia la forma de cadena por una forma tridimensional con dobleces y nuevos enlaces internos.

Los aminoácidos son moléculas que se pueden enlazar formando cadenas llamadas *péptidos* y *polipéptidos*. Un polipéptido se llama proteína cuando supera cierto tamaño, tiene una estructura tridimensional estable y una función específica.

Los nucleótidos del mARN forman grupos de tres llamados **codones** que entran al ribosoma secuencialmente, por cada codón en el ribosoma un *ARN de transferencia* (tARN) se encarga de traer el aminoácido correspondiente (de acuerdo a un diccionario llamado **código genético**), y el ribosoma lo conecta a una secuencia de aminoácidos que se va formando a partir de los codones del mARN. Cuando ha traducido todo el mARN el ribosoma suelta la nueva proteína (Narayanan, 2005).

2.1.6. El código genético

Existen 20 aminoácidos distintos para formar las proteínas y así también 20 tipos de tARN que traen los aminoácidos al ribosoma.

¿Cómo un alfabeto de cuatro letras en el ADN o en el ARN crea veinte distintos aminoácidos para la creación de las proteínas? Resulta que el código genético usa un esquema de tercias, combinaciones de tres nucleótidos llamados codones, cada uno se codifica en un aminoácido. Pero si son cadenas de 3 nucleótidos y en cada posición se permiten los 4 tipos de nucleótidos: $4^3 = 64 > 20$, la codificación permite teóricamente 64 aminoácidos pero al producirse sólo 20 distintos aminoácidos donde cada uno es traducido de más de un tipo de codon, se agrega robustez a la codificación pese a posibles mutaciones.

En el cuadro 1 puede verse en primer lugar el codón, la abreviación del aminoácido y la letra con la que se le almacena y trabaja en la bioinformática.

2a. base					
1a. base	U	C	A	G	3a. base
U	UUU Phe [F]	UCU Ser [S]	UAU Tyr [Y]	UGU Cys [C]	U
	UUC Phe [F]	UCC Ser [S]	UAC Tyr [Y]	UGC Cys [C]	C
	UUA Leu [L]	UCA Ser [S]	UAA Stop	UGA Stop	A
	UUG Leu [L]	UCG Ser [S]	UAG Stop	UGG Trp [W]	G
C	CUU Leu [L]	CCU Pro [P]	CAU His [H]	CGU Arg [R]	U
	CUC Leu [L]	CCC Pro [P]	CAC His [H]	CGC Arg [R]	C
	CUA Leu [L]	CCA Pro [P]	CAA Gln [Q]	CGA Arg [R]	A
	CUG Leu [L]	CCG Pro [P]	CAG Gln [Q]	CGG Arg [R]	G
A	AUU Ile [I]	ACU Thr [T]	AAU Asn [N]	AGU Ser [S]	U
	AUC Ile [I]	ACC Thr [T]	AAC Asn [N]	AGC Ser [S]	C
	AUA Ile [I]	ACA Thr [T]	AAA Lys [K]	AGA Arg [R]	A
	AUG Met [M]	ACG Thr [T]	AAG Lys [K]	AGG Arg [R]	G
G	GUU Val [V]	GCU Ala [A]	GAU Asp [D]	GGU Gly [G]	U
	GUC Val [V]	GCC Ala [A]	GAC Asp [D]	GGC Gly [G]	C
	GUA Val [V]	GCA Ala [A]	GAA Glu [E]	GGA Gly [G]	A
	GUG Val [V]	GCG Ala [A]	GAG Glu [E]	GGG Gly [G]	G

Cuadro 1: Tabla de traducción de codones a aminoácidos

Es necesario hacer algunas observaciones acerca del código genético: AUG codifica en M, y también es el marcador de inicio de traducción, todas las secuencias de aminoácidos empiezan con este **codón de inicio**. Por otro lado UAA, UAG y UGA son los **codones de paro** pues cuando aparece alguno el ribosoma suelta el mARN y el proceso de traducción termina.

2.2. Explicación biológica del problema: el origen de los motifs

El **ARN polimerasa** (pARN) se encarga de iniciar el proceso de transcripción de un gen, y lo hace con la ayuda de unas proteínas llamadas **factores de transcripción** (TF). Después que el pARN ha separado la doble hélice, los TF se acercan a la cadena de ADN y se enlazan a segmentos pequeños de la cadena llamados **sitios de enlace de factor de transcripción** (TFBS). Cuando el pARN llega, encuentra los TF y toma el segmento del ADN posterior a este marcador, justo donde empiezan los genes para transcribir. Esto significa que los TF son los que encuentran indirectamente dónde están los genes que se necesitan en cada momento. En la figura 5 se puede ver este proceso de transcripción como sucede en las procariotas.

Entonces los TF son las responsables, de acuerdo a las necesidades de la célula, de regular todo el tiempo cuales proteínas se sintetizarán y en qué cantidad. Los TF se enlazan a regiones específicas del genoma donde están sus TFBS, conocidas como

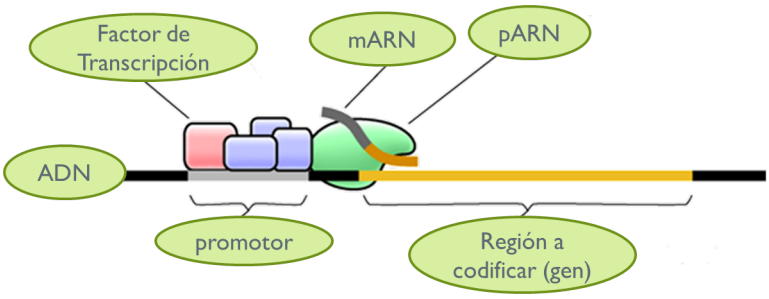

Figura 5: Proceso de transcripción

secuencias cis-regulatorias, de las que se tienen dos tipos:

- Promotores: regiones del ADN anteriores al inicio de genes. El promotor contiene varios TFBS que interactúan directamente con el pARN, el resto del promotor no tiene otra función que mantener a cierta distancia los TFBS.
- Potenciadores/represores: también llamados módulos cis-regulatorios (CRM). Secuencias que potencian o reprimen la transcripción de uno o varios genes a cierta distancia. Un CRM puede tener 100-1000 bp de longitud y se distinguen por contener muchos más TFBS juntos que un promotor, incluso algunos TFBS repetidos.

Los científicos han descubierto que los TFBS de un mismo TF contienen secuencias similares de ADN de 5 a 20 pares de bases (bp). Estos patrones en el ADN se conocen como **motifs**. A través de métodos experimentales es posible extraer los promotores, ya sea de genes homólogos de otras especies, a partir de genes en el mismo ADN con el mismo TF y por una técnica llamada *inmunoprecipitación de cromatina*. El problema consiste en encontrar la parte importante del promotor que llamamos motif (Sung, 2010).

2.2.1. El modelo de motif

Suponga que se tiene un conjunto de varios TFBS para un TF particular. Estas secuencias de ADN comparten un patrón al que llamamos **motif del factor de transcripción** que aparece en todos los TFBS. Aunque cada posición del motif suele tener preferencia por un nucleótido (a veces por más de uno), el motif no es rígido y permite variaciones en los nucleótidos que cumplen el patrón.

Cuando se tiene el conjunto de TFBS alineados por la posición de inicio de su motif, se calcula el profile o **matriz perfil** del motif, que sólo consiste en contar cuantas veces aparece cada nucleótido (A, C, G, T) en cada posición del motif. Luego de eso se tienen dos formas de representar el motif: mediante consenso de motif y por matriz de pesos posicionales (PWM).

Un **consenso de motif** contiene las características más comunes del patrón. Cuando se tiene la matriz perfil el consenso de motif representa cada posición con su tipo de nucleótido más frecuente, si se comparte la máxima frecuencia por varios tipos de nucleótidos se sigue la notación del código IUPAC-IUB (Internation Union of Biochemistry). Así el TF tiene alta afinidad con las secuencias que encajan más con el consenso y baja afinidad con las secuencias más diferentes al consenso. Se puede ver un ejemplo en la figura 6.

Nucleótidos	Código
A	A
C	C
G	G
T	T
C, G, T	B
A, G, T	D
A, C, T	H
C, G, T	V

Nucleótidos	Código
A, C	M
A, G	R
A, T	W
C, G	S
C, T	Y
G, T	K
A, C, G, T	N

Cuadro 2: Código IUPAC-IUB

Otra forma numérica de representar un motif es mediante una **matriz de pesos posicionales** (PWM) que muestra el nivel de ambigüedad de cada posición del motif. La PWM muestra la frecuencia de cada uno de los cuatro nucleótidos en cada una de las posiciones del motif. La mayoría de los algoritmos suelen cambiar la probabilidad de los nucleótidos que no aparecieron en una posición (frecuencia 0) a una pequeña llamada **pseudoconteo** debido a que no se puede estar seguro de que un tipo de nucleótido nunca aparecerá y por supuesto cambiar las demás probabilidad para que la suma de las frecuencias en cada posición sea igual a 1. Se ha trabajado con el cálculo de los pseudoconteos al definirlos como representantes de las probabilidades de cada base en el resto de la secuencia que no pertenece al motif o del resto de la secuencia que no aparece en la base de datos al mismo tiempo que cumplen la función de no influir mucho en las probabilidades a las que se adicionan, esta justifición provoca que su cálculo no obtenga números muy pequeños. Por esto mismo se encontró una formula útil (Lawrence et al., 1993) para su cálculo que utiliza la frecuencia o probabilidad de la base en el total de secuencias:

$$Pseudoconteo_A = P(A) \cdot \sqrt{NumSecuencias}$$

Una optimización a la PWM llamada **matriz de puntuaciones específicas de posiciones** (PSSM) utiliza además de la frecuencia del nucleótido en la posición, su frecuencia en las secuencias que se trabajaron para así reducir el peso de los componentes que pudieron haber sido producto del azar por ser muy comunes también fuera del motif. La nueva formula para la puntuación de cada nucleótido en cada posición es la siguiente: $S_{a,i} = log_2 \frac{P(a,i)}{P(a)}$ donde el numerador se extrae de la PWM y el denominador

es la frecuencia de ese nucleótido en el total de las secuencias dadas. Para calcular la probabilidad de una secuencia de pertenecer al motif se suman las puntuaciones en lugar de la multiplicación que se aplica en una PWM.

Una PWM o una PSSM también pueden visualizarse mediante una gráfica llamada **logo de secuencia** donde la altura de cada caracter indica la importancia relativa de ese nucleótido en esa posición. Esta representación es más fácil de manejar para el ojo humano, por lo que siempre se guarda en las bases de datos de motifs. Un ejemplo de las matrices y el logo de secuencia puede verse en la figura 7.

1	T	T	G	C	C	A	A	T
2	T	G	G	T	C	T	C	T
3	T	T	G	T	G	A	A	A
4	T	T	G	C	C	C	A	T
5	T	T	G	T	C	C	A	T
6	T	T	G	C	C	T	A	T
7	T	T	G	T	T	C	A	G
8	T	T	G	C	G	A	A	T
9	T	G	G	A	C	T	C	T
	T	T	G	Y	C	H	A	T

Figura 6: Sección de cada secuencia donde se encontró el motif y su consenso.

Ambos sistemas de representación se siguen usando por sus ventajas. El consenso tiene la principal ventaja de ser fácil de visualizar por los humanos y de buscar mediante expresiones regulares por las computadoras. La PWM entrega una mejor descripción de la situación pues es más frecuente que una secuencia "falle" en algunas posiciones a que case en todas o en ninguna y una PWM sigue considerando estas "fallas" calculándole su probabilidad en vez de descartarla; sin embargo una PWM es más elaborada que un consenso y más compleja de buscar (Bailey, 2007).

Por último se necesita una representación numérica del motif, un cuantificador que indique cuánto se distingue como patrón respecto al resto del contenido, es decir, un medidor de su contenido de información, para esto se usa una fórmula llamada **Motif Score** (Schneider et al., 1986).

$$MotifScore(P, PWM) = \sum_{i=1}^{k} \sum_{a \in \{A,C,G,T\}} P_{a,i} \cdot log_2 \frac{PWM_{a,i}}{P_{a,i}}$$

A esta formula se entregan la PWM de las apariciones del motif y la probabilidad (frecuencia) de cada caracter. Para cada celda de la matriz, se divide la probabilidad que contiene entre la frecuencia del mismo caracter (nucleótido), se le calcula el logaritmo base 2 y se multiplica por la frecuencia del mismo caracter; finalmente es una sumatoria de todos estos cálculos que se hacen por cada celda.

(a)

A:	0.00	0.00	0.00	1.00	0.00	3.00	7.00	1.00
C:	0.00	0.00	0.00	4.00	6.00	3.00	2.00	0.00
G:	0.00	2.00	9.00	0.00	2.00	0.00	0.00	1.00
T:	9.00	7.00	0.00	4.00	1.00	3.00	0.00	7.00

(b)

A:	0.00	0.00	0.00	0.11	0.00	0.33	0.78	0.11
C:	0.00	0.00	0.00	0.44	0.67	0.33	0.22	0.00
G:	0.00	0.22	1.00	0.00	0.22	0.00	0.00	0.11
T:	1.00	0.78	0.00	0.44	0.11	0.33	0.00	0.78

(c)

A:	0.12	0.12	0.12	0.18	0.12	0.29	0.53	0.18
C:	0.12	0.12	0.12	0.35	0.47	0.29	0.24	0.12
G:	0.12	0.24	0.65	0.12	0.24	0.12	0.12	0.18
T:	0.65	0.53	0.12	0.35	0.18	0.29	0.12	0.53

(d)

A:	-1.09	-1.09	-1.09	-0.50	-1.09	**0.23**	**1.08**	-0.50
C:	-1.09	-1.09	-1.09	**0.50**	**0.91**	**0.23**	-0.09	-1.09
G:	-1.09	-0.09	**1.37**	-1.09	-0.09	-1.09	-1.09	-0.50
T:	**1.37**	**1.08**	-1.09	**0.50**	-0.50	**0.23**	-1.09	**1.08**

(e)

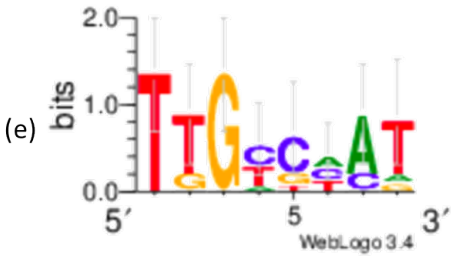

Figura 7: Matrices (a) perfil, (b) PWM, (c) PWM con pseudoconteos, (d) PSSM con los valores máximos marcados y (e) logo de secuencia del motif TTGYCHAT de la figura anterior.

2.2.2. Formalización del problema

Formalmente se simplifica y define de la siguiente manera.

Entrada: Un conjunto de secuencias de ADN posiblemente enlazadas al mismo TF obtenidas mediante técnicas experimentales.

Salida: Buscamos un solo patrón común llamado motif, representado mediante una secuencia consensus y una matriz PWM.

El planteamiento supone que es sólo uno el motif a encontrar, aunque algunas secuencias pueden tener más de un motif, uno más importante y otros más débiles, se entiende que al extraerse por experimentación resaltaron por contener un mismo motif común que es el que se va a buscar, sin importar los demás patrones ocultos que ignoraremos. A pesar de esto puede haber secuencias que no contenían el motif y fueron extraídas por error.

Existen tres principales métodos computacionales para obtener motifs. El primero consiste en buscar motifs conocidos consultables, el segundo consiste en buscar patrones frecuentes, y así descubrir nuevos motifs; y el último método es encontrarlos con la ayuda de información adicional de su entorno (Sung, 2010).

En esta tesis nos dedicaremos a un segundo método donde de nuevo ignoraremos otros patrones que aparezcan para buscar sólo un motif, y además con las ventajas de que también incluirá como entrada el tamaño del motif y que estamos seguros que se encuentra en todas las secuencias, modificando la definición del problema a la llamada **búsqueda del motif implantado**:

Entrada: Un conjunto de secuencias de ADN enlazadas al mismo TF y el tamaño del motif.

Salida: Encontrar el motif representado mediante una secuencia consensus y una matriz PWM.

Esta formalización del problema descarta algunas características del motif que fueron descritas al inicio pero por simplificación no serán tomadas en cuenta en el diseño del algoritmo. Como ya dijimos se conoce el tamaño del motif a buscar, todas las secuencias que se reciben de entrada contienen al motif sólo una vez, no se tomará cuenta la dependencia entre posiciones del motif ni los motifs discontinuos, esos motifs que aparentemente varían de tamaño porque en algunas apariciones tienen una región intermedia que no pertenece al motif, la cual tiene la misma distribución de bases que el resto que no pertenece al motif, y cuyo tamaño no está regulado, haciendo que las últimas posiciones del motif en cualquier aparición estén a distancias diferentes, desde el tamaño del motif establecido hasta muchos caracteres de distancia debido a la región intermedia con un tamaño más grande que el del motif.

Para discriminar entre las distintas combinaciones de subcadenas que obtiene el algoritmo y calificar cuál es el motif correcto se usan dos medidores comunmente: la suma del número de mutaciones debe ser la mínima posible, pues el motif debe tener pocas mutaciones en sus apariciones para seguir siendo el patrón más frecuente; y el más usado, el contenido de información, medido con el Motif Score, debe ser el máximo pues es quien determina cuánto se distinguen las apariciones del resto de las secuencias.

2.3. Bases de datos biológicas

Basadas en su contenido, las bases de datos biológicas se pueden dividir en tres categorías (Xiong, 2006):

2.3.1. Bases de datos primarias

Contienen los datos biológicos originales. Constan de archivos de secuencias llanas y datos de estructuras subidas por la comunidad científica. Existen tres principales bases de datos públicas que almacenan secuencias de ADN producidas por los investigadores a nivel mundial, los participantes de la Colaboración Internacional de Bases de datos de Secuencias de Nucleótidos: GenBank del *National Center for Biotechnology Information*

(NCBI) en Estados Unidos, la base de datos del *European Molecular Biology Laboratory* (EMBL) en Europa y el *DNA Data Bank of Japan* (DDBJ) en Japón.

Las tres bases de datos tienen su contenido disponible gratuitamente en Internet y colaboran en conjunto intercambiando sus datos a diario para que la conexión a cualquiera produzca los mismos resultados, lo más actuales posibles.

Las direcciones URL de éstas y las bases de datos en las siguientes secciones aparecen en el cuadro 3.

2.3.2. Bases de datos secundarias

Contienen información procesada y limpia, basada en la información en las bases de datos primarias.

Una de las más importantes es TrEMBL, una base de datos de secuencias de ADN almacenada en la base de datos de EMBL. Cada secuencia contiene anotaciones cuidadosamente revisadas por expertos, verificada con nueva literatura científica e ingresada cuando cumplen un nivel de calidad.

En un esfuerzo por combinar TrEMBL con dos de las mejores bases de datos secundarias de proteínas (SWISS-PROT y PIR) se condujo a la creación de la base de datos UniProt que cubre más información que cualquiera de las tres y mantiene su alto nivel de calidad.

Debido a que suele haber problemas de redundancia en las bases de datos primarias; información duplicada cuando una secuencia contiene a otra, dos secuencias se traslapan o un mal manejo de secuencias similares; el NCBI ha creado una base de datos no redundante llamada RefSeq en que todas las posibles redundancias de una secuencia de ADN o ARN se convierten en un solo registro además de enlazarse con registros de proteínas derivadas de la secuencia que se consulta.

También con la situación de que los registros en las bases de datos primarias no se pueden actualizar o corregir sin el permiso de la fuente original, el *European Bioinformatics Institute* (EBI) creó Genome Reviews, una base de datos que contiene versiones limpias de los datos en DDBJ/EMBL/GenBank con información adicional de UniProt y otras fuentes. Su principal objetivo es estandarizar las anotaciones, incluso eliminando las incompletas o carentes de bases. Esta base de datos se sincroniza constantemente con UniProt (Baxenavis, 2005).

2.3.3. Bases de datos especializadas

Son aquellas que se enfocan en un particular interés de la investigación biológica. Tienen su propia organización y formato útil para los trabajos en su campo y se suelen actualizar más pronto que las bases de datos primarias porque son propias de una más pequeña comunidad.

Ejemplos de estas bases de datos son Flybase, WormBase, AceDB que solo contienen genoma de una especie o varias especies cercanas.

Base de datos	Descripción	Enlace
DDBJ	BD primaria de Japón	www.ddbj.nig.ac.jp
EMBL	BD primaria de Europa	www.ebi.ac.uk/services
ENSEMBL	Sistema revisado de EBI	www.ensembl.org
Entrez	Sistema de búsqueda de NCBI	www.ncbi.nlm.nih.gov/sites/gquery
GenBank	BD primaria de Estados Unidos	www.ncbi.nlm.nih.gov/genbank/
JASPAR	BD especializada en TF	jaspar.genereg.net
RefSeq	BD no redundante de NCBI	www.ncbi.nlm.nih.gov/refseq/
Regulon DB	BD especializada del CCG-UNAM	regulondb.ccg.unam.mx
TRANSFAC	BD especializada en TF	www.gene-regulation.com
TrEMBL	BD revisada de EMBL	www.uniprot.org/help/uniprotkb

Cuadro 3: Lista de las bases de datos con los enlaces a sus páginas web.

En este tipo de bases de datos destacamos las bases de datos de TF motifs conocidos como son JASPAR (Sandelin et al., 2004) y TRANSFAC (Matys et al., 2003). El primero es una base de datos libre que contiene profiles revisados, no redundantes derivados de experimentos con TFBS en eucariotas. El segundo tiene una versión no actualizada gratuita y una privada, ambas trabajan con TF en eucariotas, almacenando sus TFBS revisados en experimentación, las PWM de estos y los genes que regulan (Sung, 2010).

Por último está RegulonDB (Salgado et al., 2013), una base de datos mantenida por el Centro de Ciencias Genómicas (CCG) de la UNAM que almacena los datos del sistema regulatorio compuesto por los TF y sus TFBS en el organismo *Escherichia coli K-12*.

2.3.4. Sistemas de recolección de datos biológicos

Son un tipo de sistemas de consulta amigables con el usuario que acceden a múltiples bases de datos para entregar información más completa y al mismo tiempo fácil de entender. Los casos más populares sen Entrez y Sequence Retrieval System (SRS).

Entrez es desarrollado y mantenido por el NCBI, permite búsquedas por texto de una amplia variedad de datos. Una característica de Entrez es su habilidad para integrar información proveniente de todas las bases de datos del NCBI. Por ejemplo en la entrada de una secuencia de nucleótidos uno puede encontrar enlaces a las proteínas traducidas a partir de la secuencia, su localización en mapas de genomas o la literatura de PubMed relacionada.

SRS es un sistema de recolección mantenido por el EBI, no está integrado como Entrez pero permite consultar múltiples bases de datos a la vez y ofrece accesos directos a ciertas aplicaciones de análisis de secuencias.

El proyecto ENSEMBL es un proyecto conjunto del EBI y el *Wellcome Trust Sanger Institute* para desarrollar un sistema que produce y mantiene anotaciones automáticas en genomas de células eucariotas de seleccionadas especies. Su principal buscador, muy completo, entrega información de los genes, transcripciones y proteínas de cada genoma

mediante diferentes niveles de detalle.

2.3.5. Formatos de secuencias

Los registros de las bases de datos biológicas, la mayoría secuencias de ADN, ARN y aminoácidos, se suelen guardar en archivos de texto plano con un formato establecido por la base de datos.

El formato para las secuencias de ADN o ARN en el GenBank es un archivo de texto plano que contiene información detallada junto con la secuencia: una cabecera con la definición del origen de la secuencia, el autor, el organismo del que proviene, el artículo donde fue publicado, una sección de características acerca del gen origen, regiones de la secuencia reportadas como significativas, el tamaño de la secuencia y otros datos encontrados; para el caso específico de las cadenas de ADN contiene los límites de los genes interiores, y la secuencia de la proteína en que se convierte. Al final del archivo en una tercera sección se encuentra la secuencia en sí empezando con la etiqueta ORIGIN y finalizando con una etiqueta "doble diagonal" (//).

Existen muchos otros formatos pero uno de los más simples y populares es el formato FASTA, un archivo de texto plano que contiene una línea inicial que empieza con > seguido del nombre de la secuencia y algunas veces información adicional en la misma línea. La secuencia inicia en la segunda línea y las siguientes, cada una limitada a 60 u 80 caracteres de ancho, que tampoco es obligatorio de obedecer. Un mismo archivo puede contener varias secuencias en formato FASTA usando la etiqueta que empieza con > como indicador de que la secuencia anterior terminó e inicia una nueva secuencia. Su desventaja es la carencia de anotaciones pero es la más fácil de trabajar (Xiong, 2006).

2.4. Algoritmos actuales

A continuación se hablará sobre algunos de los principales algoritmos que han surgido para resolver el problema de la búsqueda de motifs.

2.4.1. Distancia Hamming

Al trabajar con los algoritmos de búsqueda de motifs, suele encontrarse la necesidad de una función de distancia entre cadenas (subcadenas de secuencias) del mismo tamaño para comparar entre cadenas a elegir, para esto la mayoría de los algoritmos recurren a la distancia Hamming o evaden la función utilizando funciones de probabilidad que finalmente sirven para comparar.

La distancia Hamming es una función muy usada en teoría de la información para medir el mínimo de sustituciones que se requiere para convertir una cadena en la otra, o el mínimo de errores que transformaron una cadena a partir de la otra.

La función es la siguiente

$$d(cadena1, cadena2) = \sum_{i=1}^{k} d(cadena1_i, cadena2_i)$$

$$d(x, y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$$

requiere que las cadenas a comparar sean del mismo tamaño k y su rango va de 0 a ∞.

Algunas modificiones incluyen disminuir el rango a [0,1] diviendo el resultado entre k; permitir la distancia entre cadenas de distinto tamaño sumando la distancia Hamming de la cadena más corta con cada subcadena del mismo tamaño en la cadena más larga o cambiar a la distancia Levenshtein que incluye sustituciones, inserciones y supresiones.

2.4.2. CONSENSUS

Es una formalización de unas de las primeras formas que se trabajó el problema. Con este algoritmo se formalizaron las matrices de conteo y PWM, así como el uso de la **información de contenido**.

El algoritmo primero solicita el tamaño del patrón W o puede calcularlo si se le da un sesgo (*bias*) que se usa en la información de contenido.

1. Calcula el PWM de cada segmento de tamaño W de la 1a. secuencia.

2. Toma los segmentos de tamaño W de la siguiente secuencia, los combina con cada una de las PWM para obtener nuevas PWM.

3. Calcula la información de contenido de cada PWM y filtra las mejores puntuaciones.

4. Alternar entre los dos pasos anteriores hasta haber pasado por todas las secuencias. Así las PWM contienen un segmento de cada secuencia.

5. Finalmente las PWM se ordenan de acuerdo a su puntuación para dar muchos resultados probables o sólo se conserva la PWM con la mejor puntuación (Hertz and Stormo, 1999).

2.4.3. Gibbs Sampler

Uno de los primeros algoritmos que surgieron, se basa en que el patrón común, el mejor, es aquel más probable y lo busca localizando la alineación de las secuencias que maximice la proporción entre la probabilidad del patrón y la probabilidad del resto (**probabilidad de fondo**) en todas las secuencias.

La metodología de este algoritmo y las modificaciones al mismo conducen al algoritmo Markov chain MonteCarlo (MCMC), un algoritmo aleatorizado.

El algoritmo requiere que se le dé el tamaño W del patrón a buscar e inicia eligiendo al azar una posición de inicio en cada secuencia y luego itera sobre dos pasos hasta converger:

1. **Paso predictivo**: Se elige una secuencia al azar o en cierto orden y se calculan para las demás secuencias la descripción del patrón por medio de una matriz de frecuencias $q_{i,j}$ y la frecuencia de fondo p_j que son las frecuencias de las cuatro bases contando todas las posiciones que no pertenecen a las apariciones del motif. Así estamos suponiendo que el motif se encuentra en esas posiciones, calculamos sus datos y vamos a corregir la aparición en la secuencia elegida.

2. **Paso de muestreo**: Para cada segmento s de tamaño W en la secuencia seleccionada se calcula la probabilidad Q_x de generar s usando la tabla de $q_{i,j}$ anteriores y la probabilidad P_x de generarlo con el resto mediante la probabilidad de fondo, esto equivale a calcular la probabilidad de que s sea una aparición del motif. Se asigna un peso $A_x = Q_x/P_x$ a cada segmento s y estos se usan como distribución de probabilidad para una variable aleatoria que se usa para elegir uno de los segmentos s como la aparición del motif.

Un defecto del algoritmo es la tendencia a caer en máximos locales que se puede evitar insertando un paso intermedio, cada m iteraciones, se compara la posición que se analiza con unas n posiciones a la derecha e izquierda para calcular si alguna es un mejor patrón. Otra desventaja es que necesita el largo del patrón a trabajar y solo buscará patrones de ese tamaño.

Entre sus ventajas se encuentra la rapidez ya que en lugar de hacer muchas operaciones deja el encaminamiento al azar, también evita algunos máximos locales cuando en lugar de irse directamente a la aparición más probable deja espacio para elegir otra ruta y encontrar una convergencia mejor, a menos que sea el máximo global e inevitablemente volverá al camino a éste. También el algoritmo puede modificarse para buscar varios patrones simultáneamente tomándolo como un solo motif separado por porciones poco frecuentes (Lawrence et al., 1993).

2.4.4. MEME

Es un algoritmo de los más elementales en este problema que sigue siendo utilizado por su metodología distinta a la mayoría. Se basa en otro algoritmo básico llamado **Maximización de expectación**(EM), un algoritmo determinista.

Antes de explicar el algoritmo es necesario plantear que la maximización de expectación (EM) para este caso se basa en dos valores: la frecuencia de cada caracter en cada posición del patrón y la probabilidad de cada subcadena de las secuencia de ser la aparición del patrón. Entonces el método itera usando las frecuencias para calcular las probabilidades y usando las probabilidades para calcular las frecuencias hasta la convergencia; esto se explica mejor en la descripción completa del algoritmo a continuación.

Ahora, el algoritmo requiere el tamaño W del patrón a buscar, un número de patrones a buscar y un mínimo de apariciones del patrón necesarias. Luego se ejecuta un ciclo exterior por cada motif a buscar, y un ciclo interior donde se desarrolla el núcleo del algoritmo.

1. Se inicia eligiendo al azar una posición de inicio en cada secuencia y se calculan las frecuencias en cada una de las W posiciones, es decir, el PWM.

2. Expectación: Se calcula la probabilidad de cada segmento de tamaño W en cada secuencia de ser producido por el PWM.

3. Maximización: Se calcula la probabilidad de cada caracter en cada una de las W posiciones de la siguiente manera: para un caracter b en la posición n se suman las probabilidades de los segmentos que contengan el caracter b en la posición n. Con todo esto se obtiene una PWM.

4. Se itera entre los pasos de Expectación y Maximización hasta que las probabilidades de la PWM varíen poco y empiecen a converger.

Finalmente se ocultan las apariciones del patrón del conjunto de secuencias para calcular el siguiente patrón motif del ciclo exterior.

Sus ventajas particulares son que converge más rápido por no ser aleatorizado, permite evaluar el mismo motif con tamaños distintos, permite que el motif aparezca más de una vez en algunas secuencias o ninguna en otras secuencias sin afectar en las probabilidades y por último permite encontrar más de un motif diferente (Bailey and Elkan, 1995).

2.4.5. Winnower

Uno de los primeros algoritmos que definieron el problema de los motifs como **planted (l,d) motif problem** donde se busca un patrón que aparece en distintas posiciones de un conjunto de secuencias y con distintas mutaciones, l indica el tamaño del patrón y d el número de mutaciones permitidas a una aparición.

El algoritmo ataca el problema desde una perspectiva combinatoria, se centra en las mutaciones pequeñas que no crecerán a ser candidados a un patrón para podarlas hasta que sólo quede el patrón. Esto se desarrolla en una estructura de grafo multipartito donde los nodos son los segmentos que pueden ser apariciones del patrón y se conectan entre sí cuando presentan similitud suficiente basada en d cambiando a un problema de grafos donde se busca el o los nodos de mayor grado podando el resto.

1. Para cada posición en cada secuencia se crea un nodo con el segmento de tamaño l que empieza en esa posición.

2. Se conectan los nodos cuya distancia no exceda 2d, por lo general se usa la distancia Hamming.

3. Se podan aquellos nodos que no se enlazaron a nodos de todas las secuencias.

4. De cada nodo se crea otro con tamaño de un caracter más y se regresa al paso 2.

5. Se detiene el algoritmo hasta que sólo permanece el número de nodos internos igual al de patrones solicitados.

Una ventaja importante es que puede encontrar múltiples patrones frecuentes diferentes con su estructura. El algoritmo permite ejecutar podas más fuertes para encontrar más rápido el patrón (Pevzner and Sze, 2000).

2.4.6. SP-Star

Basado en el algoritmo Winnower, los autores crearon otro algoritmo con la misma metodología que disminuye los recursos computacionales a solicitar al cambiar el grafo multipartito. Además de refinar los patrones candidatos, es decir, una vez que se encuentran se les aplica un algoritmo de optimización combinatoria.

El algoritmo toma su nombre de un proceso intermedio en el cual se define un calificador para medir los patrones candidatos y elegir cuáles son más probables a convertirse en el motif correcto. El SP-Score mide la distancia entre las subcadenas que conforman un patrón.

$$SPScore(S[w_i, ..., w_n]) = \sum_{i=1, j=i+1}^{j=n} d(w_i, w_j)$$

Dado que el motif correcto contiene la menor distancia entre el patrón y cada subcadena, entonces también contiene las menores distancias entre subcadenas y como sería mucho trabajo medir todas las distancias entre todos los pares de subcadenas, nos basta con medir algunas de éstas y decir que el patrón correcto se encuentra entre los que contienen una menor suma de esas distancias que escogimos, de tal forma que usamos esta heurística.

El algoritmo solicita las secuencias donde se busca el patrón, el tamaño W del patrón a buscar y un número N de iteraciones máximo.

1. Para cada subcadena de tamaño W en cada secuencia:

   a) Encontrar la subcadena de tamaño W más parecida en cada una de las otras secuencias.

   b) Calcular el SP-Score del nuevo conjunto de subcadenas.

2. Realizar una poda conservando sólo los conjuntos de subcadenas con mayor SP-Score.

3. Para cada uno de los conjuntos filtrados:

   a) Calcular la PWM de las subcadenas.

   b) Encontrar la subcadena en cada secuencia más cercana a la PWM.

   c) Repetir los dos pasos anteriores hasta que el conjunto de subcadenas no cambie o hasta las N iteraciones.

4. Conservar sólo el conjunto de subconjunto de subcadenas con la mayor cantidad de información o Motif Score como el motif encontrado (Pevzner and Sze, 2000).

2.4.7. Motif Sampler

Es una mejora al algoritmo Gibbs sampling en dos principales modificaciones, se usa una distribución de probabilidad para estimar el número de copias de un patrón en cada secuencia y para la probabilidad de fondo usa un modelo probabilístico de mayor orden.

Para encontrar más de un motif se puede ejecutar el algoritmo varias veces y en cada ejecución ocultar las posiciones de los motifs anteriores en las secuencias.

1. El modelo de fondo de orden k se obtiene de las secuencias, se cuentan las frecuencias de los posibles segmentos de tamaño k y basado en un modelo de Markov y las convierte en una distribución de probabilidad con la que se puede trabajar.

2. Para cada segmento de tamaño W (tamaño del patrón a buscar) se le calcula la probabilidad de ser obtenido del modelo de fondo.

3. Se elige al azar una posición de inicio en cada secuencia y con éstas se construye un modelo del patrón, básicamente su PWM.

4. Se usa el modelo del patrón y el modelo de fondo para calcular la distribución de probabilidad sobre las posibles posiciones del patrón en cada secuencia. Esto quiere decir, la probabilidad de cada segmento de tamaño W de ser una aparición del patrón.

5. Del mismo modo que Gibbs Sampler se usa la distribución de probabilidad para obtener las nuevas posiciones de inicio en cada secuencia por medio de una variable aleatoria. Y con éstas contruir un nuevo modelo del patrón.

6. Se itera sobre los dos pasos anteriores hasta la convergencia, cuando el nuevo modelo del patrón no cambie respecto al anterior, o los cambios empiecen a ser mínimos. Al terminar el ciclo el modelo del patrón representa al motif encontrado.

El resultado final tiene además tres medidas: puntuación de consenso que indica la conservación del motif, la información de contenido dice cuanto difiere el motif de la distribución normal y la log-probabilidad que indica la calidad del motif (Thijs et al., 2002).

2.4.8. Weeder

Usa la misma orientación que Winnower, orientada a los patrones a través de su principal característica, un árbol de sufijos donde se almacenan las subcadenas sufijos de las secuencias, siempre almacenando en cada hoja las secuencias a las que pertenece el sufijo. Esta característica fue extraída de un algoritmo anterior (Sagot, 1998) y le añade una poda que permite llegar más rápido al resultado.

El algoritmo requiere además del conjunto de secuencias; el máximo error permitido e como un número en el rango [0,1] que indica cuánto puede mutar un patrón; y opcionalmente el mínimo de secuencias q que deben contener el patrón.

1. Almacenar las secuencias en un árbol de sufijos y empezar en la raíz del árbol.

2. Expandir: Tomar todos los nodos hijos y meterlos a una cola.

3. Explorar: Tomar el siguiente elemento de la cola y explorar caminos alternos de tal forma que con el camino actual y las caminos con mutación en el último caracter a menos de e mutaciones de distancia se tengan al menos q subcadenas, es decir, se encuentre en al menos q secuencias.

4. Validación: Si el camino actual con sus mutaciones está en al menos q secuencias se posiciona en este nodo, si no supera estos requisitos se descarta y se regresa al paso anterior.

5. Iterar entre los tres pasos anteriores hasta vaciar la cola. Al final todos los patrones que permanecen se califican para entregar el mejor como motif, o se entregan ordenados en caso de solicitar varias posibilidades de motif correcto.

Este algoritmo funciona más rápido cuando el error permitido e es más pequeño, porque permite podar más caminos y llegar a menos patrones al final, donde elegirá entre ellos al motif (Pavesi et al., 2001).

2.4.9. Otros algoritmos de búsqueda de motifs

Existen muchos más algoritmos para el problema que atacamos, aunque la mayoría están basados en los que describimos anteriormente, sobre todo en Gibbs Sampler, y consisten en modificaciones y mejoras para hacerlos más complejos y precisos. Por supuesto existen otros algoritmos que han utilizado técnicas muy diferentes como redes neuronales y algoritmos genéticos; pero al final los resultados entre los algoritmos básicos, las modificaciones y las innovaciones no hay una gran diferencia en sus resultados como se esperaría, los resultados pueden verse en artículos de comparación de estos algoritmos (Das and Dai, 2007; Tompa et al., 2005).

2.5. Evaluación de algoritmos

Para evaluar los algoritmos de búsqueda de motifs se utilizan bases de datos de secuencias de las que se conocen los motifs correctos y se calculan unas medidas basadas en:

Verdaderos positivos (TP): número de subcadenas marcadas por el algoritmo que sí contienen un motif.

Verdaderos negativos (TN): número de subcadenas no marcadas por el algoritmo que no contienen un motif.

Falsos positivos (FP): número de subcadenas marcadas por el algoritmo que no contenían un motif.

Falsos negativos (FN): número de subcadenas no marcadas por el algoritmo que sí contenían un motif.

Luego se usan tres niveles de evaluación:

Nivel de nucleótidos: Trabaja con las medidas explicadas previamente pero con un contador más específico y permisivo donde cada posición reconocida parte del motif es un TP. Para calcular la **sensibilidad** del algoritmo, esto es, la proporción de motifs pronosticados correctamente, sobre los motifs mediante $Sn = \frac{TP}{TP+FN}$ y la **especificidad** con $Sp = \frac{TP}{TP+FP}$, que es la proporción de pronosticados correctos. Y para unir ambas medidas se usa el **coeficiente de rendimiento** $PC = \frac{TP}{TP+FP+FN}$ que varía entre 0 y 1 donde una predicción perfecta tiene un valor de 1. También se usa la **medida F** para medir la precisión del algoritmo $F = \frac{2*Sn*Sp}{Sn+Sp}$ y en algunos casos se prefiere medir también el **coeficiente de correlación CC** en vez del PC, y su función es la siguiente $CC = \frac{(TP*TN)-(FN*FP)}{\sqrt{(TP+FN)(TN+FP)(TP+FP)(TN+FN)}}$ (Baxenavis, 2005).

Nivel de sitio de enlace: Requiere menos precisión que el anterior nivel en las medidas básicas, esto es, un TP sucede cuando la subsecuencia marcada tiene al menos un elemento (posición) que pertenece al motif real y así también se recalculan las demás medidas básicas. En este nivel el coeficiente más importante es su propio coeficiente de rendimiento.

Nivel de secuencia y motif: Evalua la capacidad de encontrar al menos un motif correcto. La **tasa de éxito a nivel de secuencia** es igual al número de secuencias con al menos un motif correctamente encontrado entre el número de secuencias $sSr = \frac{N_s}{N}$. La **tasa de éxito a nivel de motif**, una medida de sensibilidad del algoritmo, es igual al número de motif que fueron encontrados correctamente en al menos una secuencia entre el número total de motifs que había $mSr = \frac{Np}{N}$ (Hu et al., 2005).

Capítulo 3: Solución del problema

3.1. Preprocesamiento de los datos

Se decidió trabajar con la base de datos JASPAR (Sandelin et al., 2004) por ser la más orientada exclusivamente hacia los TF y sus TFBS y contener TF de distintas especies, por lo que se tendrá que trabajar con distintos modelos de secuencias y adaptarse a ellos para encontrar el motif; ambas son características que RegulonDB no tiene pues se dedica a una sola especie y varios datos de las celulas de esta especie. También JASPAR cuenta con una muy buena interfaz para corroborar los TFBS, los datos más actuales son de libre acceso y la misma página web permite un modo sencillo de descargar su base de datos; a diferencia de TRANSFAC cuya interfaz causa problemas para quien no conoce lo suficiente de biología molecular y el acceso a los datos más actuales está restringido a contar con una suscripción.

La base de datos de JASPAR se encuentra dividida en dos secciones, una versión del 2010 y una versión actualizada más pesada que la anterior. Ambas versiones se pueden descargar fácilmente de Internet, la versión del 2010 en
http://jaspar.genereg.net/html/DOWNLOAD/ARCHIVE/JASPAR2010/sites/
y la versión actualizada en
http://jaspar.genereg.net/html/DOWNLOAD/sites/
En el cuadro 4 se muestran en una tabla las estadísticas de ambas versiones.

	2010	Actualizada
Tamaño	15.3 Mb	177.5 Mb
Núm. de secuencias	317,059	2,535,488
Núm. de TF	452	593
Secuencia más corta	6 bp	6 bp
Secuencia más larga	7,326 bp	7,326 bp
Motif más corto	5 bp	5 bp
Motif mas largo	26 bp	26 bp

Cuadro 4: Estadísticas de las versiones de la base de datos JASPAR.

Ahora bien, ambas versiones de la base de datos están compuestas de archivos *.sites que contienen todas las secuencias donde aparece un mismo motif, siendo el nombre del archivo el identificador del motif en la base de datos. El modo en que aparecen las secuencias es en formato FASTA (explicado en el apartado 2.3.5)donde todas las secuencias aparecen escritas con minúsculas excepto la subcadena aparición del motif en cada una de ellas, la cual está escrita con mayúsculas. En la figura 8 puede verse un ejemplo de este tipo de archivos.

Como en toda base de datos fue necesario hacer una limpieza de los archivos. Fueron removidos los archivos que con caracteres faltantes, es decir aquellos donde aparecía "N" entre los caracteres de la secuencia debido a que se sabe que en esa posición va un nucleótido pero se desconoce cuál; y también se removieron los archivos de búsqueda

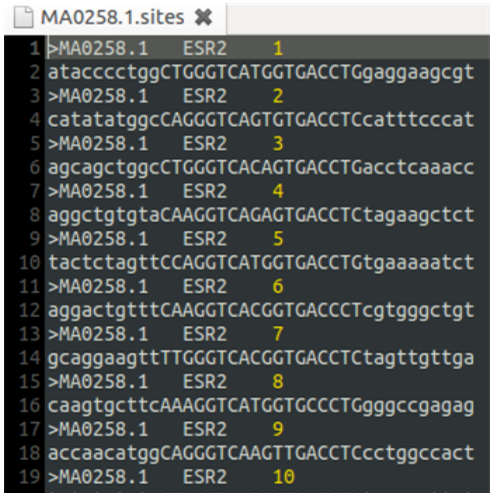

Figura 8: Archivo MA0258.1.sites de JASPAR versión 2010.

trivial, aquellos donde al menos una secuencia tuviera el mismo tamaño que el motif a buscar, ya que esto reduce la complejidad del problema por mucho. En el cuadro 5 se pueden ver las nuevas estadísticas de las versiones validadas de 2010 y actualizada.

	2010	Actualizada
Tamaño	259.7 Kb	161.7 Mb
Núm. de secuencias	6,824	1,112,029
Núm. de TF	236	361
Secuencia más corta	7 bp	7 bp
Secuencia más larga	2,000 bp	2,043 bp
Motif más corto	5 bp	5 bp
Motif mas largo	22 bp	22 bp

Cuadro 5: Estadísticas de las versiones validadas de la base de datos JASPAR.

Finalmente se cambio el formato de los archivos a uno más práctico para ser procesado por un programa de búsqueda de motifs. La extensión se nombró *.seqs y el identificador en la base de datos sigue siendo el nombre del archivo. El formato que se trabajó es el siguiente:

1. La primera línea, a modo de formato FASTA contiene el símbolo > seguido del identificador del motif y el resto de la primera etiqueta en el archivo *.sites, luego separado por una tabulación sigue el número de secuencias en el archivo y separado por una tabulación más el tamaño del motif que contienen.

2. En las líneas siguientes aparecen las secuencias que contienen el motif, una secuencia por línea, con todos los caracteres en mayúsculas.

3. Finalmente, separado por una línea en blanco, se conservan las apariciones del motif en cada una de las secuencias. Después de la línea en blanco sigue la aparición del motif en la 1a. secuencia, en la línea siguiente la aparición en la 2a. secuencia, y así sucesivamente hasta la aparición en la última secuencia.

En la figura 9 se muestra un ejemplo de este formato.

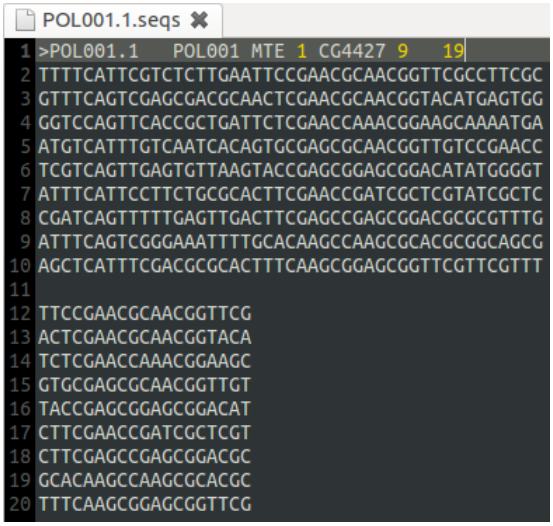

Figura 9: Archivo POL001.1.seqs de JASPAR versión 2010.

3.2. Características del nuevo método

Luego de analizar los anteriores algoritmos se llegó a la conclusión de destacar ciertas características a implementar en el algoritmo a desarrollar.

3.2.1. Orientación a patrones

La mayoría de los algoritmos tienen una orientación a perfil (Gibbs Sampler, MEME), es decir, teniendo un conjunto de secuencias tratan de encontrar una sección común; por el otro lado, la orientación a patrones se basa en buscar entre las combinaciones posibles una que se encuentre en las secuencias, como en el caso del algoritmo Winnower (Pevzner and Sze, 2000) y el algoritmo Weeder (Pavesi et al., 2001).

Empezar con una orientación a patrones permite evadir los máximos locales y manejar combinaciones que no se encontraron en primer lugar en alguna secuencia. Dada la naturaleza de los motifs es muy probable que no se encuentre una aparición del motif sin mutaciones, sino un motif el cual todas sus apariciones tengan pocas mutaciones, es decir, sea el patrón con mínimo total de mutaciones; y la orientación a patrones permite encontrar este tipo de patrones.

Se busca una ventaja en empezar en la orientación a patrones para abarcar más patrones escondidos y luego mudar el algoritmo a la orientación a perfil, pues no se ignora que ha dado buenos resultados en descalificar a los patrones más débiles.

3.2.2. La estructura de k-mer tree

Llamamos k-mer a cada subcadena de tamaño k de una secuencia, por ejemplo de la secuencia "ACCGTTAG" los 4-mers son: "ACCG", "CCGT", "CGTT","GTTA", y "TTAG".

Al momento de hacer el recorrido en las secuencias la mayoría opta por hacer un simple recorrido consecutivo por todas todas las secuencias, trabajar con el primer k-mer y al terminar volver a empezar ahora con el siguiente k-mer, hasta llegar al último. Se decidió guardar todos los k-mer de las secuencias en una estructura de árbol puesto que el alfabeto de 4 caracteres (A, C, G, T) más el tamaño de los patrones entre 5 y 20 caracteres no permite al árbol expandirse más allá de 4^{20} nodos, el cual sigue siendo una cantidad manejable. Además este árbol de k-mers o k-mer tree es una forma de programación dinámica sobre los cálculos a los k-mers, que da velocidad a estos cálculos.

La idea está basada en el árbol de sufijos (Sagot, 1998; Pavesi et al., 2001) con menos nodos y todas las hojas a la misma distancia de la raíz. De igual manera el árbol guarda la probabilidad de fondo de cada caracter y dentro de cada nodo se almacena:

- Una cadena padre que contiene la concatenación de todos los caracteres desde la raíz hasta el nodo actual.
- Una máscara de bits que indica las secuencias que llegan hasta dicho nodo.
- Y un arreglo de apuntadores a los nodos hijos, uno por cada caracter A,C,G,T siendo vacío para aristas inexistentes.

En la figura 10 se muestra un pequeño ejemplo de un k-mer tree para k=3.

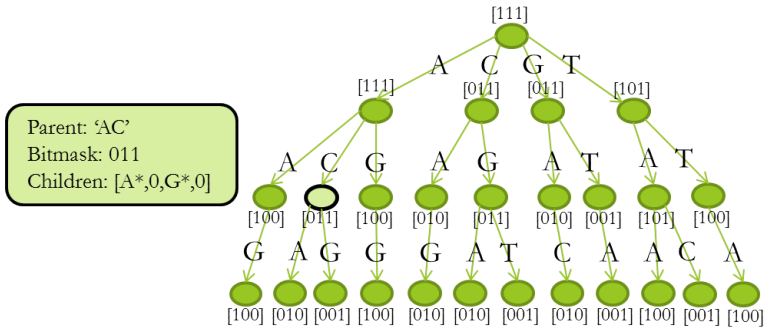

Figura 10: 3-mer tree de las secuencias
1: ACGTAC, 2: CGACAG, 3: TTAAGG;
y el contenido del nodo señalado.

3.2.3. Cambio de la función de distancia entre cadenas

Otro cambio que se decidió implementar es cambiar la función que se suele usar para medir la distancia entre dos cadenas del mismo tamaño. La función que se usa por default es la distancia Hamming pero ésta suele tener problemas de exactitud, por ejemplo la distancia entre 'CGT' y 'CGA' es 3 y la distancia entre 'CGT' y 'CGG' también es 3 sin importar si A o G aparece con mucha más frecuencia en la probabilidad de fondo. Esta característica se puede tomar como ventaja para acercarse a las subcadenas cuyos elementos se distingan respecto del fondo desde las operaciones de distancia en los primeros pasos del algoritmo.

La nueva función a la cual llamamos ProbDistance utiliza las probabilidades de fondo de cada caracter de la siguiente manera:

$$ProbDistance(seq1, seq2) = \sum_{i=1}^{k} d(seq1_i, seq2_2)$$

$$d(x, y) = \begin{cases} 0 & x = y \\ [1 - P(x)] \cdot P(y) & x \neq y \end{cases}$$

Se entiende que esta función de distancia no es simétrica pero no es necesario que lo sea para el uso que se le da en el algoritmo, el cual es comparar varias subcadenas y encontrar cuál es la que mejor se adapta al patrón en creación.

3.2.4. Simplificación de la función de distancia entre una PWM y una cadena

Otra función de distancia que arreglamos, esta vez mediante una simplificación es la distancia entre una PWM y una cadena del mismo tamaño. Por lo general se resuelve agregando la cadena al PWM y se calcula el nuevo MotifScore, resultando una función de semejanza. Se simplificó y se convirtió a una función de distancia que sigue el comportamiento inverso al MotifScore.

$$PWMDistance(PWM, seq) = \sum_{i=1}^{k} \frac{P(seq_i)}{P(PWM_i)}$$

Se entiende que $P(seq_i)$ es la probabilidad de fondo de ese caracter y que la función no es simétrica, pero igualmente no necesita serlo por ser usada sólo para comparar subcadenas y elegir la que contribuya de mejor manera a la creación de un nuevo patrón.

3.3. Diseño del nuevo algoritmo

Como se especificó en la sección 2.2.2 "El problema de encontrar motifs", el algoritmo se dirige a resolver el problema de la búsqueda del motif implantado, donde se pide el conjunto de secuencias *Seqs* donde el motif se encuentra así como el tamaño k del motif a buscar.

1. Guardar el conjunto de secuencias Seqs en un k-mer tree.

2. Para cada k-mer en Seqs:

   a) Encontrar en el k-mer tree el k-mer más cercano que pertenezca a una secuencia diferente.

   b) Formar una PWM de esos dos k-mers.

   c) Encontrar en el k-mer tree los k-mers más cercanos a la PWM que pertenezcan al resto de secuencias aun no agregadas.

   d) Formar un conjunto de k-mers, conteniendo un k-mer de cada secuencia de Seqs.

3. Calcular el MotifScore de todos los conjuntos de k-mers y usarlos para calcular un umbral que filtre sólo a los mejores conjuntos de k-mers, los que tengan mejor MotifScore.

4. Aplicar el algoritmo MEME a los mejores conjuntos de k-mers, es decir, usarlos como posiciones de inicio del algoritmo.

5. Devolver el conjunto de k-mers resultado de un algoritmo MEME con el mejor MotifScore de todos, éste será el motif encontrado.

La razón del paso 1 consiste en trabajar de manera más rápida en los siguientes pasos al recorrer varias veces todos los k-mers del conjunto de secuencias.

El paso 2 consiste en permitir a todos los k-mers iniciar un candidato a patrón, con esta variedad de inicios se debilita la posibilidad de caer en un máximo local. Las motivaciones de sus pasos interiores son primero crear un PWM cuanto antes para no buscar cercanía a una sóla subcadena sino a un conjunto de ellas, es decir, a un motif "en crecimiento" intermedio entre un conjunto de apariciones; luego que se tiene la PWM buscar el resto de subcadenas cercanas y formar un candidato a patrón.

El paso 3 filtra los conjuntos de k-mers, ya que el siguiente paso es más exhaustivo y es ineficiente entregarle todos los candidatos; por ello se hace una selección para sólo continuar con los mejores candidatos, no permitiendo que sean unos pocos para permitir más posibles caminos ni que sean muchos que alenten innecesariamente el algoritmo.

El algoritmo MEME del paso 4 es un algoritmo interno complejo, pero debido a que se inicia con unos buenos candidatos, cada ejecución no tarda mucho en converger.

3.4. Visualización de resultados

Cuando uno de los algoritmos presenta sus resultados, lo hace a través de un archivo *.motif con un formato especificado de la siguiente manera:

1. La primera línea, a modo de formato FASTA contiene el símbolo > seguido del identificador del motif y el resto de la primera etiqueta en el archivo *.sites, luego

separado por una tabulación sigue el número de secuencias en el archivo, separado por una tabulación más el tamaño del motif que contienen y separado por una tabulación más el MotifScore alcanzado por el motif encontrado.

2. En las líneas siguientes se muestran las subcadenas de las secuencias (en mayúsculas) que son las apariciones del motif encontrado ordenadas al mismo modo que las secuencias que las contienen. La 1a. subcadena pertenece a la 1a. secuencia y así sucesivamente.

3. Separado por una línea en blanco siguen las secuencias que contienen el motif, una secuencia por línea, con todos los caracteres en mayúsculas.

4. Finalmente, separado por otra línea en blanco, se conservan las apariciones correctas del motif en cada una de las secuencias. Después de la línea en blanco sigue la aparición correcta del motif en la 1a. secuencia y así sucesivamente.

En la figura 11 se muestra un ejemplo de este formato.

Figura 11: Archivo POL001.1.motif creado por el algoritmo KTreeMotif.

También se creó otro formato de archivo llamado *.score que contiene las estadísticas y evaluaciones del motif en el archivo *.motif:

1. En la primera línea se encuentran separados por tabulaciones, el identificador del motif correcto en JASPAR, el número de secuencias que lo componen, y el tamaño del motif.

2. En la segunda línea, el MotifScore alcanzado por el motif encontrado.

3. En la siguiente, el consensus del motif encontrado.

4. Luego una línea en blanco, y en la siguiente línea el profile del motif encontrado, otra línea en blanco y a continuación la PWM, otra línea en blanco y a continuación la PWM con pseudoconteos, otra línea en blanco y la PSSM en la siguiente.

5. Luego una línea en blanco y en las líneas siguientes se muestran las subcadenas (en mayúsculas) apariciones del motif encontrado ordenadas al mismo modo que las secuencias que las contienen. La 1a. subcadena pertenece a la 1a. secuencia y así sucesivamente.

6. Una última línea en blanco y por último siguen las secuencias que contienen el motif, una secuencia por línea, con todos los caracteres en mayúsculas.

En la figura 12 se muestra un ejemplo de este formato.

MA0001.1.score

```
MA0001.1        97      10
9.70251151285
CCATATATAG

        1       2       3       4       5       6       7       8       9       10
A       0.0     4.0     77.0    40.0    65.0    47.0    65.0    12.0    65.0    0.0
C       95.0    75.0    3.0     2.0     2.0     2.0     5.0     3.0     1.0     2.0
G       0.0     0.0     4.0     3.0     0.0     0.0     6.0     2.0     29.0    87.0
T       2.0     18.0    13.0    52.0    30.0    48.0    21.0    80.0    2.0     8.0

        1       2       3       4       5       6       7       8       9       10
A       0.0     0.041   0.794   0.412   0.67    0.485   0.67    0.124   0.67    0.0
C       0.979   0.773   0.031   0.021   0.021   0.021   0.052   0.031   0.01    0.021
G       0.0     0.0     0.041   0.031   0.0     0.0     0.062   0.021   0.299   0.897
T       0.021   0.186   0.134   0.536   0.309   0.495   0.216   0.825   0.021   0.082

        1       2       3       4       5       6       7       8       9       10
A       0.0     0.042   0.793   0.412   0.67    0.484   0.67    0.124   0.67    0.0
C       0.979   0.773   0.031   0.021   0.021   0.021   0.052   0.031   0.01    0.021
G       0.0     0.0     0.041   0.031   0.0     0.0     0.062   0.021   0.299   0.896
T       0.021   0.186   0.134   0.536   0.309   0.495   0.217   0.824   0.021   0.083

        1       2       3       4       5       6       7       8       9       10
A       -9.923  -3.003  1.253   0.308   1.008   0.541   1.008   -1.426  1.008   -9.923
C       2.335   1.994   -2.641  -3.221  -3.221  -3.221  -1.908  -2.641  -4.208  -3.221
G       -9.923  -9.923  -2.045  -2.458  -9.923  -9.923  -1.462  -3.039  0.808   2.392
T       -3.854  -0.703  -1.172  0.826   0.033   0.71    -0.481  1.447   -3.854  -1.87

CCATATATAG
CCATATATAG
CCATAAATAG
CCATAAATAG
CCATAAATAG
```

Figura 12: Archivo MA0001.1.score creado a partir de MA0001.1.motif creado por el algoritmo KTreeMotif.

Capítulo 4: Pruebas y resultados

4.1. Obtención de datos de prueba

En el capítulo anterior, en el apartado 3.1 se explicó el preprocesamiento que se hizo sobre la base de datos JASPAR, ahora cabe añadir que además de la validación se trabajó con una versión a la que llamaremos "2010 Actualizada", esta versión de la base de datos contiene los mismos motifs (etiquetados con su identificador) que la versión del 2010, pero cada archivo que también se encuentra en la versión actualizada se copió de esta versión, para evitar los motifs que tienen errores en la versión del 2010 y fueron corregidos más adelante, como deben encontrarse en la versión actualizada y porque la versión actualizada completa era muy pesada para el número de pruebas que se requería hacer.

En el cuadro 6 se pueden ver las nuevas estadísticas de la versión a ocupar.

	2010 Actualizada
Tamaño	239.2 Kb
Núm. de secuencias	6,467
Núm. de TF	235
Secuencia más corta	7 bp
Secuencia más larga	2,000 bp
Motif más corto	5 bp
Motif mas largo	22 bp

Cuadro 6: Estadísticas de la versión "2010 Actualizada" de la base de datos JASPAR.

4.2. Medidas de evaluación utilizadas

En el apartado 2.5 se mencionaron las medidas de evaluación que recomiendan otros artículos para verificar los algoritmos de búsqueda de motifs, en el presente trabajo se utilizaron esas medidas, aunque se tomó especial atención en sólo algunas y se agregaron otras medidas más.

Las medidas que se consideraron más importantes de las anteriores son el **coeficiente de rendimiento a nivel de nucleótidos** nPC y el **coeficiente de rendimiento a nivel de sitio de enlace** sPC, donde la función para ambas es

$$PC = \frac{TP}{TP + FP + FN}$$

Esta medida está basada en los positivos y negativos que especificaremos a continuación:

Nivel de nucleótidos:

TP: Número de posiciones de las secuencias marcadas correctamente por el algoritmo como parte del motif.

TN: Número de posiciones de las secuencias marcadas correctamente por el algoritmo como exteriores al motif.

FP: Número de posiciones de las secuencias marcadas incorrectamente por el algoritmo como parte del motif.

FN: Número de posiciones de las secuencias marcadas incorrectamente por el algoritmo como exteriores al motif.

Nivel de sitio de enlace:

TP: Número de subcadenas marcadas por el algoritmo que se intersectan con las subcadenas correctas que pertenecen al motif.

TN: Número de subcadenas correctas que pertenecen al motif que se intersectan con las subcadenas marcadas por el algoritmo.

FP: Número de subcadenas marcadas por el algoritmo que no se intersectan con las subcadenas correctas que pertenecen al motif.

FN: Número de subcadenas correctas que pertenecen al motif que no se intersectan con las subcadenas marcadas por el algoritmo.

También se trabajó con la **tasa de éxito a nivel de secuencia** "Success Rate" que indica en cuántas secuencias se encontró correctamente la aparición del motif.

$$SuccessRate = \frac{Num.SubcadenasCorrectas}{Num.Secuencias}$$

Consiste en el número de secuencias donde la subcadena aparición del motif es la misma entre la encontrada por el algoritmo y la correcta, dividido entre el número de secuencias donde se buscó el motif.

Debido a que sólo se está buscando un motif y éste se encuentra forzosamente una vez en todas las secuencias que se entregan, las demás medidas otorgan menos información porque es más fácil que entreguen valores muy altos.

También se usaron dos medidas para calificar a los algoritmos. La primera de éstas se llamó **"Equal Consensus"** y cuenta el número el número de **consensos de motif** encontrados correctamente de entre el total de motifs.

$$EqlConsensus = \frac{Num.ConsensosCorrectos}{Num.Motifs}$$

Recuérdese que cuando se tiene un motif, una de sus formas de representarlo es mediante un consenso de motif, una cadena de los caracteres más frecuentes en cada posición; esta medida cuenta cuántos motifs encontrados tienen igual consenso que su motif correcto correspondiente. Debe tomarse en cuenta que es posible formar el mismo consenso con diferentes conjuntos de subcadenas siempre y cuando los caracteres más frecuentes sean los mismos.

Otra medida usada fue el **"Equal Score"** y de similar manera al Equal Consensus cuenta el número de consensos de motif y **Motif Score** encontrados correctamente del total de motifs.

$$EqlScore = \frac{Num.ConsensosyScoresCorrectos}{Num.Motifs}$$

A diferencia de la medida anterior es muy difícil conseguir el mismo consenso y el mismo Motif Score con diferentes conjuntos de subcadenas así que es la medida más difícil de conseguir ya que más bien solicita al algoritmo encontrar correctamente todas las subcadenas que pertenecen al motif.

Por último, debido a un fenómeno que era muy común de encontrar en la búsqueda de motifs la medida "Overreach Score" mide el número de casos en donde el Motif Score del motif encontrado fue superior al del motif correcto.

$$OverreachScore = \frac{Num.ScoresExcedidos}{Num.Motifs}$$

De esta manera contamos los motifs donde la premisa de ser el patrón más frecuente del tamaño dado con que definimos un motif no se cumple.

4.3. Formato de Evaluación

Para hacer la evaluación de los motifs de la base de datos, se creó un archivo Comparative algorithm.csv con el sig. formato:

1. Por cada uno de los motifs se llenó una línea de datos, la cual contiene en cada columna:

   a) El identificador del motif, el nombre del archivo.

   b) El Motif Score del motif correcto.

   c) El consenso del motif correcto.

   d) El Motif Score del motif encontrado por el algoritmo.

   e) El consenso del motif encontrado por el algoritmo.

   f) "True" si ambos contienen el mismo consenso, "False" en caso contrario.

   g) "True" si ambos contienen el mismo consenso y mismo Motif Score, "False" en caso contrario.

h) El nombre del algoritmo si éste obtuvo un motif con mayor o igual Motif Score que el motif correcto. "Correct" si el motif correcto lo superó.

i) "Reached" si el motif obtenido por el algoritmo tiene un Motif Score igual al motif correcto. "Overreach" si el motif obtenido por el algoritmo tuvo un Motif Score superior al motif correcto. "Underreach" si el motif correcto tuvo un Motif Score superior al motif encontrado por el algoritmo.

j) El valor del coeficiente de rendimiento a nivel de nucleótidos (nPC).

k) El valor de la medida F (apartado 2.5).

l) El valor del coeficiente de rendimiento a nivel de sitio de enlace (sPC).

m) La tasa de éxito a nivel de secuencia.

2. Luego una línea en blanco y en las últimas dos líneas se calculan las medidas de evaluación para el conjunto de motifs analizados.

   a) El número de motifs con igual consenso, y debajo el valor de EqlConsensus.

   b) El número de motifs con igual consenso y Motif Score, y debajo el valor de EqlScore.

   c) El número de motifs donde el Motif Score conseguido por el algoritmo es mayor o igual al correcto, y debajo este número dividido entre el total de archivos.

   d) El número de motifs donde el Motif Score conseguido por el algoritmo es igual al correcto, y debajo este número dividido entre el total de archivos.

   e) El número de motifs donde el Motif Score conseguido por el algoritmo es mayor que el correcto, y debajo este número dividido entre el total de archivos.

   f) El promedio de los valores nPC, y debajo la fracción de los motifs que tienen nPC igual a 1, es decir, sin errores.

   g) El promedio de las medidas F, y debajo la fracción de los motifs que tienen la medida F igual a 1, sin errores.

   h) El promedio de los valores sPC, y debajo la fracción de los motifs que tienen sPC igual a 1.

   i) El promedio de las tasas de éxito, y debajo la fracción de los motifs que tienen la tasa igual a 1, esto es, sin errores.

4.4. Pruebas

Para diseñar el algoritmo KTreeMotif, se tomaron partes de distintos algoritmos, siempre que dieran los mejores resultados; por consiguiente, era necesario comparar estos componentes entre sí. Estos componentes diversos entran a formar parte en distintas

etapas del algoritmo, por lo que se hicieron varias selecciones, como a continuación se explica.

En este apartado nos redirigiremos varias veces a la explicación del algoritmo KTree-Motif en la sección 3.3 para explicar en qué etapa estamos haciendo las variaciones.

4.4.1. Comparaciones de SP-Score con Motif Score y de Gibbs Sampler con MEME

Empezaremos por el primer aspecto del algoritmo que se necesitaba corroborar, al momento de podar los conjuntos de subcadenas en el paso 3 se quiso ver si era mejor usar el SP-Score explicado por el algoritmo SP-Star (Pevzner and Sze, 2000) o el Motif Score (Schneider et al., 1986) que han venido usando la mayoría de los algoritmos.

Otro aspecto importante a tomar en cuenta en el diseño del algoritmo fue el paso 4 donde se refina cada uno de los conjuntos de subcadenas ganadores, se puede realizar mediante el algoritmo Gibbs Sampler o el algoritmo MEME. Es un paso en que se requiere alcanzar un máximo rápidamente con cada una de las opciones pues ya se están entregando buenas posiciones de entrada, y estos dos algoritmos son las dos principales metolodogías de resolución que ya existen, usar otro algoritmo más complejo hace más pesado innecesariamente al algoritmo completo en este paso final.

En la tabla 7 se puede ver la comparación en este aspecto. Se muestra cómo el Motif Score obtiene mejores resultados que el SP-Score cuando se usa el Gibbs Sampler, mientras que el SP-Score obtiene mejores resultados que el Motif Score cuando usamos el algoritmo MEME. No obstante el Gibbs Sampler en combinación con el Motif Score obtuvo los mejores resultados y por una pequeña diferencia el algoritmo MEME con el SP-Score, sobre todo en el Equal Score que es la medida más rígida. El SP-Score obtiene sus mejores resultados en combinación con el algoritmo MEME, aún así no sobresale mucho respecto a la combinación de Motif Score con el algoritmo MEME como se esperaría para demostrar su utilidad. Otra característica a tomar en cuenta es que el Motif Score es ya calculado dentro de la mecánica del algoritmo mientras que el SP-Score es una operación adicional.

	Gibbs Sampler		MEME	
	Motif Score	SP-Score	Motif Score	SP-Score
Overreach Score	117 — 49.8 %	39 — 16.6 %	103 — 43.8 %	96 — 40.9 %
Equal Consensus	158 — 67.2 %	118 — 50.2 %	152 — 64.7 %	156 — 66.4 %
Equal Score	106 — 45.1 %	42 — 17.9 %	99 — 42.1 %	105 — 44.7 %
nPC	0.902	0.790	0.893	0.895
sPC	0.975	0.926	0.972	0.976
Success Rate	0.722	0.559	0.704	0.710

Cuadro 7: Comparación del SP-Score y el Motif Score en el paso 3 cuando se usa el algoritmo Gibbs Sampler o el algoritmo MEME en el paso 4.

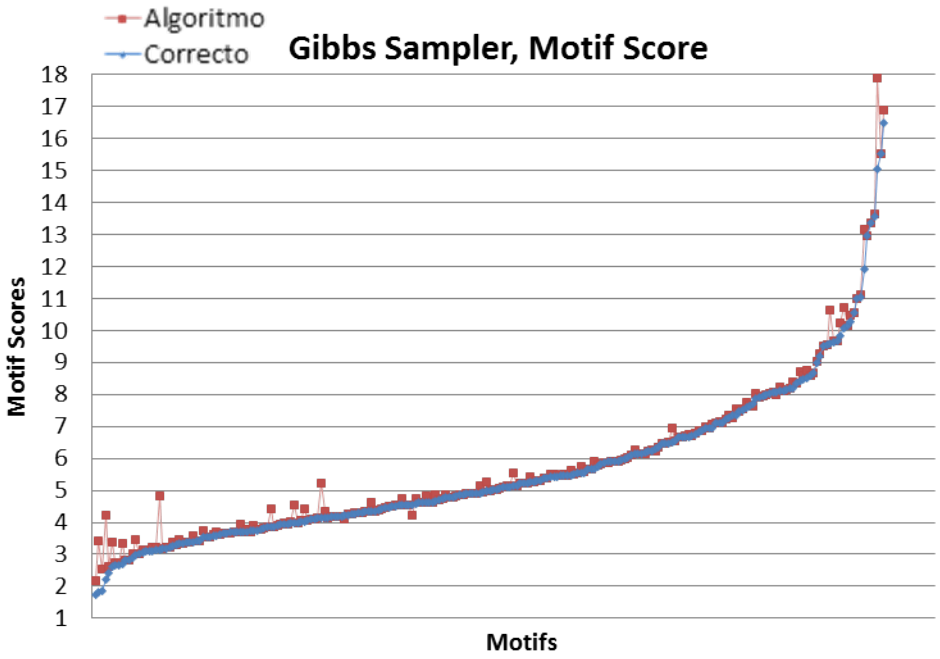

Figura 13: Resultado del algoritmo usando Gibbs Sampler y Motif Score.

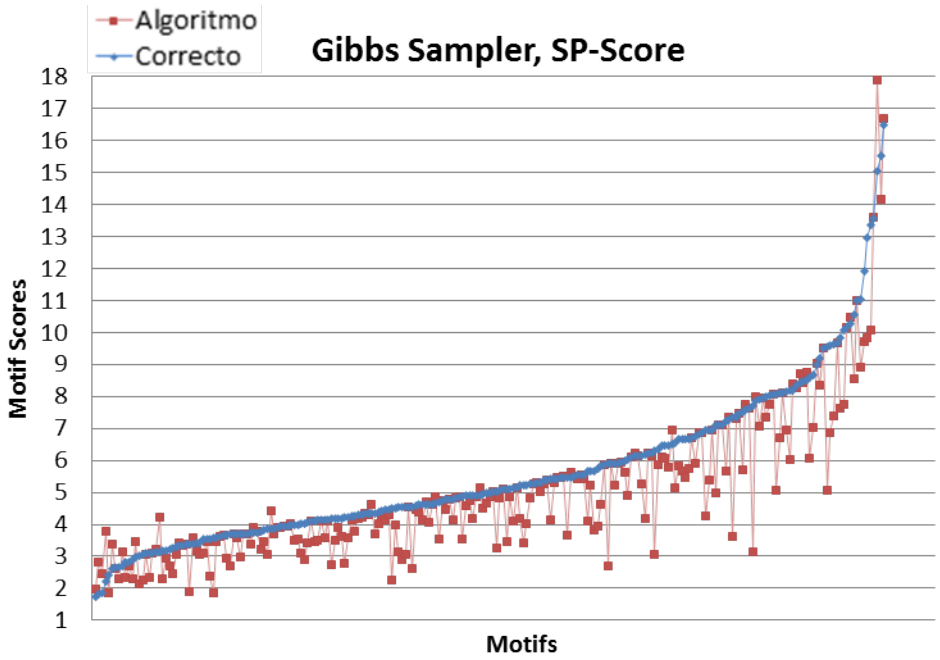

Figura 14: Resultado del algoritmo usando Gibbs Sampler y SP-Score.

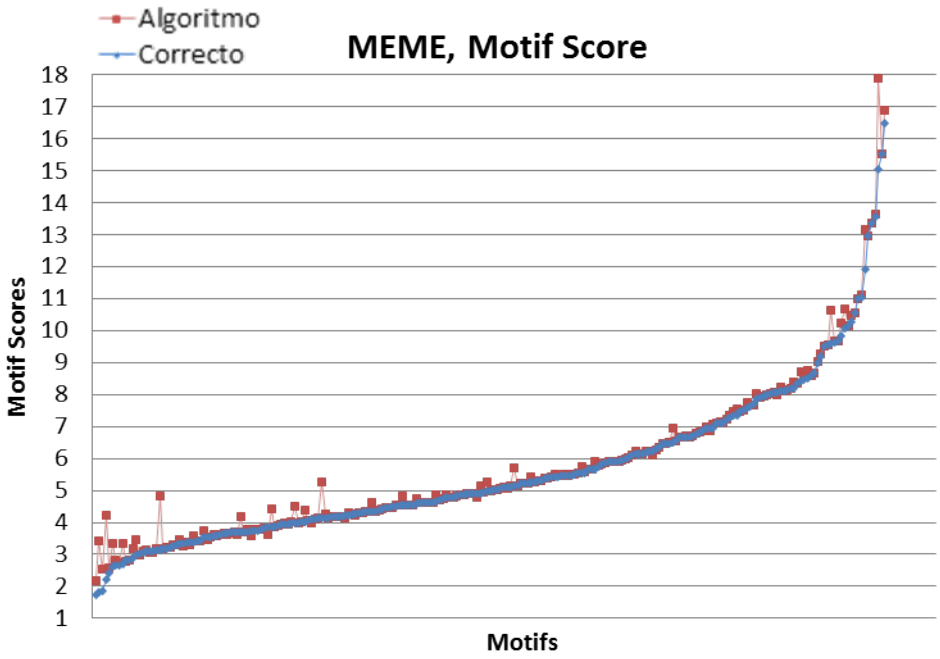

Figura 15: Resultado del algoritmo usando MEME y Motif Score.

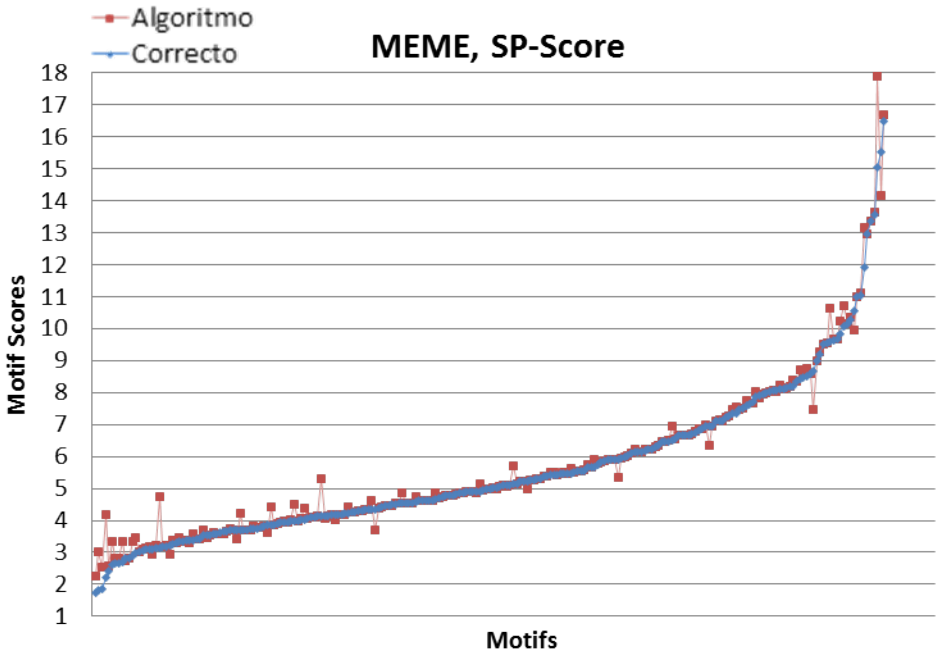

Figura 16: Resultado del algoritmo usando MEME y SP-Score.

En las gráficas 13, 14, 15 y 16 se muestra la precisión de las distintas combinaciones entre Gibbs Sampler, MEME, Motif Score y SP-Score para tener una mejor idea del resultado del algoritmo en cada prueba. Los cuadros rojos representan los Motif Score de cada uno de los motifs obtenidos por el algoritmo, mientras que los cuadros azules indican los Motif Score de los motifs correctos para los mismos; por lo que para un mismo archivo de la base de datos, un sólo motif a buscar, ambos resultados se encuentran alineados verticalmente.

A partir de todo esto decidimos continuar las pruebas con el algoritmo Gibbs Sampler en combinación con el Motif Score y el algoritmo MEME combinado con el SP-Score.

4.4.2. Comparación de la distancia Hamming y Prob-Distance

En el apartado 3.2.3 se explica una nueva medida de distancia para usar durante el algoritmo, específicamente en el paso 2.a, lo compararemos contra la distancia Hamming que es la actualmente usada por los algoritmos que se dedican al problema.

En esta ocasión se usa la distancia Hamming sin modificaciones para la comparación de cadenas, la distancia de Levenstein o de edición se reduce a entregar el mismo resultado porque las cadenas que comparamos en todo momento tienen el mismo tamaño, entonces no existen inserciones ni eliminaciones.

En la tabla 8 se realiza la comparación de ambas distancias. Se puede ver que la distancia ProbDistance obtiene mejores resultados que la distancia Hamming en ambas versiones (con el algoritmo Gibbs Sampler y con el algoritmo MEME) a pesar de no ser una gran diferencia de resultados, se debe recordar que estos dos algoritmos ocupan metodologías distintas y que esta distancia se ocupa al inicio del algoritmo pero no se vuelve a usar, esta puede ser la razón por la que los resultados cambiaron muy poco pero siguen siendo destacables por funcionar bien ante el uso con ambas metodologías.

	Gibbs Sampler y Motif Score		MEME y SP-Score	
	Hamming	ProbDistance	Hamming	ProbDistance
Overreach Score	120 — 51.1 %	117 — 49.8 %	99 — 42.1 %	96 — 40.9 %
Equal Consensus	156 — 66.4 %	158 — 67.2 %	152 — 64.7 %	156 — 66.4 %
Equal Score	101 — 43 %	106 — 45.1 %	101 — 43 %	105 — 44.7 %
nPC	0.900	0.902	0.897	0.895
sPC	0.974	0.975	0.976	0.976
Success Rate	0.721	0.722	0.703	0.710

Cuadro 8: Comparación de la distancia Hamming y la distancia ProbDistance cuando se usa el algoritmo Gibbs Sampler y Motif Score o MEME y SP-Score.

En las gráficas 17, 18, 19 y 20 se muestra la precisión de las distintas combinaciones entre Gibbs Sampler-Motif Score, MEME-SP-Score, distancia Hamming y ProbDistance para tener una mejor idea del resultado del algoritmo en cada prueba. El formato de las gráficas es igual a como se explicaron en las pruebas anteriores.

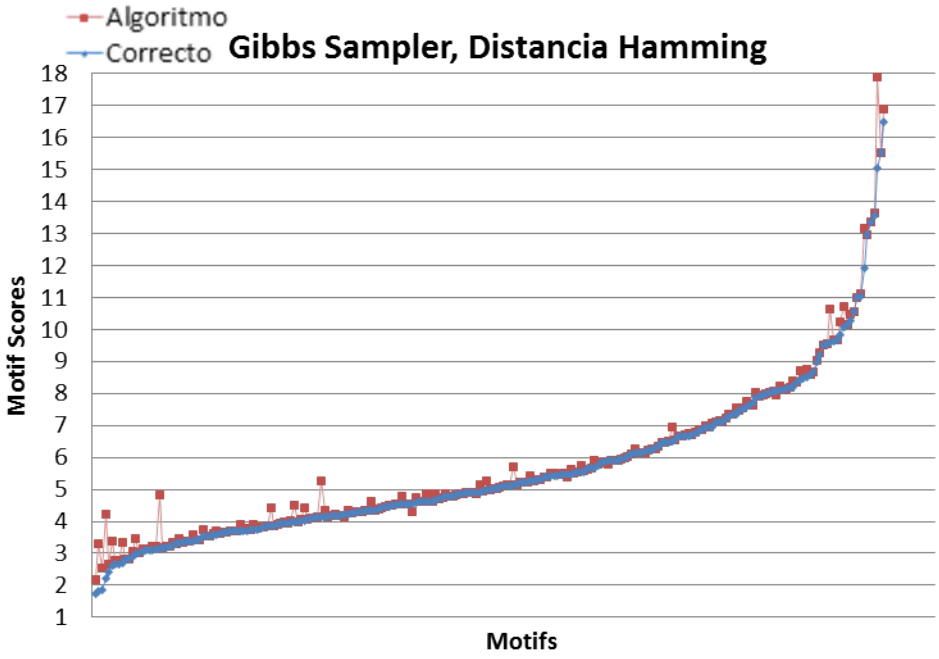

Figura 17: Resultado del algoritmo usando Gibbs Sampler, Motif Score y distancia Hamming.

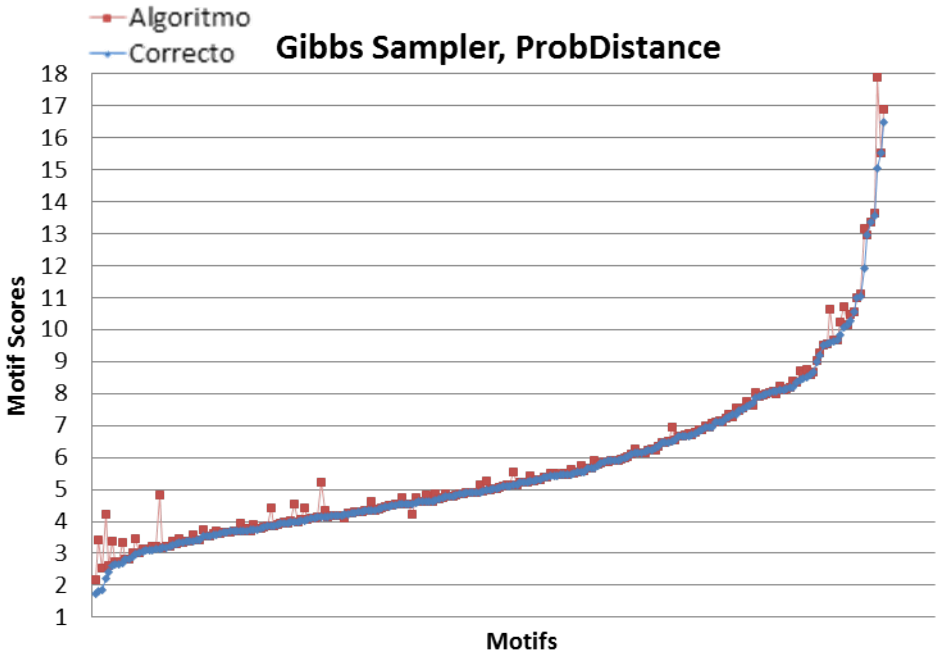

Figura 18: Resultado del algoritmo usando Gibbs Sampler, Motif Score y ProbDistance.

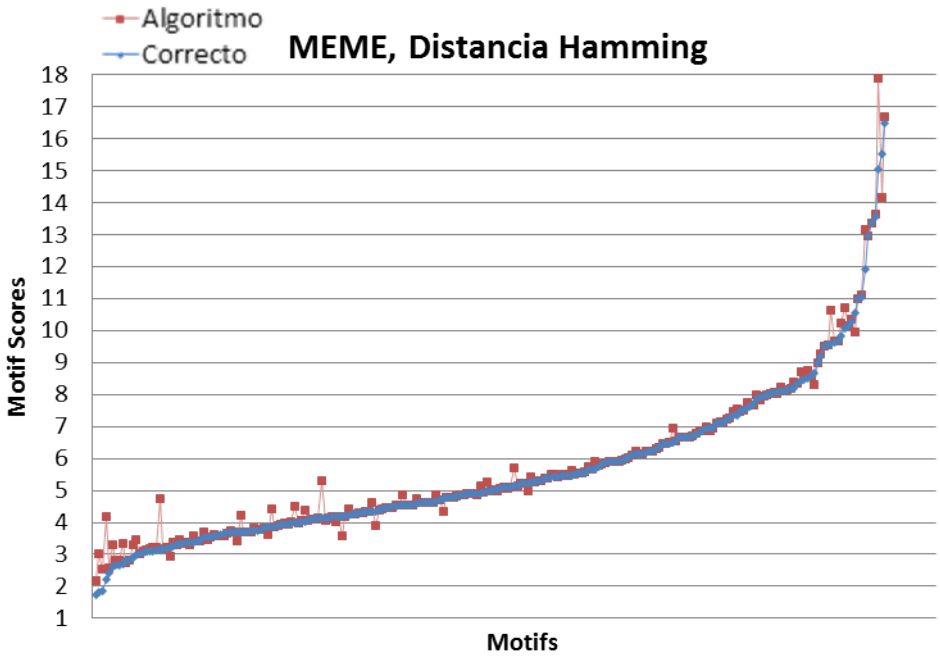

Figura 19: Resultado del algoritmo usando MEME, SP-Score y distancia Hamming.

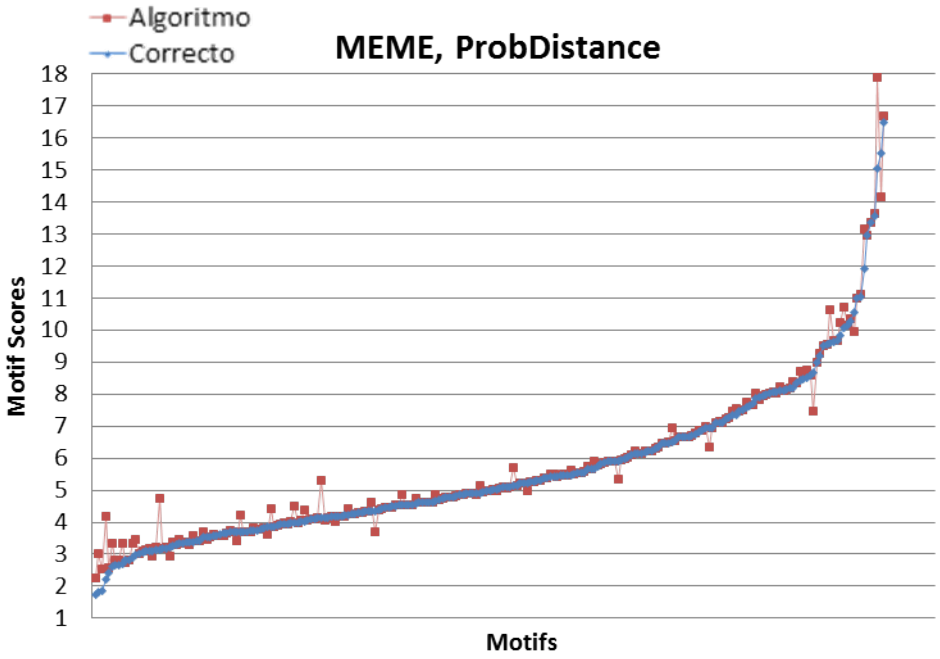

Figura 20: Resultado del algoritmo usando MEME, SP-Score y ProbDistance.

Al final debido a que la distancia ProbDistance obtuvo mejores resultados que la distancia Hamming, es la que ocuparemos en las siguientes pruebas, y debido a las razones explicadas en el apartado anterior y que de nuevo el algoritmo Gibbs Sampler obtuvo mejores resultados que el algoritmo MEME sólo se continuará con esta versión que usa el algoritmo Gibbs Sampler.

4.4.3. Comparación de similitud a PWM mediante Motif Score y distancia mediante PWM-Distance

Ahora nos dedicaremos a analizar la distancia a PWM desarrollada en el apartado 3.2.4, la cual se diseñó como una alternativa a la distancia que se obtiene mediante Motif Score. La situación de comparar una PWM contra un conjunto de k-mers del mismo tamaño se encuentra en los pasos 2.c y 4 dentro del algoritmo MEME.

Los resultados de la comparación se muestran en la tabla 9. Cabe aclarar que la propuesta obtiene mejores resultados de manera notoria que la similitud regularmente; gracias a que tenemos los datos correctos de la base de datos es posible verificar esto que se agrega como ventaja al hecho de ser una función menos compleja, razón por la que se continuará prefiriendo esta distancia respecto a la otra función.

	Similitud	PWMDistance
Overreach Score	67 — 28.5 %	117 — 49.8 %
Equal Consensus	150 — 63.8 %	158 — 67.2 %
Equal Score	93 — 39.6 %	106 — 45.1 %
nPC	0.874	0.902
sPC	0.963	0.975
Success Rate	0.696	0.722

Cuadro 9: Comparación de la similitud regular a la PWM y la distancia PWMDistance.

En las gráficas 21 y 22 se muestra la precisión de las pruebas hechas con similitud mediante Motif Score y mediante PWMDistance para tener una mejor idea del resultado del algoritmo en cada una. El formato de las gráficas es igual a como se explicaron en las pruebas anteriores.

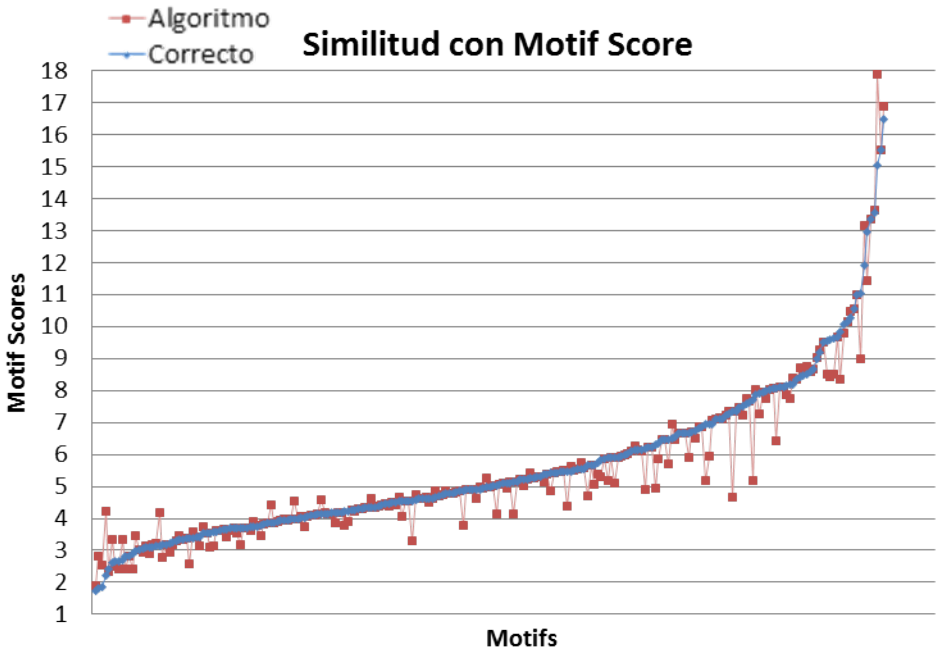

Figura 21: Resultado del algoritmo usando similitud mediante Motif Score.

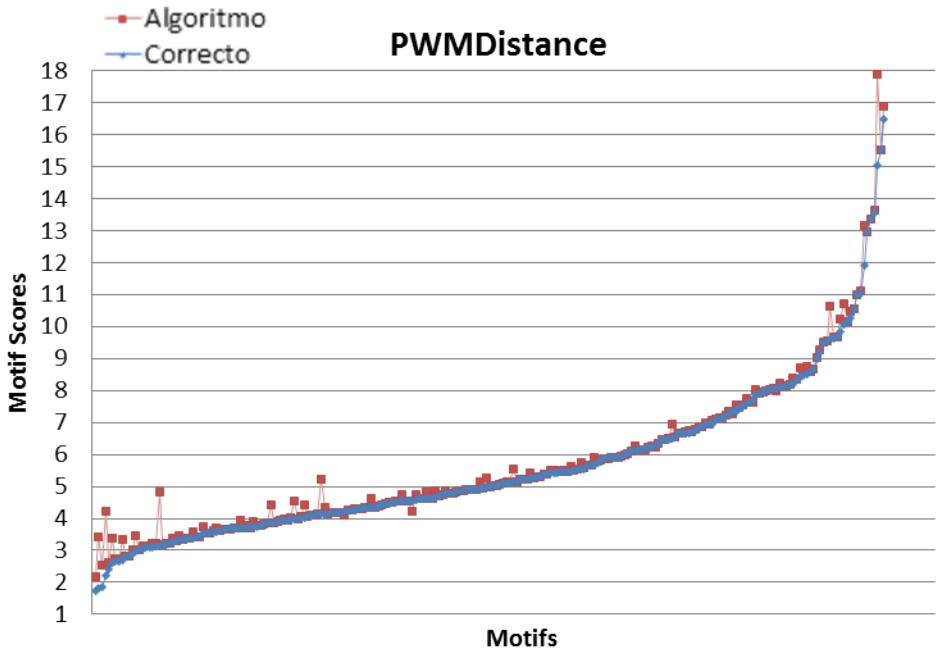

Figura 22: Resultado del algoritmo usando PWMDistance.

4.4.4. Calibración del umbral de poda

Se decidió calibrar un parámetro importante del algoritmo, el umbral de poda del paso 3, este parámetro determina a cuántos candidatos se les ofrece la oportunidad de refinarse para descubrir entre ellos el motif, como ya se indicó anteriormente se debe podar esperando no filtrar muchos candidatos al siguiente paso pero también que entre ellos se encuentre el candidato correcto o incluso el motif correcto sin llevar sólo candidatos que conduzcan a máximos locales incorrectos.

Para ello se elige un parámetro z-score que indica el punto de la distribución donde se corta la lista de Motif Scores. De esta manera, se tiene la lista de Motif Scores, se obtiene la media μ, luego la desviación estándar σ sólo de los valores superiores a la media ya que filtraremos sólo valores superiores, y finalmente se usa el z-score como parámetro para filtrar sólo los Motif Scores que cumplan la condición siguiente:

$$MotifScore \geq \mu + (zscore \cdot \sigma)$$

Como paso adicional dentro del cálculo del umbral se cuenta el número de valores que serán filtrados, si son más de 30 se elige este umbral, si son menos se hace el conteo con un z-score inferior por 0.5 y pasa por la misma validación, así se va bajando en z-scores hasta conseguir uno que filtre suficientes valores o se llegue a la media, en caso de que no haya ningun z-score superior o igual a 0.5 que reuna los 30 valores es la media la que sirve de umbral. Por ejemplo, si empezamos con z-score 2.0, bajamos a 1.5, 1.0 y finalmente 0.5, si ningun umbral consiguió filtrar 30 Motif Scores, el umbral serála media sin importar cuántos filtra.

Si se llegase a tener muchos más valores por debajo de la media calcular la desviación estándar total nos puede llegar a influenciar a podar siguiendo la distribución inferior cuando queremos saber sólo la desviación de los datos superiores para encontrar sólo los valores anómalos superiores, seguir la desviación estándar total provocaría, cuando se tienen pocos valores superiores, que al podar no queden valores o sean muy pocos. Por este motivo se decidió obtener la desviación estándar usando sólo los valores superiores a la media.

En las tablas 10 y 11 se puede ver este análisis y se concluye que el mejor umbral se puede obtener con z-score 2.5 que supera a los demás umbrales en todas las medidas excepto al usar z-score 2.0, donde la diferencia es pequeña. A pesar de la cercanía de los resultados elegimos z-score 2.5 porque supera a z-score 2.0 en la mayoría de las medidas y filtra menos candidatos hacia el siguiente paso, lo que agiliza la ejecución. No obstante la diferencia entre los valores de poda es menor de lo esperado, mostrando que el algoritmo no es tan sensible a este parámetro como parecía.

En las gráficas 23, 24, 25 y 26 se muestra la precisión de las pruebas variando el z-score entre 1.0, 2.0, 2.5 y 3.0 para tener una mejor idea del resultado del algoritmo en cada una. El formato de las gráficas es igual a como se explicaron en las pruebas anteriores.

	1.0	1.5	2.0
Overreach Score	118 — 50.2 %	119 — 50.6 %	117 — 49.8 %
Equal Consensus	156 — 66.4 %	156 — 66.4 %	155 — 66 %
Equal Score	103 — 43.8 %	103 — 43.8 %	105 — 44.7 %
nPC	0.897	0.899	0.902
sPC	0.974	0.974	0.975
Success Rate	0.711	0.714	0.722

Cuadro 10: Comparación de los valores de z-score para el umbral de poda.

	2.5	3.0	3.5
Overreach Score	119 — 50.6 %	117 — 49.8 %	118 — 50.2 %
Equal Consensus	162 — 68.9 %	155 — 66 %	157 — 66.8 %
Equal Score	104 — 44.3 %	101 — 43 %	101 — 43 %
nPC	0.902	0.898	0.900
sPC	0.976	0.972	0.975
Success Rate	0.730	0.717	0.726

Cuadro 11: Comparación de los valores de z-score para el umbral de poda (continuación).

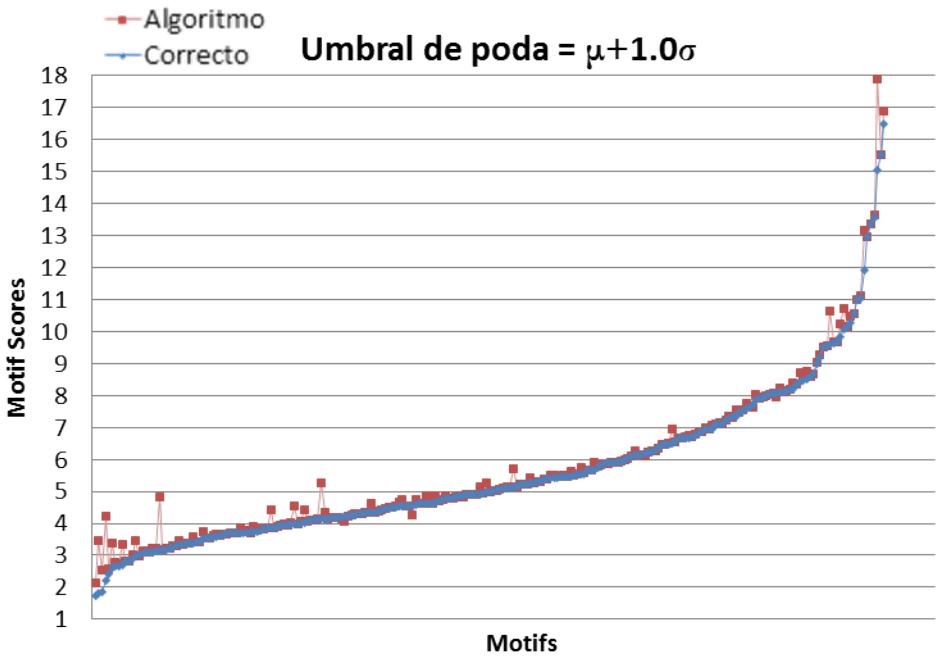

Figura 23: Resultado del algoritmo usando z-score=1.0.

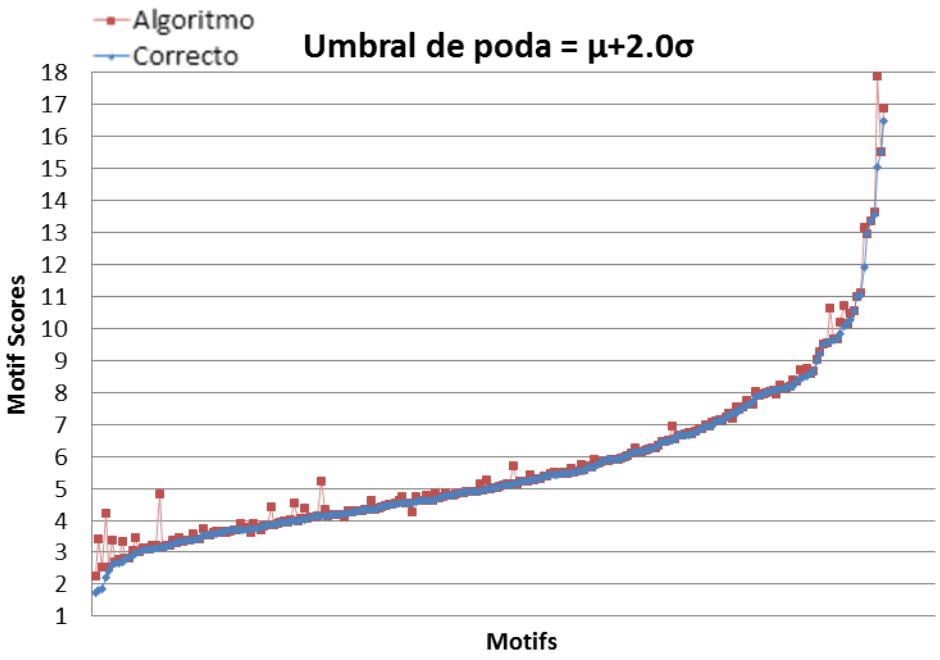

Figura 24: Resultado del algoritmo usando z-score=2.0.

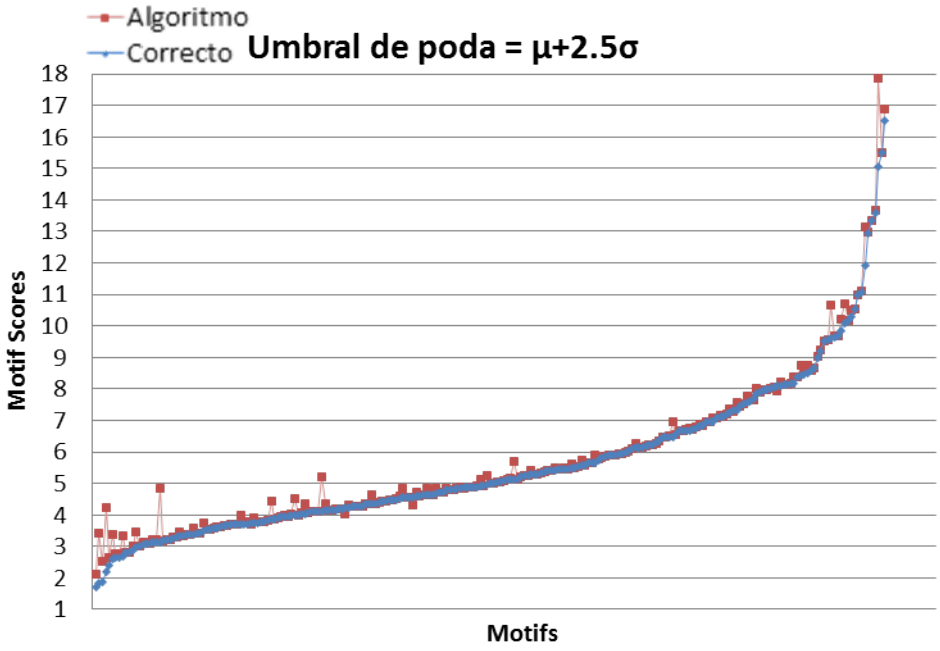

Figura 25: Resultado del algoritmo usando z-score=2.5.

4.4.5. Calibración de convergencia

En el paso 4 se ejecuta el algoritmo Gibbs Sampler como ya se decidió en una prueba anterior, pero como el Gibbs Sampler tiene sus propios parámetros para encontrar la

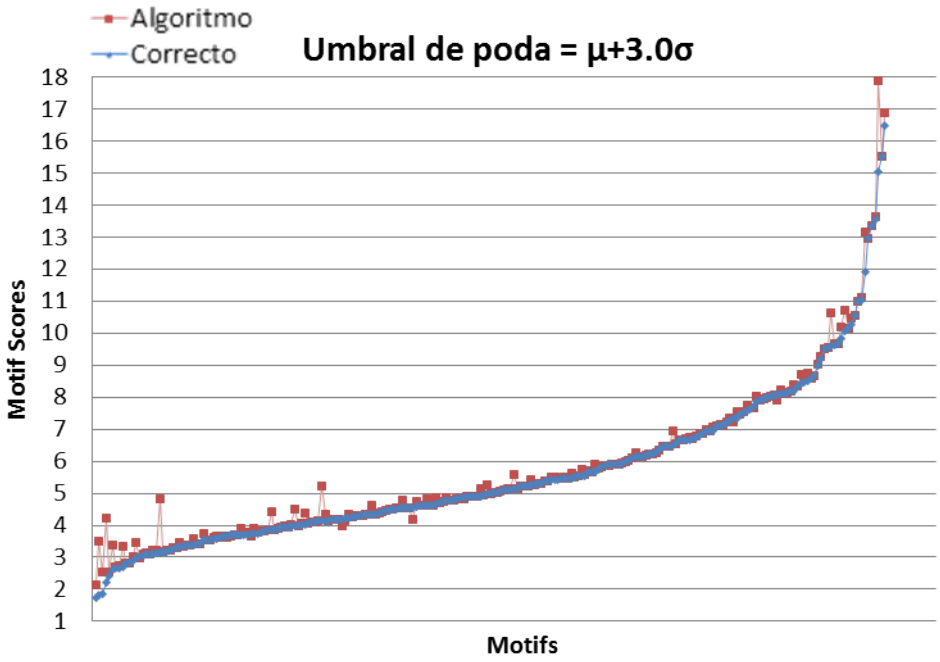

Figura 26: Resultado del algoritmo usando z-score=3.0.

convergencia (explicado en el apartado 2.4.3), estos se necesitan calibrar sobre todo para llegar rápido a la convergencia en las ejecuciones con los candidatos a motif.

Para programar las iteraciones del Gibbs Sampler se toma el Motif Score como función objetivo, y el algoritmo escapa del ciclo de iteraciones cuando los Motif Scores empiezan a resultar muy cercanos entre ellos, con diferencias pequeñas. Para esto se tienen tres parámetros: el número máximo de iteraciones N, el número mínimo de Motif Scores cercanos para terminar nmin, y la diferencia máxima entre Motif Scores continuos para definirlos cercanos dif. En las siguientes pruebas cambiaremos los tres parámetros y de acuerdo a los resultados elegiremos los mejores valores.

Primero se probará N = 50, 100, 200 con los resultados en la tabla 12.

	50	100	200
Overreach Score	117 — 49.8 %	121 — 51.5 %	124 — 52.8 %
Equal Consensus	154 — 65.5 %	157 — 66.8 %	153 — 65.1 %
Equal Score	101 — 43 %	101 — 43 %	102 — 43.4 %
nPC	0.902	0.902	0.899
sPC	0.976	0.976	0.974
Success Rate	0.726	0.721	0.714

Cuadro 12: Comparación de número máximo de iteraciones N.

Al encontrar que los resultados variaban muy poco entre distintos valores de N se concluye que este parámetro afecta muy poco el resultado, por lo que es preferible el

número menor con que se hicieron las pruebas: 50.

Ahora se procede a variar el mínimo de Motif Scores cercanos para converger nmin entre los valores 20, 30 y 40 en la tabla 13.

	20	30	40
Overreach Score	116 — 49.4 %	116 — 49.4 %	120 — 51.1 %
Equal Consensus	155 — 66 %	155 — 66 %	157 — 66.8 %
Equal Score	106 — 45.1 %	102 — 43.4 %	103 — 43.8 %
nPC	0.900	0.897	0.899
sPC	0.975	0.973	0.974
Success Rate	0.716	0.711	0.718

Cuadro 13: Comparación de número mínimo de Motif Scores cercanos nmin.

También se encontró que los resultados variaban poco entre distintos valores de nmin, con nmin=20 destacando por una pequeña diferencia, se concluye que este parámetro también afecta muy poco el resultado y se elige nmin=20 por el mismo motivo de llegar a la convergencia más rápido sin disminuir precisión.

Por último se cambia la diferencia máxima entre Motif Scores continuos para definirlos cercanos dif entre los valores 0.1, 0.01, 0.001 en la tabla 14.

	0.1	0.01	0.001
Overreach Score	115 — 48.9 %	116 — 49.4 %	114 — 48.5 %
Equal Consensus	156 — 66.4 %	159 — 67.7 %	157 — 66.8 %
Equal Score	106 — 45.1 %	106 — 45.1 %	103 — 43.8 %
nPC	0.898	0.900	0.901
sPC	0.974	0.975	0.974
Success Rate	0.717	0.717	0.730

Cuadro 14: Comparación de diferencia máxima entre Motif Scores dif.

Finalmente se encontró también una variación pequeña al modificar la diferencia máxima permitida entre Motif Scores, por lo que no es un parámetro que afecte de manera significativa el resultado final. Se elige el valor de 0.1 por ser el más permisible para encontrar la convergencia y salir de las iteraciones y aún así permitir un buen funcionamiento del algoritmo.

4.4.6. Comparación del recorrido secuencial y k-mer tree

Ahora trataremos el k-mer tree del apartado 3.2.2, una estructura para mejorar la velocidad, este árbol representa una mejora al recorrido a lo largo de las secuencias donde se analiza una por una las subcadenas de tamaño k, cada vez que se necesiten consultar. Por lo que en este caso no compararemos la precisión del algoritmo donde no se espera algun cambio, sino el cambio en la velocidad.

Se tomó el tiempo que tarda el algoritmo en los pasos 1 y 2 donde se crea el k-mer tree contra un código donde no se fabrica la estructura y cada que se necesita buscar la subsecuencia más cercana a una subsecuencia o PWM se recorren todas las subsecuencias de todas las secuencias una por una.

En las gráficas 27 y 28 se muestran los resultados, en azul se marca el tiempo que toma usar el k-mer tree y en rojo el recorrido secuencial que se explicó en el párrafo anterior. Tenemos dos gráficas para mostrar los tiempos en segundos y en log_{10} de segundos para observar mejor los tiempos más cortos.

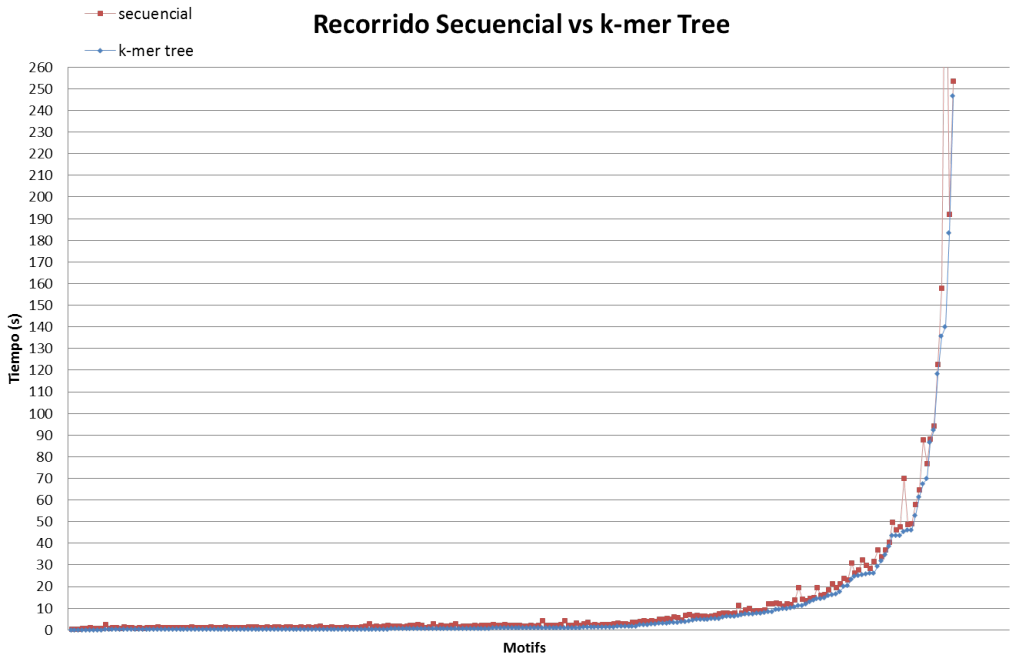

Figura 27: Comparación de tiempo del recorrido secuencial contra el uso de la estructura k-mer Tree con tiempo en segundos.

4.5. Comparación del algoritmo KTreeMotif con otros algoritmos

Además se programó un conjunto de los algoritmos explicados anteriormente, para una comparación de los resultados, sobre todo porque algunos no se pueden consultar en Internet y otros sí tienen página web para ejecutarlos pero para efectos de investigaciones almacenan los resultados de consultas previas y bases de datos, de esta forma si se les pide buscar un motif de una base de datos no realizan su algoritmo sino devuelven la consulta de la base de datos.

De esta forma se prefirió aprovechar el análisis de los algoritmos para programarlos según la explicación de los artículos. Los algoritmos que fueron programados son Consensus, Gibbs Sampler, MEME, SP-Star y Motif Sampler, este último tuvo ligeras

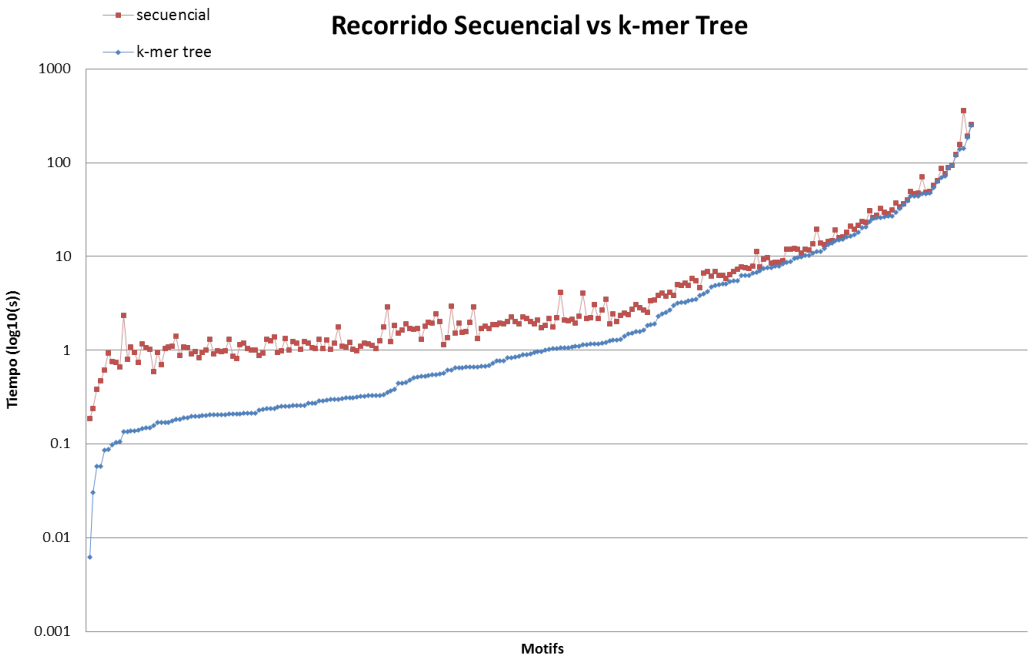

Figura 28: Comparación de tiempo del recorrido secuencial contra el uso de la estructura k-mer Tree con tiempo en logaritmo de segundos.

modificaciones personales debido a que el artículo no explicaba cómo se resolvían estos detalles.

Se eligieron estos algoritmos por ser las bases para la mayoría de los demás algoritmos de búsqueda de motifs, Consensus fue el primero con una orientación a patrones, MEME y Gibbs Sampler tienen una orientación a secuencias, pero mientras uno usa probabilidad para su metodología determinista, el otro es el primer algoritmo aleatorizado de una larga de lista de algoritmos de búsqueda de motifs con esta base para tener la ventaja de la velocidad, Motif Sampler tiene una de las mejoras a Gibbs Sampler más notorias, el uso de cadenas de Markov para un modelo y por esto fue agregado al conjunto, para observar cuánto mejora el Gibbs Sampler con modificaciones. Finalmente SP-Star es otro algoritmo basado en patrones que utiliza grafos en su idea básica, y presume de tener mucho mejores resultados que el antiguo Consensus; por pertenecer a un conjunto de algoritmos que utilizan los grafos o árboles para la resolución del problema fue considerado en este conjunto de algoritmos.

En las tablas 15 y 16 se muestran los resultados obtenidos por los distintos algoritmos y se distingue al igual que en las tablas anteriores que el Equal Score es el medidor más difícil de mejorar, ya que todos presentan un bajo índice. Además se nota que el Overreach Score es muy alto, e incluso superior al Equal Score, por lo que la mayor difícultad en la búsqueda de motifs está en encontrar entre los candidatos el correcto sin elegir un candidato con un Motif Score superior. Esto mismo nos dice que el Motif Score no es la mejor medida para representar el motif, pues el motif correcto no contiene

el mejor Motif Score entre los patrones en el conjunto de secuencias dado.

	Consensus	SP-Star	Gibbs Sampler
Overreach Score	1 — 0.4 %	124 — 52.8 %	117 — 49.8 %
Equal Consensus	28 — 11.9 %	154 — 65.5 %	155 — 66 %
Equal Score	4 — 1.7 %	102 — 43.4 %	103 — 43.8 %
nPC	0.528	0.899	0.900
sPC	0.783	0.977	0.978
Success Rate	0.259	0.716	0.708

Cuadro 15: Comparación de los algoritmos Consensus, SP-Star y Gibbs Sampler.

	Motif Sampler	MEME	kTree Motif
Overreach Score	113 — 0.4 %	93 — 39.6 %	121 — 51.5 %
Equal Consensus	157 — 11.9 %	156 — 66.4 %	158 — 67.2 %
Equal Score	106 — 45.1 %	108 — 46 %	105 — 44.7 %
nPC	0.902	0.903	0.901
sPC	0.977	0.977	0.975
Success Rate	0.730	0.716	0.723

Cuadro 16: Comparación de los algoritmos Motif Sampler, MEME y el algoritmo actual.

En las gráficas 29 y 30 se muestra la precisión de las pruebas hechas en los seis algoritmos para tener una mejor idea del resultado de cada algoritmo. El formato de las gráficas es muy similar a las pruebas anteriores, el motif correcto está indicado con los puntos negros y cada uno de los otros algoritmos se especifica en la leyenda de la gráfica.

En las conclusiones en la comparación de los algoritmos podemos señalar que el algoritmo Consensus es de los más básicos y primeros que trataron de lidiar con el problema, pero tiene una precisión muy baja. El algoritmo Motif Sampler como una versión más compleja del Gibbs Sampler obtiene mejores resultados pero no con una gran diferencia; para la complejidad que le aumenta al algoritmo básico, se puede precindir de estas mejoras y por eso mismo no se agregó el uso de cadenas de Markov al algoritmo diseñado. El MEME sigue siendo de los mejores algoritmos en la búsqueda de motifs y el algoritmo diseñado no se aleja mucho de sus resultados, aun así ambos tiene mucho que mejorar en cuestión de precisión.

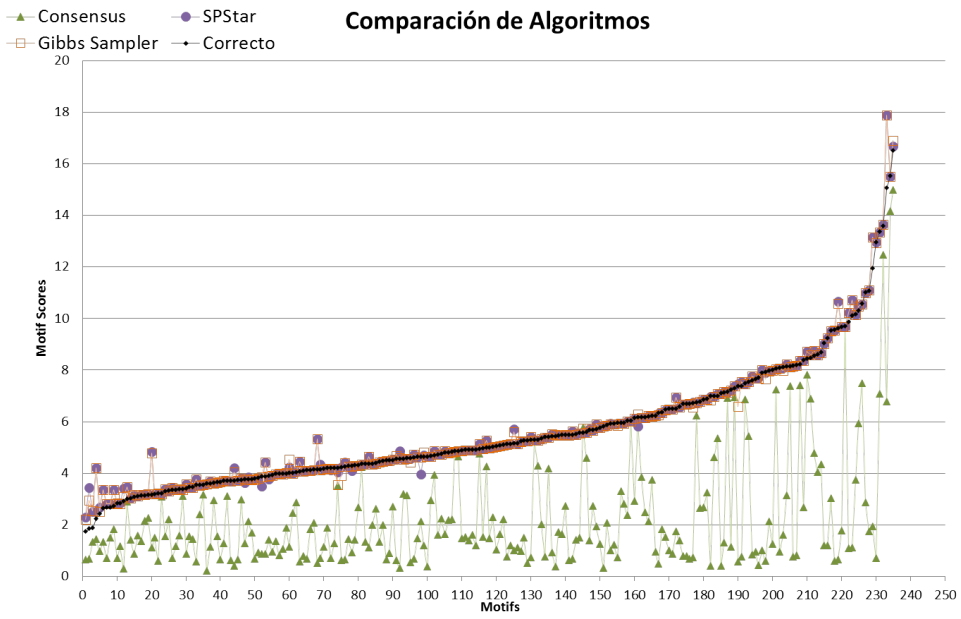

Figura 29: Resultado de los algoritmos Consensus (triángulos verdes), SP-Star (círculos morados), Gibbs Sampler (cuadros naranjas) y los resultados correctos (puntos negros).

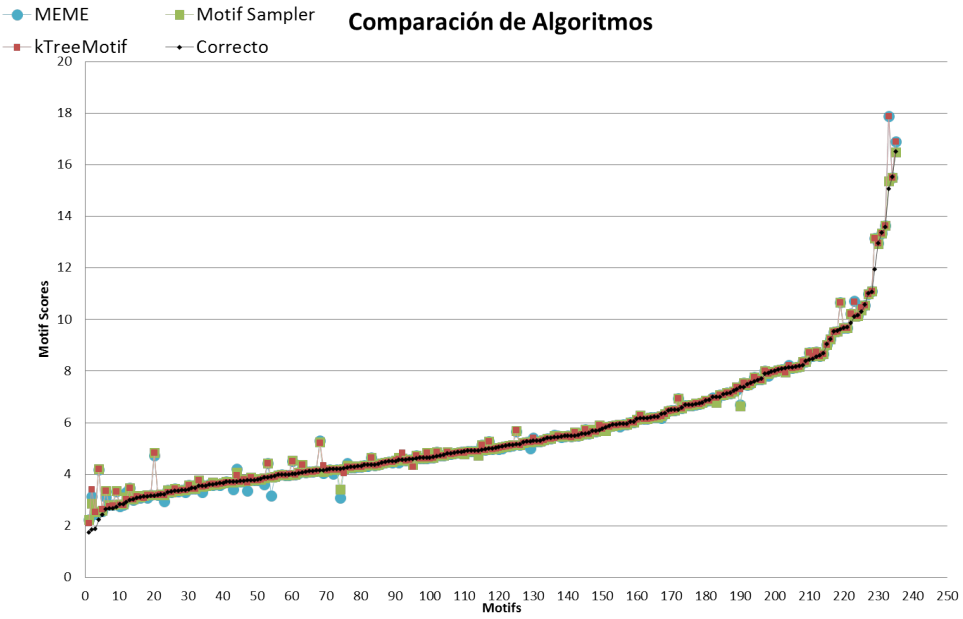

Figura 30: Resultado de los algoritmos Motif Sampler (cuadros verdes), MEME (círculos azules), KTreeMotif (cuadros rojos) y los resultados correctos (puntos negros). Esta gráfica puede verse como continuación de la gráfica anterior.

Capítulo 5: Conclusiones

5.1. Conclusiones del trabajo

1. Se encontró un resultado similar entre las metodologías determinista y estocástica. Como se vio, los principales algoritmos de búsqueda de motifs recurren a una de estas metodologías, el algoritmo MEME usa la determinista, obteniendo el mismo resultado en cada ejecución, mientras que el algoritmo Gibbs Sampler y sus derivados utilizan una metodología estocástica, por lo que es posible obtener resultados diferentes en cada ejecución, esto con el propósito de no entregar el mismo máximo local sino tomar diferentes caminos durante el algoritmo.

   Aun así al final los resultados de ambos métodos fueron muy similares, por lo que se pueden usar ambos algoritmos para verificar los resultados y estar más seguros de estos pero al momento de elegir entre una metodología o la otra. La única característica a destacar es que la metodología estocástica puede obtener resultados un poco mejores que la determinista al igual que resultados un poco inferiores, más allá de esto es no hay fuertes ventajas que apoyen a alguna de las dos.

2. Debido al tamaño del patrón la orientación a patrones no mejora mucho los resultados frente a la orientación a secuencias, sólo asegura la búsqueda del máximo global. Como se explicó anteriormente la orientación a patrones consiste en revisar que alguna combinación sea un patrón que aparece en cada secuencia, el tamaño promedio del motif hace que estas combinaciones posibles sean una cantidad computable y una orientación a considerar al diseñar el algoritmo, a diferencia de la orientación a secuencias que se suele utilizar.

   Esta orientación a patrones es una característica importante en el algoritmo diseñado, pues se busca evadir los máximos locales y asegurar que el resultado es el máximo global. Al momento de realizar las pruebas de comparación con los demás algoritmos se consiguió resultados muy similares, a pesar de tener esta otra orientación, pero en términos generales no marca una gran diferencia en precisión del resultado, de igual manera sirve para asegurar resultados correctos en un algoritmo de votación que lo incluya, por ser una metodología diferente con resultados a tomar en cuenta.

3. Para el algoritmo los mejores parámetros son la poda usando los Motif Scores y un z-score de 2.5, el refinamiento mediante Gibbs Sampler y las aportaciones hechas como se pudo concluir en la sección 4.4 de las pruebas. Las distancias incluyen por supuesto la estructura de k-mer Tree y las distancias ProbDistance y PWMDistance.

4. Características como los pseudoconteos, la reducción de la convergencia, la probabilidad del resto de la secuencia y la complejidad de las probabilidades de fondo influyen muy poco en la búsqueda del motif correcto.

   Cambiar la función para obtener los pseudoconteos y darles menor influencia, permitir que las iteraciones del último paso terminen más rápido a través de una mayor sensibilidad a tener valores similares, incluir la probabilidad de cada caracter externo al rango del motif de no ser parte del motif, o más complejo, cada combinación de caracteres externo al rango del motif de ser producto de la distribución normal de caracteres, fueron modificaciones al algoritmo que no cumplieron en añadir precisión al resultado, sólo hacían el algoritmo más lento y complejo sin aportar nada. Así que estas características fueron removidas y de ser tomado al cuenta al querer hacerle modificaciones al algoritmo o crear nuevos algoritmos; sobre todo para no buscar la mejora del algoritmo en esos detalles.

5. Actualmente es mucho más fácil encontrar el consenso o la zona donde se encuentra el motif, que encontrar todas las subcadenas exactas que lo contienen. Como se pudo ver en la comparación con otros algoritmos, en las tablas la mayoría tenía una muy buena precisión en obtener el consenso correcto y elevados índices nPC y sPC, y en las gráficas la distancia respecto al Motif Score correcto no era mucha en la mayoría de los casos, por lo que el problemas de precisión radica en que un número pequeño de subcadenas no era encontrado correctamente, en estos casos la subcadena encontrada aparecía a un par de caracteres desplazado de la subcadena correcta. lo cual seguía sin afectar mucho a obtener el consenso correcto.

   Este detalle de la precisión no sólo sucede en el algoritmo propuesto sino también en los algoritmos con que se comparó.

6. En la búsqueda de motifs la función más importante es el calificador de motif, la función objetivo, quien descarta unos candidatos sobre otros al buscar el correcto entre los generados por los algoritmos.

   Este el punto débil de los algoritmos de búsqueda de motifs y la razón de la baja precisión en las pruebas del apartado 4.5, pues la función objetivo que se usa actualmente, el Motif Score, puede encontrar el motif correcto y descartarlo por otro patrón incorrecto con un valor superior, esto es notorio en el valor del Overreach Score en las tablas, el cual es cercano al 50 %. El Overreach Score está considerando todos los resultados que cuentan con un Motif Score superior al motif correcto.

   Todos los algoritmos se ven influenciados por este detalle como se ve en las tablas de comparación de algoritmos, pues todos utilizan el Motif Score como función objetivo para elegir al mejor candidato a motif correcto.

5.2. Aportaciones del trabajo

A continuación se detallan las aportaciones que fueron descritas brevemente en la sección 1.5 y se incluye otra más.

1. Se desarrolló una estructura k-mer Tree que maneja de manera más rápida las subcadenas; esta estructura se puede implementar a otros algoritmos, siempre y cuando se aproveche ya que su construcción lleva un tiempo a tomar en consideración.

2. Se creó una función de distancia entre cadenas que funciona mejor ante la búsqueda de patrones, pues no sólo permite hacer comparaciones de distancia sino acercar los patrones significativos y alejar las cadenas que no son candidatos a patrones. Se recomienda tomarlo en cuenta al trabajar con distancias dentro de algoritmos de búsqueda de patrones.

3. Se creó una función de distancia más simple entre una PWM y una cadena y se comprobó que no afectaba negativamente al resultado final. Esta función puede utilizarse para otros futuros algoritmos de búsqueda de motifs.

4. Se consiguió un algoritmo híbrido de búsqueda de motifs que alcanza a los algoritmos actuales. Híbrido por ser la combinación de aportaciones de otros algoritmos, sobre todo de diferentes orientaciones, se incia con una orientación a patrones y se cambia a una orientación a secuencias, la cual es estocástica pero fácilmente se puede cambiar a determinista.

5. Se hizo una prueba de los algoritmos con una base de datos real más objetiva. La mayoría de las pruebas de algoritmos trabajan con datos artificiales y sólo muestran casos donde cierto algoritmo sale con mejores resultados, en este caso que los datos fueron extraídos de una base de datos real, se encontró fallas en los algoritmos, sobre todo en la función objetivo que funciona bien con patrones frecuentes implantados en secuencias artificiales pero no modela de manera correcta un patrón motif real de un conjunto de secuencias resultando en patrones mejores que el motif real.

5.3. Trabajo futuro

1. Desarrollar una mejora al algoritmo para la búsqueda de motifs sin conocer el tamaño.

   Existen algunos algoritmos que implementan esta característica que trabajan sobre otro algoritmo más simple; lo que hacen es probar con distintos tamaños en algún paso de las instrucciones mientras no se debilite mucho la calificación del patrón candidato. Es posible implementar esta mejora para el algoritmo diseñado para que sea más útil a casos reales donde no hay tanta información como en una base de datos.

2. Desarrollar una mejora al algoritmo que verifique la certeza del motif encontrado respecto a sus vecinos.

   Debido a que se encontró en varios casos el hecho de estar a pocas posiciones de distancia de la subcadena correcta en algunas de las secuencias, es posible mejorar la precisión del algoritmo verificando que las subcadenas de un patrón candidato produzcan un candidato más fuerte que otro patrón formado con subcadenas vecinas, ya sea cambiando todas las subcadenas a la vez, o sólo algunas.

   Para que esta aportación funcione y en verdad mejore la certeza del algoritmo es necesario implementarla junto con la siguiente idea.

3. Buscar un calificador que caracterice de mejor manera el motif y provoque un resultado más exacto en todos los algoritmos.

   La principal debilidad de los algoritmos actualmente sigue siendo la función objetivo, el Motif Score, si se implementa una función que modele de mejor manera el motif aumentará en gran medida la exactitud de cualquiera de los algoritmos, y a partir de ahí hacer las demás modificaciones al algoritmo para adaptarlo a una mayor diversidad de patrones en secuencias.

   Se propone crear una función que trabaje con las probabilidades interiores al motif y las probabilidades exteriores de una distinta manera, pues el principal problema pareciera ser cuando las secuencias son de tamaño cercano al del motif a buscar, las probabilidades exteriores serán muy cercanas a las interiores, esto es porque en estos casos no hay muchos caracteres de dónde obtener la probabilidad normal de cada base para de ahí obtener las subsecuencias que resaltan de esta normalidad; logicamente también se lidia con el caso contrario donde se tienen muchos caracteres externos al motif para formar una probabilidad normal que debería ser mejor aprovechada.

Bibliografía

Bailey, T. L. (2007). *Discovering sequence motifs*, volume Methods in molecular biology: comparative genomics, chapter 17, pages 271–292. Humana Press.

Bailey, T. L. and Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2):51–80.

Baxenavis, B. F. F. O. . A. D., editor (2005). *Bioinformatics: a practical guide to the analysis of genes and proteins*. John Wiley & Sons, Inc.

Das, M. K. and Dai, H.-K. (2007). A survey of dna motif finding algorithms. *BMC Bioinformatics*, 8(7):1–13.

Hertz, G. Z. and Stormo, G. D. (1999). Identifying dna and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7):563–577.

Hu, J., Li, B., and Kihara, D. (2005). Limitations and potentials of current motif discovery algorithms. *Nucleic Acids Res*, 33(15):4899–4913.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science*, 262:208–214.

Luscombe, N. M., Greenbaum, D., Gerstein, M., et al. (2001). What is bioinformatics? a proposed definition and overview of the field. *Methods of information in medicine*, 40(4):346–358.

Matys, V., Fricke, E., Geffers, R., Gößling, E., Haubrock, M., Hehl, R., Hornischer, K., Karas, D., Kel, A. E., Kel-Margoulis, O. V., Kloos, D.-U., Land, S., Lewicki-Potapov, B., Michael, H., Münch, R., Reuter, I., Rotert, S., Saxel, H., Scheer, M., Thiele, S., and Wingender, E. (2003). Transfac®: transcriptional regulation, from patterns to profiles. *Nucleic Acids Research*, 31(1):374–378.

Narayanan, E. K. . A. (2005). *Intelligent bioinformatics. The application of artificial intelligence techniques to bioinformatics problems*. John Wiley & Sons, Ltd.

Pavesi, G., Mauri, G., and Pesole, G. (2001). An algorithm for finding signals of unknown length in dna sequences. *Bioinformatics*, 17(suppl 1):S207–S214.

Pevzner, P. A. and Sze, S.-H. (2000). Combinatorial approaches to finding subtle signals in dna sequences. In Bourne, P. E., Gribskov, M., Altman, R. B., Jensen, N., Hope, D. A., Lengauer, T., Mitchell, J. C., Scheeff, E. D., Smith, C., Strande, S., and Weissig, H., editors, *ISMB*, pages 269–278. AAAI.

Sagot, M.-F. (1998). Spelling approximate repeated or common motifs using a suffix tree. In Lucchesi, C. and Moura, A., editors, *LATIN'98: Theoretical Informatics*, volume 1380 of *Lecture Notes in Computer Science*, pages 374–390. Springer Berlin Heidelberg.

Salgado, H., Peralta-Gil, M., Gama-Castro, S., Santos-Zavaleta, A., Muñiz-Rascado, L., García-Sotelo, J. S., Weiss, V., Solano-Lira, H., Martínez-Flores, I., Medina-Rivera, A., Salgado-Osorio, G., Alquicira-Hernández, S., Alquicira-Hernández, K., López-Fuentes, A., Porrón-Sotelo, L., Huerta, A. M., Bonavides-Martínez, C., Balderas-Martínez, Y. I., Pannier, L., Olvera, M., Labastida, A., Jiménez-Jacinto, V., Vega-Alvarado, L., del Moral-Chávez, V., Hernández-Alvarez, A., Morett, E., and Collado-Vides, J. (2013). Regulondb v8.0: omics data sets, evolutionary conservation, regulatory phrases, cross-validated gold standards and more. *Nucleic Acids Research*, 41(D1):D203–D213.

Sandelin, A., Alkema, W., Engström, P., Wasserman, W. W., and Lenhard, B. (2004). Jaspar: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research*, 32(suppl 1):D91–D94.

Schneider, T. D., Stormo, G. D., Gold, L., and Ehrenfeucht, A. (1986). Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology*, 188(3):415–431.

Stormo, G. D. and Hartzell, G. W. (1989). Identifying protein-binding sites from unaligned dna fragments. *Proceedings of the National Academy of Sciences*, 86(4):1183–1187.

Sung, W. K. (2010). *Algorithms in bioinformatics: a practical introduction*. Chapman & Hall/CRC mathematical and computational biology.

Thijs, G., Marchal, K., Lescot, M., Rombauts, S., Moor, B. D., Rouzé, P., and Moreau, Y. (2002). A gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *Journal of Computational Biology*, 9(2):447–464.

Tompa, M., Li, N., Bailey, T. L., Church, G. M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y., Kent, W. J., Makeev, V. J., Mironov, A. A., Noble, W. S., Pavesi, G., Pesole, G., Regnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C., and Zhou, Z. (2005). Assessing computational tools for the discovery of transcription factor binding sites. *Nature biotechnology*, 23(1):137–144.

Wong, L., editor (2004). *The practical bioinformatician*. World Scientific Publishing Co.

Xiong, J. (2006). *Essential Bioinformatics*. Cambridge University Press.