UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MAESTRÍA EN CIENCIAS EN COMPUTACÍON AVANZADA Y ELECTRÓNICA

TESIS

Plataforma computacional para el descubrimiento de motifs en secuencias de ADN empleando técnicas de minería de patrones frecuentes

Para obtener el grado de
Maestro en Ciencias en Computación Avanzada y Electrónica

PRESENTA

LCC. Victor Ignacio Sobrevilla Solis

Director (a)

Dra. Anilú Franco Arcega

Codirector (a)

Dr. Luis Heriberto García Islas

Comité tutorial

Dr. Omar López Ortega
Dra. Anilú Franco Arcega
Dr. Esteban Rueda Soriano
Dr. Luis Heriberto García Islas

Mineral de la Reforma, Noviembre 2022

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Área Académica de Computación y Electrónica
Department of Electronics and Computer Science

Mineral de la Reforma Hidalgo, a 26 de octubre de 2022

Número de control: ICBI-AACyE/1615/2022
Asunto: Autorización de impresión de tema de tesis.

MTRA. OJUKY DEL ROCIÓ ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

El Comité Tutorial de la tesis titulada "**Plataforma computacional para el descubrimiento de motifs en secuencias de ADN empleando técnicas de minería de patrones frecuentes**", realizado por el sustentante **LCC. Víctor Ignacio Sobrevilla Solís** con número de cuenta 314246, perteneciente al programa de la Maestría en Ciencias en Computación Avanzada y Electrónica, una vez que se ha revisado, analizado y evaluado el documento recepcional de acuerdo a lo estipulado en el artículo 110 del Reglamento de Estudios de Posgrado, tiene a bien extender la presente.

AUTORIZACIÓN DE IMPRESIÓN

Por lo que el sustentante deberá cumplir con los requisitos del Reglamento de Estudios de Posgrado y con lo establecido en el proceso de grado vigente.

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

Atentamente
"Amor, Orden y Progreso"

Dra. Anilu Franco Arcega
Directora de Tesis

Dr. Luis Heriberto García islas
Codirector de Tesis

Instituto de Ciencias Básicas e Ingeniería
Área Académica de Computación y Electrónica

Comité Tutorial

Dr. Omar López Ortega	Presidente	UAEH
Dr. Esteban Rueda Soriano	Secretario	UAEH
Dra. Anilu Franco Arcega	Vocal	UAEH
Dr. Luis Heriberto García Islas	Suplente	UAEH

AFA/APL

Ciudad del Conocimiento
Carretera Pachuca-Tulancingo km 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo, México. C.P. 42184
Teléfono: +52 (771) 71 720 00 ext. 2250, 2251
Fax 2109
aacye_icbi@uaeh.edu.mx

Agradecimientos

Esta investigación ha sido extenuante y requerido mucho empeño, no solo mía, sino de todos aquellos que han participado directa o indirectamente. Sin su apoyo este trabajo de tesis no se hubiera podido cumplir.

Primero quiero agradecer a mis padres por apoyarme incondicionalmente, no solo en estos dos años sino en todo momento de mi vida, y sacarnos adelante a mis hermanos y a mí, independientemente de las circunstancias. Sepan que todo ha valido la pena.

A mis hermanos Alex, Alfredo y Cesar junto a mis cuñadas Ivon, Malú e Itzel por acompañarme en este recorrido de dos años en los que se vivieron distintas aventuras y en las cuales me brindaron su cariño, apoyo y confianza. A Daniel por ser ese hermano adicional. También a Samuel, Santiago y el próximo sobrino que viene en camino, por llenar de alegría la vida de todos en la familia. Verlos felices es lo mejor.

A mis abuelos Samuel y Tita por compartir sus vivencias y su sabiduría para ser una mejor persona.

Tambien quiero agradecer a mis compañeros y amigos por compartir en aula de clases, experiencias, conocimiento y su amistad.

Un agradecimiento especial a mis asesores de tesis, la Dra. Anilú Franco Arcega y el Dr. Luis Heriberto Garcia Islas, por permitirme trabajar con ustedes en este proyecto de tesis y formar parte de su línea de investigación, depositando toda su confianza y apoyo en mí incluso cuando las situaciones no me favorecían. Además de bridarme de sus conocimientos, experiencias, consejos y llamadas de atención no solo para este trabajo, sino también para desarrollarme en el plano académico, profesional y personal.

De igual forma al comité tutorial, el Dr. Omar López Ortega y el Dr. Esteban Rueda Soriano por brindarme de su retroalimentación, ideas, perspectivas y consejos para complementar este proyecto de tesis. Así mismo a la Dra. María de los Ángeles Alonso Lavernia (Q.E.P.D)

por ofrecer su conocimiento, su apoyo, y sus consejos que aporto en su momento a este trabajo. Desde donde se encuentre, creo que está feliz de verme finalizando este proyecto.

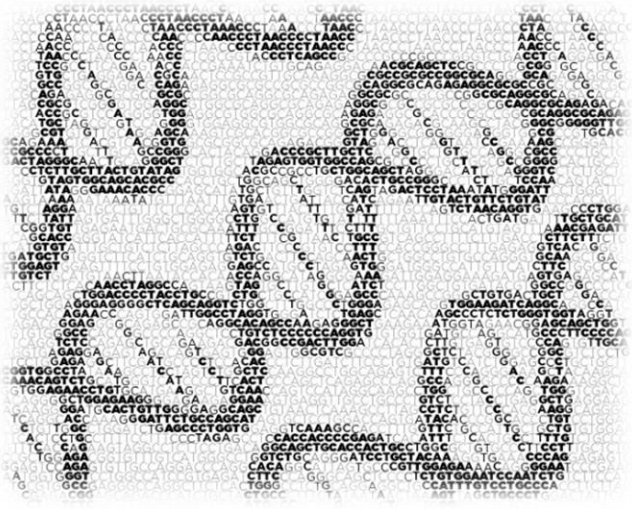
Al cuerpo académico de la maestría en computación avanzada y electrónica por compartir sus conocimientos y experiencias que me han servido a lo largo de mi estadía en la maestría.

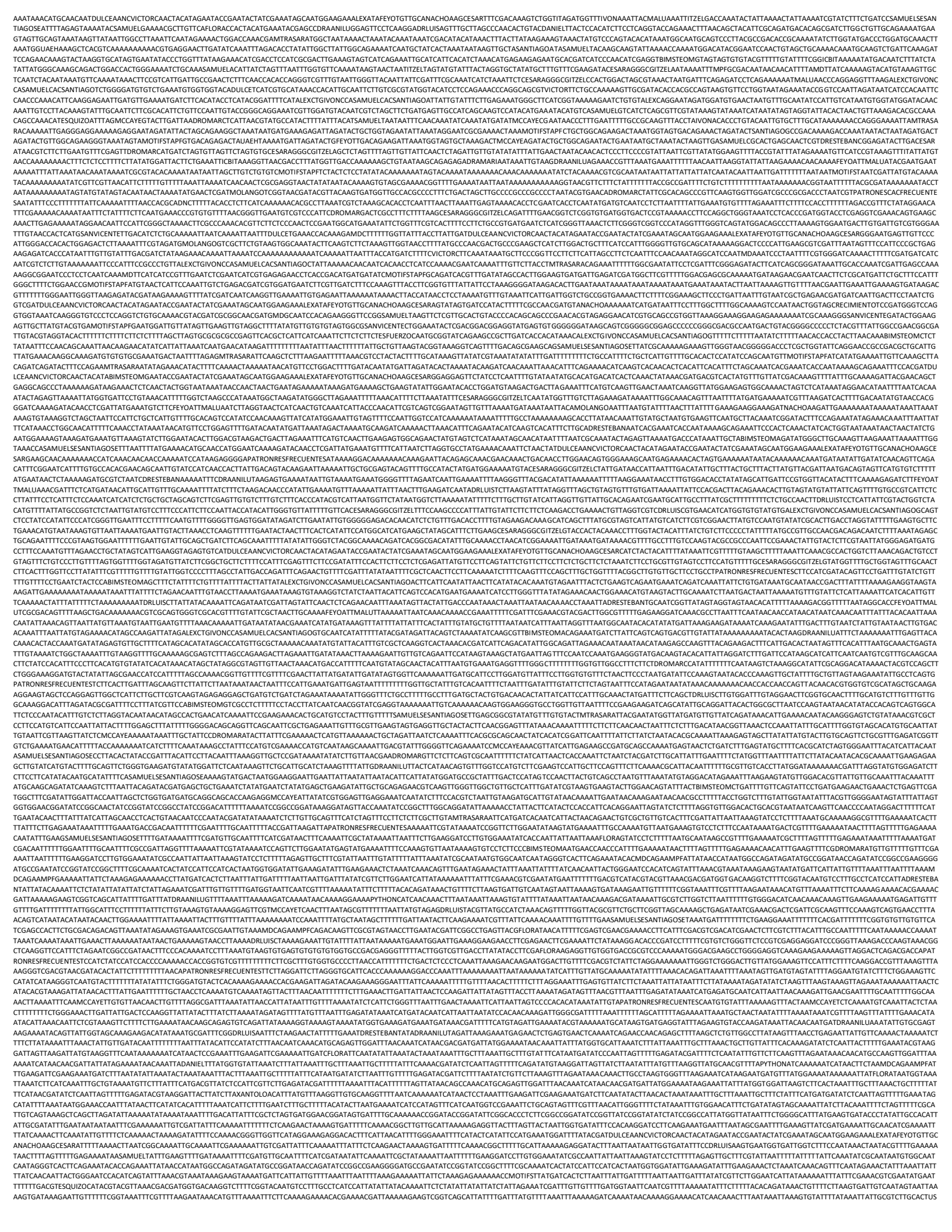
Agradezco a mi madrina Flora por darme su apoyo en el ámbito profesional, así como en lo personal, además de compartir su conocimiento.

Una mención especial a la Mtra. Sara Centeno Mendoza por ser una mentora profesional, y guiarme e inculcarme el amor por la computación y la tecnología.

Muchas gracias a todos, inclusive aquellos que no he nombrado, pero me han apoyado, me dieron ánimo para salir adelante y cumplir mis metas.

Victor Ignacio Sobrevilla Solis





Índice general

Lista de Figuras

Lista de Tablas

Resumen

La computación ha logrado desarrollar diferentes herramientas con gran potencial que permiten a diversas áreas de aplicación descubrir conocimiento relevante que apoya la toma de decisiones es estos ámbitos. Un caso particular es la Biología, y específicamente, la especificación de motifs, los cuales permiten dar indicios sobre el comportamiento de organismos vivos. Este trabajo de tesis presenta el desarrollo de una plataforma computacional enfocada al descubrimiento de motifs utilizando algoritmos de minería de patrones frecuentes. Se propone el uso de este tipo de procedimientos debido a que consisten en buscar elementos recurrentes en grandes volúmenes de datos a través del manejo eficiente de estructuras de datos, reducción en el espacio de búsqueda o en el conteo de elementos. Para el desarrollo de la plataforma, se empleó una metodología que permitió una construcción gradual de ella, considerando cinco fases principales:

- Entrada de secuencias de ADN
- Pre-procesamiento de datos
- Análisis de datos biológicos
- Estructuración de resultados
- Despliegue de resultados

Para la creación de dicha plataforma, se utilizaron diferentes herramientas de codificación como HTML, CSS, JavaScript y Python. Con estas acciones se logró desarrollar una plataforma computacional web, la cual posee una interfaz sencilla, donde el usuario puede utilizar diferentes algoritmos para analizar de forma eficiente grupos de secuencias de ADN de alto volumen, con la finalidad de hallar sus respectivos patrones frecuentes y, posteriormente, el conjunto de motifs asociado a ese grupo de secuencias.

La obtención de patrones frecuentes en este trabajo se realiza implementando una mejora en su identificación dentro de un grupo de secuencias. Para la obtención de motifs, se propone en esta investigación una nueva forma de búsqueda, comprobando experimentalmente que obtiene más motifs que otras estrategias reportadas en la literatura, empleando un menor tiempo.

Introducción

La aplicación de la computación en diferentes campos de la ciencia y conocimiento humano ha permitido tener un gran avance en el desarrollo tecnológico de la humanidad en las últimas décadas. Esto es gracias a la facilidad de automatizar tareas o resolver problemas e implementar soluciones a través de la creación de programas computacionales, también llamado software, que contienen las instrucciones necesarias y métodos computacionales para ejecutar correctamente alguna tarea en un dispositivo electrónico, con el objetivo de obtener y procesar datos de interés de esta tarea o problemática.

La diversificación del uso de software para diferentes propósitos, así como su facilidad de uso y escalabilidad, ha permitido a más personas emplear software como una herramienta de trabajo, entretenimiento, aprendizaje, entre otros usos, y a su vez acrecentando la captación de datos. Hardy y Williams mencionan que la información digital generada en el planeta en el 2011 fue de alrededor de 1800 exabytes (EB) (1 EB = 10^{18} bytes) [14]. En el caso particular de diferentes aplicaciones como Twitter, aproximadamente se producen más de 12 terabytes de información (TB) (1 TB = 10^{12} bytes); en Facebook se agregan cerca de 15 TB de datos; y dentro de los servidores de Google se genera alrededor de 1 Petabyte de datos (PB) (1 TB = 10^{15} bytes) cada hora, durante el 2014 [21]. Ante estos hechos, dentro de la computación se desarrolló una rama que cambiaría el tratamiento de la información digital, llamada Minería de Datos (MD). Esta disciplina está enfocada a emplear diferentes técnicas inteligentes en grandes cantidades de datos para descubrir conocimiento útil, nuevo y valioso [13]. Los datos que puede procesar pueden ser del tipo alfanumérico, binario o cadenas de caracteres extraídos de diferentes acciones u objetos de la vida real, tales como transacciones de compra, descripción de productos, transferencias bancarias, expedientes médicos, calificaciones de alumnos, experimentos científicos, mediciones geográficas, entre otras, que en gran parte están contenidos en bases de datos.

El interés de emplear técnicas de MD a datos provenientes de distintos ámbitos, ha llevado a numerosos centros de investigación, gobiernos y empresas de diferentes ramos a invertir en infraestructura, así como en el desarrollo de herramientas

a medida, para la obtención de conocimiento de gran importancia. Para lograr esto, se necesitan tener los conocimientos necesarios para interpretar la información obtenida, requiriendo el apoyo de uno o varios especialistas en el tema. Por ejemplo, la bioinformática, que se considera como la intersección de las áreas de ciencias computacionales, biología, matemáticas y estadística, tiene como finalidad implementar soluciones computacionales y estadísticas a problemas biológicos teóricos o prácticos [29]. Una de las tareas importantes dentro de la bioinformática es el análisis del ADN, debido a que es la "plantilla" molecular que contiene extensa información genética necesaria para la creación y funcionamiento de cualquier ser vivo [36]. A su vez, en el ADN se pueden hallar subsecuencias recurrentes de corta longitud llamados motifs, los cuales pueden indicar funciones biológicas, sitios de unión para proteínas como factores de transcripción, e incluso son parte del proceso de formación del ARN [8].

Para realizar la búsqueda y hallazgo de motifs dentro de una o varias secuencias de ADN de manera automatizada, se emplea software especializado que integra técnicas computacionales basadas en el alineamiento múltiple de secuencias, lo que la hace importante debido a que puede arrojar una identificación cercana a su contra parte biológica. Sin embargo, el proceso de descubrimiento de motifs tiende a ser extenso por las operaciones computacionales que realiza, especialmente en el caso de que las secuencias de ADN sean muy extensas o sean un gran grupo de secuencias de ADN a analizar, perjudicando el tiempo de ejecución ya que se vuelve más tardado de lo normal, se puede no concluir el proceso o incluso existe la posibilidad de perdida de la información [4]. Para atacar este problema, las plataformas limitan su entrada de datos, ya sea en la longitud o en el número de elementos en un grupo de secuencias de ADN. Esto significa que el software enfocado al descubrimiento de motifs dentro de secuencias de ADN contiene procesos que no están diseñados para grandes volúmenes de datos.

Los métodos dirigidos al descubrimiento de motifs en secuencias de ADN que están actualmente implementados en distintos software, no están diseñados para analizar conjuntos de datos extensos. Teniendo en cuenta que un motif es un elemento repetitivo de corta longitud, que se puede hallar en n número de secuencias de ADN de l longitud de nucleótidos, se puede desarrollar software que implemente métodos diseñados para procesar conjuntos de datos extensos. Lo anterior da la oportunidad a emplear y/o proponer diferentes estrategias que sean acordes para realizar esta tarea, como los algoritmos de minería de patrones frecuentes, los cuales se dedican a la detección de elementos repetitivos en grandes volúmenes de información.

Considerando que las secuencias de ADN pueden contener una cantidad inmensa de caracteres, y el motif es una subsecuencia recurrente de corta longitud, y de gran relevancia en la bioinformática y la biología, se plantea el uso de algoritmos que permitan la identificación de estas subsecuencias, como los algoritmos de patrones frecuentes. De ahí que surge la siguiente hipótesis:

Usando un conjunto de patrones frecuentes como base, es posible desarrollar un algoritmo que descubra un conjunto de motifs asociado a un grupo de secuencias de ADN, que obtenga motifs más rápido que algoritmos reportados en literatura. Además, el algoritmo resultante puede ser implementado en una plataforma computacional.

Se plantea el siguiente objetivo general para el cumplimiento de este trabajo.

Desarrollar una plataforma computacional con la implementación de algoritmos de minería de patrones frecuentes en grandes volúmenes de datos biológicos para el descubrimiento de motifs en secuencias de ADN de una forma eficiente, adecuada y comprensible.

Con el fin de alcanzar el objetivo general planteado, se definen una serie de objetivos específicos que engloban las diferentes actividades a desarrollar. Esos objetivos son:

- Revisar los conceptos relacionados con el caso práctico a través del estudio de la literatura, para establecer un marco de referencia respecto a la definición y uso de los motifs en secuencias de ADN.
- Diseñar estrategias de implementación de la plataforma a desarrollar por medio de la planeación de sus componentes, haciendo uso de metodologías ágiles de software para la construcción gradual de la misma.
- Implementar algoritmos de identificación de patrones frecuentes para el tratamiento de secuencias de ADN a través de la investigación y codificación de ellos.
- Desarrollar mecanismos para procesar la entrada de datos biológicos en formatos estandarizados, así como para la visualización del conjunto de motifs obtenidos, por medio de módulos que permitan su configuración y despliegue de resultados obtenidos por los algoritmos.
- Integrar los algoritmos y los mecanismos desarrollados para la identificación de motifs tomando como base la entrada de secuencias de ADN por diversos medios.

Estructura del documento

Este documento está compuesto por cinco capítulos. En el capítulo uno, se presenta el marco teórico, el cual expone el conocimiento relacionado a este proyecto. Entre los temas principales que trata están: Ingeniería de software, Minería de datos y Bioinformática. Dentro del capítulo dos, llamado trabajos relacionados, se exponen las características de aquellos algoritmos convencionales de descubrimiento de motifs.

En el tercer capítulo, llamado método de identificación de patrones basado en índices, se presenta un método nuevo de reconocimiento de patrones frecuentes en un grupo de secuencias de ADN, que basa sus procesos en el algoritmo Basado en índices. Por su parte, el capítulo cuatro describe una nueva estrategia para la obtención de motifs de una o varias secuencias de ADN, basada en el conjunto de patrones frecuentes encontrados en dichas secuencias. La plataforma de identificación de motifs desarrollada en este trabajo se introduce en el capítulo cinco, en el que se muestra el proceso de creación de la plataforma computacional dentro de las primeras cinco fases de desarrollo y el detalle del funcionamiento de la plataforma computacional resultante. Finalmente, se presentan las conclusiones alcanzadas en este proyecto, así como el trabajo futuro que se plantea seguir en esta línea de investigación.

Capítulo 1

Marco teórico

En el primer apartado de este capítulo se presentan los fundamentos de la ingeniería de software para el desarrollo de herramientas computacionales. En el segundo apartado se habla sobre la utilidad de la minería de datos para la revelación de detalles importantes que no se muestran a simple vista dentro de grandes volúmenes de información. Además, se expone el uso minería de patrones frecuentes como una herramienta que se encuentra presente en la minería de datos y cómo se emplea en casos donde surge la necesidad de obtener normas repetitivas. En el apartado tres se describe el uso de la bioinformática y en qué consiste, también se habla acerca de secuencias de ADN como un elemento que posee información biológica, junto con los procesos en los que está implicado. En el cuarto y último apartado, se presenta la identificación y utilidad de los motifs dentro de las secuencias de ADN.

1.1. Ingeniera de software

De manera general, la composición de algún *software* (programa de computadora) u algún otro producto computacional no solo consiste del trabajo de codificación, sino también de entender el caso de uso con base en las necesidades del problema, el tiempo de desarrollo, integración, pruebas, mantenimiento, entre otras tareas relacionadas con la planificación y ejecución de un proyecto de fabricación de software. Esta planeación se realiza con la finalidad de que todos los miembros involucrados en un proyecto tengan claro el propósito del programa a desarrollar, considerando en que ámbito se empleará y los usuarios a los que es dirigido. A esto se le llama Ingeniería de Software (IDS).

Pantaleo y Rinaudo comentan que "*IDS es la disciplina de la computación que se enfoca en el estudio de la elaboración, ejecución y manejo del software. Esto incluye la participación del cliente y/o los usuarios finales desde la identificación de la problemática hasta la elaboración, pruebas e implementación del sistema*" [27]. Por lo que esta disciplina se

considera pieza fundamental en la elaboración de todo tipo de software, ya sea, simple, complejo, empresarial, institucional, administrativo, científico, etc.

Esto se debe a que el uso de cualquier aparato electrónico como una computadora, tablet, smartphone, entre otros dispositivos, emplean software para su funcionamiento. Por ende, esta herramienta electrónica está presente en todas las ramas del conocimiento, por ejemplo (i) una infraestructura nacional eléctrica es gestionada por medio de sistemas computacionales, (ii) la administración de los impuestos se realiza mediante un sistema en línea, (iii) la gestión aeronáutica en un aeropuerto se lleva a cabo con el uso de un sistema en línea, (iv) sistemas computacionales que manejan información detallada sobre especies de animales de un área determinada, entre otros usos que se les puede dar a los programas informáticos [35].

Para la construcción de cualquier software se tiene que tener en cuenta que se ejecutan varias actividades, las cuales dependen del enfoque y tamaño del software a desarrollar. Por lo general, se siguen cuatro procesos fundamentales, como lo plantea Sommerville [35]:

1. ***Especificación:*** Se definen las funcionalidades del software y sus limitantes de operación.
2. ***Desarrollo:*** Una vez conocidas las especificaciones del software, este se fabrica.
3. ***Validación:*** Se verifica que el software cumpla con las necesidades del cliente.
4. ***Evolución:*** Se conocen las nuevas necesidades del cliente y se agregan al software ya existente.

Estas actividades tienden a ser complicadas, ya que involucran subtareas que abarcan desde la planeación del proyecto, recolección de requerimientos, diseño, codificación, pruebas, entre otros. Cabe mencionar que también se deben tener en cuenta los alcances y limitaciones del proyecto, ya que si no se establecen estas pautas, el proyecto posiblemente podría fracasar o llevarlo a un objetivo erróneo. También se involucra a un equipo de personas que toman diferentes roles en el proyecto, algunos por ejemplo son: [35]:

- ***Project Manager***
- ***Administrador de configuración***
- ***Arquitectos de software***
- ***Diseñadores de software***

- ***Programadores***
- ***Testers***

Cada miembro del equipo de desarrollo tiene asignadas tareas correspondientes a su rol, las cuales se establecen antes, durante y después del proyecto de desarrollo de software.

Con las pautas establecidas dentro del equipo de desarrollo, se comienza a realizar la obtención de especificaciones que definirán las funcionalidades del proyecto como la interacción de los usuarios, el tipo de entrada de datos, los cálculos a realizar, la conexión a bases de datos, la interacción con otros sistemas, etc. Estos se pueden obtener de entrevistas, diagramas o documentos que escriban la funcionalidad que se desee en el programa; por lo general, el *Project Manager* es el encargado de recopilar toda esta información, y de que se traduzca en tareas con prioridad para asignarlas a los miembros restantes del equipo, siendo útiles estas actividades durante las siguientes fases. En la etapa de desarrollo, los arquitectos de software se encargan de extraer los requerimientos para maquetar y estructurar todas la funcionalidades del software para que los diseñadores creen las interfaces gráficas del software, empleado herramientas especializadas, y los programadores construirán las funcionalidades internas utilizando diferentes herramientas, como un lenguaje de programación, IDE's (Entorno de Desarrollo Integrado), librerías especializadas, API's (Interfaz de programación de Aplicaciones), hardware especializado, etc. Conforme se vayan construyendo las partes funcionales del software, los *testers* o probadores se encargan de verificar que esas piezas del sistema funcionan de manera correcta [10]. Estas fases se pueden repetir y evolucionar dependiendo de diferentes circunstancias, como estipulaciones entre el cliente y el equipo de software o el tiempo de vida del producto.

En la práctica puede llegar a ser complejo, ya que se suelen tomar decisiones difíciles que pueden o no afectar al proyecto, además de diferentes factores a considerar como la magnitud, el número de miembros del equipo, el tiempo a desarrollar, el tiempo para la entrega de producto(s), la experiencia del equipo, etc. Con lo anterior, se puede decir que no existe un proceso preestablecido para desarrollar todo tipo de software, pero dentro de la IDS existen metodologías que permiten especificar las actividades y organizar los procesos que se deben llevar a cabo durante la construcción del software, cumpliendo con las pautas y restricciones establecidas, tal como lo mencionan Pantaleo y Rinuado [27].

Existen diferentes metodologías de creación de software que se clasifican en dos principales categorías: las metodologías conducidas por planes y las metodologías

ágiles. Cada estrategia de trabajo tiene diferente enfoque: las conducidas por planes son ideales para proyectos que ya han sido desarrollados, se tiene uno o varios antecedentes, o bien el enfoque del software es muy concreto, como el caso de software de puntos de venta o inventarios de productos. Por el contrario, los procedimientos de trabajo ágil están enfocados a proyectos con cambios, donde la idea del producto pueda ser volátil, además de que se puede aplicar a desarrollos que sean novedosos como herramientas de aprendizaje, plataformas personalizadas, software científico, entre otros.

1.1.1. Metodologías conducidas por planes

Desde los años 50, los programas de computadora se construían para resolver un problema específico y eran utilizados por el mismo desarrollador. Esto se consideraba un trabajo artesanal que llevaba cierto tiempo realizar, debido principalmente al hardware que tenían en ese momento. Con el paso de los años, entre las décadas de 1960 y 1970, la tecnología avanzó junto con la complejidad de los programas que pasaban de ser algo más "casero" para un solo usuario, a sistemas de administración con multiusuarios que necesitaban mantenimiento y desarrollos propios [27]. Fue también la época donde la gente involucrada en el proceso de desarrollo de software necesitaba reglas de trabajo para tener mayor control en los procesos. Ante esto surgieron los primeros marcos de trabajo, principalmente los inspirados en la ingeniera de los sistemas militares [35]. Este tipo de metodologías tienen procesos estrictos y burocráticos como la creación y seguimiento de planes, o la documentación previa a todas y cada una de las tareas que se llevan a cabo, lo que dificulta realizar cambios durante el desarrollo [27]. Algunos de estos modelos de trabajo que están diseñados bajo estas ideologías son:

- Cascada
- Prototipado
- DRA (Desarrollo Rápido de Aplicaciones)
- Incremental
- Espiral

Cabe mencionar que a estas metodologías se les conoce como *Tradicionales*. Décadas después, con la llegada del nuevo mileno y más avances tecnológicos, los sistemas computacionales habían ingresado a satisfacer necesidades en otras áreas como la construcción, la economía, el gobierno, el transporte, la administración, entre otras. Esta diversidad causó que se crearan más metodologías de desarrollo, con la finalidad de que fuesen mas fluidas en los procesos y con más flexibilidad en los cambios.

1.1.2. Metodologías Ágiles

Debido al gran crecimiento de la tecnología computacional, aparecieron nuevas herramientas para programación como lenguajes o librerías. Los métodos tradicionales de desarrollo de software presentaban algunas deficiencias como la no tolerancia a los cambios en los procesos y no adaptabilidad del mismo. Por estas razones, surgieron nuevas metodologías que buscaban cubrir las necesidades anteriormente mencionadas, dando paso a la creación de nuevas metodologías llamadas Ágiles. Las ideas de estos marcos de trabajo están descritas dentro del *manifiesto ágil* [27]. Los principales enfoques de este manifiesto son [9]:

- Los individuos y las interacciones entre ellos son más importantes que los procesos o las técnicas a utilizar.
- La prioridad es crear un producto computacional funcional.
- Son más importantes las aptitudes y la fortaleza ante los cambios que apegarse a un plan.

Dentro de las principales metodologías ágiles se encuentran:

- Extrem Programming (XP)
- Lean Development software
- Crystal
- Feature Driven Development
- SCRUM

Siendo SCRUM la metodología ágil que más se emplea en la industria del desarrollo de software. Y esto se debe a sus diferentes características que se enfocan en visualizar el progreso del proyecto y facilidad de implementación. En la siguiente sección, se presenta a detalle esta metodología.

SCRUM

SCRUM fue creada por Ken Schwaber, Jeff Sutherland y Mike Beedle [25]. El nombre de esta metodología se debe a una formación de Rugbi del mismo nombre, donde los jugadores de un equipo se agrupan y atenazan entre sí uniendo fuerzas para empujar a un obstáculo, en este caso, al equipo adversario [27].

Para el funcionamiento de la metodología se necesita un equipo de personas, a las cuales, a cada una se le asigna un rol para realizar tareas especificas. Estos roles pueden ser [27]:

- Product Owner: Persona que contiene los conocimientos del negocio o problemática.
- Scrum Master: Coordinador y facilitador del equipo. A su vez, tiene más contacto con el Product Owner para acordar la operatividad del proyecto.
- Team: Equipo de desarrollo conformado por programadores que se encargarán de la construcción del software.
- Product Manager: Responsable de administrar el proyecto.

En la ejecución de SCRUM, es importante tener en cuenta la creación de diferentes activos para el funcionamiento de la metodología, que contendrán aspectos importantes como los requerimientos, las tareas a realizar y la productividad. Estos son [27]:

- Product Backlob: Listado de requerimientos que son priorizados con anterioridad.
- Sprint Backlog: Subconjunto de requerimientos que se dividen en tareas y se realizan en un ciclo (sprint).
- Burndown Char: Diagrama de productividad de un sprint o el proyecto en general.

SCRUM emplea iteraciones llamadas *sprint*, en las que se realizan un conjunto de tareas previamente planeadas y asignadas en un lapso de tiempo establecido, que generalmente es de una a cuatro semanas, dando como resultado un producto funcional. Estos *sprints* se realizan de forma iterativa, y cada iteración tiene diferentes tareas que incrementan las funcionalidades del producto [25]. Para esto, se sigue la serie de actividades que se presentan a continuación [27]:

- Proyect Planning: Proceso de planificación del proyecto.
- Sprint Planning: Proceso de planificación del Sprint (iteración).
- Sprint: Iteración de duración fija.
- Daily Scrum: Reunión diaria de corta duración entre el Team y el Scrum Master en que se revisan las actividades del día.
- Release: Código funcional.
- Demo Meeting: Presentación de las nuevas funcionalidades integradas al producto.

- Sprint Retrospective: Reunión entre el Team y el Scrum Master en la que examinan el desempeño del sprint anterior e identifican las oportunidades de mejora

Con base en estas características se ha elegido a SCRUM como la metodología de desarrollo de software a utilizar en este proyecto, remarcado que algunos de sus elementos no serán necesarios de emplear y se adaptarán de acuerdo a las necesidades.

Cabe mencionar que la gran mayoría del software tiene el propósito de administrar y gestionar datos del entorno en el que es implementado. Estos pueden ser de diferentes tipos de datos como números, palabras, valores de verdad o falsedad, e inclusive imágenes. En muchas ocasiones, estos datos son resguardados para realizar procesos de extracción de conocimiento que pueda ser de utilidad, a este proceso se le llama Minería de Datos.

1.2. Minería de datos

Los datos son características que describen a un objeto y pueden estar formados por valores tales como números (0, 1, 2, 3, 4, 5, 7, 8, 9), letras (A, B, C, D, E, . . . , X, Y, Z), binarios (Si o No) y las cadenas de caracteres que pueden englobar todas las anteriores (“Hola Mundo”, “Juan Pérez”, “3.- Rojo”, “Si”, etc.). Por ejemplo, para realizar una solicitud de empleo se tomarían características de los solicitantes, tales como su nombre, edad, sexo, lugar y fecha de nacimiento, CURP (Clave Única de Registro de Población), escolaridad, etc. En la actualidad existen diversas fuentes que producen grandes cantidades de datos, desde cientos de formularios en papel con los datos escritos a mano hasta los sistemas informáticos más sofisticados. En ambos casos, la información obtenida se resguarda en bases de datos de diferentes dimensiones, contenidas en numerosas computadoras o servidores. El propósito principal de contar con estos datos es estudiarlos y obtener conocimiento que no se muestra a simple vista y que puede contener detalles beneficiosos como patrones o comportamientos que pueden servir para la toma de decisiones futuras.

Para la obtención de ese conocimiento se lleva a cabo un proceso iterativo llamado Knowledge Discovery in Databases o Descubrimiento de conocimiento a partir de bases de datos (KDD por sus siglas en inglés), en el que involucran diversos pasos secuenciales que se tienen que seguir, como lo expone Han, Jamber y Pei y se presenta de forma gráfica en la Figura 1.1 [13]:

1. **Limpieza de datos:** Consiste en la eliminación de ruido e inconsistencia en los datos.

2. **Integración de los datos:** Incorpora las diversas fuentes de datos como lo pueden ser las bases de datos, la web, ficheros, entre otros.
3. **Selección de los datos:** A partir de las tareas de análisis se obtienen datos relevantes de la base de datos.
4. **Transformación de los datos:** Mediante operaciones de reducción o agregación, los datos son adaptados para aplicarse técnicas de minado de datos.
5. **Minería de datos:** Es el paso más importante de todo el proceso, debido a que se aplican técnicas especializadas para la extracción de patrones de datos.
6. **Evaluación de los patrones:** Verifica la exactitud de los patrones encontrados con base en el conocimiento de interés.
7. **Representación del conocimiento:** Se emplean técnicas de representación enfocadas en conocimiento descubierto para la comprensión de los usuarios.

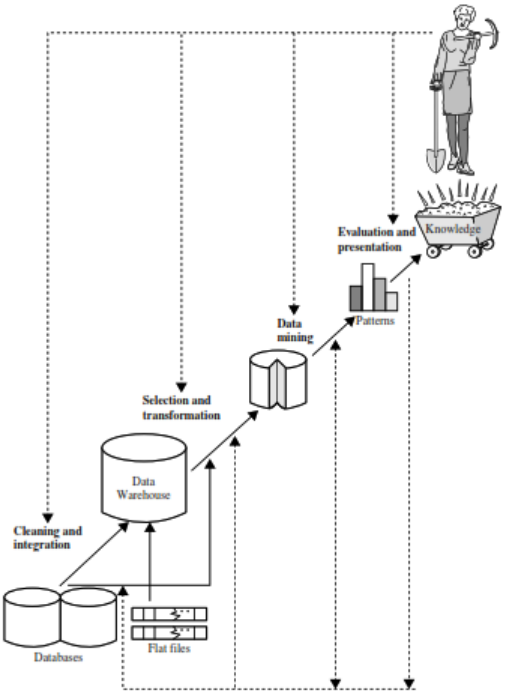


FIGURA 1.1: Diagrama del proceso KDD [13].

Dentro de KDD, uno de los pasos más importantes es el de minería de datos (MD), tanto que, cuando se usa el término MD a menudo se engloba el proceso KDD. Adicional a lo anterior, la MD es un proceso iterativo que usa técnicas inteligentes

y herramientas especializadas para el análisis exploratorio en grandes volúmenes de información, con el fin de encontrar conocimiento útil, novedoso y comprensible [22].

Añadiendo a lo anterior, menciona Kantardzic que " *los principales objetivos de la MD son dos: la* **descripción** *busca esquemas que expliquen los datos que puedan ser interpretados por humanos; por otro lado, la* **predicción** *involucra el uso de algunos atributos del conjunto de datos para pronosticar valores desconocidos de otras variables más importantes*" [22]. A partir de estos objetivos, son catalogadas las técnicas de MD en tareas primarias :

- **Clasificación:** Es la aplicación de funciones de aprendizaje predictivas que clasifican a cada uno de los elementos del conjunto de datos en una de las clases preestablecidas.
- **Regresión:** Emplea funciones de aprendizaje predictivo que mapea elementos del conjunto de datos para realizar una predicción de un valor real de una variable.
- **Agrupamiento:** Es una tarea descriptiva usual que consiste en identificar grupos finitos dentro del conjuntos de datos.
- **Resumen:** Emplea métodos para encontrar descripciones compactas para un grupo de datos.
- **Modelado de dependencia:** Consiste en la búsqueda de estructuras que describan significativamente las dependencias entre variables o valores de un conjunto de datos o una fracción del mismo.
- **Detección de cambios:** Es la labor de identificar los cambios más significativos dentro de un conjunto de datos.
- **Identificación de patrones frecuentes:** Se enfoca en hallar relaciones entre los elementos de un conjunto de datos.

Como se muestra, tanto el proceso KDD, como la MD están conformados por técnicas para el tratamiento de datos enfocados a la búsqueda y hallazgo de patrones que tengan un peso significativo dentro de extensos conjuntos de datos.

Los patrones son una pieza esencial para el descubrimiento, como los menciona Luna y Ventura: " *se pueden describir como subestructuras que se presentan regularmente en los datos, lo que representa consistente y sustancial propiedad en los conjuntos de datos.*" Existen métodos para encontrar estos elementos dentro de los conjuntos de datos,

para esto la DM tiene un componente dedicado al hallazgo de este tipo de pautas recurrentes llamada minería de patrones frecuentes [39].

1.2.1. Minería de patrones frecuentes

La minería de patrones frecuentes (MPF) consiste en la búsqueda de relaciones habituales entre los elementos de un conjunto de datos. Como lo mencionan Aggarwal y Han, el problema de MPF se plantea de la siguiente manera: "Dado un conjunto de datos D con transacciones $T_1...T_i$, se pueden determinar todos los patrones P que estén presentes en una fracción s de las transacciones" [1]. Donde $T_1...T_i$ son i número de vectores, y cada uno tiene un segmento de elementos del conjunto de datos que están involucrados en una operación, acompañados de un número de identificación que los distingue. Por otra parte, el parámetro s es una parte del número total de transacciones realizadas sobre los datos.

Un ejemplo de detección de patrones frecuentes se puede observar en un conjunto de compras (transacciones) realizadas en una tienda en la que están involucrados los productos del conjunto p, donde $p = \{leche, cereal, soda, harina, huevo\}$ [11]. La Tabla 1.1 muestra un conjunto de s transacciones, donde $s = T_1...T_5$. Cada transacción muestra el conjunto de productos que fueron adquiridos en ella. La idea es identificar patrones de compras frecuentes.

TID	Transacción
T_1	leche, cereal, soda, harina, huevo
T_2	cereal, soda, harina, huevo
T_3	leche, soda, harina
T_4	leche, soda, huevo
T_5	cereal, soda, huevo

TABLA 1.1: Grupo de transacciones con los objetos

Para detectar un patrón dentro de este grupo de transacciones, se realizan conteos de aparición de cada producto. De primera instancia, se define que un patrón se compone de solo un elemento, y posteriormente se le van agregando más elementos para encontrar patrones más grandes.

De inicio, en este ejemplo, se hace el conteo de cada producto en individual, de ahí que el producto leche aparece en tres transacciones; el producto cereal también en tres; el producto soda aparece en las cinco transacciones; la harina en tres; y por último el huevo en cuatro. Para determinar que un patrón sea frecuente, se fija un umbral min_sup con un valor que especifica el mínimo número de veces que debe

aparecer este patrón en las transacciones. Para este ejemplo, $min\_sup = 2$, por lo que como se puede apreciar en la Tabla 1.2, todos los productos que en individual se consideran patrones.

Producto(s) (conjunto f_j (j=0))	Apariciones
leche	3
cereal	3
soda	5
harina	4
huevo	4

TABLA 1.2: Apariciones de los productos en individual dentro de las transacciones

El conjunto F está conformado por los grupos de elementos frecuentes f que se van hallando en cada ciclo de búsqueda j, describiendo esto como $F = f_0, f_1, ...f_n$. Una vez conocida la frecuencia de cada producto en individual que sea mayor o igual al min_sup, f_0 contendrá dichos productos al igual que un conjunto base b, en el que $b = \{leche, cereal, soda, harina, huevo\}$ servirá para formar nuevos subconjuntos candidatos a partir de los patrones hallados en un ciclo anterior $(j - 1)$.

En la segunda iteración, el conjunto f_j cuenta con los nuevos candidatos a buscar, resultado del producto cartesiano del conjunto base b y los elementos frecuentes de la iteración anterior $f_{(j-1)}$, representado como $f_j = b * f_{(j-1)}$, mostrándose en la Tabla 1.2 los elementos que conforman dicho conjunto con su respectiva frecuencia dentro del grupo de transacciones s.

Producto(s) (conjunto f_j (j=1))	Apariciones
(leche, leche)	0
(leche, cereal)	1
(leche, soda)	2
(leche, harina)	0
(leche, huevo)	0
(cereal, leche)	0
(cereal, cereal)	0
(cereal, soda)	3
(cereal, harina)	0
(cereal, huevo)	0
(soda, leche)	0
(soda, cereal)	0
(soda, soda)	0
(soda, harina)	3
(soda, huevo)	2
(harina, leche)	0
(harina, cereal)	0
(harina, soda)	0
(harina, harina)	0
(harina, huevo)	2
(huevo, leche)	0
(huevo, cereal)	0
(huevo, soda)	0
(huevo, harina)	0
(huevo, huevo)	0

TABLA 1.3: Frecuencia de los productos del grupo f_j (j=1) en la segunda iteración

Dentro de este grupo f_j, solo los pares de productos (leche, soda), (cereal, soda), (soda, harina), (soda, huevo), (harina, huevo) tienen una frecuencia dentro del conjunto s mayor o igual a min_sup, eliminando al resto de los elementos como lo muestra la Tabla 1.4. Dado que existen elementos frecuentes en f_j, se realiza un nuevo ciclo de búsqueda para hallar más elementos frecuentes.

Elementos frecuentes hallados	
Producto(s) (conjunto f_j (j=1))	Apariciones
(leche, soda)	2
(cereal, soda)	3
(soda, harina)	3
(soda, huevo)	2
(harina, huevo)	2

TABLA 1.4: Productos del grupo f_j (j=1) que tienen un valor de frecuencia mayor o igual al *min_sup* establecido

Dentro de este tercer ciclo de búsqueda, se crea un nuevo subconjunto f_j resultado del producto cartesiano de $f_j = b \times f_{(j-1)}$ con el objetivo de identificar los elementos habituales dentro del grupo s, mostrándose en la Tabla 1.5.

Producto(s) (conjunto f_j (j=2))	Apariciones
(leche, soda, leche)	0
(leche, soda, cereal)	0
(leche, soda, soda)	0
(leche, soda, harina)	1
(leche, soda, huevo)	1
(cereal, soda, leche)	0
(cereal, soda, cereal)	0
(cereal, soda, soda)	0
(cereal, soda, harina)	2
(cereal, soda, huevo)	1
(soda, harina, leche)	0
(soda, harina, cereal)	0
(soda, harina, soda)	0
(soda, harina, harina)	0
(soda, harina, huevo)	2
(soda, huevo, leche)	0
(soda, huevo, cereal)	0
(soda, huevo, soda)	0
(soda, huevo, harina)	0
(soda, huevo, huevo)	0
(harina, huevo, leche)	0
(harina, huevo,cereal)	0
(harina, huevo, soda)	0
(harina, huevo, harina)	0
(harina, huevo, huevo)	0

TABLA 1.5: Frecuencia de los productos del grupo f_j (j=2) en la tercera iteración

Las tercias de productos de f_j, (cereal, soda, harina) y (soda, harina, huevo) tienen una frecuencia dentro del conjunto s mayor o igual al *min_sup*. Teniendo en cuenta que hay elementos de f_j a evaluar, se procede a realizar una nueva iteración aumentando la longitud de los elementos a 4 productos.

La Tabla 1.6, muestra los elementos de f_j actuales en el cuarto ciclo, siendo (cereal, soda, harina, huevo) el único individuo en tener un valor de frecuencia similar al de *min_sup*, dando paso a una nueva iteración para el hallazgo de patrones frecuentes.

Producto(s) (conjunto f_j (j=3))	Apariciones
(cereal, soda, harina, leche)	0
(cereal, soda, harina, cereal)	0
(cereal, soda, harina, soda)	0
(cereal, soda, harina, harina)	0
(cereal, soda, harina, huevo)	2
(soda, harina, huevo, leche)	0
(soda, harina, huevo, cereal)	0
(soda, harina, huevo, soda)	0
(soda, harina, huevo, harina)	0
(soda, harina, huevo, huevo)	0

TABLA 1.6: Frecuencia de los productos del grupo f_j (j=3) en la cuarta iteración

En la última y quinta iteración se ha alcanzado el tope de búsqueda porque ningún elemento de f_j cuenta con el número de apariciones dentro del conjunto s que supere o iguale el *min_sup* establecido.

Producto(s) (conjunto f_j (j=4))	Apariciones
(cereal, soda, harina, huevo, leche)	0
(cereal, soda, harina, huevo, cereal)	0
(cereal, soda, harina, huevo, soda)	0
(cereal, soda, harina, huevo, harina)	0
(cereal, soda, harina, huevo, huevo)	0

TABLA 1.7: Frecuencia de los productos del grupo f_j (j=4) en la segunda iteración

La Tabla 1.7 muestra el conjunto descartado de grupos de productos candidatos a patrones frecuentes, quedando el conjunto F de la siguiente manera: $F = \{f_0, f_1, f_2, f_3\}$ Obteniendo los patrones frecuentes de la colección de transacciones s, presentados en la Tabla 1.8.

Producto(s)	Apariciones	Conjunto de F
leche	3	f_0
cereal	3	f_0
soda	5	f_0
harina	4	f_0
huevo	4	f_0
(leche, soda)	2	f_1
(cereal, soda)	3	f_1
(soda, harina)	3	f_1
(soda, huevo)	2	f_1
(harina, huevo)	2	f_1
(cereal, soda, harina)	2	f_2
(soda, harina, huevo)	2	f_2
(cereal, soda, harina, huevo)	2	f_3

TABLA 1.8: Patrones frecuentes hallados en el grupo de transacciones s

Con el valor de umbral establecido en $min\_sup = 2$ se encontraron 13 patrones frecuentes contiguos dentro del grupo de transacciones s. Cinco de estos patrones frecuentes contienen un producto, seis elementos cuentan con dos productos, tres individuos con tres productos y uno elementos con cuatro productos, siendo la soda con cinco apariciones el insumo más repetido. Por otro lado el grupo (cereal, soda), (soda, harina) son los ejemplares no unitarios con mas apariciones, con tres cada uno.

Si solo se toman en cuenta las apariciones que tienen esos subconjuntos, los patrones frecuentes serian P = *(leche, cereal, soda, harina, huevo, (leche, soda), (cereal, soda), (soda, harina), (soda, huevo), (harina, huevo), (cereal, soda, harina), (soda, harina, huevo), (cereal, soda, harina, huevo))*. Cabe recalcar que para efectos prácticos en este ejemplo se tomó el criterio más simple para la detección de patrones frecuentes, pero no significa que sea la única ya que puede haber otros como el precio de cada producto, la categoría a la que pertenece, cantidad, entre otras.

El objetivo de la MPF es hallar patrones que sean de un alto aporte al usuario (a consideración de un experto) con un manejo eficiente de estructuras de datos, reducción de espacio de búsqueda de candidatos o eficiencia en el conteo de elementos. Algoritmos como GSP, PrefixSpan, FreeSpan, AprioriAll, Basado en índices [1] entre algunos otros, están dentro del grupo de la MPF y son empleados para diferentes tareas como análisis de transacción de usuarios, análisis de bug de software o minería web. Además, la aplicación de la MPF se puede extender a otras áreas del

conocimiento como la química y la biología dado que sus datos pueden ser procesados por este tipo de métodos y pueden ser representados como patrones [1].

Como esta, muchas técnicas y aplicaciones de las ciencias computaciones se han desarrollado y empleado en diferentes ramas del conocimiento con el fin de satisfacer las necesidades de cada una mediante el uso de computadoras u otro dispositivo electrónico. Esto ha dado pie a la combinación de nuevas áreas con el propósito de investigar y resolver tareas de forma mas efectiva. Entre ellas está la bioinformática.

1.3. Bioinformática

Es una disciplina que combina cuatro ramas importantes del conocimiento: matemáticas, ciencias computacionales, biología y estadística. Según Rodríguez: *"Estudia el desarrollo de procesos computacionales y técnicas matemáticas para resolver problemas prácticos y teóricos enfocados al manejo del procesamiento de la información biológica"* [30]. Esto incluye tareas de adquisición, almacenamiento, análisis y visualización de datos biológicos.

Para el Centro Nacional para la información Biotecnología o National Center for Biotechnology Information (NCBI por sus siglas en inglés), la bioinformática se desenvuelve en tres facetas importantes [29]:

- Desarrollo de nuevos algoritmos computacionales o estadísticos para evaluar relaciones en los datos biológicos.
- Análisis e interpretación de diferentes tipos de datos biológicos como por ejemplo: nucleótidos y secuencias de aminoácidos, dominios de proteínas y estructuras de proteínas.
- Desarrollo e implementación de herramientas que permitan el eficiente acceso y administración de diferentes tipos de información biológica.

Una pieza fundamental en la biológica molecular, por ende, en la bioinformática es el ADN, debido a que incluye la información genética de todo ser vivo, y con ella la formación de nuevos seres vivos. A continuación se presenta una descripción de este concepto.

1.3.1. ADN

El Ácido Desoxirribonucleico, mejor conocido como ADN (también se puede encontrar en la literatura como DNA por sus siglas en inglés, pero para la compresión

de este documento se tomarán sus siglas en español) es la molécula que se encuentra en todas las células de cualquier ser vivo (e inclusive se encuentra en algunos virus que viven en el ambiente). Es la pieza fundamental que contiene la información molecular heredada por antepasados que dicta el funcionamiento biológico de todo organismo vivo, debido a que está involucrado en sus procesos biológicos, pasando por diferentes etapas que comienzan desde la concepción, seguido del crecimiento, la madurez físico-mental, la adultez, la reproducción, entre otras actividades fisiológicas [20].

El ADN es un elemento extenso formado por dos cadenas secuenciadas y enlazadas en forma de hélices compuestas por unidades llamadas nucleótidos. Dicha estructura fue hallada en el año 1953 por el biólogo James Watson y el biofísico Francis Crick. Junto a ellos, la bioquímica Rosalind Franklin confirmaría dicha estructura mediante procedimientos de rayos X para obtener una imagen detallada del mismo [36].

Dentro de las secuencias del ADN existen dos tipos de nucleótidos: purinas y pirimidinas; dentro de las purinas las bases adenina y guanina; por otro lado, citosina y timina dentro de las pirimidinas como se ve en la Figura 1.2. Debido a que el ADN sólo se conforma por cuatro bases, el lenguaje del mismo elemento solo está conformado por cuatro letras: **A** (Adenina), **G** (Guanina), **T** (Timina) y **C** (Citosina) [28].

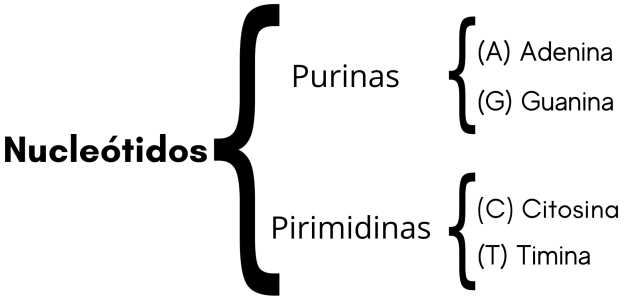


FIGURA 1.2: Nucleótidos que conforman el ADN

El orden en el que se ubican los nucleótidos dentro de las cadenas de ADN dependerá de la especie. Además de dos secuencias que son complementarias, es decir, cada base de cada secuencia forman un par o unión con la base de la secuencia de acuerdo con el patrón de enlace del ADN: A con T o T con A, y C con G o G con C [36], como se muestra en la Figura 1.3.

FIGURA 1.3: Unión de patrones de enlace entre dos cadenas de nucleótidos para formar el ADN [36]

Un proceso en el que está involucrado el ADN es llamado Expresión Genética, la cual consiste en decodificar la información contenida dentro de las cadenas secuenciales en varias etapas para producir piezas con utilidades mecánicas o de apoyo que sean participes en las funciones biológicas del ser vivo. Estas piezas son llamadas Proteínas y para obtenerlas es necesario transformar el ADN en Ácido Ribonucleico (ARN) [19].

Cabe recalcar que el ADN es como una guía biológica que tiene los datos necesarios para la construcción y mantenimiento de un ser vivo. Es escrito con solo cuatro letras (A, T, G, C) representando a los tipos de nucleótidos que forman dos hilos de estas unidades, ordenados secuencialmente y que se complementan entre si. Esta información genética está dividida en subconjuntos de secuencia llamados Genes. De los genes se pueden obtener proteínas, pero se necesita convertir a ARN [36].

Al igual que el ADN, el ARN está compuesto por una secuencia de nucleótidos pero por un solo hilo y no en dos. La clasificación de nucleótidos es similar, sólo que la base timina se convierte en uracilo; para las purinas están las bases adenina y guanina; y para las pirimidinas citosina y uracilo. Estas bases están representadas por las letras (A, G, C, U) [20] tal como se puede apreciar en la Figura 1.4.

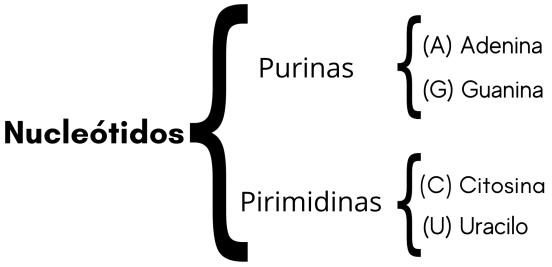


FIGURA 1.4: Nucleótidos que conforman el RNA

La función del ARN es fundamental dentro del proceso de expresión genética debido a que toma la secuencia del gen, lo transforma y lo utiliza para ensamblar proteínas. Para obtener ese resultado, existen tres tipos principales de ARN: ARN mensajero (ARNm), ARN de transferencia (ARNt) y ARN rebosoma (ARNr), que intervienen cada uno en una etapa específica de la expresión [36].

Primero se empieza con la Transcripción, una enzima se coloca en el sitio donde se encuentra el gen del ADN y emplea una de las cadenas como modelo para la construcción del ARNm. Al inicio de este proceso, las dos cadenas del gen se encuentran como unidades, y al final, las mismas dos cadenas terminan unidas. El ARNm resultante se puede dividir en subsecuencias de tres letras llamadas codones. La expresión de un codón es clave para identificar a un aminoácido, componente básico de las proteínas. Dados los tres lugares de un codón y las cuatro letras para la formación del ARN (A, G, C, U), existen 64 combinaciones de letras para un codón. Dicho grupo de 64 codones conforman el Código genético, de los cuales 61 codones ayudan a identificar 20 tipos de aminoácidos y tres para detener el proceso de traducción, el cual consta de tres principales fases [36]:

- **Inicio:** La unidad del ARNt se une con el ARNm, junto con dos unidades de ARNr.
- **Elongación:** El ARNt identifica y complementa los pares bases del codón para formar una unidad de aminoácido dentro del ARNr. Se realiza el mismo acto con el siguiente codón, el nuevo aminoácido se une con el anterior formando una cadena, y el ARNt anterior se desecha, junto con el codón anterior. Este subproceso se repite hasta llegar al próximo.
- **Terminación:** Cuando el ARNr detecte a un codón de detención, la cadena de aminoácidos es liberada.

La cadena resultante de aminoácidos de la expresión genética es una proteína. Esta pieza molecular es un componente fundamental y numeroso dentro de las células; se halla en procesos y componentes más importantes dentro del sistema biológico, que se refleja en sus productos físicos, como la queratina en el cabello, y también se puede convertir en anticuerpos, enzimas, hormonas, proteínas estructurales, entre otros [19]. Este producto es resultado de la acción biológica donde el ADN es la materia prima para la formación de proteínas a través del ARN, como se ve observa en la Figura 1.5.

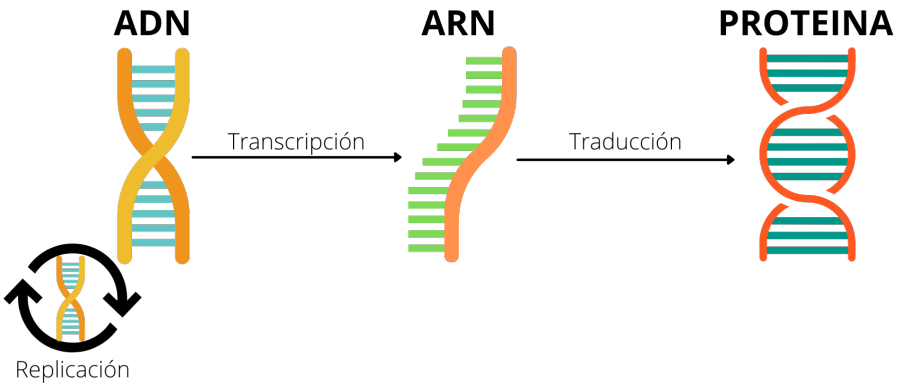


FIGURA 1.5: La expresión genética y la replicación: El flujo de información genética que va desde el ADN y el proceso de replicación para dar paso al proceso de formación de proteínas a través del ARN

Tanto el proceso de replicación del ADN como el proceso de la expresión genética son parte del flujo de la información biológica contenida en el ADN, el cual es la pieza fundamental para producir descendencia y fabricar organismos celulares funcionales para el ser vivo.

En la teoría, este proceso de información biología se muestra de forma simple, pero representa uno de los mayores retos en la biología. Una de las labores en este reto es hallar sitios de unión en el ADN para factores de transcripción. Para esto es necesario identificar patrones recurrentes dentro de un grupo de secuencias de ADN, llamados motifs. Por medio de métodos especializados y el uso de recursos computacionales se realiza su hallazgo dentro de las secuencias de ADN, lo que también representa un desafío para la bioinformática [7].

1.4. Motif

Para D'haeseleer un motif [*] es un patrón recurrente de secuencia corta (entre 5 a 20 pares bases aproximadamente) que se pueden encontrar dentro de las secuencias de ADN y que suponen tener una función biológica. A menudo también representan sitios de unión de secuencias específicas para proteínas tales como nucleasas y

[*]La palabra motif también se puede hallar en la literatura en su traducción al español como "motivo"pero para efectos prácticos y no confundir al lector, se empleará el primer término.

factores de transcripción (TF por sus siglas en inglés) que controlan la transcripción de la información contenida en el ADN hacia el ARNm. También, están incluidos en los procesos al interior del ARN, incluyendo uniones de ribosoma, terminación de transcripción y proceso de formación del ARNm [8].

La tarea de hallazgo de un motif, inicia con un grupo de secuencias de ADN que contengan elementos repetitivos de corta longitud y que de esta manera sean candidatos para ser sitios de unión a factores de transcripción o transcription factor binding sites (TBFS por sus siglas en inglés). Bajo el planteamiento de este problema, se han desarrollado procesos que se toman como base para la búsqueda de motifs dentro de secuencias de ADN [7].

1.4.1. Procesos base para identificación de motifs

Para Das y Dai existen dos principales tipos de procesos para el descubrimiento de motifs [7]:

- **Con base en palabras:** Consiste en la búsqueda exhaustiva mediante conteo o comparación de letras o grupos de letras. Estas técnicas son apropiadas para motifs cortos o idénticos, debido a que pueden ser veloces junto con una implementación de optimización de estructura de datos, como los árboles sufijo. Agregando a esto, Hashim, Mabrouk y Atabany mencionan que para el funcionamiento de este tipo de métodos, el usuario necesita ingresar algunos parámetros como la longitud del motif a buscar, el número de comparaciones permitidas y el número de secuencias en las que el motif puede aparecer [17].
- **Métodos probabilísticos:** Estos métodos utilizan potentes operaciones matemáticas probabilísticas, y al contrario de los métodos basados en palabras, pueden emplearse grupos de secuencias de ADN para hallar motifs más extensos que contengan información relevante. La ventaja de estos métodos es que requieren pocos parámetros para su búsqueda, pero que a su vez pueden ser sensibles con respecto a pequeños cambios en los datos de entrada.

Una vez que los algoritmos terminan su ejecución, presentan un número de sub secuencias candidatas que pueden representar el patrón en forma de cadena de letras o en una matriz de probabilidad con la finalidad de entregar una salida comprensible para el usuario.

1.4.2. Representación de un motif

Existen diferentes procedimientos para representar un grupo de n número de motifs, entre los que destacan la Expresión regular (ER), Matriz de Conteo por Posición (MCP), Matriz de Probabilidad por Posición (MPro), Matriz Peso por Posición (MPP) y la Matriz de Información Contenida (MIC).

Expresión Regular

La ER describe un conjunto de caracteres en un patrón de secuencia, formado posición por posición, con base en las letras que componen al ADN (A, G, C, T) y las conjunciones que puedan existir entre las mismas. Estas conjunciones se presentan en posiciones donde posiblemente pueden ser representadas por 2 a 4 letras que ocupan misma posición, empleando un diccionario de códigos de nucleótidos del International Union of Pure and Applied Chemistry o La Unión Internacional de Química Pura y Aplicada (IUPAC por sus siglas en inglés) [4], el cual se presenta en la Tabla 1.9.

En esencia, se toman todas las letras de una posición del conjunto de secuencias hasta formar la expresión con base en el código de secuencias de la IUPAC. Un ejemplo serían las expresiones **T-A-T-A-W-T** y **T-A-C-N(2,4)-G-T-A**; en el caso de la primera, resultan dos secuencias: **T-A-T-A-A-T** o **T-A-T-A-T-T**, la diferencia existen en la letra de la quinta posición. La siguiente expresión indica que comienza con las letras T-A-C, después *N* indicando cualquier nucleótido, seguido de la longitud de la subsecuencia que va de dos a cuatro, indicando que dentro de ese rango pueden ser de dos a cuatro letras cuales sea, lo que se refleja en la longitud de la expresión del motif que puede ser de 8 a 10 letras, y finalizando con las letras G-T-A [4]. Este tipo de representación es empleado para sintetizar en un patrón el resultado de algoritmos de identificación de motifs base en palabras.

Nucleótido Código	Base
A	Adenina
C	Citosina
G	Guanina
T	Timina
R	A o G
Y	C o T
S	G o C
W	A o T
K	G o T
M	A o C
B	C o G o T
D	A o G o T
H	A o C o T
V	A o C o G
N	Cualquier nucleótido
. o −	gap

TABLA 1.9: Lista del código de Nucleótidos de la IUPAC para secuencias de ADN [37].

Además de representar a un motif con una expresión regular, también se puede realizar de forma numérica mediante una matriz de conteo por posición para conocer a detalle la distribución de los nucleótidos en cada posición del motif.

Matriz de conteo por posición

La MCP es la representación numérica básica de un motif, que contiene la frecuencia de cada nucleótido en cada posición del motif. Por lo general, esta matriz es de n por w, donde n es el número de nucleótidos del ADN, 4 elementos, y w es el número de posiciones que componen al motif [38].

Matriz de probabilidad por posición

Por otra parte, MPro contiene los valores que determinan la probabilidad de que cada nucleótido (A,G,C,T) ocupe una posición en específico dentro del motif. Para obtener la probabilidad de cada elemento del arreglo se aplica la siguiente fórmula:

$$P_{n,w} = \frac{f_{n,w}}{N}$$

Donde $P_{n,w}$ representa el promedio de cada letra n en la posición w dentro de MCP; N es el número total de secuencias de ADN involucradas; y $f_{a,i}$ es la frecuencia de la letra de acuerdo a la posición w [4].

Matriz peso por posición

La MPP o también conocida como matriz peso por posición específica o matriz de puntaje por posición específica, contiene el "puntaje o calificación"que cada nucleótido en cada posición del motif a formar, mediante el cálculo del logaritmo binario de los valores de MPro sobre las probabilidades de fondo de cada nucleótido, lo cual se expresa en la siguiente fórmula:

$$S_{n,w} = \frac{P_{n,w}}{b}$$

Donde $S_{n,w}$ es el puntaje que se obtiene del nucleótido n en la posición w y b es la probabilidad de fondo con un valor uniforme de $\frac{1}{4}$ (la probabilidad que tiene cada letra de estar en una posición), $P_{n,w}$ se obtiene de MPro [38].

Matriz de Información Contenida

MIC contiene el total de información contenida en cada posición, el cual indica el nivel de conservación en la misma. Para esto se calcula el total posible de información contenida (IC). El total de IC se basa en la longitud del alfabeto del ADN ($k = 4$) con la siguiente fórmula [38]:

$$IC_{total} = \log_2(k)$$

Dando como resultado $IC = 2$. Seguido, se obtiene la incertidumbre actual en cada posición U. Esta operación se basada en la Entropía de Shanoon, que es una medida de incertidumbre de un modelo y lo impredecible que sería una secuencia que genera [18], y se aplica con la siguiente fórmula:

$$U_w = -\Sigma_{n=A}^{T} P_{n,w} * \log_2(P_{n,w})$$

El logaritmo es en base 2, y los resultados se presentan en 2 bits, el contenido de información se asocia a una posición con respecto a los nucleótidos (A, C, G, T) [18]. Después se obtiene el IC_{final} restando del IC_{total} a U [38]:

$$IC_{final} = IC_{total} - U$$

Finalmente, se obtiene la información de conservación de cada letra en cada posición, multiplicando el IC_{final} por $P_{n,w}$ [38]:

$$IC_{n,w} = P_{n,w} * IC_{final}$$

Una vez realizadas estas operaciones, se puede obtiene una representación visual del contenido del MIC, llamado LOGO de secuencias.

Gráfico LOGO de secuencias

Un logo es una pieza gráfica que contiene la cantidad de información presente en cada posición de la secuencia, medida en bits. La creación de un logo provine a partir de un conjunto de secuencias alineadas, ya sean de ADN, ARN o proteínas [31]. La Figura 1.6 es un gráfico LOGO de ejemplo obtenido a partir del grupo de secuencias [GCTGAAACTTA, GCTGAAACTTA, GCTGAAACTTA, GCTGAAACTTA, GCTGAAACTTA, GCTGAAACTCA, GGTGAAACGCA, GCTGAAACCGA].

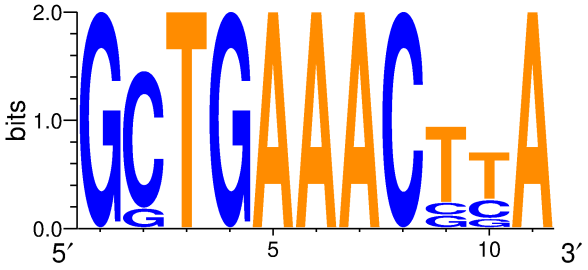


FIGURA 1.6: Gráfico Logo de secuencias de ejemplo, con 11 nucleótidos de longitud

Lo siguiente es mostrar un ejemplo en el cual se tiene un conjunto de secuencias, se obtienen las matrices antes descritas y al final se presenta un gráfico que muestra de forma visual la información que se preserva en cada posición del grupo de secuencias.

Caso de representación de motifs

Tremblay [38] presenta un ejemplo de representación de un motif en el que toma un grupo de secuencias ordenadas conformadas por seis cadenas de seis nucleótidos. La Tabla 1.10 muestra el grupo de secuencias que será representado en ER, MCP, MPro, MPP y MIC, para finalizar con una representación visual de la información que se conserva en las secuencias de entrada.

#	Secuencias
1	AAGAAT
2	ATCATA
3	AAGTAA
4	AACAAA
5	ATTAAA
6	AAGAAT

TABLA 1.10: Grupo de secuencias de ejemplo

Con el grupo de secuencias ya presentado, se puede iniciar el proceso de representación de un motif. Para este ejemplo, se comenzará con la ER para representar al grupo de secuencias de entrada en una cadena de caracteres, que indica la presencia de los nucleótidos en una posición con base en el código de Nucleótidos de la IUPAC (Tabla 1.9). La ER es la siguiente:

$$A - W - B - W - W - W$$

Esta indica que A está presente en todas las secuencias dentro de la primera posición; La W denota la presencia de los nucleótidos A y T en las posiciones dos, cuatro, cinco y seis; Por último, en la posición tres se encuentra una B, que indica la presencia de C, G y T.

Se obtiene la MCP de la Tabla 1.11 del grupo de secuencias de ejemplo.

Posiciones	1	2	3	4	5	6
A	6	4	0	5	5	4
C	0	0	2	0	0	0
G	0	0	3	0	0	0
T	0	2	1	1	1	2

TABLA 1.11: MCP del conjunto de secuencias del ejemplo

Se puede apreciar que en MCP, A domina en la primera posición; En la posición dos y seis, A y T tiene una frecuencia de cuatro y dos respectivamente; C, G y T se hallan en la posición tres en $2, 3, 1$ ocasiones, respectivamente; En las posiciones cuarto y cinco, A tiene una frecuencia igual a cinco y T a uno. El MCP resultante permite obtener la probabilidad de cada nucleótido en cada posición para formar el MPro, tal como se muestra en la Tabla 1.12

Posiciones	1	2	3	4	5	6
A	1.00	0.67	0.00	0.83	0.83	0.67
C	0.00	0.00	0.33	0.00	0.00	0.00
G	0.00	0.00	0.50	0.00	0.00	0.00
T	0.00	0.33	0.17	0.17	0.17	0.33

TABLA 1.12: MPro del conjunto de secuencias del ejemplo

En el caso de la MPro del ejemplo, La letra A tiene una probabilidad de ocupar la posición uno del 100 %; Las posiciones dos y seis comparten las mismas probabilidades de la presencia de $A = 67\,\%$ y $T = 33\,\%$, siendo A la de mayor porcentaje; Para la posición tres, $C = 33\,\%$, $G = 50\,\%$ y $T = 17\,\%$ en la probabilidad de aparición, en la que G tiene mayor oportunidad de aparecer que a C y T; Y para las posiciones cuatro y cinco, A obtuvo un 83 % y C un 17 %, siendo A la de mayores posibilidades de aparición en ambas posiciones. Con estos datos, se realiza la obtención de la MPP que se muestra en la Tabla 1.13

Posiciones	1	2	3	4	5	6
A	2	1.425	-Inf	1.737	1.737	1.415
C	-Inf	-Inf	0.415	-Inf	-Inf	-Inf
G	-Inf	-Inf	1.000	-Inf	-Inf	-Inf
T	-Inf	0.415	-0.585	-0.585	-0.585	0.415

TABLA 1.13: MPP del conjunto de secuencias del ejemplo

Dentro de la primera columna del MPP, A obtuvo un puntaje máximo de 2; En las posiciones dos y tres, A obtuvo la mayor puntuación 1.425 y T 0.415; Dentro de la posición tres C 0.415 de puntaje, G de 1.000 y T de -0.585. Por lo que G fue el obtuvo el mayor puntaje en la posición; Para las posiciones 4 y 5, C obtuvo - 0.585 y A un 1.737, siendo esta última la que obtuvo un puntaje mayor. El valor de -inf de los nucleótidos en diferentes posiciones es porque su valor es menor a la probabilidad de fondo. Para la siguiente matriz, estos valores se normalizan a cero para evitar errores en las operaciones.

Posiciones	1	2	3	4	5	6
A	2.000	0.721	0.000	1.125	1.125	0.721
C	0.000	0.000	0.180	0.000	0.000	0.000
G	0.000	0.000	0.270	0.000	0.000	0.000
T	0.000	0.361	0.090	0.225	0.225	0.361

TABLA 1.14: MIC del conjunto de secuencias del ejemplo

La Tabla 1.14 presenta los resultados de la matriz MIC: En la posición uno, *A* tiene una alta conservación, abarcando los 2 bits de información. En las posiciones dos y seis, *A* obtuvo un valor de IC de 0.721 y *T* de 0.361, dando a *A* el nucleótido con mayor IC en las respectivas posición; En la posición tres, *C* tiene un IC igual a 0.180, *G* de 0.270, y *T* igual a 0.090. Por lo tanto *G* tiene el valor de IC alto entre los nucleótidos de la posición pero es el mas bajo con respecto a otras posiciones, como *A* en la posición 1; En las posiciones cuatro y cinco, *A* obtuvo un mayor IC con un valor de 1.125. Por otro lado, *T* tiene un valor de 0.225, por lo que *A* tiene el IC más representativo en esas posiciones. Estos valores se ven reflejados de forma gráfica en el LOGO de secuencias que se presenta en la Figura 1.7

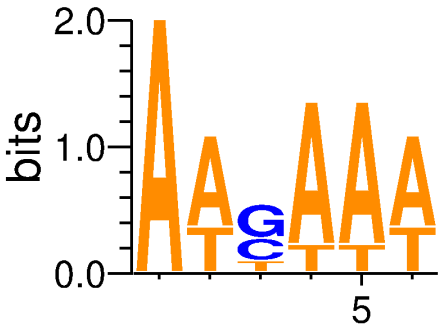


FIGURA 1.7: Logo de secuencias para la representar el ejemplo del motif AWBWWW

Con base en los valores que se obtuvieron en el MIC, se formó el LOGO de las secuencias de la Figura 1.7 que muestra la IC del motif AWBWWW. Como se había descrito con anterioridad, el núcleotido *A* domina en la primera posición, abarcando los dos 2 bits de información. En el caso de las posiciones dos y seis, *A* convive con *T*, pero siendo *A* la que contiene más información en las posiciones. En la posición tres, se ve una clara disminución de un IC en los tres nucleótidos que conviven ahí: *C*, *G*, *T*. Y en las posiciones cuatro y cinco, *A* tiene más IC con respecto a *T*.

Capítulo 2

Trabajos Relacionados

A lo largo de este capítulo se analizan algunos algoritmos enfocados al descubrimiento de motifs en secuencias de ADN. Estos algoritmos aplican distintas estrategias de procesamiento, como pueden ser los métodos basados en enumeración o estadísticos [17]. Algunos otros algoritmos aplican otro tipo de operaciones, tales como procesos de minería de datos, los cuales se enfocan en la búsqueda de patrones frecuentes o subsecuencias que se presentan de manera recurrente en un conjunto de datos secuenciales [15]. En este capítulo se mostrarán algunos trabajos relacionados al tema de este proyecto investigación, los cuales implementan operaciones antes mencionadas.

2.1. MEME

Multiple Expectation maximization for Motif Elicitation o MEME por sus siglas en inglés, es el algoritmo que tiene como objetivo detectar motifs significativos en un grupo de secuencias donde se desconoce su existencia [7]. MEME emplea el algoritmo Expectation-Maximization (EM) para obtener la probabilidad de los posibles sitios o segmentos dentro de la(s) secuencia(s) de ADN donde existan esos motifs [3].

El algoritmo EM tiene dos pasos principales: Expectativa y Maximización. El primero calcula los valores de algunos conjuntos desconocidos (la probabilidad de las subsecuencias) basados en la MPP; El segundo usa esos valores para recalcular esta matriz. Para el caso del procesamiento de secuencias de ADN para el hallazgo de motifs, EM es empleado para identificar subsecuencias "desordenadas", teniendo en cuenta que el sitio que cubre todas las secuencias tiene segmentos idénticos [17].

MEME realiza el siguiente procedimiento [17]:

- Inicia desde un único sitio dentro de la(s) secuencia(s) de ADN (ya sea aleatorio o específico), se genera la MPP y se estima un motif inicial a partir de ella.

- Usando este modelo inicial resultante se calcula la probabilidad de cada ubicación posible de este motif dentro de las secuencias de entrada, y se detectan subsecuencias relacionadas
- EM estima la probabilidad de las subsecuencias relacionadas al motif modelo dentro de las secuencias de ADN, y re-evalúa dicho motif modelo en MPP. Esto debe ser realizado iterativamente para cada subsecuencia que pertenece a las secuencias objetivo, se elige el mejor motif proveniente del sitio y se itera hasta que se alcance la coincidencia en MPP.
- Después de hallar un nuevo motif, se borran los viejos motifs hallados.

El proceso de MEME da como resultado la siguiente información de cada motif hallados, los cuales pueden ser de distintos tamaños: la expresión regular representativa del motif y la información contenida en cada posición del mismo [3].

Para la ejecución de MEME, se requiere que las secuencias de ADN sean seleccionadas cuidadosamente, considerando que sean lo más cortas posible y con el menor ruido en ellas (es decir, secuencias que posiblemente no tengan motifs). De acuerdo a los autores, esto facilitará la obtención de los motifs a medida que crezcan en longitud y en número de subsecuencias similares. En caso contrario, si las secuencias de entrada son largas, será más difícil que se encuentren subsecuencias del motif que sean similares [2].

2.2. Gibbs sampler

Gibbs Sampler o muestreo de gibbs emplea cadenas de Márkov para simplificar el cálculo en el alineamiento múltiple de secuencias de ADN y la identificación de secuencias recurrentes [42]. Su objetivo es identificar al motif más probable, un patrón en común en el conjunto de secuencias de ADN [24]. El núcleo principal de este método es la generación de MPP a partir del ordenamiento de subsecuencias, formando la distribución de frecuencias por nucleótidos, sitio por sitio del motif. [40]

Gibss Samper inicia las posiciones aleatorias de inicio dentro de un conjunto N de secuencias $S_1, S_2..., S_N$ de ADN, suponiendo de que existe un motif en cada secuencia de longitud W, sus respectiva posiciones a_k, donde k va desde 1 a N. Además se inicializa MPP considerado la longitud de W y las N secuencias; las probabilidades de fondo P de cada nucleótido (A, C, G,y T) Con esta información se procede a ejecutar los pasos de gibbs sampler [24]:

1. Paso de actualización predicativa: Se elige una secuencia Z de N, ya sea a la azar o en orden especifico, y se excluye de este paso. A partir de las demás

secuencias de N, se calcularan la descripción del patrón por medio de las probabilidades P y el MPP, .

2. Pase de simplificación: Dentro de la secuencia Z, cada subsecuencia de longitud W , z , es considerada como una posible subsecuencia relacionada al motif. Con esto en cuenta, se calculan las probabilidad Q_z y la probabilidad de fondo P_z de cada z_i acorde al conjunto $q_{i,j}$ y P. Seguido, se obtiene los pesos $A_z = Q_z / P_z$ de cada segmento z, para después un segmento z sea seleccionado y su posición se guarde en a_k, considerándolo como subsecuencia relacionada al motif de la secuencia z.

La idea central es que sea mas exacta la descripción del motif a través de la MPP y las probabilidades de fondo en el paso 1; y en el paso dos sea más exacto la elección de la localización de la subsecuencia relacionada al motif. Esto dará como resultado un motif de longitud W. [24]. Estos pasos se realizan de forma iterativa hasta llegar al máximo número de iteraciones o hasta que las probabilidades no mejoren [17].

2.3. DREME

Discriminative Regular Expression Motif Elicitation, por sus siglas en inglés, es un algoritmo enfocado a la búsqueda rápida de motifs relativamente cortos (hasta de longitud de 8 letras) y sin espacios, buscando motifs en forma de expresión regular (ER), que se construye con base al código de nucleótidos de la IUPAC (véase la Tabla 1.9), y este algoritmo considera como comodines aquellas letras que representan a dos o mas letras base del ADN. Por ejemplo **B** representa a A o C o G en una posición dentro de una formación de secuencias de ADN. Esta técnica sólo admite dos secuencias de ADN de igual longitud como datos de entrada y un umbral de significación [5].

Para conseguir estos motifs, el algoritmo comienza con una búsqueda de haz para hallar con un conjunto inicial de semillas de motifs en ER de longitud 3 a 8 letras sin comodines, mediante la prueba exacta de Fisher calcula el valor p de cada palabra de las semillas , dando como resultado un nuevo conjunto de semillas conformado 100 palabras más significativas que se utilizara en un ciclo interno para mejorar la búsqueda [5].

- Ciclo interno: En esta etapa se reciben las 100 palabras más significativas. Si todas las palabras que coinciden con una generalización de la semilla, tienen valor de p es igual o menor a 0.01, se estima la estadística significante de la ER generalizada, sino la generalización se descarta. Cuando ER han sido generalizada, Dreme clasifica el nuevos, y más general ERs en base a su significación,

y luego se valoran la significación de las 100 palabras para emplearlos como semillas para la siguiente iteración.

- Ciclo externo: el algoritmo que realiza una búsqueda heurística de motifs de ER, resultando en el mayor motif significativo, el cual se reporta y se borran todas sus repeticiones en los conjuntos de ADN. Esta etapa se repite hasta que el E-value del nuevo motif sea menor al del umbral de significación.

DREME se enfoca en buscar el espacio los motifs unicos en forma de mas simple ER, sin contener comodines, para hallar a los mas significativos pero con un longitud de 4 a 8 nucleótido. Lo cual beneficia en un tiempo de ejecución más corto, pero puede limitar una búsqueda mas extensa de los motifs que incluya diferentes longitudes y variaciones.

Ordenarlosalgoritmosporao -

2.4. Mining Frequent Patterns of Biological Spaced Motifs

Este trabajo propone un método para el hallazgo de motifs espaciados basándose en técnicas de MD. El proceso que realiza el algoritmo se divide en dos pasos: El primer paso implementa una técnica de búsqueda por niveles para extraer los conjuntos de patrones de motifs en las secuencias de ADN que cumplan con cierta frecuencia, para posterior identificar motifs espaciados en las secuencias. En el segundo paso, se transforman los motifs en un entorno categórico para ser procesados por un algoritmo enfocado a descubrir reglas de asociación relacionadas a los motifs [26]. Para entender el funcionamiento del algoritmo es necesario conocer las siguientes definiciones:

- T es el conjunto de secuencias de nucleótidos con diferentes longitudes.
- M es el conjunto de patrones recurrentes de motifs identificados en T.
- $minS$ es el mínimo soporte (una parte del porcentaje de las secuencias T en el que un patrón ocurre)
- $minF$ es el mínimo soporte de frecuencia ($minF < |T|$)
- El subconjunto S ($S \subseteq 2^M$), es el subconjunto de motifs espaciados pertenecientes a M extraídos de T y cumplen con $minF$.
- Sea el subconjunto s ($s \in S$) y su subconjunto de apoyo t ($t \subseteq T$), en la cual s contiene al motif espaciado y t sus frecuencias.

El primer paso está enfocado en hallar el conjunto de los elementos frecuentes de motifs M sin considerar su ubicación en T. Comenzando con una búsqueda exhaustiva por patrones de motif por niveles, a partir de los elementos más pequeños (de longitud uno), pasando por la generación de los patrones frecuentes de longitud i gracias a los de longitud $i-1$, hasta llegar con los más grandes (de i-ésima longitud) que pertenecerán a M. Después se extraen los elementos de s en T, buscando aquellos patrones que estén a x nucleótidos de distancia, y t guarda el identificador de las secuencias de ADN en donde tiene presencia s, que será utilizado para evaluar la frecuencia de s con $minF$ [26].

Para el segundo paso, se tiene como objetivo descubrir los patrones frecuentes distribuidos del conjunto S extraídos previamente, y representarlos en forma de reglas de asociación (AR). Para esto t transformará su contenido para representar la información en términos de motifs, es decir, anotar la distancia de forma numérica entre los patrones frecuentes de motifs que estén estrictamente consecutivos en las secuencias que tenia almacenadas t; Cabe mencionar que las distancias en ocasiones pueden modificar, si son cercanas. Después los valores numéricos de las distancias pasan a ser valores categóricos con base en un conjunto de intervalos cerrados (ID). Con t ya formando, lo siguiente es descubrir los patrones frecuentes de motifs espaciados (FP) en forma de AR mediante SPADA, un algoritmo multi-relacional capaz de extraer AR a partir de datos relacionados, siendo las secuencias de ADN los objetos principales, y los motifs M y sus distancias ID.

Este algoritmo da como resultado un conjunto de motifs en forma de Reglas de Asociación, considerando valores categóricos que indican el espacio que separa a dos elementos consecutivos, con los cuales se quiere enriquecer la información.

2.5. YMF

Buscador levado de motif o Yeast Motif Finder (YMF por sus siglas en inglés), es un algoritmo para el descubrimiento de motifs cortos con una alta valoración [34].

Para iniciar la ejecución se define el número de caracteres sin espacios para el motif representado por l, y el número máximo de espacios dentro del motif representado por w. Con estos parámetros y un modelo generado previamente por el método de cadenas de Márkov, se define el espacio de búsqueda de todos los motifs que serán evaluados. Este espacio considera a todos los motifs de longitud l que contengan los caracteres A,C,G,T,R,Y,S,W, y con (**N**) espacios de medio entre 0 y w. [33]

Primero se analizan las secuencias de entrada obteniendo el número N_s de apariciones de cada motif s dentro del grupo de secuencias de entrada. Por cada motif s

que N_s >0, se calcula la desviación media y estándar del recuento del motif utilizando la ecuación 2.1 [33]:

$$z_s = (N_s - E(X_s))/\sigma(X_s) \tag{2.1}$$

Donde:

- X_s: Es una variable aleatoria que contiene el número de motif s en las secuencias generadas aleatoriamente X.
- $E(X_s)$: Es la desviación media.
- $\sigma(X_s)$: Es la desviación estándar.
- N_s: El número de apariciones del motif.

YMF enumera todos los motifs hallados en un espacio de búsqueda, y asegura que se producen motifs de un *puntaje* z superiores, empleando la representación del consenso, reportándolos, junto con sus ocurrencias ocurrencias dentro de las secuencias de ADN [33].

2.6. Modified PrefixSpan Method for Motif Discovery in Sequence Databases

Este trabajo usa modificación del algoritmo de MPF, PrefixSpan para extraer un gran número de patrones frecuentes desde un conjunto de secuencias anotadas conformados por identificador de la secuencia, la secuencia, y un conjunto de elementos. Pero con la diferencia que este método permite extraer elementos con gaps o comodines.

Al igual que PrefixSpan, el usuario determinara el umbral para determinar que subsecuencia es un patrón frecuente con base a su numero de apariciones dentro de un conjunto de datos, e incluye una modificación del algoritmo se podrá determinar el número máximo de gaps que pueden contener los patrones frecuentes, y en el de que no se ingrese un valor el algoritmo lo tomara un número de gaps infinito, lo que podria afectar a la frecuencia de las subsecuencias para ser patrones frecuentes dependiendo el tipo de manejo de gaps [23]:

- **Método de longitud variable de gaps**: Determina la frecuencia de aquellas subsecuencias que están representadas dentro de la(s) secuencia(s) sin importar que sus elementos son contiguos o estén separados hasta el máximo numero de gaps. El ejemplo que muestra Kitakami y Yamazaki es el siguiente [23]: Dado un grupo de secuencias: *[100:MFKALRTIPVILNMNKDSKLCPN,*

200:MSPNPTNIHTGKTLR] y la lista de elementos de longitud uno con su frecuencia: *[<M>: 2, <K>: 2, <L>: 2, <R>: 2, <T>:2, <I>: 2, <P>:2, <N>: 2, <S>: 2, <P>: 2].* La obtención de patrones frecuentes de longitud dos con el prefijo **M** dio como resultado la siguiente lista: *[<MI>:2, <MK>:2, <ML>:2, <MN>:2, <MP>:2, <MT>:2]* y donde el patron **MK** esta representado dentro de la primera secuencia como **M*K**, y en la segundo como **M**********K**.
Sin importar el número de gap entre ambas subsecuencias representativas solo no se tiene que superar el máximo establecido por el usuario.

- **Método de longitud fija de gaps**: Determina la frecuencia de aquellas subsecuencias que están representadas dentro de la(s) secuencia(s) y tenga un un numero establecido de gaps entre sus elementos. El ejemplo que muestra Kitakami y Yamazaki es el siguiente [23]:
Dado un grupo de secuencias: *[100:MFKALRTIPVILNMNKDSKLCPN, 200:MSPNPTNIHTGKTLR]* y la lista de elementos de longitud uno con su frecuencia: *[<M>: 2, <K>: 2, <L>: 2, <R>: 2, <T>:2, <I>: 2, <P>:2, <N>: 2, <S>: 2, <P>: 2].* Del prefijo **K** se obtiene el patrón <K*L>:2, y del prefijo **K*L** se obtiene el patrón frecuente <K*LR>.

Después se extraen los patrones frecuentes que cumplan con con un soporte mínimo de conteo y un máximo número de gaps. Dando como resultado motifs de diferentes longitudes y con distinta información añadida.

2.7. Discusión sobre el trabajo relacionado

Como se muestra en este capítulo, existen diferentes algoritmos con distintas formas de descubrir motifs en grupos de secuencias de ADN. Cada estrategia presenta distintos parámetros de ejecución y procesos de obtención de motifs con base en operaciones matemáticas relacionadas con la biología o con métodos de otra perspectiva, enfocados al procesamiento y análisis de datos, como la Minería de Datos y la Minería de Patrones Frecuentes.

En la Tabla 2.1 se muestra la comparación entre los algoritmos presentados en este capítulo y el que se propone en este trabajo de tesis, mostrando los aspectos principales, que incluyen el tipo de operación y la clasificación a la que pertenecen.

Comparación de algoritmos			
Algoritmo	Operación	Principal Clasificación	Resultados
Mining Frequent Patterns of Biological Spaced Motifs	Búsqueda por nivel y SPADA	Minería de datos	Motifs en forma de reglas de asociación
Modified PrefixSpan Method for Motif Discovery in Sequence Databases	PrefixSpan modificado	Minería de datos	Conjunto de motifs con información de gaps y longitudes
MEME	Expectation Maximization	Probabilística	Motifs con información probabilística y de longitudes
Gibbs sampler	Gibbs sampler	Probabilística	Un motif de las características requeridas por el usuario y su ubicación dentro de la(s) secuencia(s)
DREME	Búsqueda de haz y heurística, junto con el test	Enumeración	Un conjunto de motifs de corta longitud
YMF	Cadenas de Márkov y operaciones probabilistica	Enumeración	Un conjunto de motifs de corta longitud
Propuesta de esta investigación	Mapeo de ubicaciones con generación de candidatos y crecimiento con base en tolerancia	Minería de datos	Conjunto de motifs incluyendo información de longitud, ocurrencia, posiciones, patrones, MCP y MIC.

TABLA 2.1: Tabla comparativa de algoritmos enfocados al descubrimiento de motifs en secuencias de ADN

Para esta comparación se presentan 7 algoritmos: tres basados en MPF como BIMS + EOM (Véase la secciones 3.3 y 4), Biological Spaced Motifs y Modified PrefixSpan; dos basados en información probabilística: MEME y Gibbs Sample; y por ultimo dos de operaciones de enumeración: DREME e YMF, dando cuatro algoritmos relacionados con procesos biologicos.

Menciona Hashim, Mabrouk y Al-Atabany que el reto de hallar motifs en secuencias de ADN es difícil para la bioinformática, debido a que no siempre son idénticos entre ellos, son subsecuencias descocidos, al igual que su ubicación que no esta relacionada con otras secuencias del conjunto, además que su existencia es aleatoria. Es por ello que han creado procesos matemáticos relacionados con su contra parte biología que se pueden dividir en dos: Enumeración y Probabilísticos. En el caso de los enumeración se enfocan en hallar de manera exhaustiva la mayoría motifs de corta longitud dependiendo de los parámetros ingresados por el usuario. Por su parte, los probabilísticos se enfocan en hallar los motifs estadísticamente relevante, siendo mas rápidos que los enumeración pero dejando de lado el no encontrar a todos los motif [17].

También se han integrado algoritmos que emplear otras técnicas como las de MD y MPF para la tarea de descubrimiento de motifs en secuencias de ADN, a través de sus operaciones se enfocan en hallar los patrones frecuentes que están dentro de grandes cantidades de información como pueden ser los conjuntos de secuencias de ADN, entregando motifs relevantes para el biólogo en un tiempo razonable en comparación con los métodos que se emplean para tradicionalmente [16].

Los algoritmos que se basan en técnicas de MD y MPF pueden ser más atractivos para descubrimiento de motifs en secuencias de ADN, porque pueden procesar mayor cantidad de información, ser más rápidos que los métodos relacionados con las operaciones, y entregando un número mayor de motifs que pueden ser relevantes para el usuario. Pero los algoritmos probabilísticos y enumeración arrojan motifs cercanos a su contra parte biológica. Ambas estrategias son funcionales y pueden adaptarse las necesidades y circunstancias de esta tarea en cuestión.

Dentro de los trabajos que se presentan caen en las siguientes categorias

- MPF que son *Mining Frequent Patterns of Biological Spaced Motifs* (MFP Spaced Motifs para fines prácticos) y Modified PrefixSpan Method for Motif Discovery in Sequence Databases (Modified PrefixSpan para fines prácticos).
- Dos de Enumeración como DREME e YMF
- Dos de tipo probabilístico como MEME y Gibbs Sampler

Cada uno de estos algoritmos presentan distintos procesos y formas de presentar sus resultados como MPF Spaced Motifs el cual busca patrones frecuentes contiguos que tengan unos cuantos nucleótidos de distancia para después ser transformados por el algoritmo SPADA en reglas de asociación que representan al motif; Para Modified PrefixSpan altera el algoritmo de PrefixSpan para que pueda procesar patrones frecuentes con un valor fijo o variable de gaps, dando como resultado motifs con gaps; En el caso de DREME emplea dos diferentes tipos de búsqueda para hallar a los motifs de corta longitud (4 a 8 nucleótidos) mas significativos; Al igual que YMF solo que se buscan a los motifs con base a las características que ingrese el usuario y los que contengan los valores más altos de *puntaje_z*, serán los motifs finales; Por otro lado el proceso de que realiza Gibbs sampler le permite ubicar a las subsecuencias relacionadas en cada secuencia de ADN del conjunto de entrada al motif que se desea formar y que presentara como resultado; Por ultimo MEME emplear el algortimo EM para evaluar los sitios dentro de un grupo de secuencias de ADN para hallar a su conjunto de motifs de distintas longitudes.

Para tener un buen punto de comparación y que sea lo mas parecido a BIMS+EOM, se eligió a MEME como el algoritmo comparar ya que en su ejecución da como resultado un conjunto de motifs de distintas longitudes sin presentar elementos adicionales como lo pueden ser lo gaps, además que cuenta con una implantación disponible para su uso a través de su plataforma web con el mismo nombre (MEME suite). Algo que los demás algoritmo no cumplen ya sea porque presentan sus resultados en otra forma de representación, son muy chicos, presentan alteraciones como los gaps o el número de motifs es reducida.

Capítulo 3

Método de identificación de patrones basado en índices

Dentro de la MPF existen diversos algoritmos que se han desarrollado para encontrar estructuras concurrentes y aplicarse a distintos escenarios como análisis transacciones de compra, análisis bugs de software, web mining, análisis de elementos biológicos y químicos, entre otros [1]. Dentro del análisis de elementos biológicos, una de las aplicaciones mas importantes es el descubrimiento de subsecuencias que resulten en patrones frecuentes dentro de secuencias de ADN [4] ya que aportan información relevante para los investigadores del área de biología. Algoritmos como GSP se han utilizado para esta tarea *CITA*, pero con el paso de los años han surgido nuevos algoritmos con mecanismos innovadores como el algoritmo basado en índices [12], el cual presenta un método eficiente de búsqueda para obtener los patrones frecuentes de una secuencia. El presente capítulo propone un nuevo algoritmo que introduce una mejora del trabajo antes mencionado con el fin de identificar los patrones que servirán como base para la identificación de motifs, tomando como fuente de datos una secuencia de ADN o bien un grupo de ellas.

3.1. Definición del problema

Para entender los procedimientos que lleva a cabo este algoritmo, García Islas cita las siguientes definiciones, ejemplos y propiedades relacionados al problema de búsqueda de patrones frecuentes en secuencias de ADN [12].

Definición 3.1: *sea* $\Sigma = ACGT$ *el conjunto conformado por cuatro letras que representan a los nucleótidos : Adenina* (A)*, Citocina* (C)*, Guanina* (G)*, Tiamina* (T)*. Siento estas unidades las que conforman las secuencias de ADN.*

Definición 3.2: *Sea una secuencia de ADN* $S = s_1, s_2, ...s_n$ *con* $s_i \in \Sigma, (i = 1, .., l)$*, donde l es la longitud de S. En la computación, una secuencia de ADN es considerada como*

una cadena de caracteres de extensa longitud finita, que se construye a partir del diccionario de nucleótidos Σ. Además sea $s_1, s_2, s_3, ..., s_n$ cada uno de los nucleótidos pertenecientes a S

Ejemplo 3.1: *Sea $\Sigma = \{A, C, T, G\}$, y $S =< ACGTGTAAAACTCTTGTT >$, S es una secuencia de $longitud(S)$ = 18, por ende esta constituido por 18 nucleótidos.*

Definición 3.3: *Sea $S = s_i | s_i \in \Sigma$ una secuencia de ADN y $s_1 = S_{[j]} | s_{[j]} \in s_i$ es una subsecuencia de S. Así mismo, S es una súper secuencia de s_1.*

Ejemplo 3.2: *Sea $S =< CTAAGTCCGTAGCCGACT >$ y $s_1 =< TAA >$. s_1 es una subsecuencia de S, y a su vez S es una súper secuencia de s_1.*

Definición 3.4: *Sea $S = s_i | s_i \in \Sigma$ una secuencia de ADN y $s_1 = S_{[j]} | s_{[j]} \in s_i$ es una subsecuencia de S. $f_{support}(s_j)$ es una función de s_j y es definido como el número de ocurrencias de la subsecuencia s_j dentro de una secuencia S.*

Ejemplo 3.3: *Sea las subsecuencias $s_1 =< TAA >$ y $s_2 =< CCG >$ pertenecientes a la secuencia S.*

Definición 3.5: *El parámetro umbral es un parámetro definido por el usuario utilizado para valorar la(s) aparición(es) de una subsecuencia s_j entro de una secuencia S.*

Ejemplo 3.4: *Sea el valor ingresado por el usuario $umbral = 2$.*

Definición 3.6: *Sea s_j la subsecuencia considerada como candidata, y es definida como aquella que puede ser evaluada por medio de la función $f_{support}(s_j)$, que calcula el numero de apariciones de la subsecuencia dentro de la secuencia de ADN.*

Ejemplo 3.5: *Se calcula el $f_{support}(s_j)$ de cada subsecuencia, obteniendo $f_{support}(s_1) = 1$ y $f_{support}(s_2) = 2$*

Definición 3.7: *Sea $pf = s_j | f_{support}(s_i) \geq umbral$ un patrón frecuente.*

Ejemplo 3.6: *Una vez aplicada la función $f_{support}$ sobre las subsecuencias s_1 y s_2, la única que cumple con un valor mayor o igual a umbral es s_2, por lo tanto esta subsecuencia es un patrón frecuente.*

Definición 3.8: *Un patrón frecuente pf es utilizado para generar nuevas subsecuencias candidatas para la siguiente iteración empleando la propiedad anti-monótona de soporte*

Propiedad 3.1: *La propiedad anti-monótona de soporte considera dos condiciones:*

1. *Si una subsecuencia candidata es un patrón frecuente, entonces todas sus subsecuencias también deben ser frecuentes.*

2. *Si una subsecuencia no es frecuente, entonces todas sus supersecuencias no serán frecuentes.*

   Esta propiedad propiedad se describe en la ecuación 3.1

$$\forall X, Y | (X \subseteq Y) \Longrightarrow f_{support}(X) \geq f_{support}(Y) \tag{3.1}$$

Donde:

- *X es una subsecuencia de Y*
- $f_{support}(X)$ *y* $f_{support}(Y)$ *es la función de soporte de X y Y, respectivamente. Para este caso, representan el número de ocurrencias de aquellas subsecuencias dentro de la secuencia.*

Ejemplo 3.8: *Sea las subsecuencias* $s_1 =< TAA >$ *y* $s_2 =< CCG >$ *pertenecientes a S.* s_2 *es un patrón frecuente, por ende todas sus subsecuencias (*$< CC >$ *y* $< CG >$*) también son frecuentes. Por otro lado,* s_1 *es una subsecuencia candidata que no cumple con el umbral requerido para ser un patrón frecuente, por lo tanto cualquier supersecuencia derivada de esta subsecuencia candidata (<ATAA>, <CTAA>, <GTAA>, <TTAA>, <TAAA>, <TAAC>, <TAAG>, <TAAT>, ...) tampoco será un patrón frecuente.*

Definición 3.9: *Sea* $PF = \{pf_1, pf_2, pf_3, ..., pf_n\}$ *el conjunto de todos patrones frecuentes hallados en una secuencia S de ADN .*

Ejemplo 3.9: *Sea* $PF = \{< AA >, < AC >, < CT >, < GT >, < TT >, < TG >, < AAA >< CCG >< TGT >\}$ *es el conjunto de todos los patrones frecuentes hallados en S.*

Definición 3.10: *Sea el subconjunto* $PF_p \in PF$*, cuyos integrantes son de una iteración p y empleados para generar nuevas subsecuencias candidatas para* $p + 1$*.*

Ejemplo 3.10: *Sea* $pf_1 =< CCG > | pf_1 \in PF_1$*, el empleado para generar las próximas subsecuencias candidatas (*$< CCGA >, < CCGC >, < CCGG >, < CCGT >$*) a ser evaluadas en la siguiente iteración.*

3.2. Algoritmo para una secuencia

Basado en índices localiza las posiciones de los nucleótidos de la secuencia ADN a procesar, para realizar búsquedas rápidas, después llevar a cabo ciclos de búsqueda y evaluación de subsecuencias candidatas a patrones frecuentes. Este proceso está segmentado en tres etapas principales: Mapeo de la secuencia, Generación de secuencias candidatas y Evaluación de secuencias candidatas [12], las cuales son descritas a continuación.

3.2.1. Etapa 1: Mapeo de la secuencia

Este primer paso consiste en obtener las posiciones de cada nucleótido dentro de la secuencia de ADN y ordenarlos en una tabla. Cada fila representa a un elemento de Σ, es decir, habrá cuatro filas, una por cada nucleótido. El contenido de las filas son pares formados de la siguiente manera: *(Pos, nextE)*, en la que *Pos* representa la posición del nucleótido dentro de la secuencia y *nextE* representa el siguiente elemento. La Figura 3.1 muestra el flujo del proceso de creación del mapa.

3.2.2. Etapa 2: Generación de subsecuencias candidatas

Esta etapa se realiza en dos pasos: El primero ocurre inmediatamente después de la etapa anterior en la que se generan 16 subsecuencias candidatas iniciales de longitud 2, debido a que los cuatro nucleótidos base (A,C,G,T) se combinan consigo mismos. En las siguientes iteraciones *i*, este primer paso utilizará un número *t* de pf de longitud *n* obtenidos en la iteración *i-1* (provenientes de la tercera etapa, descrita en la sección 3.2.3) para generar $t * 4$ subsecuencias candidatas, dado que a cada patrón frecuente se le concatena un carácter del alfabeto de nucleótidos Σ, dando cuatro nuevos candidatos de longitud *n+1*. El ejemplo que plantea García Islas [12] es si pf =«AA», se crearán cuatro subsecuencias { «AAA» , «AAC» , «AAG» , «AAT» }, que serán evaluadas por la propiedad anti-monótona de soporte para evitar generar candidatos que no superen el umbral [12]. En la Figura 3.2 se muestra el proceso de generación de subsecuencias candidatas.

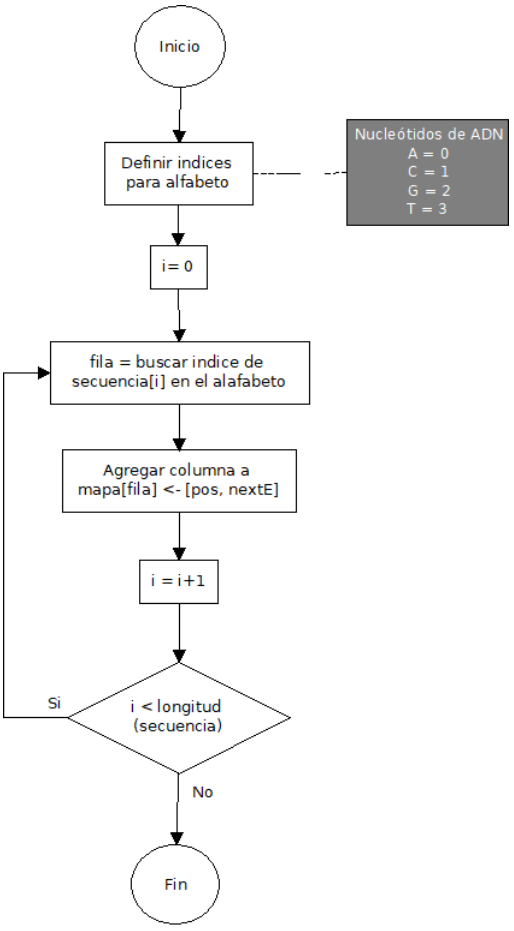


FIGURA 3.1: Generación del mapa de posiciones [12]

3.2.3. Etapa 3: Evaluación de subsecuencias candidatas

Justo después de que las subsecuencias candidatas se han generado, se procede a la evaluación de cada una mediante una novedosa perspectiva de obtención de frecuencias, aplicando un proceso de identificación de pares (posición, nextChar) en el mapa. Se revisa cada nucleótido de la subsecuencia candidata para calcular su número de apariciones dentro de una secuencia y dando así el número de ocurrencias igual a la frecuencia de cada subsecuencia candidata. Posteriormente, se crea un subconjunto PF_j, es decir, aquellas subsecuencias candidatas que cumplen con la regla $f_{support}(subsecuencia\_canditada) \geq umbral$. Este proceso se ilustra en la Figura 3.3. Cabe mencionar que este algoritmo se detiene cuando ninguna subsecuencia candidata cumpla con la frecuencia requerida para ser patrón frecuente, dejando sin elementos a PF_j.

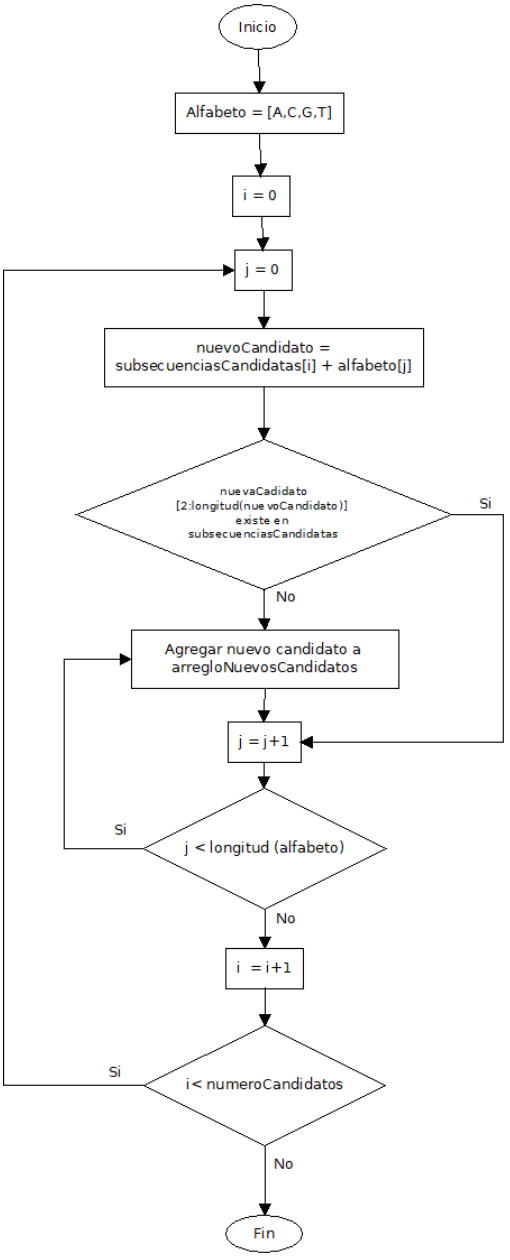


FIGURA 3.2: Producción de subsecuencias candidatas [12]

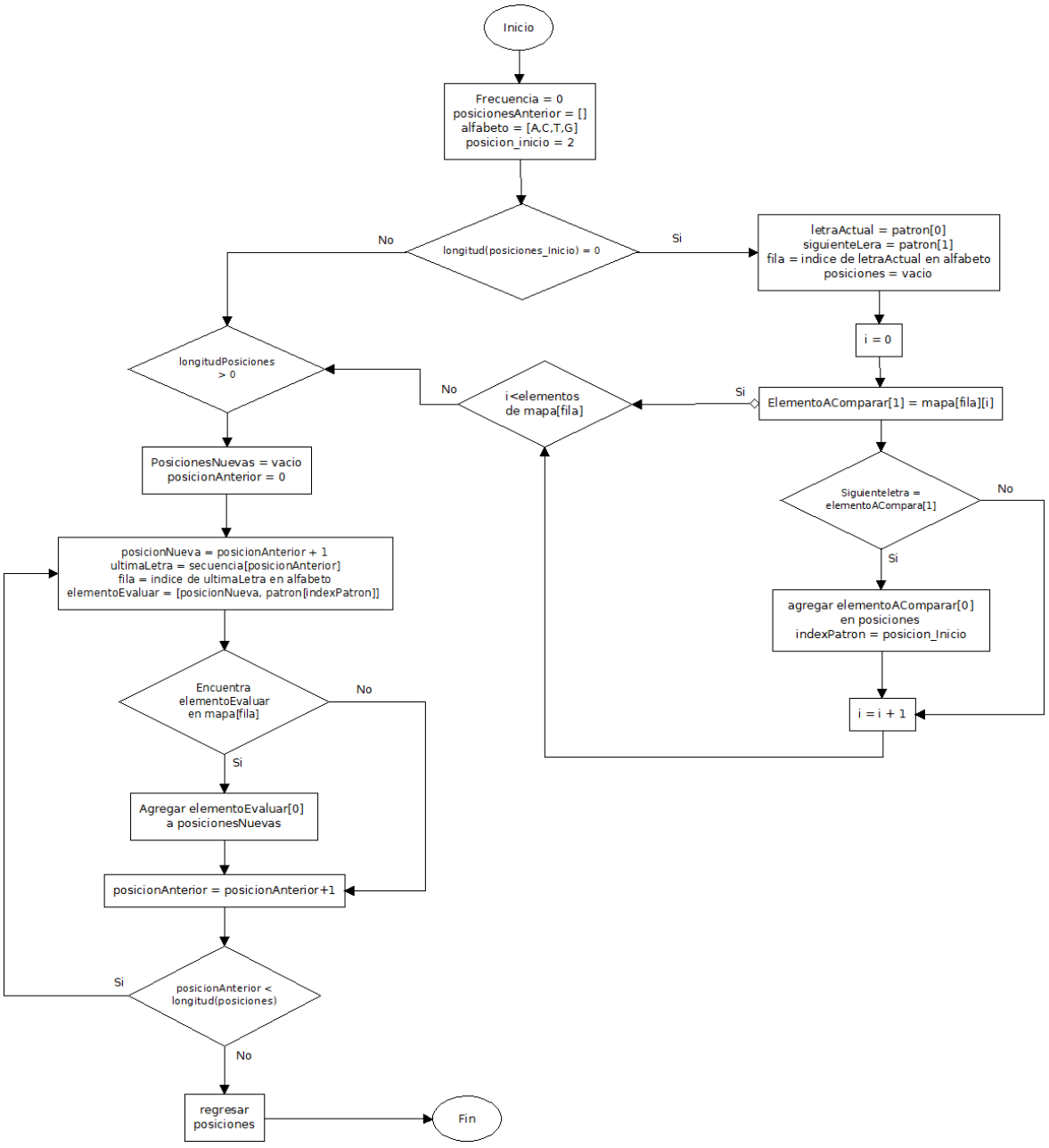


FIGURA 3.3: Obtención de frecuencias de las subsecuencias candidatas [12]

3.3. Propuesta de algoritmo basado en índices en múltiples secuencias

Originalmente el algoritmo Basado en índices está diseñado para trabajar con una secuencia de ADN con el fin de hallar los patrones frecuentes, cuyos elementos que los conforman son contiguos. Teniendo en cuenta que una sola secuencia de ADN es una extensa cadena de caracteres formada por las letras de los nucleótidos adenina (A), citosina (C), guanina (G) y timina (T), se tiene la representación de una gran cantidad de información. La propuesta en este trabajo es crear un método para que pueda procesar n número de secuencias, es decir, en una sola ejecución del algoritmo encontrar los patrones frecuentes en un grupo de múltiples secuencias de ADN.

Para explicar esta nueva estrategia de generación de patrones frecuentes en múltiples secuencias, es necesario presentar nuevas definiciones que rodean a este algoritmo Basado en Índices en Múltiples Secuencias (llamado en adelante BIMS, por sus siglas), sin dejar de lado los conceptos del algoritmo base. Cabe mencionar que se continúa con las definiciones pautadas en la sección 3.1.

Definición 3.11: *sea* $\kappa = S_1, S_2, ..., S_n$ *un grupo de secuencias de ADN a analizar, donde n es el número de secuencias total.*

Ejemplo 3.11: *Sea* $\kappa = \{S_1, S_2\}$ *donde* $S_1 =< ACGTGTAAAACTCTTGTT >$ *y* $S_2 =< CTAAGTCCGTAGCCGACT >$

Cabe mencionar que cada elemento del grupo de secuencias κ tiene su propia longitud, tal como está descrito en la definición 3.2 de la sección 3.1. Los ejemplos 3.12 y 3.13 presentan estos casos.

Ejemplo 3.12: *Sea* $\kappa = \{< ACGTGTAAAACTCTTGTT >, < CTAAGTCCGTAGCCG >\}$ *donde la* $longitud(S_1) = 18$ *y* $longitud(S_2) = 15$

Ejemplo 3.13: *Sea* $\kappa = \{< ACGTGTAAAACTCTTGTT >, < CTAAGTCCGTAGCCGACT >\}$ *donde la* $longitud(S_1) = 18$ *y* $longitud(S_2) = 18$.

De acuerdo a la definición 3.3 descrita en la sección 3.1, se pueden encontrar subsecuencias s_j en una secuencia S de ADN. Para este caso, una subsecuencia s_j puede estar presente en f número de secuencias pertenecientes al conjunto κ, donde $f <= n$. A su vez que las f secuencias del conjunto κ son supersecuencias de s_j.

Ejemplo 3.12: *Sea* $\kappa = \{< ACGTGTAAAACTCTTGTT >, < CTAAGTCCGTAGCCGACT >$ $\}$ *y* $s_1 =< GTA >$, $s_2 =< AAC >$, *y* $s_3 =< CCG >$. *La subsecuencia* s_1 *está presente en las dos secuencias del conjunto* κ; s_2 *solo se aparece en la secuencia* S_1; *por último,* s_3 *solo se presenta en la secuencia* S_2

Tomando como referencia la definición 3.5 de la sección 3.1, una subsecuencia candidata s_j es una subsecuencia que es evaluada con base en las apariciones dentro de S, mediante la función $f_{support}(s_j)$. Para este nuevo algoritmo se presenta la siguiente definición:

Definición 3.12: *Sea la función* $f_{support}$ *que cuenta la cantidad de apariciones de una subsecuencia candidata* s_j *dentro del conjunto de secuencias* κ. *Es decir, el número de secuencias del conjunto* κ *en las que exista dicha subsecuencia, sin importar las veces que se encuentre en cada secuencia.*

Ejemplo 3.15: *Se calcula el* $f_{support}$ *de cada subsecuencia candidata* s_j, *obteniendo* $f_{support}(s_1) = 2$, $f_{support}(s_2) = 1$, $f_{support}(s_3) = 1$

Definición 3.13: *Sea la variable umbral, un valor ingresado por el usuario, y empleado para determinar si una subsecuencia candidata es un patrón frecuente o no, basándose en las* f *apariciones dentro del conjunto* κ. *A su vez,* $umbral <= n$.

Ejemplo 3.16: *Sea el valor ingresado por el usuario* $umbral = 2$.

Tomando la definición 3.7 de la sección 3.1, un patrón frecuente pf es aquella subsecuencia candidata que $f_{support}(s_j) >= umbral$. Con base en esto, se propone la siguiente definición para este algoritmo.

Definición 3.14: *Sea* $pf = s_j | f_{support}(s_j) >= umbral \therefore f_{support}(s_j) <= n$.

Ejemplo 3.17: *Ya aplicada la función* $f_{support}$ *para obtener el numero de apariciones de* s_1, s_2 *y* s_3, *la subsecuencia candidata* s_1 *aparece en la secuencias* S_1 *y* S_2 *del conjunto* κ; *la subsecuencia* s_2 *solo aparece en* S_1; s_3 *dos veces solo en* S_2. *Por lo tanto* s_2 *y* s_3 *obtuvieron un valor* $f_{support}$ *menor a umbral, Por otro lado,* s_1 *pasa a ser un* pf *ya que su número de apariciones es igual a umbral.*

3.3.1. Procedimiento

Los procedimientos del nuevo algoritmo BIMS siguen el mismo flujo que se muestra en la Figura 3.4 y de igual forma, el proceso general se divide en tres etapas principales: mapeo de las secuencias, generación de subsecuencias candidatas y evaluación de subsecuencias candidatas.

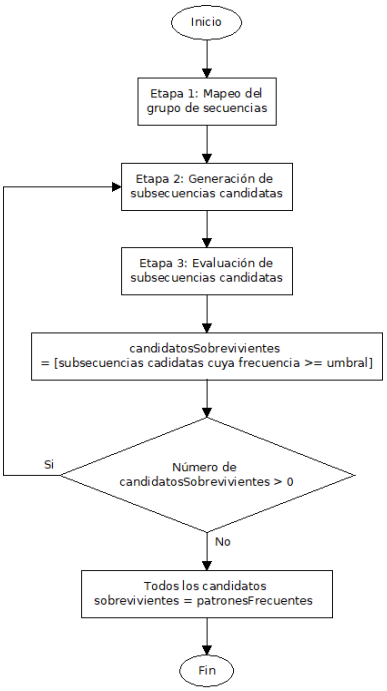


FIGURA 3.4: Flujo general de ejecución de BIMS

El primer paso consiste en mapear las posiciones de los nucleótidos dentro del conjunto de secuencias de ADN para integrarlos en un mapa de búsqueda que será empleado durante las siguientes etapas del algoritmo; Enseguida, se ejecutan procesos iterativos en los cuales se generan subsecuencias candidatas para ser evaluadas con un umbral establecido por el usuario (parámetro *umbral*), que indica el número de secuencias de ADN en las que tienen que aparecer las subsecuencias aspirantes a patrones frecuentes. Si y solo si, el número de apariciones de una subsecuencia candidata es igual o mayor a *umbral*, pasa a ser un patrón frecuente y servirá para generar más subsecuencias candidatas para posteriores iteraciones. Este algoritmo finaliza cuando ya no existen más subsecuencias candidatas que cumplan con *umbral*, y todas los patrones frecuentes hallados en κ junto con sus posiciones son integradas a un conjunto de resultados.

3.3.2. Etapa 1: Mapeo del grupo de secuencias

A diferencia del proceso de mapeo del algoritmo base, en esta etapa se obtienen todas las posiciones específicas de los nucleótidos (identificador de la secuencia, posición y siguiente elemento) dentro del grupo κ, de la siguiente forma:

- *(elmtn)*: refiere a los nucleótidos del ADN, es decir, a los elementos del conjunto Σ.
- *(Seq)*: es el número de la secuencia en la que se ubica el nucleótido.
- *(Pos)*: la posición del nucleótido dentro de la secuencia.
- *(nextE)*: representa el siguiente nucleótido de la posición.

La Figura 3.5 muestra el proceso de construcción del mapa.

3.3.3. Etapa 2: Generación de subsecuencias candidatas

Esta etapa inicia con el ciclo iterativo del algoritmo. Este proceso de generación de subsecuencias es igual al del algoritmo base, el cual se describe en la sección 3.2.2.

3.3.4. Etapa 3: Evaluación de subsecuencias candidatas

Es el segundo proceso dentro de los ciclos iterativos y el último del algoritmo. Para evaluar una *subsecuencia_candidata* se identifica dentro del mapa creado, en la Sección 3.3.2, la tupla que contiene la secuencia (*(Seq)*) , la posición (*(Pos)*) y el siguiente elemento (*(nextE)*), con base en la fila correspondiente a su penúltimo nucleótido, con el fin de obtener el número de apariciones dentro del grupo de secuencias κ, el cual representará la frecuencia de la subsecuencia candidata. Enseguida se evaluará su frecuencia mediante la función $f_{support}$, y aquellas subsecuencias candidatas que cumplan con $f_{support}(subsecuencia\_candidata) \geq umbral$ pasan a ser un patrón frecuente y son empleadas en la siguiente iteración para formar más subsecuencias candidatas.

La Figura 3.6 muestra el diagrama de flujo de este procedimiento de evaluación para subsecuencias candidatas dentro de un grupo de secuencias de ADN. El algoritmo continuará con las iteraciones hasta que ya no se tengan subsecuencias candidatas que superen el umbral, lo que permite a las subsecuencias sobrevivientes ser consideradas el conjunto de patrones frecuentes de κ.

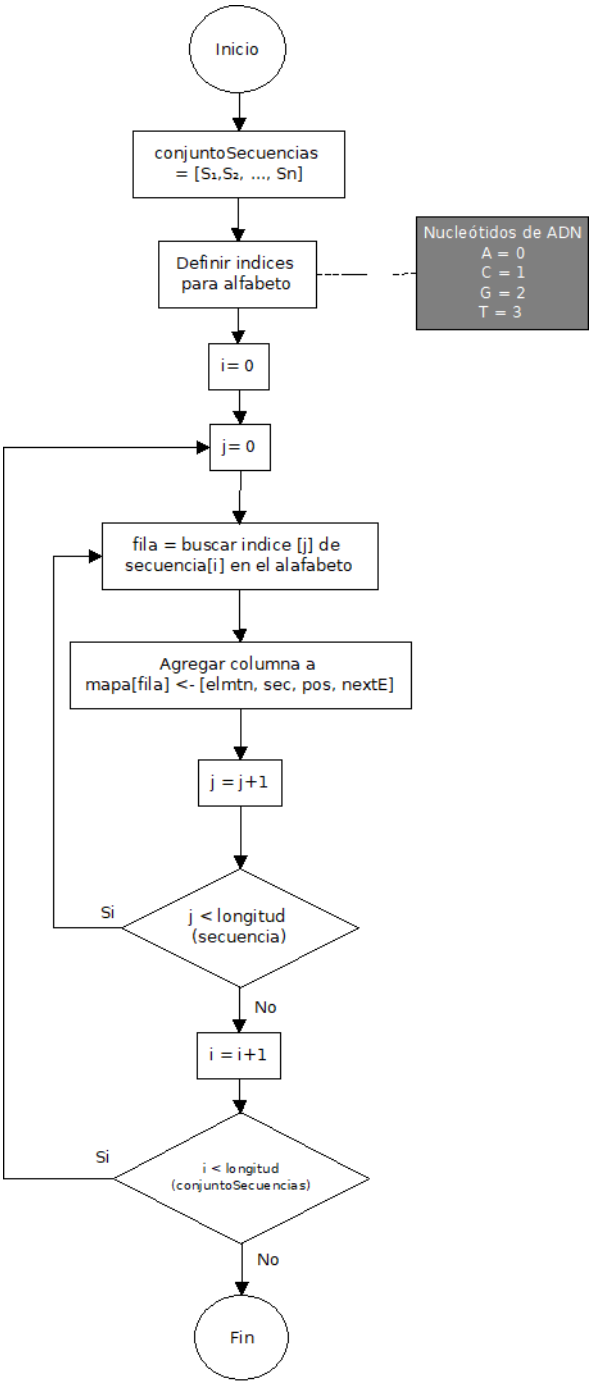


FIGURA 3.5: Generación del mapa de posiciones dentro de un conjunto de secuencias

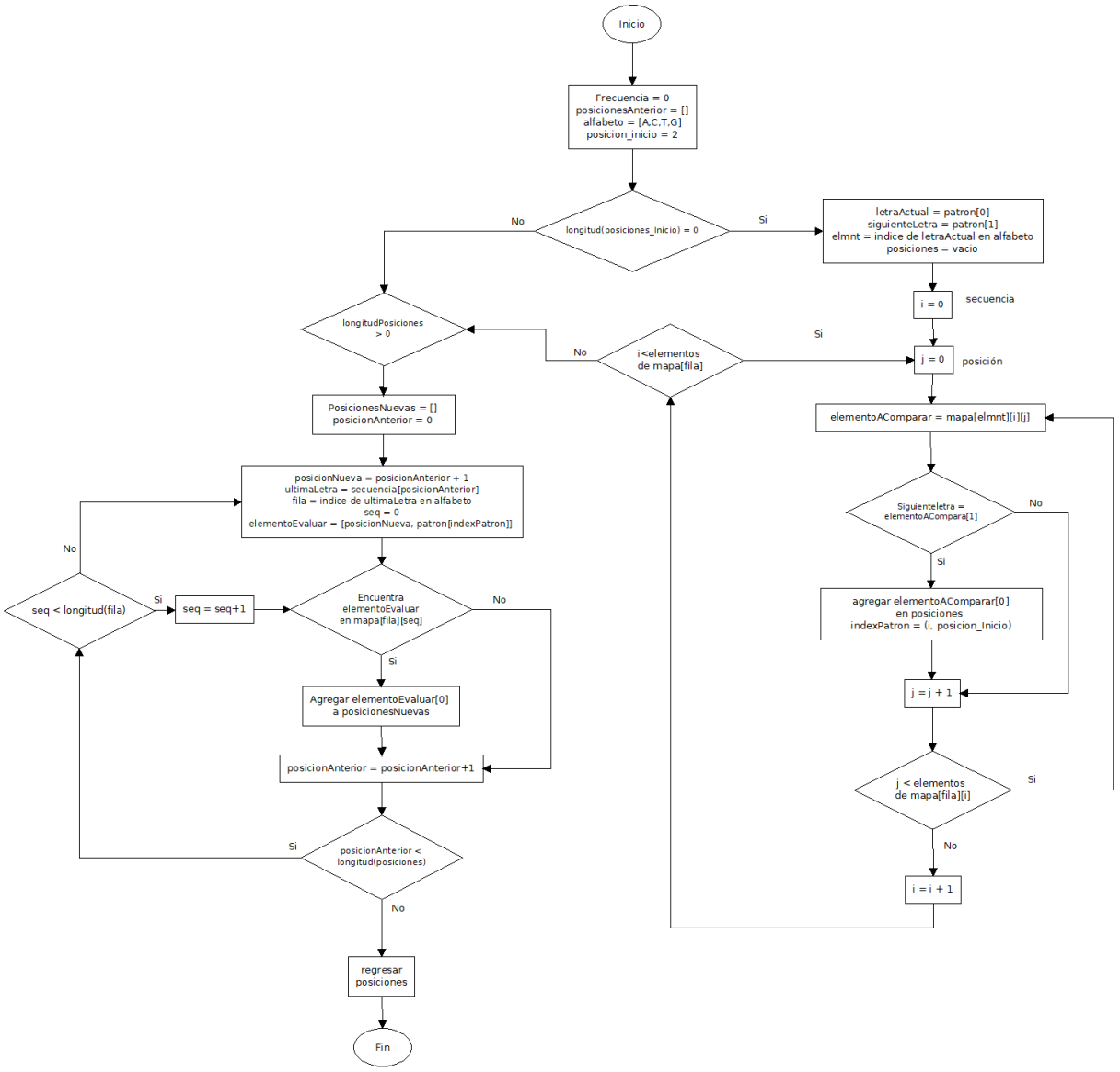


FIGURA 3.6: Extracción de frecuencias de las subsecuencias candidatas, a través del mapa de posiciones

3.4. Ejemplo de aplicación de BIMS

Sea el grupo de tres secuencias de ADN κ = { <*ACGTGTAAAACTCTTGTT*>, <*CTAAGTCCGTAGCCGACT*>, <*GGATCCAATCGCTAATCG*>}, donde $longitud(S_1, S_2, S_3)$ = 18 y cuyo *umbral* asignado a este ejemplo es de 2. Para comenzar el proceso de búsqueda de los patrones frecuentes se mapearán todos los nucleótidos de cada secuencia del grupo κ.

3.4.1. Mapeo del grupo de secuencias

Para el mapeo de los nucleótidos dentro del grupo de secuencias κ, se buscará la secuencia en la que se encuentre el nucleótido, su posición y el siguiente elemento en la secuencia. La Tabla 3.1 muestra el mapa que se construye de κ.

Nucleótido	Secuencia	Posición	Siguiente elemento
A	*S*_1	0	C
		6	A
		7	A
		8	A
		9	C
	*S*_2	2	A
		3	G
		10	G
		15	C
	*S*_3	2	T
		6	A
		7	T
		13	A
		14	T
C	*S*_1	1	G
		10	T
		12	T
	*S*_2	0	T
		6	C
		7	G
		12	C
		13	G
		16	T

...

	S_3	4 5 9 11 16	C A G T G
G	S_1	2 4 15	T T T
	S_2	4 8 11 14	T T C A
	S_3	0 1 10 17	G A C N/A
T	S_1	3 5 11 13 14 16 17	G A C T G T N/A
	S_2	1 5 9 17	A C A N/A
	S_3	3 8 12 15	C C A C

TABLA 3.1: Mapeo de los nucleótidos dentro del grupo de secuencias κ

Una vez mapeadas todas las posiciones de los nucleótidos base (A,C,G,T) dentro del conjunto κ que se muestra en la Tabla 3.1, se proceden a realizar las iteraciones necesarias para encontrar todos los patrones frecuentes de las secuencias.

3.4.2. Iteración 1

Generación de subsecuencias candidatas

Para esta primera iteración se tomarán los nucleótidos base (A,C,G,T) para generar las subsecuencias candidatas de longitud 2 dando como resultado las siguientes subsecuencias:

- **AA**
- **AC**
- **AG**
- **AT**
- **CA**
- **CC**
- **CG**
- **CT**
- **GA**
- **GC**
- **GG**
- **GT**
- **TA**
- **TC**
- **TG**
- **TT**

Evaluación de subsecuencias candidatas

En esta primera evaluación, a cada una de las subsecuencias candidatas generadas en el proceso anterior se les contará las apariciones dentro del conjunto de secuencias κ, a partir de sus posiciones contenidas dentro del mapa presentado en la Tabla 3.1. Para determinar si una subsecuencia candidata será una subsecuencia sobreviviente, esta tendrá que superar o igualar su número de secuencias en las que aparece con respecto al parámetro *umbral*. Esta evaluación se presenta en la Tabla 3.2

Subsecuencias candidatas	Posiciones	Frecuencia	$F_{support}(secuencia\_candidata) = umbral$
AA	S_1: 6,7,8 S_2: 2 S_3: 6, 13	3	True
AC	S_1: 0, 9 S_2: 15 S_3: N/A	2	True
AA	S_1: 6,7,8 S_2: 2 S_3: 6, 13	3	True
AC	S_1: 0, 9 S_2: 15 S_3: N/A	2	True

...

AG	S_1: N/A S_2: 3, 10 S_3: N/A	1	False
AT	S_1: N/A S_2: N/A S_3: 2, 7, 14	1	False
CA	S_1: N/A S_2: N/A S_3: 5	1	False
CC	S_1: N/A S_2: 12 S_3: 4	2	True
CG	S_1: 1 S_2: 7, 13 S_3: 16	3	True
CT	S_1: 10,12 S_2: 0,16 S_3: 11	3	True
GA	S_1: N/A S_2: 14 S_3: 1	2	True
GC	S_1: N/A S_2: 11 S_3: 10	2	True
GG	S_1: N/A S_2: N/A S_3: 0	1	False
GT	S_1: 2, 4, 15 S_2: 4, 8 S_3: N/A	2	True
TA	S_1: 5 S_2: 1,9 S_3: 12	2	True
TC	S_1: 11 S_2: 5 S_3: 3, 8, 15	2	True

. . .

TG	S_1: 3, 14 S_2: N/A S_3: N/A	1	False
TT	S_1: 13, 16 S_2: N/A S_3: N/A	1	False

TABLA 3.2: Número de apariciones de las subsecuencias candidatas de la primera iteración

Como se muestra en la Tabla 3.2, se encontraron las primeras subsecuencias que sobrepasan el parámetro *umbral* = *2* con base en su número de apariciones, las cuales son: **AA, AC, CC, CG, CT, GA, GC, GT, TA y TC**. Por lo tanto, estas subsecuencias formarán parte de las primeras subsecuencias sobrevivientes (*subsecuencias_sobrevivientes = { AA, AC, CC, CG, CT, GA, GC, GT, TA, TC }*) que se emplearán en la siguiente iteración.

3.4.3. Iteración 2

Generación de subsecuencias candidatas

Dentro de la segunda iteración, se tomarán las subsecuencias sobrevivientes de la iteración anterior ({ AA, AC, CC, CG, CT, GA, GC, GT, TA, TC }) para formar las siguientes subsecuencias candidatas combinando los nucleótidos base (A,C,G,T), y dando como resultado las siguientes subsecuencias:

- **AAA**
- **AAC**
- **AAG**
- **AAT**
- **ACA**
- **ACC**
- **ACG**
- **ACT**
- **CCA**
- **CCC**
- **CCG**
- **CCT**
- **CGA**
- **CGC**
- **CGG**
- **CGT**
- **CTA**
- **CTC**
- **CTG**
- **CTT**
- **GAA**
- **GAC**
- **GAG**
- **GAT**
- **GCA**
- **GCC**
- **GCG**
- **GCT**
- **GTA**
- **GTC**
- **GTG**
- **GTT**
- **TAA**
- **TAC**
- **TAG**
- **TAT**
- **TCA**
- **TCC**
- **TCG**
- **TCT**

Evaluación de subsecuencias candidatas

En esta iteración se evaluará cada subsecuencia candidata generada anteriormente tomando como base el número de secuencias del conjunto κ en las que aparece. Aquellas que superen o igualen el valor del parámetro *umbral* serán consideradas las subsecuencias sobrevivientes, tal como se muestra en la Tabla 3.3.

Subsecuencias candidata	Posiciones	Frecuencia	$F_{support}$ $(secuencia\_candidata)$ $= umbral$
AAA	S_1: 6, 7 S_2: N/A S_3: N/A	1	False
AAC	S_1: 8 S_2: N/A S_3: N/A	1	False
AAG	S_1: N/A S_2: N/A S_3: N/A	0	False
AAT	S_1: N/A S_2: N/A S_3: 6, 13	0	False
ACA	S_1: N/A S_2: N/A S_3: N/A	0	False
ACC	S_1: N/A S_2: N/A S_3: N/A	0	False
ACG	S_1: 0 S_2: N/A S_3: N/A	0	False
ACT	S_1: 9 S_2: 15 S_3: N/A	2	False
CCA	S_1: N/A S_2: N/A S_3: 4	1	False
CCC	S_1: N/A S_2: N/A S_3: N/A	0	False

...

Subsecuencias candidata	Posiciones	Frecuencia	$F\_\{support\}(secuencia\_candidata) = min\_sup$
CCG	S_1: N/A S_2: 6, 12 S_3: N/A	1	False
CCT	S_1: N/A S_2: N/A S_3: N/A	1	False
CGA	S_1: N/A S_2: 13 S_3: N/A	1	False
CGC	S_1: N/A S_2: N/A S_3: 9	1	False
CGG	S_1: N/A S_2: N/A S_3: N/A	0	False
CGT	S_1: 1 S_2: 7 S_3: N/A	2	True
CTA	S_1: N/A S_2: 0 S_3: 11	2	True
CTC	S_1: 10 S_2: N/A S_3: N/A	1	False
CTG	S_1: N/A S_2: N/A S_3: N/A	0	False
CTT	S_1: 12 S_2: N/A S_3: N/A	1	False
GAA	S_1: N/A S_2: N/A S_3: N/A	0	False
GAC	S_1: N/A S_2: 14 S_3: N/A	1	False

...

Subsecuencias candidata	Posiciones	Frecuencia	$F\_\{support\}(secuencia\_candidata) = min\_sup$
GAG	S_1: N/A S_2: N/A S_3: N/A	0	False
GAT	S_1: 1 S_2: N/A S_3: N/A	1	False
GCA	S_1: N/A S_2: N/A S_3: N/A	0	False
GCC	S_1: N/A S_2: 11 S_3: N/A	1	False
GCG	S_1: N/A S_2: N/A S_3: N/A	0	False
GCT	S_1: N/A S_2: N/A S_3: 1	1	False
GTA	S_1: 4 S_2: 8 S_3: N/A	2	True
GTC	S_1: N/A S_2: 4 S_3: N/A	1	False
GTG	S_1: 2 S_2: N/A S_3: N/A	1	False
GTT	S_1: 15 S_2: N/A S_3: N/A	1	False
TAA	S_1: 5 S_2: 1 S_3: 12	3	True
TAC	S_1: N/A S_2: N/A S_3: N/A	0	False

...

Subsecuencias candidata	Posiciones	Frecuencia	$F\_\{support\}(secuencia\_candidata) = min\_sup$
TAG	S_1: N/A S_2: 9 S_3: N/A	1	False
TAT	S_1: N/A S_2: N/A S_3: N/A	0	False
TCA	S_1: N/A S_2: N/A S_3: N/A	0	False
TCC	S_1: N/A S_2: 5 S_3: 3	2	True
TCG	S_1: N/A S_2: N/A S_3: 17	1	False
TCT	S_1: N/A S_2: N/A S_3: N/A	1	False

TABLA 3.3: Número de apariciones de las subsecuencias candidatas de la segunda iteración

En la Tabla anterior se muestran las subsecuencias candidatas con su respectivo número de apariciones dentro del grupo de secuencias κ, y solo las que tengan el valor de **True** en la columna de $F_{support}(secuencia\_candidata) = umbral$ son las que superan esta condición, las cuales son: **ACT, CGT, CTA, GTA, TAA, TCC**. Estas serán empleadas en la siguiente iteración y agregadas al grupo de subsecuencias sobrevivientes, quedando hasta el momento de la siguiente forma:*subsecuencias_sobrevivientes = { AA, AC, CC, CG, CT, GA, GC, GT, TA, TC, ACT, CGT, CTA, GTA, TAA, TCC}*

3.4.4. Iteración 3

Generación de subsecuencias candidatas

En este ciclo se generarán las subsecuencias candidatas empleado las sobrevivientes de la iteración anterior { ACT, CGT, CTA, GTA, TAA, TCC }, combinándolas con los nucleótidos base (A,C,T,G). El resultado son las siguientes subsecuencias:

- **ACTA**
- **ACTC**
- **ACTG**
- **ACTT**
- **CCGA**
- **CCGC**
- **CCGG**
- **CCGT**
- **CGTA**
- **CGTC**
- **CGTG**
- **CGTT**
- **CTAA**
- **CTAC**
- **CTAG**
- **CTAT**
- **GTAA**
- **GTAC**
- **GTAG**
- **GTAT**
- **TAAA**
- **TAAC**
- **TAAG**
- **TAAT**
- **TCCA**
- **TCCC**
- **TCCG**
- **TCCT**

Evaluación de subsecuencias candidatas

Una vez formadas las subsecuencias candidatas, se buscarán en el mapa de posiciones y se evaluarán las apariciones de cada una de ellas, tal como se muestra en al Tabla 3.4.

Subsecuencias candidata	Posiciones	Frecuencia	$F_{support}$ $(secuencia\_candidata)$ $= umbral$
ACTA	S_1: N/A S_2: N/A S_3: N/A	0	False
ACTC	S_1: 9 S_2: N/A S_3: N/A	1	False
ACTG	S_1: N/A S_2: N/A S_3: N/A	0	False
ACTT	S_1: N/A S_2: N/A S_3: N/A	0	False
CGTA	S_1: N/A S_2: 7 S_3: N/A	1	False
CGTC	S_1: N/A S_2: N/A S_3: N/A	0	False
CGTG	S_1: 1 S_2: N/A S_3: N/A	1	False

...

CGTT	S_1: N/A S_2: N/A S_3: N/A	0	False
CTAA	S_1: N/A S_2: 0 S_3: 11	2	True
CTAC	S_1: N/A S_2: N/A S_3: N/A	0	False
CTAG	S_1: N/A S_2: N/A S_3: N/A	0	False
CTAT	S_1: N/A S_2: N/A S_3: N/A	0	False
GTAA	S_1: 4 S_2: N/A S_3: N/A	1	False
GTAC	S_1: N/A S_2: N/A S_3: N/A	0	False
GTAG	S_1: N/A S_2: 8 S_3: N/A	1	False
GTAT	S_1: N/A S_2: N/A S_3: N/A	0	False
TAAA	S_1: 5 S_2: N/A S_3: N/A	1	False
TAAC	S_1: N/A S_2: N/A S_3: N/A	0	False
TAAG	S_1: N/A S_2: 1 S_3: N/A	1	False

. . .

TAAT	S_1: N/A S_2: N/A S_3: 12	1	False
TCCA	S_1: N/A S_2: N/A S_3: 3	1	False
TCCC	S_1: N/A S_2: N/A S_3: N/A	0	False
TCCG	S_1: N/A S_2: 5 S_3: N/A	1	False
TCCT	S_1: N/A S_2: N/A S_3: N/A	0	False

TABLA 3.4: Número de apariciones de las subsecuencias candidatas de la tercera iteración

Como se muestra en la tabla anterior, solo una subsecuencia candidata iguala al parámetro *umbral*, y es **CTAA** que se empleará en la siguiente iteración y será agregada al grupo de subsecuencias sobrevivientes, quedando de la siguiente forma: *subsecuencias_sobrevivientes = { AA, AC, CC, CG, CT, GA, GC, GT, TA, TC, ACT, CGT, CTA, GTA, TAA, TCC, CTAA}*

3.4.5. Iteración 4

Generación de subsecuencias candidatas

Para esta producción de subsecuencias candidatas se emplea el elemento **CTAA**, combinándola con los nucleótidos base (A,C,T,G), dando como resultado las siguientes subsecuencias candidatas:

- **CTAAA**
- **CTAAC**
- **CTAAG**
- **CTAAT**

Evaluación de subsecuencias candidatas

Ya formadas las subsecuencias candidatas, se buscarán en el mapa de posiciones y se contabilizará el número de secuencias en las que se halle cada una de ellas, tal como se muestra en la Tabla 3.5.

Subsecuencias candidata	Posiciones	Frecuencia	$F_{support}$ $(secuencia\_candidata)$ $= umbral$
CTAAA	S_1: N/A S_2: N/A S_3: N/A	0	False
CTAAC	S_1: N/A S_2: N/A S_3: N/A	0	False
CTAAG	S_1: N/A S_2: 0 S_3: N/A	1	False
CTAAT	S_1: N/A S_2: N/A S_3: 1	1	False

TABLA 3.5: Número de apariciones de las subsecuencias candidatas de la cuarta iteración

Como se muestra en la tabla anterior, ninguna de las subsecuencias cumple con el número de apariciones requerido para ser considerada como una subsecuencia sobreviviente. Por lo tanto, la ejecución del algoritmo termina en la iteración 4, quedando con un total de 17 subsecuencias sobrevivientes que pasarán a ser patrones frecuentes. Dando como resultado el siguiente mapa de patrón - posiciones:

$patrones\_frecuentes = subsecuencias\_sobrevivientes =$
$\{(AA, [(S_1, 6), (S_1, 7), (S_1, 8), (S_2, 2), (S_3, 6), (S_3, 13)]),$
$(AC, [(S_1, 0), (S_1, 9), (S_2, 15)]),$
$(CC, [(S_1, 0), (S_1, 9), (S_2, 15)]),$
$(CG, [(S_1, 1), (S_2, 7), (S_2, 13), (S_3, 16)]),$
$(CT, [(S_1, 1), (S_2, 7), (S_2, 13), (S_3, 16)]),$
$(GA, [(S_2, 14), (S_3, 1)]),$
$(GC, [(S_2, 11), (S_3, 10)]),$
$(GT, [(S_1, 2), (S_1, 4), (S_1, 15), (S_2, 4), (S_2, 8)]),$
$(TA, [(S_1, 5), (S_2, 1), (S_2, 9), (S_3, 12)]),$
$(TC, [(S_1, 11), (S_2, 5), (S_3, 3), (S_3, 8), (S_3, 15)]),$
$(ACT, [(S_1, 9), (S_2, 15)]),$
$(CGT, [(S_1, 1), (S_2, 7)]),$
$(CTA, [(S_2, 0), (S_3, 11)]),$
$(GTA, [(S_1, 4), (S_2, 8)]),$
$(TAA, [(S_1, 5), (S_2, 1), (S_3, 12)]),$
$(TCC, [(S_2, 5), (S_3, 3)]),$
$(CTAA, [(S_2, 0), (S_3, 11)])\}.$

3.5. Valoración de parámetro

BIMS contiene solo un parámetro de entrada llamado *umbral*, valor que se utiliza para determinar si una subsecuencia es patrón frecuente o no dependiendo de su número de apariciones. Para conocer el comportamiento de BIMS con distintos valores de *umbral*, se realizaron una serie de ejecuciones con distintos valores que van de dos hasta llegar al número de secuencias de ADN a analizar, para ello se utilizaron dos conjuntos de 15 secuencias de ADN: uno con secuencias de la especie *plankton*, con una longitud de 1000 nucleótidos por cada secuencia que lo integra; El otro conjunto de secuencias proviene del espécimen *Homo sapiens*, cuyos elementos tienen distintas longitudes que van desde 274 hasta 2000 nucleótidos. Estos conjuntos de datos se obtuvieron de la base de datos biológica GenBank [6]. La Tabla 3.6 muestra las características antes mencionadas de los grupos de secuencias que se emplearán en los experimentos.

#	Secuencias de igual longitud		Secuencias de diferente longitud	
	Secuencia	Longitud	Secuencia	Longitud
1	HM103452.1	1000	AF145047.1	2000
2	HM103461.1	1000	JA379397.1	1976
3	HM103463.1	1000	HV753254.1	1903
4	HM103487.1	1000	BBHT01000174.1	1857
5	JN825704.1	1000	CAAEXK010000072.1	1609
6	KF528823.1	1000	CAAEZI010000253.1	1609
7	HM103458.1	1000	CAAEBU010000593.1	1488
8	XM_001417990.1	1000	HQ247728.1	1410
9	HM103737.1	1000	HM838795.1	1372
10	XM_001415668.1	1000	HM305513.1	1348
11	XM_001416413.1	1000	GQ873568.1	1023
12	XM_001416515.1	1000	CY112440.1	863
13	XM_001416672.1	1000	GC861521.1	601
14	XM_001417212.1	1000	FU502510.1	274
15	XM_001417499.1	1000	FU502755.1	274

TABLA 3.6: Información de los conjuntos de secuencias de ADN empleados en los experimentos

Una vez conocidas las características de los grupos de secuencias de ADN, se presentan los experimentos realizados para conocer el proceder de BIMS con diferentes valores de *umbral*.

Experimento con conjunto de datos de igual longitud

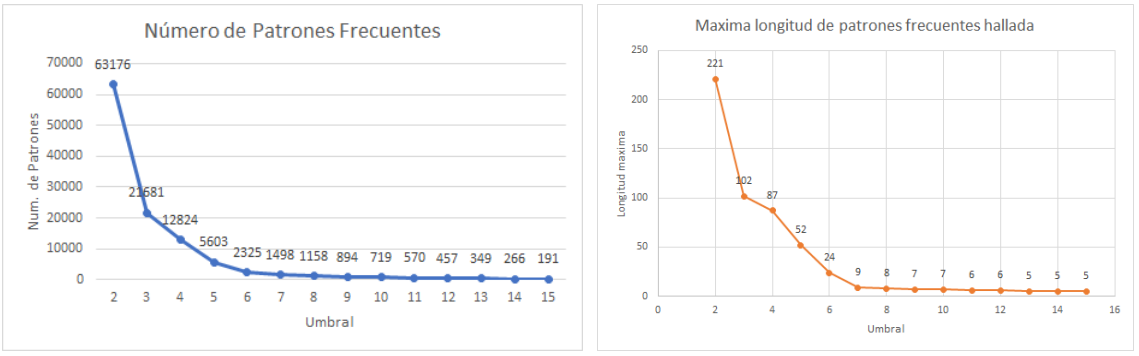
En este primer experimento se realizaron 14 ejecuciones de BIMS en las que cada una tiene un valor de umbral diferente, comenzando con dos e incrementando en cada ejecución hasta llegar a un valor de umbral de 15, misma cantidad de secuencias de ADN del conjunto de entrada. La Tabla 3.7 contiene la información de cada aplicación de BIMS como el umbral, la longitud máxima de los patrones frecuentes hallados, y el número de patrones frecuentes encontrados dentro del grupo de secuencias.

No. ejecución	Umbral	Máxima longitud hallada en los Patrones Frecuentes	Número de Patrones Frecuentes
1	2	221	63176
2	3	102	21681
3	4	87	12824
4	5	52	5603
5	6	24	2325
6	7	9	1498
7	8	8	1158
8	9	7	894
9	10	7	719
10	11	6	570
11	12	6	457
12	13	5	349
13	14	5	266
14	15	5	191

TABLA 3.7: Ejecuciones del algoritmo BIMS con el valor de umbral distinto en cada una de ellas y un conjunto de entrada de elementos de longitud similar

Los datos del primer experimento contenidos en la Tabla 3.7 muestran decremento del número de patrones frecuentes hallados y la máxima longitud a medida que el valor de *umbral* aumenta. Por ejemplo en la primera ejecución que tiene un *umbral* de dos, 63176 patrones hallados y la longitud máxima fue de 221 nucleótidos; En ejecuciones siguientes como la cuarta que tiene un valor de umbral de cinco, 52 nucleótidos de longitud máxima y 5603 patrones frecuentes que están en cuatro o más secuencias de ADN, un número menor que en la primera ejecución; La ejecución ocho con un umbral de nueve arrojó 894 patrones frecuentes con una longitud

máxima de siete, valores demasiado menores en comparación a las ejecuciones anteriores; En la última ejecución que alcanza el valor tope de 15, el número de patrones frecuentes fue de 191 que están dentro de todas las secuencias de ADN y una longitud máxima de 5. Este comportamiento de los patrones frecuentes se visualiza de forma gráfica en la Figura 3.7, en la que se puede apreciar un decremento del número de patrones que se obtienen a medida que el valor de *umbral* aumenta hasta llegar al valor máximo permitido.



((A)) Gráfico de linea del números patrones frecuentes obtenidos por el parámetro umbral

((B)) Gráfico de linea de la longitud máxima obtenida por el parámetro umbral

FIGURA 3.7: Gráficos de linea de los números patrones frecuentes y longitud máxima hallados sobre el parámetro umbral en el conjunto se secuencias de igual longitud

Las gráficas de la Figura 3.7 muestran la tendencia anteriormente mencionada, si *umbral* es igual a dos se encuentra la mayor cantidad de patrones frecuentes de un grupo de secuencias de ADN, y a medida que aumenta el valor del parámetro, BIMS excluye a las subsecuencias cuya frecuencia sea menor a *umbral*, lo que repercute en la cantidad de patrones frecuentes hallados y la longitud máxima sobre los mismos, como se muestra en las Figuras 3.7(a) y 3.7(b) respectivamente. Adicionalmente, el grupo total de patrones frecuentes incluye a todos los elementos de frecuencias de dos hasta 15, como lo muestra la gráfica circular de la Figura 3.8, donde se pueden apreciar los 63176 elementos recurrentes hallados, divididos con su respectiva frecuencia (valor de *umbral*).

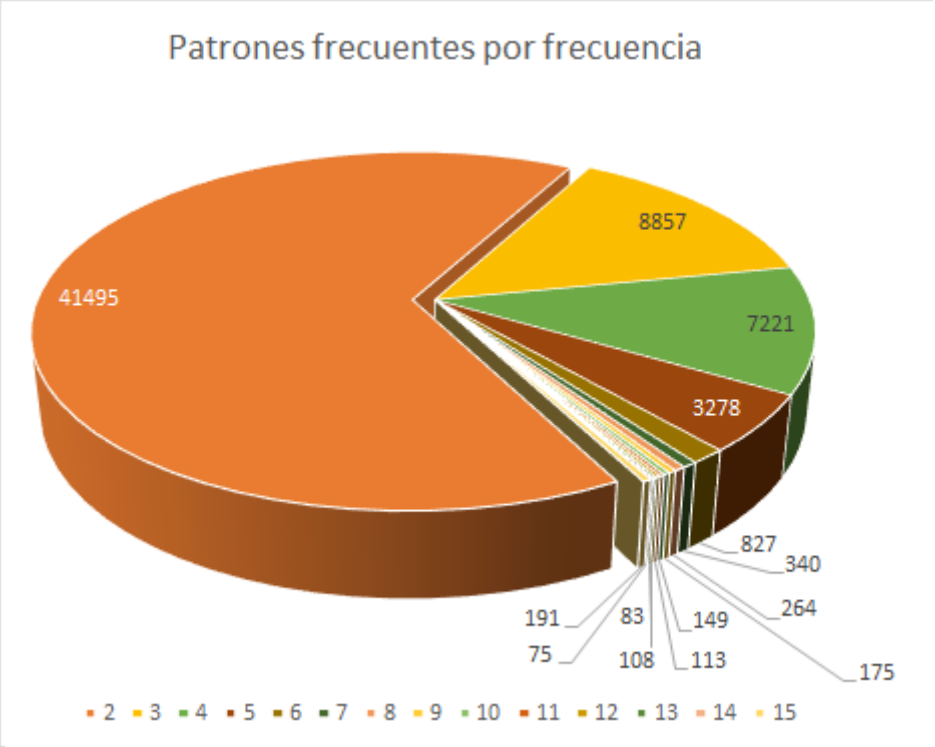


FIGURA 3.8: Conjunto máximo de patrones frecuentes, agrupados por valor de *umbral*, obtenidos con secuencias de ADN de longitud similar

La gráfica 3.8 presenta visualmente la fragmentación del conjunto total de patrones frecuentes por frecuencia, y como el valor de *umbral* influye en BIMS al momento de omitir subsecuencias por su frecuencia. Por ejemplo, si $umbral = 3$, BIMS no tomaría en cuenta a las subsecuencias de recurrencia dos, por tanto, en este caso la sección amarilla del gráfico se excluye y se mantiene el resto de las secciones, que representan los 21681 patrones frecuencia con frecuencias que van de tres hasta 15. Además, a medida que la frecuencia aumenta, el número de patrones frecuentes correspondientes a su respectivas repeticiones decrementa, a excepción de la última frecuencia que aumenta ligeramente. Por ejemplo: 41495 elementos son de frecuencia dos que son la población mas grande el grupo, 7221 patrones frecuentes con frecuencia cuatro, con frecuencia seis 827 patrones frecuentes, 175 elementos con nueve repeticiones, y 191 elementos con frecuencia 15.

En el siguiente experimento se ejecutara BIMS con diferentes valores de *umbral*, pero con un conjunto de secuencias de ADN con diferentes longitudes.

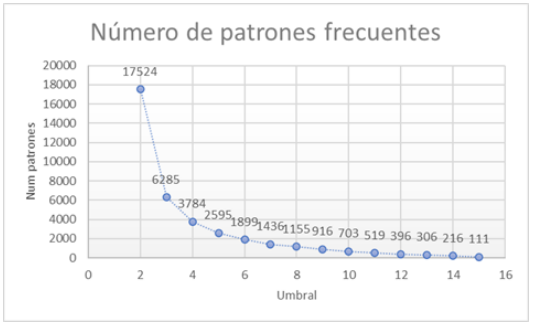
Experimento con conjunto de datos de diferente longitud

Para este segundo experimento, al igual que el primero, se hicieron 14 ejecuciones con el mismo modelo. La primera ejecución con un umbral igual a dos e incrementándose hasta llegar al mismo número de secuencias de ADN del grupo de entrada.

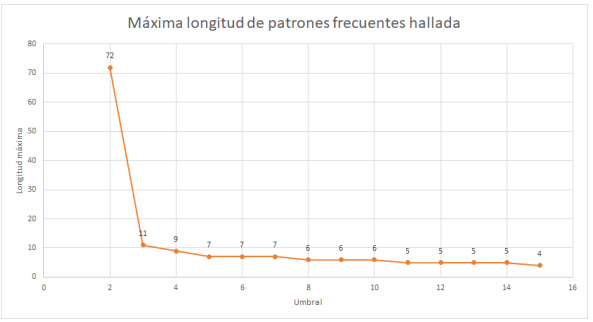
No. ejecución	Umbral	Máxima longitud hallada en los Patrones Frecuentes	Número de Patrones Frecuentes
1	2	72	17524
2	3	11	6285
3	4	9	3784
4	5	7	2595
5	6	7	1899
6	7	7	1436
7	8	6	1155
8	9	6	916
9	10	6	703
10	11	5	519
11	12	5	396
12	13	5	306
13	14	5	216
14	15	4	111

TABLA 3.8: Ejecuciones del algoritmo BIMS con el valor de umbral distinto en cada una de ellas y un conjunto de entrada de elementos de longitud diferente

Considerando que la longitud de los patrones frecuentes hallados depende del tamaño de las secuencias de ADN, la Tabla 3.8 presenta el comportamiento del número de patrones frecuentes similar al experimento anterior, es decir, entre más grande sea el umbral menor será el número de patrones frecuentes. Este comportamiento se puede observar en la primera ejecución, con un umbral de dos, el algoritmo arrojó 17524 patrones frecuentes con una longitud máxima de 72 nucleótidos; ejecuciones posteriores como la siete, con un umbral de ocho, se hallaron 1155 patrones frecuentes con una longitud máxima de 6, cantidad menor que en las ejecuciones anteriores; en la última ejecución, se obtuvieron 111 patrones frecuentes con una longitud máxima de cuatro, debido al valor del umbral. la Figura 3.9 refleja este comportamiento en un gráfico de línea.



((A)) Gráfico de linea del números patrones frecuentes obtenidos por el parámetro umbral



((B)) Gráfico de linea de la longitud máxima obtenida por el parámetro umbral

FIGURA 3.9: Gráficos de los número de patrones frecuentes y longitud máxima hallados sobre el parámetro umbral en el conjunto de secuencias de diferente longitud

De igual forma que el experimento previo, los gráficos de la Figura 3.9 muestran un comportamiento similar: con el mínimo valor de *umbral* que es dos, se obtiene el mayor número posible de patrones frecuentes y longitud máxima, los cuales disminuyen conforme el valor de *umbral* incrementa hasta llegar al tope máximo posible, que en este caso es 15, como lo se muestra en las Figuras 3.9(a) y 3.9(b). Además, el conjunto máximo de patrones frecuentes hallado con *umbral* igual a dos, contiene a todos los elementos encontrados de todas las frecuencias a partir de tres hasta 15. Visualmente este grupo total de elementos recurrentes se muestra en la Figura 3.10.

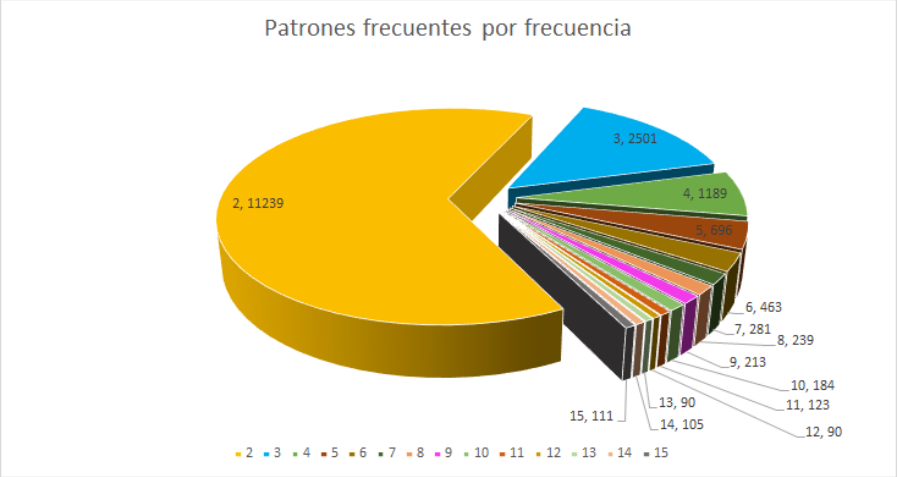


FIGURA 3.10: Conjunto máximo de patrones frecuentes, agrupados por valor de *umbral*, obtenidos con secuencias de ADN de longitud diferente

En el gráfico antes mencionado se muestran los 17524 patrones frecuentes agrupados por su frecuencia correspondiente. Gran parte de estos elementos tiene una frecuencia de 2, y al igual que en el experimento anterior, entre más grande sea la frecuencia, menor número de patrones frecuentes serán encontrados, teniendo solamente un ligero incremento en los últimos conjuntos, como en este caso en las dos frecuencias finales (14,15).

Con base en los experimentos realizados, se observan los siguientes comportamientos:

- A medida que valor de *umbral* es cercano al número de secuencias de ADN del conjunto de entrada, el número de patrones frecuentes hallados y la longitud máxima serán menores.
- Si la frecuencia aumenta, habrá menos elementos recurrentes que tengan dicho número de repeticiones.
- Con $umbral = 2$ se pueden hallar todos los patrones frecuentes de un conjunto de secuencias de ADN. De igual modo, gran parte de estos patrones frecuentes tienen una frecuencia igual a este valor.
- Si el umbral es igual al número de secuencias de ADN del grupo de entrada, se hallará el mínimo número de patrones frecuentes, pero cada uno de ellos estará presente en todas las secuencias del grupo.

- En el caso que los grupos de secuencias de ADN posean distintas longitudes, es posible que la secuencia de menor longitud puede influir en la longitud máxima de los patrones frecuentes.

Con estas observaciones, se presenta lo siguiente: con una *umbral* de menor valor se obtiene una mayor cantidad de patrones frecuentes, entre los que se encuentran los de mayor longitud de nucleótidos, los cuales presentan mayor información, y el caso contrario, un *umbral* mayor arroja una cantidad menor de patrones frecuentes de longitud inferior. Por lo tanto, si se desean extraer patrones frecuentes con mayor información biológica, u obtener el conjunto completo de estos elementos recurrentes de un grupo de secuencias de ADN, ejecutar BIMS con $umbral = 2$ es lo idóneo. Por otra parte, si solo se quieren hallar los patrones frecuentes a partir de una cierta frecuencia hacia delante, se puede emplear BIMS con un *umbral* específico.

3.6. Comparación entre algoritmos de generación de patrones frecuentes

Se realizaron una serie de experimentos con el fin de comprobar la efectividad de *BIMS* frente a otros métodos que se enfocan en encontrar patrones frecuentes de elementos contiguos dentro de conjuntos de datos secuenciales. Las consideraciones que se tomaron para elegir al o los algoritmos a comparar fueron: que el parámetro inicial de frecuencia sea similar al *umbral* de *BIMS*; que permitan generar patrones frecuentes con una longitud mínima de dos; que los patrones frecuentes resultantes tengan elementos frecuentes contiguos; y que permitan obtener el conjunto máximo de patrones frecuentes de un grupo de secuencias de ADN.

Con base en estos requerimientos, se eligió al algoritmo *GSP* (Patrones Frecuentes Generalizados, por sus siglas en inglés) para comparar los resultados del algoritmo propuesto en este trabajo, ya que GSP busca de forma exhaustiva los patrones frecuentes dentro de un conjunto de elementos secuenciales, entregando aquellos que cumplan con su mínimo soporte. Otros algoritmos como PrefixSpan, generan un mapa de posibles candidatos, usando prefijos como elementos únicos que conforman el conjunto y sufijos que representan todas las posibles combinaciones a crear, con el fin de concatenarlos y buscar su frecuencia dentro de un grupo de elementos secuenciales. Sin embargo, si en este algoritmo no se define un límite entre los máximos y mínimos que tienen que tener las subsecuencias a evaluar, este proceso puede tomar demasiado tiempo, o bien si se define un límite máximo pequeño no se podría encontrar el conjunto completo de patrones frecuentes [32].

Las características del equipo que se utilizó para realizar estas pruebas fue:

- **Procesador:** AMD A10-4655M APU 2.00 GHz
- **Memoria RAM:** 12 GB DDR3 1333 MHz
- **SO:** Windows 10
- **Lenguaje de programación:** Python 3.8

Los parámetros con los cuales se ejecutaron los algoritmos fueron: *BIMS* con $umbral = 2$, y *GSP* con $min\_sup = 2$, con la intención de obtener el conjunto completo de patrones frecuentes. Se realizaron siete experimentos, considerando en cada uno diferentes combinaciones de *número_secuencias - longitud_secuencias*, con el fin de observar el desempeño de los algoritmos aplicados a grupos de datos de entrada con diferentes cantidades de secuencias de ADN de distintas longitudes.

La Tabla 3.9 muestra las características de los datos a considerar en los experimentos a ejecutar, como el número de secuencias en cada experimento con sus respectivos identificadores, y la longitud de cada una. Además, se puede apreciar que en los primeros cuatro grupos de secuencias sus elementos son de longitud similar y los últimos tres de longitud variable.

La Tabla 3.10 presenta la información de los resultados encontrados de los conjuntos de secuencias de ADN con los que se ejecutaron los algoritmos *BIMS* y *GSP*. La columna de *Núm. de patrones hallados* indica el número total de patrones frecuentes hallados por los algoritmos. En este caso tanto *BIMS* como *GSP* obtienen la misma cantidad de patrones, ya que al configurar sus parámetros a 2, se solicita hacer una búsqueda exhaustiva y hallar el conjunto completo. Por su parte, la columna *Longitud máx. hallada* visualiza la longitud del patrón frecuente más extenso hallado en cada experimento, esto con el fin de mostrar que dependiendo de cada conjunto de secuencias a procesar, los resultados que se obtienen varían. Finalmente, se muestra la comparación entre los tiempos de ejecución empleados por ambos algoritmos y la diferencia entre estos tiempos, usando el formato hh:mm:ss.ms.

#	Secuencias	Núm de Secuencias	Longitudes
1	AJ318502.1-GQ379194.1-GQ379195.1-AB591821.1-KY706226.1-KY706228.1-MT477783.1-FJ805422.1-GQ996210.1-KU880700.1	10	100-100-100-100-100-100-100-99-99-99
2	GU437440.1-DQ990946.2-AY734690.1	3	1500-1500-1500
3	PTJW01000095.1-PTKA01000272.1	2	3750-3750
4	DQ249187.1-AJ964963.1-AJ965001.1-AJ965002.1-AJ965076.1	5	500-500-500-500-500
5	X04693.1-JC682396.1-KY394541.1-KP332653.1-MK095555.1	5	1000-882-791-660-453
6	CS591378.1-OK188798.1-KX902639.1-DQ080639.1-JB321525.1-JB331440.1	6	750-658-504-335-183-70
7	CS591988.1-AY714251.1-JB321574.1-KT934290.1-MN647657.1-CS592150.1-HC469933.1	7	250-192-140-100-90-70-69

TABLA 3.9: Información de los conjuntos de datos a analizar en los experimentos

#	Núm. de patrones hallados	Longitud máx. hallada	*BIMS*	*GSP*	Diferencia
1	882	74	00:00:00.119976	00:00:01.178200	**00:00:01.058**
2	5438	118	00:00:00.705876	00:00:31.790674	**00:00:31.085**
3	2115	30	00:00:00.439790	00:00:20.283136	**00:00:19.843**
4	6012	223	00:00:00.730803	00:00:19.566574	**00:00:18.836**
5	2317	36	00:00:00.335016	00:00:10.895151	**00:00:10.560**
6	1554	35	00:00:00.353145	00:00:05.275923	**00:00:04.923**
7	993	64	00:00:00.128000	00:00:01.268235	**00:00:01.140**

TABLA 3.10: Experimentos realizados con múltiples secuencias entre *BIMS* y *GSP*.

De acuerdo con los resultados de la Tabla 3.10, a pesar de que se obtiene el mismo número de patrones por ambos algoritmos, cada uno emplea distinto proceso de obtención de patrones frecuentes, lo que se ve reflejado en el tiempo de ejecución que emplea cada uno. La búsqueda exhaustiva que realiza *GSP* se lleva a cabo en segundos, mientras que *BIMS* la obtiene entre 10 y 45 veces más rápido que *GSP*.

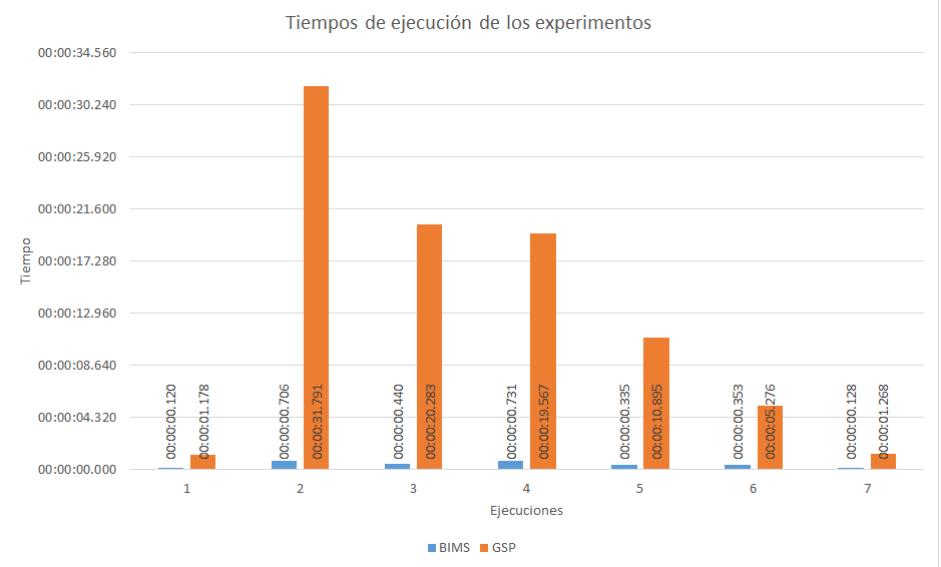


FIGURA 3.11: Gráfica comparativa de ejecuciones entre *BIMS* y *GSP*

La gráfica de la Figura 3.11 muestra en azul los tiempos de ejecución de *BIMS* y en naranja los de *GSP*. En el caso del experimento uno, *GSP* tardó *01.178* segundos contra *00.120* segundos de *BIMS* siendo un 89,81 % mas rápido; En el experimento dos, *BIMS* analizó el conjunto de datos en un tiempo de *00.706* segundos a diferencia de *GSP* que lo hizo en *31.791* segundos, dando una deferencia de 97,78 % a favor de *BIMS*; En el experimento 3, *GSP* demoró *20.283* segundos en analizar el grupo de secuencias de ADN a comparación de *BIMS* que lo hizo en *00.440*, costando a *GSP* un 97,83 % más de tiempo; Para el experimento 4, *BIMS* tardó en realizarlo *00.731* segundos, lo que representa un 3,74 % de lo que hizo *GSP*, que fue de *19.567* segundos; Para el experimento 5, *GSP* empleó *10.895* segundos y *BIMS 00.335* segundos teniendo una reducción en tiempo de 96,93 %; Para el experimento 6, BIMS tardó *00.353* segundos y *GSP 05.276* segundos, teniendo una diferencia de *04.923* segundos o lo que equivale a una mejora de 93,31 % a favor de *BIMS*, finalmente, en el experimento 7, *GSP* tardó *01.268* en hallar los 993 patrones frecuentes dentro del grupo de secuencias de ADN, a comparación de *BIMS* que tardó *00.128* segundos, teniendo una diferencia de *89.91 %*.

La eficacia de *BIMS* frente a *GSP* muestra una mayor celeridad que va desde **89.91 %** hasta **97.83 %**, tomándole a *BIMS* de un **10 %** a un **3 %** de lo que a *GSP* le tomaría procesar un conjunto de secuencias de ADN, beneficiando la búsqueda de patrones frecuentes con un algoritmo más rápido, que obtiene el mismo conjunto de patrones frecuentes.

Capítulo 4

Estrategia de obtención de motifs

Como se ha explicado en el marco teórico, un motif es un patrón recurrente de longitud corta que puede indicar una función biológica. Tradicionalmente, el descubrimiento de un motif se puede llevar a cabo utilizando dos vertientes: conteo de palabras o procesos probabilísticos. En ambos casos, se crea una alineación de n número de subsecuencias de longitud m, es decir, una matriz de nucleótidos $n\ x\ m$. A partir de esta alineación y calculando las modas sobre dicha matriz, se pueden identificar los componentes que formarán un motif.

Este trabajo propone una nueva forma de generar motifs, tomando como base el conjunto de patrones frecuentes contiguos obtenidos en una secuencia o en un grupo de ellas. La detección de los patrones frecuentes dentro de una o varias secuencias de ADN, puede arrojar una gran cantidad de subsecuencias recurrentes que pueden indicar la ubicación de diferentes motifs. Considerando la posición relativa de un patrón frecuente en la(s) secuencia(s), se propone una estrategia que guié en el descubrimiento de motifs, buscando patrones cercanos a este, que puedan ser concatenados para acrecentar la información del motif descubierto. Teniendo en cuenta lo anterior, se plantea la siguiente pregunta de investigación: ¿Se pueden unir dos o más patrones frecuentes, considerando una pequeña variación entre ellos, para enriquecer la información biológica y formar así un motif que proporcione mayor información?

El presente capítulo describe la propuesta del nuevo algoritmo de generación de motifs, así como la evaluación experimental del mismo.

4.1. Procedimiento

El proceso que se realiza en esta nueva estrategia de obtención de motifs comienza con el procesamiento de cada patrón frecuente, obtenidos a partir de los resultados del algoritmo propuesto para la búsqueda de patrones frecuentes en múltiples

secuencias BIMS (véase la sección 3.3). De manera general, para cada patrón frecuente i, se revisa si se puede generar un motif. Primero se hace una búsqueda de todas las apariciones de i en la(s) secuencia(s) y estas se organizan en una lista de tuplas donde cada elemento tendrá el patrón frecuente i, el identificador de la secuencia y la posición. Después se realiza una búsqueda de patrones frecuentes cercanos hacia el frente de cada aparición del patrón i; enseguida se realiza otra búsqueda de elementos recurrentes pero hacía atrás del patrón en cuestión. Una vez terminadas ambas exploraciones, se procede a formar una matriz que ordene todos los elementos de la lista, a fin de crear una matriz de dimensión (x, y), donde x es el número de apariciones del patrón i, y y la longitud de los elementos de la lista. Esta matriz permitirá obtener el conjunto de nucleótidos que representan al motif creado. Cada posición y del motif será evaluada con base en el diccionario de nucleótidos de la IUPAC (Tabla 1.9) para conseguir cada letra que los representará y que será parte del motif final. Por último, se evalúa la longitud de cada motif encontrado, y si cumple o supera el valor de mínima longitud ingresado por el usuario, este será agregado a los resultados finales junto con su respectivo mapa de ubicaciones y subsecuencias. La Figura 4.1 muestra el procedimiento general que se lleva a cabo para encontrar el conjunto de motifs, el cual finaliza cuando todos los patrones frecuentes son analizados.

A continuación se describe a detalle el proceso propuesto para la identificación de motifs en una o varias secuencias de ADN. Para ello, se presentan algunas definiciones que sirven como apoyo para comprender los procesos que lleva a cabo el método.

4.1.1. Definiciones iniciales

Definición 4.1: *Sea* ***longitud_mínima*** *el valor ingresado por el usuario, el cual determina la mínima extensión para ser considerado un motif.*

Ejemplo 4.1: *Sea* ***longitud_mínima*** *= 6*

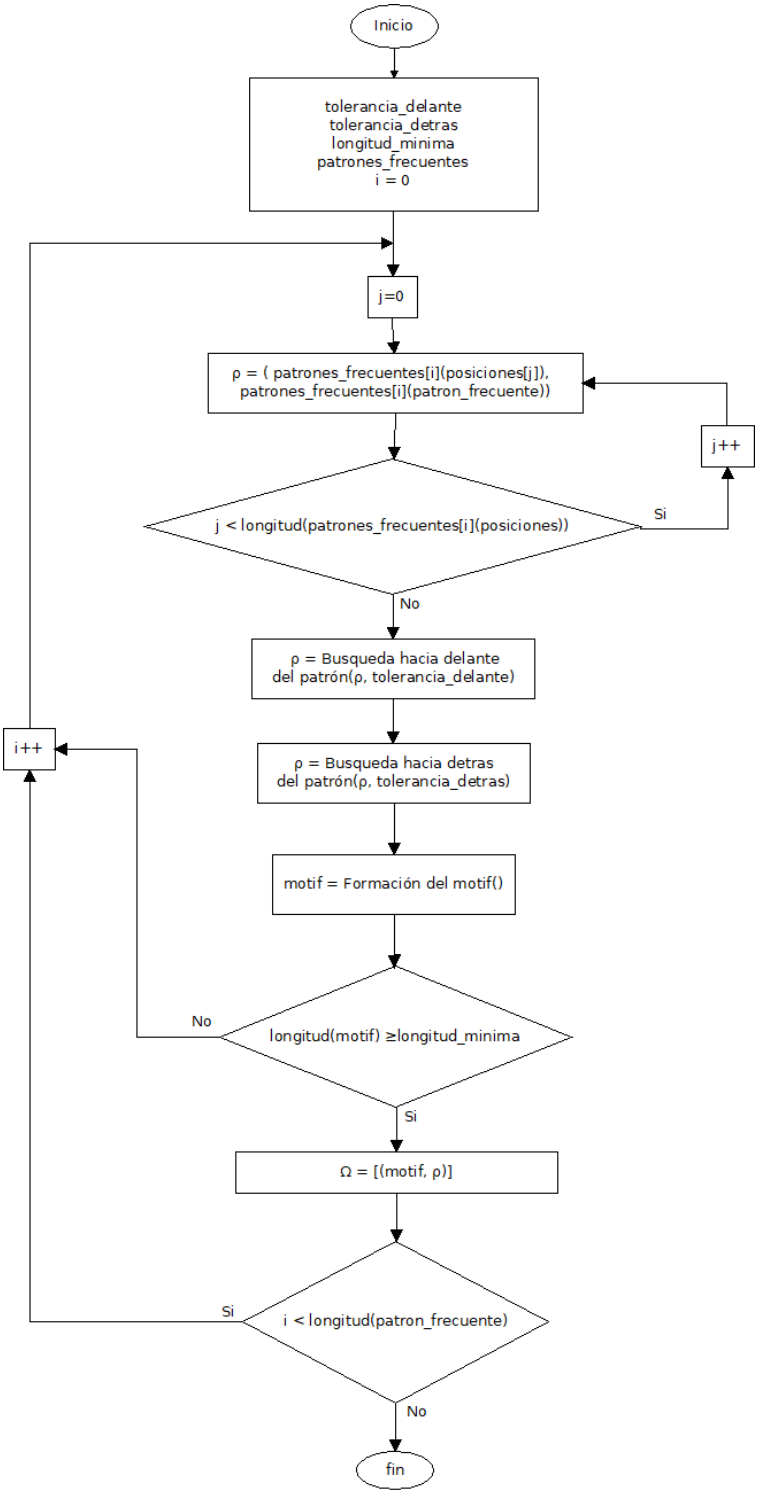


FIGURA 4.1: Diagrama de flujo general de la estrategia propuesta para la obtención de motifs

Definición 4.2: *Se le denomina **posición concreta** a aquellas posiciones específicas dentro de un grupo de subsecuencias ordenadas en las que solo existe el mismo nucleótido, ya sea A o C o G o T.*

Ejemplo 4.2: *La Figura 4.2 presenta cuatro secuencias ordenadas y cuya longitud de cada una de ellas es la misma. En las posiciones 1, 8, 13 y 15 marcadas con color azul domina solo un nucleótido. En la posición 1 la A, en la 8 la G, dentro del lugar 13 la T, y en la posición 15 la C, estas serían las denominadas posiciones concretas.*

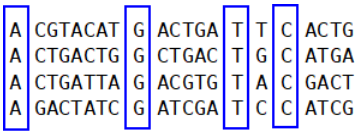


FIGURA 4.2: Ejemplo de posiciones concretas con nucleótidos de un mismo tipo, dentro de un grupo de subsecuencias

Definición 4.3: *Se le denomina **posiciones no concretas** a aquellas posiciones dentro de un grupo de subsecuencias ordenadas en las que existan nucleótidos que no coinciden entre si. En estas ubicaciones se pueden encontrar 2, 3 o 4 nucleótidos base, sin importar su frecuencia, con una sola aparición es suficiente.*

Ejemplo 4.3: *La Figura 4.3 presenta cuatro secuencias ordenadas. En las posiciones 2, 7 y 17, que están marcadas con rojo, se encuentran nucleótidos no concretos. En la posición 2 se hallan los nucleótidos **C y G**; Dentro de la posición 7 se ubican todos los nucleótidos base **A, C, T, G**; Por ultimo, en la posición 17 se localizan **A, T, C** .*

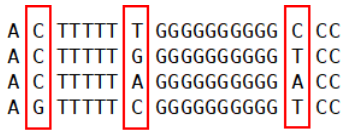


FIGURA 4.3: Ejemplo de posiciones no concretas con nucleótidos diferentes, dentro de un grupo de subsecuencias

Para representar a una posición no concreta se utiliza el código de nucleótidos de la IUPAC, mostrados en la Tabla 1.9. En este ejemplo, en la posición 2 el símbolo IUPAC es *S*, en la posición 7 es *N*, y en la 17 es *H*.

Definición 4.4: *Sea el parámetro ingresado por el usuario **tolerancia_delante**, que determina el número de posiciones no concretas (es decir, que no coinciden), que puede haber entre dos elementos recurrentes por delante del patrón considerado.*

Ejemplo 4.4: *Sea el valor* ***tolerancia_delante*** *= 2*

Definición 4.5: *Sea el parámetro ingresado por el usuario* ***tolerancia_detrás****, que determina el número de posiciones no concretas que puede haber entre dos subsecuencias recurrentes por detrás del patrón considerado.*

Ejemplo 4.5: *Sea el valor* ***tolerancia_detrás*** *= 2*

De acuerdo a las definiciones 3.11 y 3.14, existe un grupo de secuencias de ADN llamado κ y de él se obtiene un conjunto PF de patrones frecuentes (Véase la sección 3.3).

Ejemplo 4.7: *Sea* $\kappa = \{ TCTAAGCGTCACCATCCTGCTCGCCTGGACACCCCCGCCTTCAACATCCTGGACAACCCC \}$ *donde longitud*(S_1)*=61*

Ejemplo 4.8: *Sea el* $patron\_frecuente_i = < CATCCTG >$ *de la secuencia* S_1 *del grupo* κ

4.1.2. Ordenamiento de los datos de acuerdo a un patrón frecuente

A partir de los resultados provenientes de la búsqueda de patrones frecuentes, el proceso comienza organizando todas las ubicaciones del patrón frecuente mediante una lista de tuplas denominada ρ, donde cada elemento contiene la subsecuencia que lo representa como tal, el identificador de la secuencia y la posición. El proceso de creación del conjunto se muestra en la Figura 4.4.

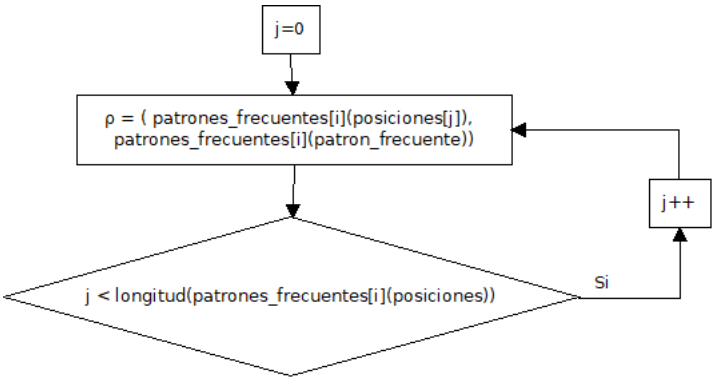


FIGURA 4.4: Diagrama de flujo general de la estrategia de obtención de motifs

El conjunto ρ será utilizado en operaciones posteriores, que se explicarán más adelante.

Definiciones

Definición 4.7: *Sea el conjunto* $\rho \mid \rho_i \in \rho \Longrightarrow \rho_i = (id\_secuencia, posicion, subsecuencia)$, *donde* ρ_i *contiene las ubicaciones (id_secuencia y posicion) dentro de* κ *donde representa al* $patron\_frecuente_i$ *, junto con cada subsecuencia de que representa.*

Ejemplo 4.9: *Sea* $\rho = \{(S_1, 12, < CATCCTG >), (S_1, 44, < CATCCTG >)\}$, *el cual contiene las ubicaciones y subsecuencias que representan al* $patron\_frecuente$

Definición 4.8: *Sea el subconjunto* $\chi \in \rho$ *que contiene solo a las subsecuencias de* ρ

Ejemplo 4.10: *Sea* $\chi = \{< CATCCTG >, < CATCCTG >\}$

4.1.3. Crecimiento hacia delante del patrón

Una vez formado el conjunto de datos de entrada, el primer proceso de análisis a realizar es la búsqueda y adición de los elementos recurrentes por delante del $patron\_frecuente_i$, es decir, a la derecha de las subsecuencias, tal como lo presenta la Figura 4.5 en el que se encierra en un cuadro verde el grupo de subsecuencias que representan al patrón frecuente (subconjunto χ), y las flechas indican la dirección que se toma para realizar esta búsqueda.

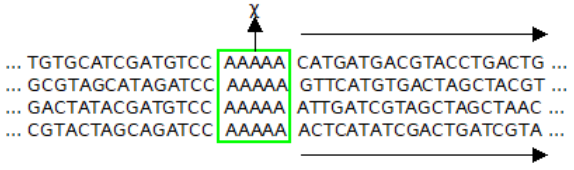


FIGURA 4.5: Planteamiento sobre la búsqueda hacia al frente del patrón frecuente

A partir de las ubicaciones donde termina cada subsecuencia que representa al patrón, se realizará la exploración a través del conjunto de subsecuencias μ que contendrá a los nucleótidos por enfrente (o la derecha) y se añadirán al subconjunto χ de forma iterativa, tomando en cuenta las siguiente condiciones:

- Cada elemento de μ, m tiene una longitud igual o menor a $tolerancia\_delante$+1.

- La adición de nucleótidos a cada m se realiza posición por posición. Si el último elemento añadido de todos los m es una posición no concreta, se siguen añadiendo nucleótidos a μ hasta encontrar una posición concreta y sin revisar la longitud establecida, tal como se presenta en al Figura 4.6.

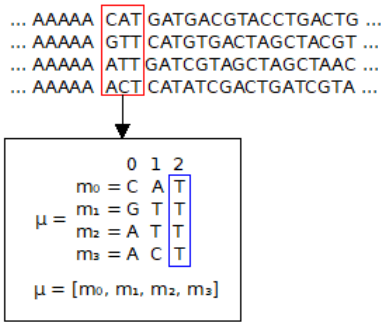


FIGURA 4.6: Ejemplo de concatenación de nucleótidos hasta encontrar una posición concreta, teniendo como parámetro *tolerancia_delante* = 2

El ejemplo de la Figura 4.6 marca en un rectángulo rojo las subsecuencias en las que se realizó la primera búsqueda y que fueron agregadas posición por posición a cada elemento m del conjunto μ hasta encontrar la posición concreta marcada en el recuadro azul.

- Si el conjunto μ tiene una posición concreta, y con la extensión adecuada, se añadirá por delante al subconjunto de subsecuencias χ perteneciente al conjunto ρ, tal como lo muestra la Figura 4.7. Una vez añadidos los elementos de μ con χ, Se comenzará una nueva búsqueda con μ reiniciado desde 0, es decir, eliminando todo los elementos que lo conforman.

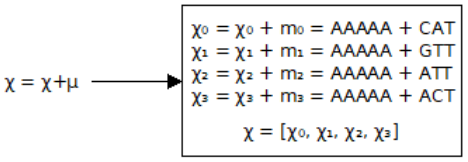


FIGURA 4.7: Ejemplo de concatenación de los elementos de μ con los elementos de la subsecuencia χ

- Si todos los elementos del conjunto μ tiene la longitud igual a *tolerancia_delante*+1 y están conformados por posiciones no concretas, se descarta todo el conjunto, se termina este proceso de búsqueda, y se regresa el conjunto ρ con con su subconjunto de subsecuencias χ.

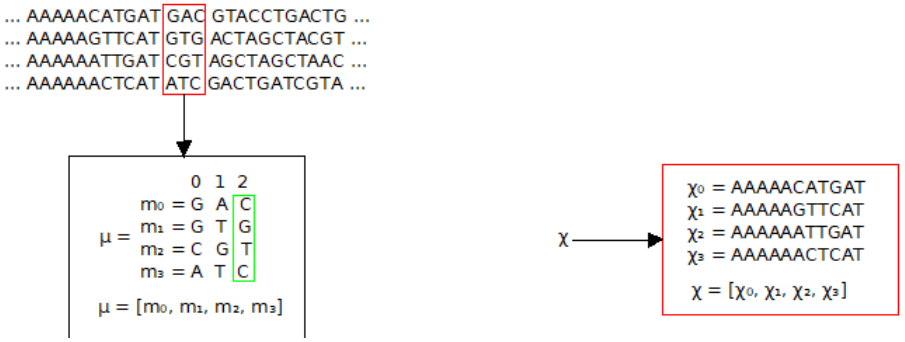


FIGURA 4.8: Ejemplo de terminación de la búsqueda hacia delante del patrón con un conjunto de búsqueda μ conformado por posiciones no concretas.

Como lo muestra la Figura 4.8, se realizó la última búsqueda sin hallar una posición en concreto, en la que se marca en color verde la ultima posición j del conjunto μ. Por ende, ya no se realiza la concatenación con el conjunto χ y este último se retornara con las modificaciones hechas a las subsecuentes para ser analizadas.

- La búsqueda no tiene que superar el límite de la(s) secuencia(s) que le corresponde a cada subsecuencia. En caso de que ocurra esta situación, la búsqueda termina.

Este proceso se muestra en el diagrama de transición de estados presentado en la Figura 4.9, conociendo a detalle las operaciones que se realizan en esta primera búsqueda.

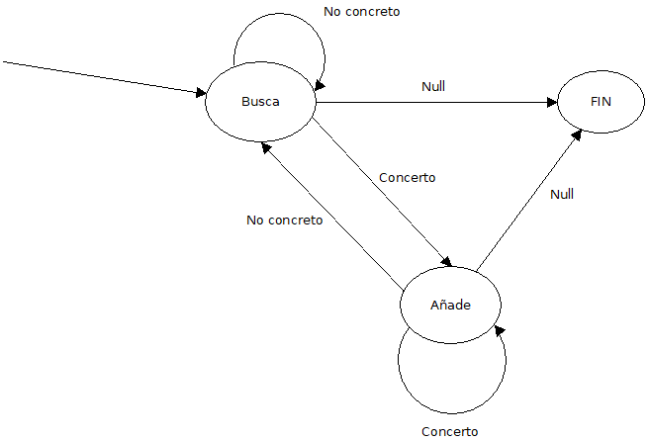


FIGURA 4.9: Diagrama de transición de estados de una unidad m del conjunto μ

Cada elemento m del conjunto μ puede tener dos estados: se comienza con **busca** mientras la entrada sea una posición no concreta, y esta puede cambiar de estado con una posición concreta a **añade**. Si se sigue encontrado una posición concreta, no cambiara de **añade** hasta que se tenga una posición no concreta que lo regresa a un modo de **busca** nuevamente. En ambos estados puede tener un resultado *null*, lo cual finaliza las operaciones del elemento m, y esto puede ser por: Si llegó a la longitud establecida y con todas las posiciones no concretas, o se llegó al límite de alguna secuencia de ADN durante la búsqueda. Establecido el diagrama de transición de estados, la expresión regular que define a es: $m = (A|C|G|T)* \{tolerancia\_delante+\}$, donde puede iniciar con cualquier nucleótido base (A,C,G,T) y puede ampliar con esos mismos nucleótidos hasta $tolerancia\_delante + 1$.

Al final de este procedimiento se retorna el conjunto ρ con las modificaciones realizadas al subconjunto de subsecuencias χ.

Definiciones

Definición 4.9: *Sea el grupo de subsecuencias μ integrado por las subsecuencias que están por delante del patrón frecuente a crecer y cuyo número de elementos es igual al del conjunto ρ.*

Definición 4.10: *Sea el elemento $m \in \mu | m \in \kappa$, y longitud(m) $\leq$ **tolerancia_delante**+1.*

Definición 4.11: *Sea la función $f_{busqueda\_delante}$, en la que aplica $\forall m | m_i = m_i + \kappa[\rho_i[secuencia\_id]]$ $[\rho_i[posicion]+ \rho_i[subsecuencia] + longitud(m_i)] \iff \forall m_i[longitud(m_1)] = \exists!(A|C|G|T) \wedge$ $longitud(m_i) \leq tolerancia\_delante + 1 \therefore \chi_i = \chi_i + m_i$.* De forma iterativa, todos los elemento de μ, m, se les añadirá el nucleótido siguiente a la $\chi_i = \rho_i[subsecuencia]$ respectivamente, sin que la longitud de m_i supere a la **tolerancia_delante**+1. En caso que de se encuentre una posición concreta al final de todos los elementos m y sus longitudes sean menores o iguales a **tolerancia_delante**+1, estos de añadirán al final de cada χ_i. En caso de que no se halle una similitud de un nucleótido en específico al final de todas las m, la función terminará y retornará ρ con sus respectivas modificaciones.

Ejemplo 4.11: *Sea la función $f_{busqueda\_delante}(\rho)$, en la cual se realizaron las siguientes búsquedas*

- **Primera búsqueda:** la primera como resultado las subsecuencias $\mu = \{< CTC >, < GAC >\}$. Dentro de su alineamiento, en la primera posición están las letras <C>y <G>respectivamente, seguido de los nucleótidos <T>y <A>y por último

la letra <C>, la cual es una posición concreta que se encuentra al final de los elementos μ, y finalizando esta primera etapa de búsqueda. Se procederá a añadir estas subsecuencias al final de los miembros del subconjunto χ, quedando de la siguiente forma: $\chi = \{< CATCCTGCTC >, < CATCCTGGAC >\}$.

- **Segunda búsqueda:** da como resultado las siguientes subsecuencias $\mu = \{< GCT >, < AAC >\}$, de las cuales no tienen una posición que predomine, por lo tanto la búsqueda hacia delante se finaliza.

Antes de finalizar la búsqueda, se retorna el mismo conjunto ρ, dando resultado: $\rho = \{(S_1, 12, < CATCCTGCTC >), (S_1, 44, < CATCCTGGAC >)\}$.

4.1.4. Crecimiento hacia atrás del patrón

Teniendo en cuenta lo explicado en la sección 4.1.3, se tomará al conjunto ρ como datos de entrada dando inicio la segunda etapa de búsqueda y concatenación de elementos recurrentes, pero esta vez por detrás de los patrones frecuentes, es decir, por la izquierda del subconjunto χ como lo indican las flechas en la Figura 4.10.

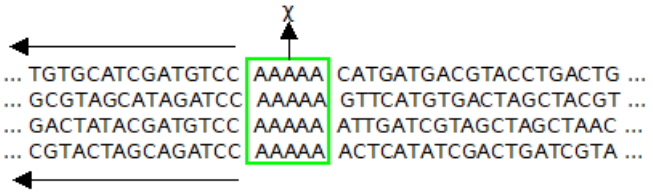


FIGURA 4.10: Planteamiento sobre la búsqueda hacia atrás del patrón frecuente

A diferencia del proceso anterior, la búsqueda se realiza por detrás hasta no encontrar posiciones concretas o, que la exploración llegue al límite de la(s) secuencia(s) donde se encuentra el patrón. Para ello se emplea otro conjunto denominado ν conformado por los elementos n_i que tiene a los nucleótidos hallados atrás de cada una de las subsecuencias del subconjunto χ. Cuando se halle una posición en concreto se añadirá el conjunto ν por detrás de las subsecuencias de χ, pero se tienen que tomar en cuenta ciertas consideraciones al igual que en el crecimiento hacia delante:

- Los elementos n_i pertenecientes a ν tiene una longitud igual o menor a *tolerancia_detras* + 1.

- La adicción de los nucleótidos en cada n_i se realiza posición por posición, situándose al principio de estos elementos. Esta suma de nucleótidos a n_i continúa si hay posiciones no concretas, y termina hasta hallar una posición concreta, cumpliendo con la longitud del punto anterior, tal como lo muestra la Figura 4.11.

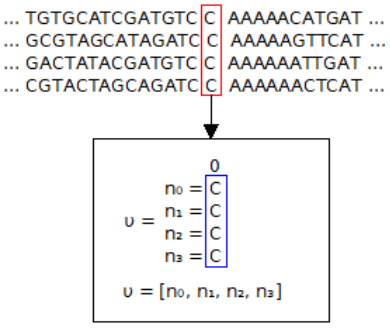


FIGURA 4.11: Ejemplo de concatenación de un solo nucleótido dentro de una posición concreta, teniendo como parámetro $tolerancia\_detras = 2$

El caso de la Figura 4.11 muestra que posterior al patrón se encontró una posición concreta, teniendo la característica para ser añadido por detrás de los elementos de χ. Otro caso se presenta en la Figura 4.12, en el que existe un grupo de subsecuencias de longitud igual a $tolerancia\_detras + 1$ y su concatenación es hacia detrás de los elementos de ν, hasta llegar a una posición concentra, que se ubica al principio del conjunto y está marcado por un recuadro azul.

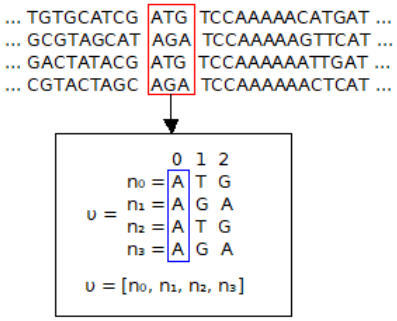


FIGURA 4.12: Ejemplo de concatenación de nucleótidos hasta encontrar una posición concreta, teniendo como parámetro $tolerancia\_detras = 2$

- Tomando en cuenta las características descritas en el punto anterior, si se presenta una posición concreta dentro de ν y sus elementos tienen la longitud adecuada, este conjunto se concatena por detrás de χ, como lo ejemplifica la Figura 4.13.

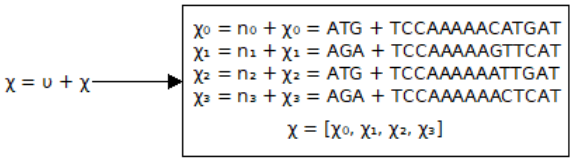


FIGURA 4.13: Concatenación de los conjuntos ν y χ

- Las posiciones de cada uno elementos de ρ se actualizan con base en la longitud de los elementos ν, siguiendo la fórmula: $\forall \rho_i | \rho_i[posicin] = \rho_i[posicin] - longitud(\nu_i)$

El comportamiento que sigue cada n del conjunto de ν es igual al que se presenta en la Figura 4.9. Teniendo cada elemento dos estados: Comenzado con el primero, **busca**, en donde si la posición no es concreta, se queda en el mismo estado; pero si es concreta, pasa al estado de **añadir** hasta que se ubique de nuevo una nueva posición no concreta, lo que provoca regresar al estado anterior (**busca**). De igual manera, la expresión regular que define a los elementos n es la siguiente: $n = (A|C|G|T) * \{tolerancia\_detras + 1\}$, donde puede iniciar con cualquier nucleótido base (A,C,G,T) y puede ampliar con esos mismos nucleótidos hasta $tolerancia\_detras + 1$.

Al igual que en la búsqueda hacia delante del patrón, se retorna el conjunto ρ.

Definiciones

Definición 4.12: *Sea el grupo de subsecuencias ν integrado por las subsecuencias que están por detrás del patrón frecuente a crecer y cuya longitud es igual a la del conjunto ρ.*

Definición 4.13: *Sea el elemento $n_i \in \nu | n_i \in \kappa$, y $longitud(n_i) \leq$* ***tolerancia_detras****+1.*

Definición 4.14: *Sea la función $f_{busqueda\_detras}$, en la que se realiza $\forall n | n_i = \kappa[_i[secuencia\_id]]$ $[\rho_i[posicion] - longitud(n_i)] \iff \forall n_i[0] = \exists !(A|C|G|T) \wedge longitud(n_i) \leq tolerancia\_detras + 1 \therefore \chi_i = n_i + \chi_i$.* De forma similar a la búsqueda hacia delante, se realiza el hallazgo repetitivamente de aquellos nucleótidos y subsecuencias vecinas que dominan una

posición especifica y estén por detrás del patrón frecuente; todos los n que pertenecen a ν se les añadirá el nucleótido(s) que estén detrás de $\rho_i[subsecuencia]$ respectivamente, sin que la longitud de n_i supere a la **tolerancia_detras**+1, y en caso de que se encuentren uno de los nucleótidos base (A, C, G, T) dentro de una posición particular, estos de añadirán al principio de χ respecto a su índice tal como se presenta en la siguiente expresión: $\chi = (A|C|G|T)\{1, tolerancia\_delante + 1\} + \rho_i[subsecuencia]$ y se actualiza la posición de la subsecuencia de ρ, al menos con la longitud de n ($\rho_i[posicion] = \rho_i[posicion] - longitud(n_i)$). En caso que de se cumpla $longitud(n) =$ **tolerancia_detras** $+ 1$, y dentro del ordenamiento de todos los n no exista un nucleótido que no domine una posición en específico, la actual función terminará y retornará ρ con las modificaciones correspondientes.

Ejemplo 4.12: *Sea la función $f_{busqueda\_detras}(\rho)$, donde:*

- ***Primer búsqueda:*** *Da como resultado $\nu = \{< AC >, < AA >\}$, donde nucleótido <A>domina en las primeras posiciones de ν y con una $longitud(v\_i) = 2$. Por lo que se modificarán las "posiciones" y "subsecuencias" de ρ, quedando de la siguiente manera: $\rho = \{(S_1, 10, < ACCATCCTGCTC >), (S_1, 42, < AACATCCTGGAC >)\}$.*
- ***Segunda búsqueda:*** *Se encuentra la letra <C>que predomina la posición posterior a la nueva organización de los elementos de $\rho[subsecuencia]$, $\nu = \{< C >, < C >\}$. Esto se añadirá las "subsecuencias"de ρ y la resta de una unidad a "posiciones", $\rho = \{(S_1, 9, < CACCATCCTGCTC >), (S_1, 41, < CAACATCCTGGAC >)\}$.*
- ***Tercera búsqueda:*** *La siguiente búsqueda muestra al conjunto ν con al elemento <T>que domina la posición a χ, $\nu = \{< T >, < T >\}$, eso significa que se modificaran los elementos de ρ, quedando de la siguiente forma: $\rho = \{(S_1, 8, < TCACCATCCTGCTC >), (S_1, 40, < TCAACATCCTGGAC >)\}$*
- ***Cuarta búsqueda:*** *El conjunto ν muestra la búsqueda actual de los elementos que le anteceden a las subsecuencia ordenadas de ρ: $\nu = \{< CG >, < CT >\}$, eso significa que se modificarán los elementos de ρ, respectivamente: $\rho = \{(S_1, 6, < CGTCACCATCCTGCTC >), (S_1, 38, < CTTCAACATCCTGGAC >)\}$*
- ***Quinta iteración:*** *La quinta búsqueda da como resultado $\nu = \{< AAG >, < CGC >\}$, en que ninguna subsecuencia tiene nucleótidos que coincidan en una posición, por lo que termina la búsqueda y da pie a la formación del motif.*

4.1.5. Formación y Evaluación del motif

Ya con las búsquedas y modificaciones realizadas al conjunto ρ, el siguiente paso es obtener el motif con base en el conjunto de subsecuencias, para ello se ordenan

en el subconjunto χ y posición por posición se evalúa conforme al diccionario de nucleótidos de la IUPAC (*véase la Tabla* 1.9).

La Figura 4.14 muestra la creación de un motif tomando a un grupo de subsecuencias ordenadas, y que son evaluadas posición por posición para obtener su letra correspondiente, las posiciones 1 y 7 por ejemplo, que están marcadas con rojo, obtuvieron la letra **A** porque la posición es concreta con dicho nucleótido. Por otro lado, algunas otras posiciones, como la 3 y 13 marcadas con azul, contienen diferentes nucleótidos, por lo que se tiene que buscar una letra que represente la información de la posición, siendo **K** para la posición 3 y **H** en la 13 las letras correspondientes.

FIGURA 4.14: Construcción de un motif a partir de una organización de subsecuencias.

Por último se toma el motif previamente formado y se evalúa su longitud considerando el valor de $longitud\_minima = 6$. Tomando el motif formado **AKRTCCAAAAAVHTSAT**, y cuya longitud es igual a **17** nucleótidos, por lo que pasa al conjunto ω que contiene el mapa de ubicaciones de los motifs, el cual incluye: la subsecuencia, el identificador de la secuencia y la posición.

Definiciones

Definición 4.15: *Sea motif el resultado del contenido del ordenamiento de todas las "subsecuencias"de ρ. Se evalúa el contenido de cada posición con base en el código de nucleótidos de la IUPAC, y se obtiene una palabra que representa al motif.*

Ejemplo 4.13: *Sea e de las subsecuencias de ρ, en la cual se evalúa cada posición del alineamiento, como lo muestra la Tabla 4.1.*

Secuencia	Subsecuencias\Posiciones	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S_1	$subsecuencia_1$	C	G	T	C	A	C	C	A	T	C	C	T	G	C	T	C
S_1	$subsecuencia_2$	C	T	T	C	A	A	C	A	T	C	C	T	G	G	A	C
Resultado		**C**	**K**	**T**	**C**	**A**	**M**	**C**	**A**	**T**	**C**	**C**	**T**	**G**	**S**	**W**	**C**

TABLA 4.1: Organización de las subsecuencias pertenecientes a ρ

Esta estructuración de subsecuencias crecientes da como resultado el ***motif = <CKTCAMCATCCTGSWC>***

Definición 4.16: *Sea el conjunto de datos ω, el cual resguarda la información del motif (el conjunto π y el $motif$), si y solo si longitud(motif) $\geq$ longitud_minima.*

Ejemplo 4.14: ***motif = <CKTCAMCATCCTGSWC>*** *junto con su respectiva información de alineación de secuencias: $\rho = \{(S_1, 6,$<CGTCACCATCCTGCTC>$),$ $(S_1, 38,$ <CTTCAACATCCTGGAC>$)\}$. La $longitud(< CKTCAMCATCCTGSWC >) = 16 \geq longitud\_minima = 6$, por lo tanto, el motif como su información de crecimiento serán agregados al conjunto ω, resultando lo siguiente: $\omega = [(< CKTCAMCATCCTGSWC >, \{(S_1, 6, < CGTCACCATCCTGCTC >)\ (S_1, 38, < CTTCAACATCCTGGAC >)\})]$*

4.2. Ejemplo del crecimiento de un patrón

Para este ejemplo solo se utilizará un $patron\_frecuente_i$, obtenido en un conjunto κ de secuencias de ADN, para ello se tienen los siguientes datos de entrada:

- $\kappa = \{CTAAGTCCGTAGCCGACT, GGATCCAATCGCTAATCG\}$
- $patron\_frecuente_i =< GC >$
- $patrones\_frecuentes_i = (GC, [(S_1, 11), (S_2, 10)])$
- $tolerancia\_delante = 2$
- $tolerancia\_detras = 2$
- $longitud\_minima = 6$

Definidos los datos de entrada, se procede al paso uno de estructuración de los datos del $patron_frecuente_i$ para su análisis.

Ordenamiento de los datos del patrón

Se tomó la información del mapa de ubicaciones del $patron\_frecuente_i$ para obtener el conjunto ρ, quedando de la siguiente manera: $\rho = [(GC, S_1, 11), (GC, S_2, 10)]$.

4.2.1. Crecimiento hacia delante del patrón ejemplo

Con el conjunto ρ ya creado, se realiza de manera iterativa la primera búsqueda por delante de los patrones frecuentes, utilizando el subconjunto de ρ, χ que solo

contiene el apartado de las subsecuencias ($\chi = [GC, GC]$) y el subconjunto μ para resguardar las subsecuencias a concatenar con χ. En este ejemplo, se efectuaron las siguientes búsquedas.

- **Primer Búsqueda:** Después de pasar por dos posiciones no concretas, se ubicó el primer elemento concreto, resultando en $\mu = [CGA, TAA]$ que se añadirán a los elementos de χ como lo muestra la Figura 4.15.

$$\chi = \chi + \mu \rightarrow \begin{pmatrix} GCCGA \\ GCTAA \end{pmatrix} = \begin{pmatrix} GC \\ GC \end{pmatrix} + \begin{pmatrix} GC \\ GC \end{pmatrix}$$

FIGURA 4.15: Resultado de la primera búsqueda de elementos hacia delante del patrón frecuente GC

- **Segunda Búsqueda:** En esta iteración de la búsqueda hacia delante, se llegó al tope de la secuencia S_1, quedando solo con el conjunto $\mu = [CT, TC]$ contando solo con posiciones no concretas.

Al llegar a límite la secuencia S_1 y como el conjunto μ tiene subsecuencias con posiciones no concretas, se finaliza esta búsqueda. El conjunto ρ retorna con los siguientes valores: $\rho = [(GCCGA, S_1, 11), (GCTAA, S_2, 10)]$.

4.2.2. Crecimiento hacia detrás del patrón ejemplo

Como datos de entrada para esta búsqueda, el conjunto ρ añadirá los elementos por detrás de sus subsecuencias, para ello se usa el conjunto ν y de nuevo el subconjunto χ. Cabe mencionar que este proceso de búsqueda también actualiza cada posición de ρ, en caso de que se lleguen a encontrar y añadir los elementos recurrentes.

- **Primera búsqueda:** En esta primera búsqueda se localizó una posición concreta con una longitud de subsecuencia de dos, quedando $\nu = [TA, TC]$ que se concatenan al frente del patrón, tal como lo muestra la Figura 4.16.

$$\chi = \nu + \chi \rightarrow \begin{pmatrix} TCGCCGA \\ TAGCTAA \end{pmatrix} = \begin{pmatrix} TC \\ TA \end{pmatrix} + \begin{pmatrix} GCCGA \\ GCTAA \end{pmatrix}$$

FIGURA 4.16: Resultado de la primera búsqueda de elementos hacia detrás del patrón frecuente GC

Además, se actualizan las posiciones de las subsecuencias restándoles la longitud de los elementos de ν (en este caso dos), resultando en: $\rho = [(TCGCCGA, S_1, 9), (TAGCTAA, S_2, 8)]$.

- **Segunda Búsqueda:** En esta segunda búsqueda, $\nu = [CCG, CAA]$ tiene a las subsecuencias que se hallaron por detrás de los elementos actualizados de χ teniendo a C como una posición en concreto. Esto da como resultado la concatenación mostrada en la Figura 4.17.

$$\chi = \nu + \chi \rightarrow \begin{pmatrix} CCGTCGCCGA \\ CAATAGCTAA \end{pmatrix} = \begin{pmatrix} CCG \\ CAA \end{pmatrix} + \begin{pmatrix} TCGCCGA \\ TAGCTAA \end{pmatrix}$$

FIGURA 4.17: Resultado de la segunda búsqueda de elementos detrás del patrón frecuente GC

Las posiciones de las subsecuencias se actualizaron a menos tres, quedando el conjunto ρ de la siguiente manera: $\rho = [(CCGTCGCCGA, S_1, 6), (CAATAGCTAA, S_2, 5)]$.

- **Tercera Búsqueda:** En esta iteración se encontró a las subsecuencias $\nu = [AGT, ATC]$, las cuales se concatenarán al principio de los elementos de χ como lo presenta la Figura 4.18.

$$\chi = \nu + \chi \rightarrow \begin{pmatrix} AGTCCGTCGCCGA \\ ATCCAATAGCTAA \end{pmatrix} = \begin{pmatrix} AGT \\ ATC \end{pmatrix} + \begin{pmatrix} CCGTCGCCGA \\ CAATAGCTAA \end{pmatrix}$$

FIGURA 4.18: Resultado de la tercera búsqueda de elementos por detrás del patrón frecuente GC

Las posiciones de las subsecuencias se actualizaron a menos tres unidades, quedando de la siguiente manera: $\rho = [(AGTCCGTCGCCGA, S_1, 3), (ATCCAATAGCTAA, S_2, 2)]$.

- **Cuarta búsqueda:** En esta etapa de la búsqueda se llegó al principio de la secuencia S_2, resultando que $\nu = [TA, GG]$. Por lo que no se ubicó una posición concreta, terminando el proceso de búsqueda.

Este proceso dio como resultado el siguiente conjunto: $\rho = [(GCCGA, S_1, 3), (GCTAA, S_2, 2)]$ que será retornado para emplearse en el siguiente proceso.

4.2.3. Formación y evaluación del motif

Para este procedimiento, se ordenó el subconjunto χ con la finalidad de obtener el valor de cada posición, con base en el código de nucleótidos de la IUPAC. Esto se muestra en la Tabla 4.2.

Secuencia	**Subsecuencias\ Posiciones**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
S_1	$subsecuencia_1$	A	G	T	C	C	G	T	A	G	C	C	G	A
S_1	$subsecuencia_2$	A	T	C	C	A	A	T	C	G	C	T	A	A
Resultado		**A**	**K**	**Y**	**C**	**M**	**R**	**T**	**M**	**G**	**C**	**Y**	**R**	**A**

TABLA 4.2: Organización de las subsecuencias de ρ

Esta organización dio como resultado al motif
$< AKYCMRTMGCYRA >$ con una longitud de 13 nucleótidos, lo que significa que se añadirá al conjunto de resultados ω, quedando de la siguiente manera: $\omega = [(AKYCMRTMGCYR, [(GCCGA, S_1, 3), (GCTAA, S_2, 2)])]$.

4.3. Valoración de parámetros

Al igual que en la sección 3.5, se analizarán los valores necesarios para la ejecución de la estrategia de obtención de motifs (EOM abreviada a partir de ahora) con el objetivo de entender su comportamiento con distintos valores de entrada de *longitud mínima del motif, y las tolerancias hacia delante y atrás*. Al igual que con BIMS, se tomaron los mismos dos conjuntos de 15 secuencias de ADN presentados en la Tabla 3.6 para realizar estas ejecuciones de experimentación. Cabe mencionar que esta estrategia necesita como entrada patrones frecuentes, es por eso que también se empleo BIMS con un $umbral = 2$ para obtener el conjunto total de patrones frecuentes.

Experimento con conjunto de datos de igual longitud

Para este primer experimento, se realizaron nueve ejecuciones para los parámetros de *tolerancia hacia delante y atrás* con valores que van desde 2 hasta 10 en ambos. Para la *longitud mínima del motif* fueron 14 ejecuciones con valores que van desde 6 hasta 20. Cabe mencionar que al momento de realizar la experimentación con un parámetro, el otro tenía su valor por defecto, por ejemplo, durante la serie de ejecuciones enfocado a la *tolerancia hacia delante y atrás* el valor preestablecido de *longitud mínima del motif* no fue modificado, y a su vez, a lo largo de las ejecuciones en las *longitud mínima del motif* cambiaba su valor, la *tolerancia hacia delante y atrás* mantuvieron su valor predeterminado.

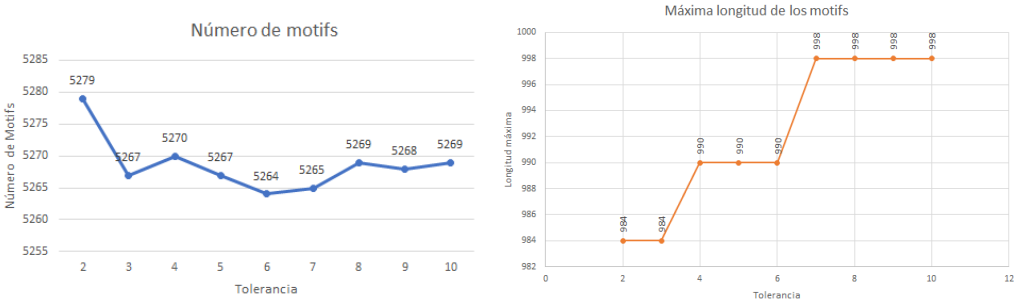
Primero se muestran las ejecuciones enfocadas a la *tolerancia hacia delante y atrás*, cuyos datos están contenidos en la Tabla 4.3, la cual presenta el número de ejecución, las secuencias de ADN que fueron procesadas, la longitud de las secuencias, las tolerancias empleadas (tanto hacia delante o hacia atrás) en cada ejecución, la longitud máxima hallada dentro de los motifs descubiertos, y el número de motifs.

No. ejecución	Tolerancia hacia delante y atras	Maxima longitud hallada en los motifs	Número de motifs
1	2	984	5279
2	3	984	5267
3	4	990	5270
4	5	990	5267
5	6	990	5264
6	7	998	5265
7	8	998	5269
8	9	998	5268
9	10	998	5269

TABLA 4.3: Ejecuciones para EOM con los valores de tolerancia hacia delante y atrás diferentes en cada una de ellas, aplicado a un conjunto de secuencias de ADN de igual longitud

Dentro de la Tabla 4.3 se muestra que a medida que la tolerancia aumenta, la longitud máxima del grupo de motifs descubiertos aumenta también pero forma escalonada, aumentando solo 14 nucleótidos a lo largo de estas ejecuciones. Este comportamiento se muestra forma gráfica se expone en la sub figura 4.19(b), perteneciente a la Figura 4.19.

Asimismo, con el aumento de las tolerancias en cada ejecución, el número de motifs desciende irregularmente teniendo diferencias entre 9 a 15 motifs. Debido a que se aumenta el número de tolerancias, algunos motifs se unen con otro(s) para formar motifs más extensos, y la formación de nuevos motifs a partir de la unión de patrones frecuentes pequeños. Por ejemplo: Entre la ejecución 1 y 2 hubo un descenso en la cantidad de motifs hallados por la concatenación de algunos motifs cercamos gracias al aumento de la tolerancia; y entre la ejecución dos y tres, aumento levemente el número de motifs porque se añadieron nuevos motifs a partir de patrones frecuentes con una mayor tolerancia. Los datos descritos anteriormente se muestran en la gráfica de la Figura 4.19(a).



((A)) Gráfico lineal del número de motifs sobre el parámetros de tolerancia hacia delante y hacia atrás

((B)) Gráfico lineal de la longitud máxima de los motifs sobre el parámetro de tolerancia hacia delante y hacia atrás

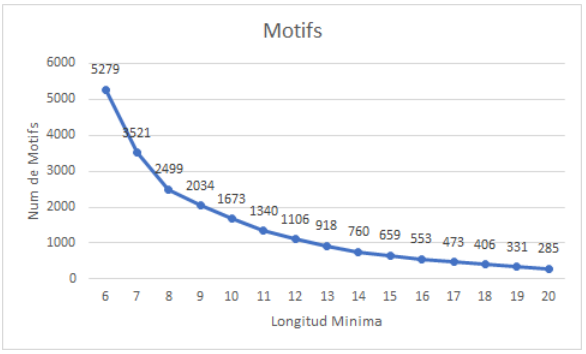
FIGURA 4.19: Gráficos del número de motifs y la longitud máxima hallada sobre los parámetros tolerancia hacia delante y hacia atrás en el primer experimento

Por otro lado, se presentan las ejecuciones relacionadas con el parámetro *longitud mínima del motif* en la Tabla 4.4 en las que presenta la información de cada una de ellas como el número de ejecución, la longitud mínima del motif, la longitud máxima hallada y el número de motifs resultantes.

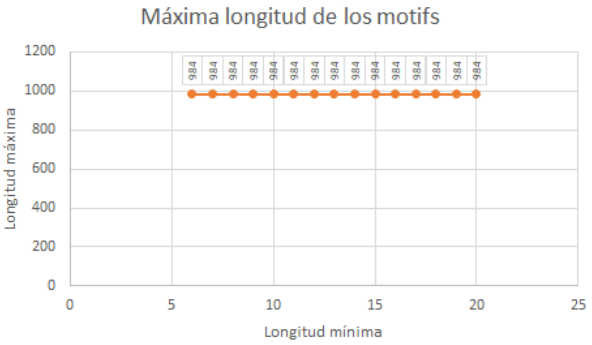
No. ejecución	Longitud máxima del motif	Máxima longitud hallada en los motifs	Número de motifs
1	6	984	5279
2	7	984	3521
3	8	984	2499
4	9	984	2034
5	10	984	1673
6	11	984	1340
7	12	984	1106
8	13	984	918
9	14	984	760
10	15	984	659
11	16	984	553
12	17	984	473
13	18	984	406
14	19	984	331
15	20	984	285

TABLA 4.4: Ejecuciones para EOM con el valor de longitud mínima del motif diferente en cada una de ellas, aplicado a un conjunto de secuencias de ADN de igual longitud

De igual manera que paso con *umbral* de BIMS dentro de la sección 3.5, el parámetro *longitud mínima del motif* a medida de incrementa, el número de motifs disminuye, para este caso va de 5279 a 285 motifs en la última ejecución pero con la diferencia que se mantiene la máxima longitud hallada de 984 nucleótidos en todas las ejecuciones. El comportamiento descrito se muestra visualmente en la Figura 4.20.



((A)) Gráfico lineal del número de motifs sobre el parámetro longitud mínima



((B)) Gráfico lineal de la longitud máxima de los motifs sobre el parámetro longitud mínima

FIGURA 4.20: Gráficos del número de motifs y la longitud máxima hallada sobre el parámetro longitud mínima de motif del primer experimento

Los gráficos de la Figura 4.20, especial la gráfica 4.20(a), muestra que a medida que el valor de *longitud mínima del motif* incrementa, la cantidad de motifs hallados desciende, haciendo que EOM sea más restrictivo con la longitud de los motifs a medida que este parámetro aumenta, pero la longitud máxima a lo largo de las ejecuciones del mantienen, tal como lo presenta el gráfico de la Figura 4.20(b). Por ejemplo, en la primera ejecución con una *longitud mínima del motif* de seis, EOM arrojó 5279 motifs; en la quinta aplicación de EOM que incluye un valor de *longitud mínima del motif* de 10, dio como resultado 1673 motifs; En la ejecución 10 con un valor de parámetro de 15, se presentaron 659 motifs; En la última ejecución se presentaron 285 motifs de longitud mínima de 20. El grupo mayor de motifs se pueden hallar a partir del valor default del parámetro que es de seis, es decir, este grupo contempla a los 5279 motifs de longitud seis hasta la máxima extensión, en este caso de 984 nucleótidos. En la gráfica circular de la Figura 4.21 se muestra la distribución del grupo completo de motifs con base en la longitud.

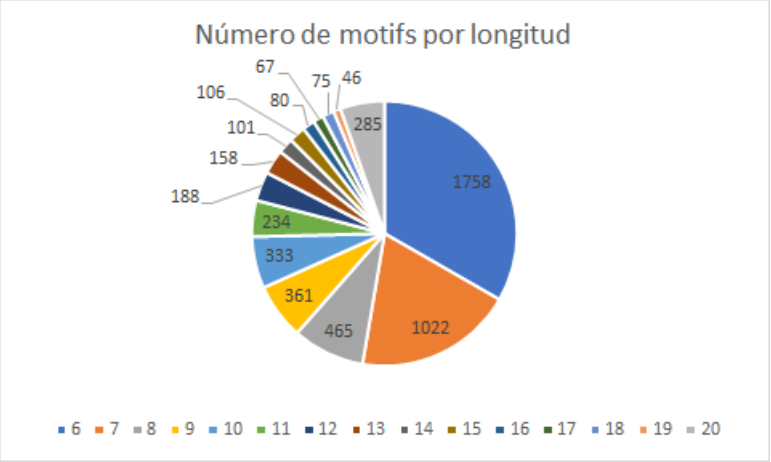


FIGURA 4.21: Gráfico circular de la división del número de motifs sobre su longitud correspondiente

Internamente el grupo completo de motifs se divide con base en la longitud, y a medida que crece la longitud, el número de motif correspondiente a esta extensión decrementa de forma gradual, tal como lo muestra el gráfico 4.21. Por ejemplo, los motifs de longitud dos son 1758, 351 son de longitud nueve, 158 de longitud 15, 80 de longitud 16, 46 de longitud 19, y por último 285 de longitud 20 hasta la extensión máxima de 984.

En el siguiente experimento se aplica EOM en un grupo de secuencias de ADN de diferente longitud, cambiando los valores de *tolerancia hacia delante y hacia atrás* y *longitud mínima del motif* en cada ejecución.

Experimento con conjunto de datos de diferente longitud

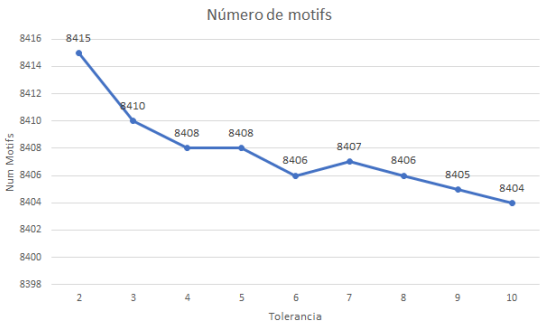
Al igual que el primer experimento, se realizaron nueve ejecuciones para *tolerancia hacia delante y hacia atrás* y 15 para *longitud mínima del motif*. De igual manera, al momento de realizar la experimentación con un parámetro, el otro tenia su valor por defecto, como se menciona en el experimento anterior.

Inicialmente se presentan las ejecuciones centradas en la *tolerancia hacia delante y hacia atrás*, cuyos datos se muestran en la Tabla 4.5 conteniendo el valor de tolerancia utilizado en cada ejecución, la longitud máxima de los motifs, y el número de motifs hallados.

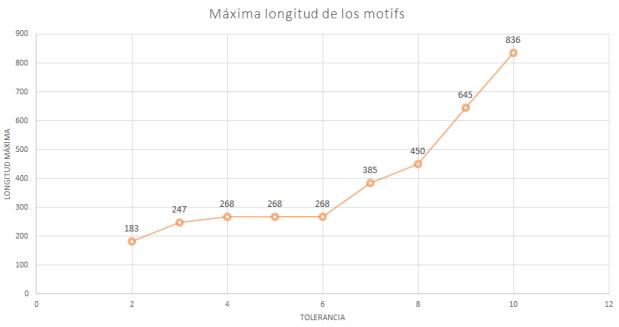
No. ejecución	Tolerancia hacia delante y atras	Máxima longitud hallada en los motifs	Número de motifs
1	2	183	8415
2	3	247	8410
3	4	268	8408
4	5	268	8408
5	6	268	8406
6	7	385	8407
7	8	450	8406
8	9	645	8405
9	10	836	8404

TABLA 4.5: Ejecuciones para EOM con el valores de tolerancia hacia delante y atrás diferentes en cada una de ellas, aplicado a un conjunto de secuencias de ADN de diferente longitud

En este caso, a medida que la tolerancia aumenta, también la máxima longitud de los motifs aumenta y el número de motifs hallados decrementa ligeramente. Por un lado, el decremento en el número de motifs se debe a la unión de uno o varios motifs para formar uno más completo, además que la cantidad total de apariciones de los patrones frecuentes aumenta, haciendo más difícil la concatenación con otros elementos frecuentes, tal como se ve reflejado en el gráfico de la Figura 4.22(a). En el caso de la longitud de los motifs, se debe a que conforme el valor de tolerancia aumenta, el espacio permitido para que dos patrones frecuentes vecinos se unan para formar un motif más completo, aumenta en cada ejecución. Por ejemplo, en la ejecución uno con *tolerancia hacia delante y atrás* de dos, solamente dos nucleótidos como máximo pueden tener de separación dos patrones frecuentes; otro ejemplo es la ejecución cuatro con un valor de tolerancia de cinco, donde el número de posiciones discordantes en que dos patrones frecuentes pueden estar separados aumenta, formando motifs más largos. Ligado con este parámetro, cada ejecución genera diferentes motifs con una distancia de separación entre dos patrones frecuentes distinta, aumentando la longitud de las mismos. El comportamiento de la máxima longitud de los motifs se muestra en la gráfica de linea 4.22(b), de la Figura 4.22.



((A)) Gráfico lineal del número de motifs sobre el parámetros de tolerancia hacia delante y hacia atrás



((B)) Gráfico lineal de la longitud máxima de los motifs sobre el parámetro de tolerancia hacia delante y hacia atrás

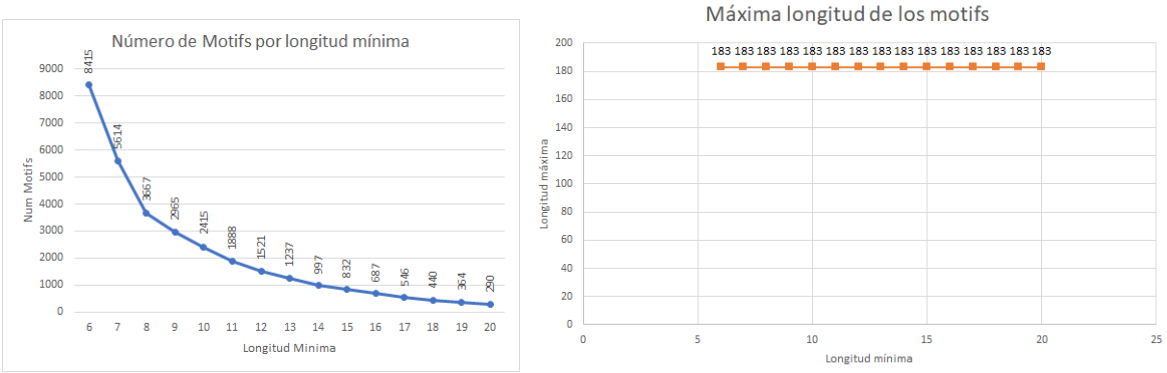
FIGURA 4.22: Gráficos del número de motifs y la longitud máxima hallada sobre los parámetros tolerancia hacia delante y hacia atrás del segundo experimento

Para el parámetro *longitud mínima del motif* se realizaron 15 ejecuciones, cuya información esta contenido en la Tabla 4.6, mostrando el valor del parámetro, la máxima longitud de los motifs y el número de motifs.

No. ejecución	Longitud maxima del motif	Máxima longitud hallada en los motifs	Número de motifs
1	6	183	8415
2	7	183	5614
3	8	183	3667
4	9	183	2965
5	10	183	2415
6	11	183	1888
7	12	183	1521
8	13	183	1237
9	14	183	997
10	15	183	832
11	16	183	687
12	17	183	546
13	18	183	440
14	19	183	364
15	20	183	290

TABLA 4.6: Ejecuciones para EOM con el valor de longitud mínima del motif diferente en cada una de ellas, aplicado a un conjunto de secuencias de ADN de diferente longitud

En este parámetro, los datos de la Tabla 4.6 muestran el mismo comportamiento del experimento mostrado en la Tabla 4.4, a medida que el valor de *longitud mínima del motif* aumenta, la cantidad de motifs hallados se reduce, lo cual se ve reflejado en los datos de la Tabla 4.6 teniendo en la primera ejecución de 8415 motifs, a 290 en la última. Este comportamiento se muestra visualmente en la gráfica de la Figura 4.20.



((A)) Gráfico del número de motifs sobre el parámetro longitud mínima

((B)) Gráfico de la longitud máxima de los motifs sobre el parámetro longitud mínima

FIGURA 4.23: Gráfico del número de motifs y longitud máxima hallada sobre el parámetro longitud mínima de motif del segundo experimento

De la misma forma que sucede en el experimento anterior, la Figura 4.24 expone el comportamiento descrito anteriormente. El gráfico de la Figura 4.23(a) muestra que a medida que el valor de *longitud mínima del motif* incrementa, el número de motifs hallados decrementa y EOM se vuelve más restrictivo con la longitud mínima de los motifs que se están formando, pasando de 8415 en la primera ejecución con un valor en el parámetro de seis a 290 motifs en la última ejecución con un valor de 20. Asimismo, el conjunto de 8415 motifs hallados en la primera ejecución es el mayor grupo resultante que contempla a los motifs de longitudes que van desde seis hasta 20 como máximo. Adicionalmente, la longitud máxima hallada dentro de los motifs resultantes se conservo a lo largo de las 15 ejecuciones del experimento, dando un valor de 183 nucleótidos, tal como lo muestra la Figura 4.23(b).

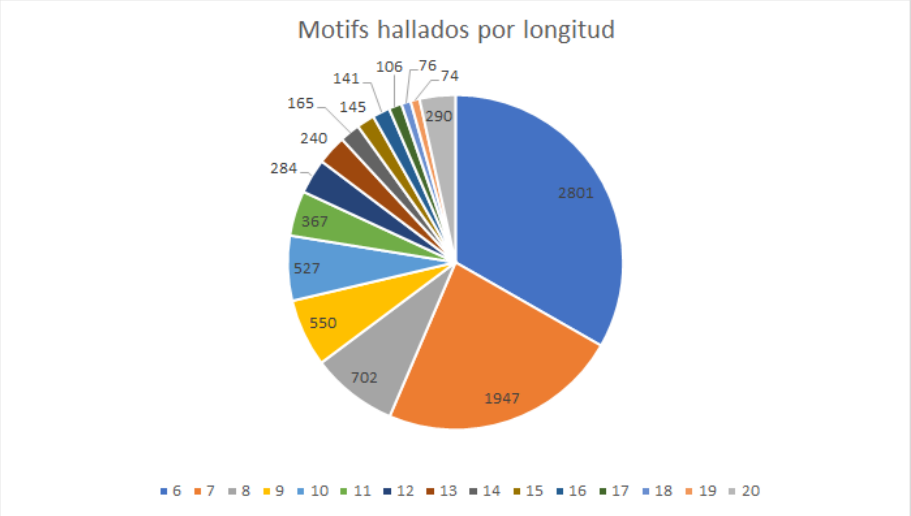


FIGURA 4.24: Gráfico circular de la división del número de motifs sobre su longitud, correspondiente al segundo experimento

La Figura 4.24 muestra la distribución del conjunto con mayor número de motifs con base en su longitud. De igual forma, visto en el experimento pasado, a medida que la longitud crece la cantidad de motifs con dicha longitud disminuye. Por ejemplo: los de longitud 6 tienen una gran mayoría con 2801; los de longitud 10 tienen 527 motifs; 240 motifs de longitud 15; y por ultimo 290 motifs de longitudes que van desde 20 hasta 183 como máximo.

Con los experimentos realizados, se observan los siguientes comportamientos:

- Conforme el valor de *tolerancia hacia delante y atrás* aumenta, la longitud de los motifs aumentará también, pero el número de motifs descenderá ya que algunos se unirán para formar motifs más extensos. También, es posible que a partir de otros patrones frecuentes se presenten nuevos motifs no mostrados en ejecuciones anteriores.
- Un valor de tolerancia puede presentar motifs distintos que los de otro valor.
- A medida que el valor de *longitud mínima del motif* aumenta, la cantidad de motifs resultantes disminuye pero la longitud máxima hallada será la misma.
- Con el valor default *longitud mínima del motif*, que es seis, se hallan todos los motifs de todas longitudes, que van desde seis hasta el máximo permitido.
- Con el valor superior en *longitud mínima del motif* se presentarán los motifs de mayor longitud.

Con un valor *tolerancia hacia delante y atrás* superior, se obtienen una mayor longitud de los motifs lo que significa mayor información, pero la cantidad de motifs resultantes disminuirá. Al igual con el parámetro *longitud mínima del motif* que al aumentar su valor, la cantidad de motifs disminuye pero la longitud máxima de los motifs igual, provocando que el proceso de EOM sea más restrictivo con esta característica. En primera instancia, si se quiere obtener un conjunto resultante de motifs con una cantidad y longitud equilibradas, dejar los valores por defecto de ambos parámetros es lo idóneo. En caso de que se desee obtener motifs con mayor longitud sin importar que la cantidad de motifs disminuya ligeramente, se debe de aumentar el número de tolerancia; junto con esto, si se quieren mostrar solo motifs más extensos se debe de incrementar el parámetro *longitud mínima del motif*.

A comparación de *umbral* de BIMS, la *longitud mínima del motif* y la *tolerancia hacia delante y atrás* no tiene un valor límite porque no se sabe cuántos motifs pueda contener un grupo de secuencias de ADN y cómo se conformarán, solo dependerá de estos dos parámetros y de la configuración que le de el usuario.

4.4. Comparación de resultados

Se realizaron una serie de experimentos con el propósito de comparar los resultados obtenidos por esta estrategia de obtención de motifs. Para ello se tomaron algunas consideraciones, con el fin de que estas pruebas estuvieran en posibles circunstancias similares:

- Las ejecuciones se realizarán con los valores predeterminados.
- La cantidad de motifs resultante sea diferente a uno.
- Los resultados se presentan de diferentes formas (texto, gráficas, archivos, etc.)
- Los resultados serán comparables con los que presenta esta estrategia de obtención de motifs.

Considerando lo anterior se eligió el algoritmo MEME para realizar estas comparaciones, ya que otros algoritmos que están enfocados a esta tarea solo entregan un solo motif, como es el caso de DAMBE o el algoritmo Gibbs Sample; o no están disponibles o están descontinuados para su uso, como YMF o DREME.

Para este experimento se empleará el método BIMS junto con la estrategia de obtención de motifs propuesta en esta tesis (BIMS+EOM, a partir de aquí) comparándolo con lo que realiza el algoritmo MEME dentro de MEME suite. La Tabla 4.7 presenta los parámetros de estos algoritmos, con sus valores predeterminados para obtener

el conjunto máximo de motifs de cada experimento a procesar. En ambos casos, se tienen parámetros similares como el *Umbral* o *Mínimo de sitios* o el parámetro de *longitud mínima* para ambos casos, pero la diferencia es que MEME puede trabajar con el complemento de las secuencias de ADN; delimitar el número de apariciones por secuencia; o poner un límite tope de crecimiento de los motifs.

MEME	BIMS+EOM
Longitud mínima = 6 Longitud máxima = 50 Mínimo de sitios = 2 Máximo de sitios = 50 Uso de la secuencia complementaria del ADN Distribución de aparición = cero o una ocurrencia por secuencia	Umbral = 2 Tolerancia_delante = 2 Tolerancia_detras = 2 Longitud_minima = 6

TABLA 4.7: Valores predeterminados para la ejecución de MEME y BIMS+EOM

Con lo que respecta a los conjuntos de datos que se emplearán para estas experimentaciones, son los mismos que se presentan en la Tabla 3.9, y se toma el proceso completo que realiza cada algoritmo dentro de sus respectivos software, como el análisis de las secuencias de ADN para obtener motifs, y la generación de archivos o gráficos.

La Tabla 4.8 presenta la información de la ejecución de los experimentos MEME y BIMS+EOM en dos columnas principales, en ambas se puede hallar el número de motifs hallado, la longitud máxima encontrada, el tiempo de ejecución en formato hh:mm:ss.000 y la producción entre el número motifs descubiertos en un segundo. Del lado izquierdo se encuentra la numeración del experimento que es correspondiente a la de los conjuntos de datos.

	BIMS +EOM				MEME			
#	Número de motifs	Longitud máxima	Tiempo	Motifs por segundo	Número de motifs	Longitud máxima	Tiempo	Motifs por segundo
1	140	74	00:00:09.478	15.56	32	41	00:00:10.490	3.91
2	993	118	00:01:12.416	13.79	169	50	00:02:28.630	0.34
3	890	30	00:01:10.721	12.54	414	18	00:13:32.520	0.02
4	407	223	00:00:55.088	7.40	64	50	00:00:33.470	1.49
5	975	36	00:01:00.887	15.98	162	45	00:02:23.270	0.31
6	627	35	00:00:39.657	15.68	116	27	00:01:06.860	0.40
7	124	64	00:00:09.571	12.40	37	47	00:00:10.730	4.38

TABLA 4.8: Análisis de experimentos entre BIMS+EOM de obtención de motifs y MEME

A grandes rasgos, la información de la Tabla 4.8 presenta que MEME generó un número de motifs menor a los de BIMS+EOM, llegando a ser hasta seis veces más en un tiempo menor o igual. Esto se ve reflejado en el número de motif/segundo que producen ambos procesos, siendo BIMS+EOM el que fabrica más rápido sus propios motifs, llegando a un aproximado de 15.5 motifs por segundo, cinco veces mas con respecto al experimento de MEME. En lo que respecta a la longitud Máxima hallada dentro de los motifs creados, MEME dentro de sus parámetros establece una longitud máxima de 50 nucleótidos, que llega alcanzar en los experimentos 2 y 4, mismos en los que BIMS+EOM obtuvieron un largo de 118 y 223 nucleótidos respectivamente.

Es claro que BIMS+EOM presenta una nueva forma de producir motifs a partir de patrones frecuentes, llegando a generar un número significativo en un lapso de tiempo similar al de MEME, pero eso no quiere decir que sea un reemplazo del mismo, sino que se brinda una nueva forma de descubrir motifs empleando los algoritmos de minería de patrones frecuentes y uniendo a los elementos recurrentes hallados con la finalidad de obtener motifs con mayor información biológica.

Capítulo 5

Plataforma de Identificación de motifs

En este capítulo se muestra el proceso de desarrollo de la plataforma computacional. Comenzando con la descripción de la metodología de desarrollo, en la que se incluyen las etapas que se realizarán, la obtención de requerimientos, las herramientas de desarrollo que fueron empleadas, y la ejecución de cada fase de la metodología. Posteriormente, se analiza el producto finalizado en las etapas de elaboración de software. Finalmente, se realiza una comparativa de características con otras plataformas enfocadas a esta tarea.

5.1. Descripción de de los procesos de la plataforma

Es necesario conocer cuáles son las acciones principales que realizará la plataforma computacional y quién o quiénes son los individuos involucrados en las actividades. Con base en el estudio del tema y las necesidades de ejecución de los algoritmos que se integrarán, se ha realizado un diagrama de caso de uso que muestra la relación entre los usuarios y el sistema [27], como lo muestra la Figura 5.1.

El diagrama muestra el funcionamiento general que tendrá la plataforma en la que solo está presente un actor, **el usuario**, el cual podrá realizar dos actividades principales: **Iniciar experimento** y **Consultar experimento**. Para el primero, se tienen acciones obligatorias señaladas con la etiqueta «*Include*» que son el *Añadir secuencia(s) de ADN* y *Verificar secuencia(s) de ADN*; Por otro lado, la *Configuración de parámetros*, *Guardar experimento* y *Mostrar experimento* con «*Extend*» son opcionales. La segunda actividad solo tiene las acciones opcionales de *Guardar experimento* y *Mostrar experimento*.

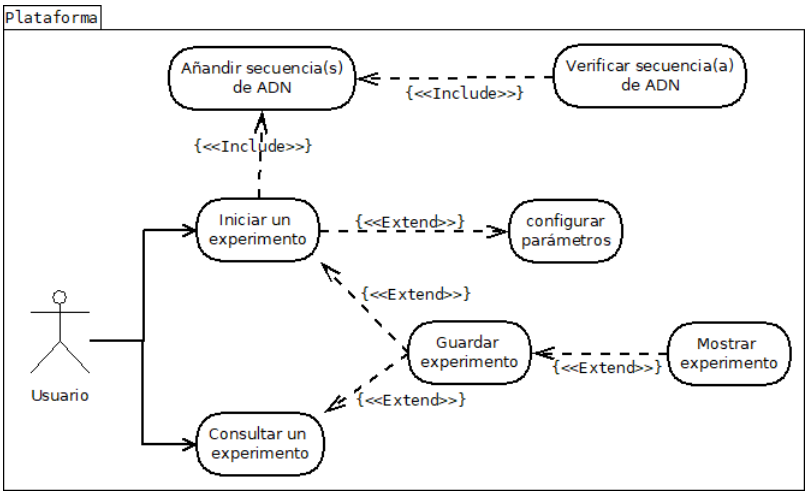


FIGURA 5.1: Diagrama de caso de uso de la interacción general con la plataforma

En cada actividad se realiza lo siguiente:

- **Iniciar experimento**: En esta actividad se podrán ingresar las secuencias de ADN, configurar los parámetros y ejecutar los algoritmos, así como mostrar los resultados como los patrones frecuentes y los motifs
  - *Añadir secuencia(s) de ADN*: Es el proceso en el que se añaden las secuencias de ADN en un formado adecuado.
  - *Verificar secuencia(s) de ADN*: Es el mecanismo que asegura que la secuencia de ADN ingresada sea integra.
  - *Configuración de parámetros*: Es el proceso de ingresar los parámetros que se deseen, pero no es necesario ya que se puede ejecutar con los valores default.
  - *Guardar experimento*: Se guardan los resultados del experimento.
  - *Mostrar experimento*: En este se despliegan los resultados del experimento.
- **Consultar experimento**: Se pueden consultar los experimentos realizados anteriormente.
  - *Guardar experimento*: Se guardan los resultados del experimento.
  - *Mostrar experimento*: En este se despliegan los resultados del experimento.

Teniendo en cuenta estas acciones que realiza la plataforma, se organizaron cinco módulos a desarrollar, donde cada uno está enfocado a una función en particular de

la plataforma y sirven para definir una metodología de investigación. Estos módulos son: entrada de secuencias de ADN, pre-procesamiento de datos, análisis de datos biológicos, estructuración de resultados y despliegue de resultados. Definidos los módulos de la plataforma a desarrollar, se estructura la metodología a emplear que consta de cinco etapas, mismas que conforman a la plataforma computacional, y cada uno cuenta con tareas enfocadas a la construcción de los módulos que conforman a la plataforma. Las fases mencionadas de la metodología base son las siguientes:

- **Fase de entrada de secuencias de ADN:** En esta fase se crean los mecanismos para permitirle al usuario ingresar las estructuras de ADN a analizar, ya sea de forma manual o archivos FASTA. Así mismo, se configuran los parámetros de la ejecución. Para ello se realizan las siguientes actividades:
  - Identificar los medios de entrada de datos y parámetros necesarios para la ejecución adecuada de la plataforma.
  - Diseñar la interfaz de entrada de datos biológicos.
  - Programar la funcionalidad de la interfaz que permita la entrada de datos biológicos.
- **Fase de pre-procesamiento de datos:** Esta etapa se encarga de desarrollar los procesos de validación de datos para evitar errores en la ejecución de la plataforma, además de ordenarlos para su análisis. Esta fase la conforman las siguientes tareas:
  - Diseñar los mecanismos de análisis de verificación para integridad de los datos biológicos.
  - Programar los mecanismos de verificación para datos biológicos.
- **Fase de análisis de datos biológicos:** En esta fase se implementan los algoritmos de minería de patrones frecuentes y el método de búsqueda de motifs aptos para esta tarea.
  - Identificar a los algoritmos de minería de patrones frecuentes aptos para su aplicación.
  - Codificar algoritmos de patrones frecuentes previamente seleccionados.
  - Codificar los procesos necesarios para la obtención de motifs.
- **Fase de estructuración de resultados:** Durante este ciclo se crean los procedimientos para resguardar los patrones frecuentes identificados y los parámetros de ejecución en ficheros JSON, con el objetivo de que sea de fácil acceso y pueda ser empleado en otras herramientas. Además, se tiene un repositorio de experimentos en el que se podrán consultar ejecuciones anteriores.

  - Examinar los datos resultantes del análisis de secuencias de ADN que serán resguardados en formato JSON.
  - Codificar los mecanismos necesarios para la creación de archivos JSON con un formato accesible y comprensible para el usuario.

- **Fase de despliegue de resultados:** En esta fase se crean los mecanismos para mostrarle una forma comprensible al usuario de los resultados obtenidos en una ejecución.

  - Diseñar la interfaz de salida de los datos empleando componentes gráficos y librerías.
  - Desarrollo de la funcionalidad de la interfaz de salida que permita visualizar los datos obtenidos.

5.2. Aplicación de la metodología

Para la aplicación de esta metodología fue necesario integrar otro marco de trabajo que permitiera manejar de la mejor forma los procesos de gestión del desarrollo de la plataforma computacional. SCRUM fue el marco de trabajo seleccionado para esta integración ya que brinda todas las ventajas de una metodología ágil, junto con una serie de actividades que se pueden completarse con otras metodologías. Pantaleo y Rinaudo en el libro *Ingeniería de Software* explican la integración de las metodologías SCRUM y XP para la realización de un proyecto, en que SCRUM contiene todas las actividades de proceso de gestión como la recolección de requerimientos, asignación de roles, realización de reuniones, administración de tareas a efectuar, retroalimentación, entre otros. Por otro lado, XP contiene las actividades relacionadas con el proceso técnico como la forma en que se programa el software, las pruebas a realizar, tiempo de trabajo, entre otras actividades que le permiten conocer al programador las condiciones de desarrollo [27].

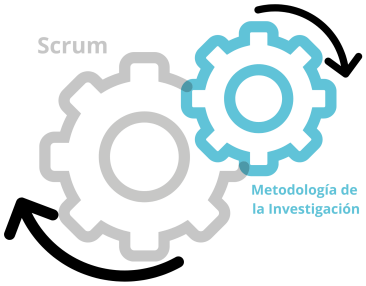


FIGURA 5.2: Integración entre Scrum y la metodología de la investigación

Para este caso en particular, la integración ambas metodologías (como se muestra en la Figura 5.2) se llevará a cabo mediante las fases que conforman la metodología de investigación y se convertirán en sprints para SCRUM, tal como se muestra en la Tabla 5.1. Esto permitirá ejecutar las fases cuantas veces sea necesario para el proyecto, y obtener una versión de la plataforma en un lapso corto de tiempo a medida que se integran las funcionalidades con el paso de las iteraciones.

Fase	Sprint
Fase de entrada de secuencias de ADN	Sprint 1: Entrada de secuencias de ADN
Fase de pre procesamiento de datos	Sprint 2: Pre-procesamiento de datos
Fase de análisis de datos biológicos	Sprint 3: Análisis de datos biológicos
Fase de estructuración de resultados	Sprint 4: Estructuración de resultados
Fase de despliegue de resultados	Sprint 5: Despliegue de resultados

TABLA 5.1: Incorporación de fases de la metodología de investigación a sprints para la aplicación de la metodología SCRUM

Con esta estructuración de las fases y el funcionamiento en conjunto de una metodología base y un marco de trabajo ágil para el desarrollo de software, se tendrá un entorno flexible ante los cambios que se presenten durante el desarrollo de la plataforma, lo cual para este tipo de proyectos es ideal. Lo siguiente es la elección de las herramientas que se emplearán para la creación de la plataforma computacional.

5.2.1. Elección de herramientas

Con base en las tareas que se llevan a cabo en cada ciclo, se decidió que la plataforma sea de tipo web debido las características de este tipo software que encajan

con el proyecto, como el ejecutar en un navegador sin necesidad de utilizar algún otro complemento gráfico. Junto con esto, se utiliza el lenguaje de programación Python, ya que es un lenguaje multiplataforma y utilizado para diferentes propósitos como análisis de datos, computo científico o programación web, siendo estas características idóneas para este proyecto. Para la construcción de esta plataforma se emplean las siguientes herramientas:

- **HTML y CSS(Bootstrap)**: Con estos dos lenguajes se desarrollarán paginas web con las que el usuario va interactuar.
- **JavaScript**: Este se encargará de darle funcionalidad a las paginas web y comunicación con el servidor.
- **Python**: Este lenguaje de programación se empleará para la creación del servidor y la codificación de los algoritmos de minería de patrones frecuentes que se utilizaran.
  - LogoMaker: Es una librería de python para la generación de LOGOS de secuencias.
  - Flask: Es el framework de Python enfocado a la creación de servidores web ligeros y a medida.
  - Archivos py: Estos archivos contendrán a las clases de los algoritmos de minería de patrones frecuentes que se utilizarán en el proyecto.
- **Archivos JSON**: Se utilizarán para estructurar los resultados de las ejecuciones. Además de enviar y recibir datos entre la página y el servidor.
- **Archivos CSV**: Se utilizarán para el resguardo de información resultante del proceso de obtención de patrones frecuentes.
- **Archivos XLS**: Resguardarán la información resultante del proceso de obtención de motifs.

Cada una de estas herramientas está enfocada a un funcionamiento en particular en la plataforma, como se ha descrito anteriormente. De forma gráfica, la Figura 5.3 muestra la interacción y aplicabilidad de estos lenguajes de la plataforma.

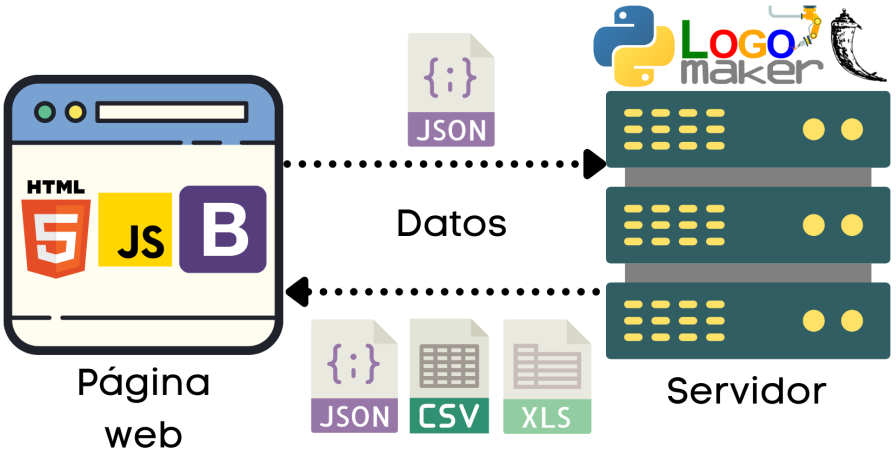


FIGURA 5.3: Interacción de las herramientas para la creación de la plataforma computacional

Una vez que se eligieron las herramientas para la creación de la plataforma computacional, y en que partes del sistema se van a aplicar, se procede a la ejecución de los sprint, comenzando con el Sprint 1: entrada de secuencias de ADN.

5.2.2. Sprint 1: entrada de secuencias de ADN

En este primer sprint se identificaron los medios de entrada para que el usuario pueda analizar las secuencias de ADN que desee. Estos medios son:

- Ingreso manual: Se ingresa la cadena de caracteres que conforman la secuencia de ADN en un campo dentro de la interfaz. Un ejemplo es la siguiente cadena de caracteres:

  TCGAAGCCTGCCCAACAGAATGACCCGCGAACATGTTCATCGA

- Subir archivo FASTA: Este tipo de archivos tiene una estructura simple, conteniendo a la(s) secuencia(s) de ADN a analizar junto con una cabecera que brinda la información puntual del conjunto. Un ejemplo de la estructura de este tipo de archivos se presenta en la Figura 5.4, donde el primer renglón contiene información puntual de los datos biológicos, seguido de la secuencia de ADN, representada por una cadena de caracteres.

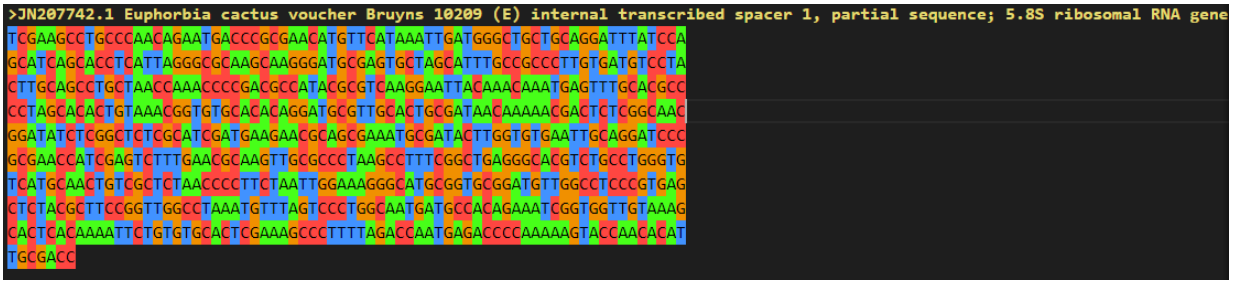


FIGURA 5.4: Archivo FASTA

Una vez conocidas las formas de entrada, también se determinaron los siguientes parámetros y el orden de ingreso que se enumeran a continuación:

1. **Tipo de entrada de datos**: Es la opción que le permite al usuario elegir si ingresa las secuencias de ADN de forma manual o mediante un archivo FASTA.

2. **Selección de algoritmo** : Esta le permite al usuario elegir qué algoritmos de minería de patrones frecuentes desea utilizar.

3. **Umbral**: Es el valor numérico ingresado por el usuario, utilizado para determinar si un elemento es un patrón frecuente.

4. **longitud_mínima**: El valor que determina el usuario para la mínima extensión para ser considerado un motif.

5. **tolerancia_delante**: Valor ingresado por el usuario y representa el número de posiciones no concretas (es decir, posiciones que no coinciden con un nucleótido en concreto dentro de ordenamiento de secuencias) que puede haber entre dos elementos recurrentes, por delante del patrón considerado.

6. **tolerancia_detrás**: valor que ingresa el usuario y es el número de posiciones no concretas que puede haber entre dos subsecuencias recurrentes por detrás del patrón considerado.

7. **imprimir_gráfica**:Es un valor binario en el que el usuario decide si desea imprimir las gráficas de los motifs resultantes (Logos de secuencia) o no.

Para el caso particular del parámetro **Tipo de entrada de datos**, se plantean las siguientes funciones para que se adapte al tipo de entrada de datos que se elija: se despliega un campo para ingresar la secuencia, un botón para agregar a la cadena de ADN y un área de texto en donde pueda visualizar las secuencia ingresadas en caso de elegir la opción de *"Ingreso manual"*. Esta funcionalidad descrita se presenta en la Figura 5.5.

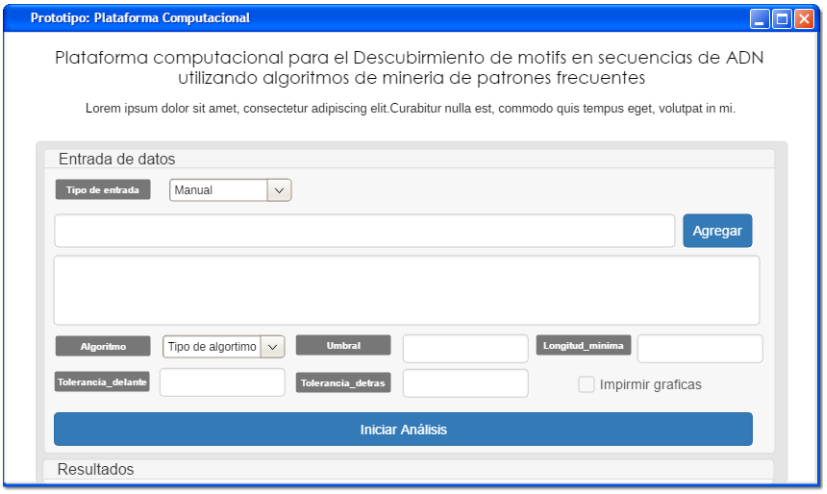


FIGURA 5.5: Prototipo de la opción **Tipo de entrada de datos** con el valor "Ingreso manual"

Para la opción "Archivos FASTA" es necesario un botón para cargar el archivo tal como se muestra en la Figura 5.6.

FIGURA 5.6: Prototipo de la opción **Tipo de entrada de datos** con el valor "Archivos FASTA"

Una vez establecidos los mecanismos y el diseño que llevará la interfaz, mediante el uso de tecnologías Web (HTML, CSS y JavaScript) se construyó la interfaz gráfica a partir de los prototipos de las Figuras 5.5 y 5.6, junto con los mecanismos descritos con anterioridad, los cuales le permiten al usuario ingresar los valores de los parámetros anteriormente mencionadas. El resultado es la interfaz gráfica web que se muestra en la Figura 5.7, en donde se muestra la implementación del aspecto de "Ingreso manual".

También la Figura 5.8 presenta la respectiva implementación en la plataforma para la opción de "archivos FASTA".

Lo siguiente fue diseñar y programar los mecanismos de verificación de los datos biológicos que se realizaran en el próximo sprint.

FIGURA 5.7: Opción de "Entrada manual" en la Plataforma Computacional

FIGURA 5.8: Opción de "Archivo FASTA" en la Plataforma Computacional

5.2.3. Sprint 2: pre-procesamiento de datos

Para evitar errores durante el análisis de las secuencias de ADN, se tiene que asegurar que estas sean correctas, es decir, que las cadenas de caracteres que representan a las secuencias solo estén formadas por las letras de los nucleótidos base: ACGT. Además, se tiene que ordenar la información de entrada en una estructura de datos adecuada para su análisis.

Con base en lo mencionado, se diseñaron los mecanismos de preprocesado en los que se asegura:

- Las cadenas de caracteres solo tengan las letras A, C, G yT. En caso contrario, se le notifica al usuario que modifiqué su entrada de datos.
- Para la forma manual de entrada de datos y archivos FASTA, el formato de las secuencias tiene que ser en mayúsculas.
- En el caso de los archivos FASTA, que tengan el formato sea el adecuado para su correcta lectura.
- Ingresar las secuencias de ADN previamente verificadas a la estructura de datos de tipo *lista* para el proceso de análisis con algoritmos de minería de patrones frecuentes

Estos mecanismos fueron implementados a través de JavaScript, añadiéndose como parte de las funcionalidades de la interfaz de entrada de datos, con el objetivo

de que se le notifique al usuario algún error que se descubra antes que se envíe la información de entrada al servidor, a su vez ordenar los datos de entrada verificados en una estructura adécuela. Ejemplo de la aplicación de estos mecanismos es la Figura 5.9, en la que se muestra al usuario una ventana emergente indicándole que fue descubierto un error en la entrada de datos.

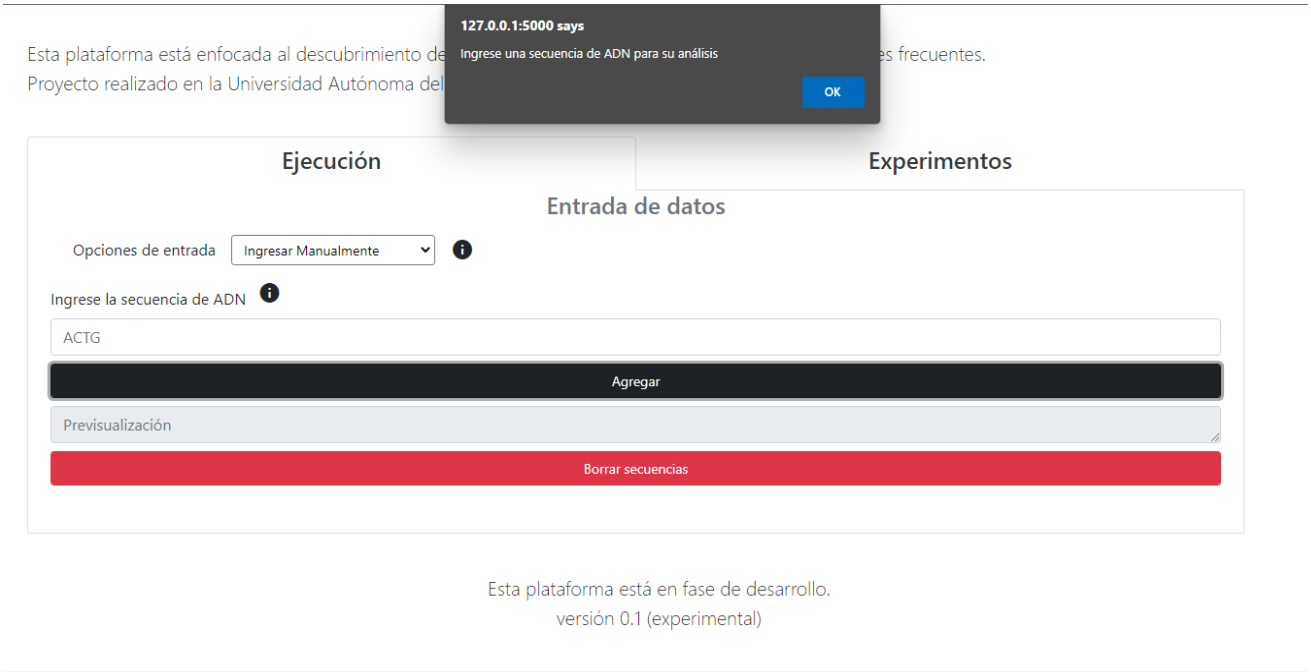


FIGURA 5.9: Verificador de contenido de secuencias activo en la plataforma

Con estas validaciones y el proceso de organización de datos ya creados, se procede a la ejecución del tercer sprint que consistirá en identificar y poner en marcha los algoritmos de minería de patrones frecuentes aptos para esta tarea.

5.2.4. Sprint 3: análisis de datos

Se buscaron diferentes métodos bajo el idea de analizar grandes volúmenes de datos para encontrar elementos recurrentes, siendo los algoritmos de minería de patrones frecuentes ideales para esta tarea. Para la construcción de este proyecto se requirió que los procesos de los algoritmos generen candidatos que se encuentren

dentro del grupo de secuencias de ADN y arrojen como resultado subsecuencias frecuentes relevantes que se convertían en motifs, junto con las posiciones exactas dentro del conjunto. GSP, Basado en índices y Basado en índices en múltiples secuencias son aquellos algoritmos que cumplen con la pauta antes mencionadas añadiendo los procesos de conteo eficiente, siendo estos los motivos para ser implementadas dentro de la plataforma. Junto con los algoritmos, se implemento un método de obtención de motifs basándose en la unión de elementos frecuentes cercanos con posiciones concretas cercanas a un patrón frecuente (véase la sección 4).

Para la codificación de los algoritmos y EOM se utilizo Python, junto con el framework flask para la elaboración de un servidor web para comunicar la pagina con los métodos a través del envió los datos de entrada y desplegar los resultados. Para ello, es necesario organizar los patrones obtenidos en un formato que sea compresible tanto para el usuario como para la plataforma, por lo que los archivos JSON tienen el formato apto para este propósito. Para complementar la información obtenida, a partir de los JSON que se forman, se añadieron dos formatos mas: CSV y XLS; En el caso de CSV almacenara la información resultante de los patrones frecuentes, y XLS de los motifs con su respectivo gráfico de secuencias LOGO.

Dentro del sprint cuatro se analizaran y crearan los mecanismos adecuados para la estructuración de resultados en dicho formato.

5.2.5. Sprint 4: Estructuración de los resultados

Se examinaron los valores resultantes del análisis de las secuencias de ADN junto con los parámetros de entrada de la ejecución, dividendo en dos principales conjuntos de resultados: los patrones frecuentes y motifs. Primero se presentaran los patrones frecuentes, organizando la información en dos etiquetas principales: *"Configuración" y "Patrones"*, a su vez identificando sus respectivos atributos, quedando de la siguiente manera,

- **Configuración**: Contiene los valores con los que se realizo la ejecución de la plataforma.
    - *Algoritmo*: Es nombre del algoritmo que se empleo.
    - *Min_sup*: El umbral para determinar a las subsecuencias como patrones frecuentes.
    - *Entrada*: El tipo de entrada, ya sea manual o mediante un archivo FASTA.

- *Sequencias_ananlizadas*: El número de secuencias analizadas.
- *Patrones_hallados*: El número de patrones frecuentes hallados en el grupo de secuencias.

- **Patrones**: que contiene la información a los patrones frecuentes descubiertos dentro de las secuencias de ADN.
  - *Patrón*: Es el elemento recurrente hallado dentro de las secuencias.
  - *Longitud*: Es la extensión del patrón.
  - *Ocurrencia*: El número de ocurrencias del patrón.
  - *Posiciones*: Representa cada ubicación en especifico del patrón dentro del grupo de secuencias de ADN, teniendo los atributos:
    - *secuencia*: el identificador de la secuencia de ADN del grupo.
    - *posición*: la posición dentro de la secuencia.

Con los atributos definidos, se programaron los mecanismos con el lenguaje Python que permitan guardar esta información en un archivo JSON y CVS dentro del servidor. La imagen 5.10 muestra un JSON archivo creado, con las características antes mencionadas.

```
{
  "Configuracion":
  {"Algoritmo": "Basado en indices en multiples secuencias",
   "Min_sup": 2,
   "Entrada": "Archivo-test_sequence.fasta",
   "Secuencias_ananlizadas": 3,
   "Patrones_hallados": 17},

  "Patrones":
     [{"Patron": "AA", "Longitud": 2, "Ocurrencias": 3, "Posiciones":
        [{"sequencia": "000000", "posicion": 7},
         {"sequencia": "000000", "posicion": 8},
         {"sequencia": "000000", "posicion": 9},
         {"sequencia": "000001", "posicion": 3},
         {"sequencia": "000002", "posicion": 7},
         {"sequencia": "000002", "posicion": 14}]},
      {"Patron": "AC", "Longitud": 2, "Ocurrencias": 2, "Posiciones":
        [{"sequencia": "000000", "posicion": 1},
         {"sequencia": "000000", "posicion": 10},
         {"sequencia": "000001", "posicion": 16}]},
      {"Patron": "CC", "Longitud": 2, "Ocurrencias": 2, "Posiciones":
        [{"sequencia": "000001", "posicion": 7},
         {"sequencia": "000001", "posicion": 13},
         {"sequencia": "000002", "posicion": 5}]},
      {"Patron": "CG", "Longitud": 2, "Ocurrencias": 3, "Posiciones":
        [{"sequencia": "000000", "posicion": 2},
         {"sequencia": "000001", "posicion": 8},
         {"sequencia": "000001", "posicion": 14},
         {"sequencia": "000002", "posicion": 10},
         {"sequencia": "000002", "posicion": 17}]},
      {"Patron": "CT", "Longitud": 2, "Ocurrencias": 3, "Posiciones":
        [{"sequencia": "000000", "posicion": 11},
         {"sequencia": "000000", "posicion": 13},
         {"sequencia": "000001", "posicion": 1},
```


FIGURA 5.10: Archivo JSON resultante con patrones frecunetes

Por otro lado los motifs, solo contienen las subsecuencias y su información organizada en un archivo JSON diferente al anterior, y además de un archivo XLS, teniendo la siguiente estructura:

- **Alineaciones:** El conjunto de datos de los motifs hallados.
    - **alineamientos:** el grupo de subsecuencias que conforman al motifs, además incluye la secuencia y posición de origen.
    - **Expresión regular:** es el patrón de búsqueda del motifs.
    - **Longitud**: La extensión del motif.
    - **Matriz de conteo:** La es matriz que contiene la frecuencia de cada nucleótido en cada posición
    - **Motif:** La palabra de representa al motif.
    - **Ocurrencia:** El número de apariciones del motif.
    - **Patrón:** El patrón frecuente de origen del motif.
    - **Número de alineaciones:**

Con ambos archivos en los formatos correspondientes, el usuario podrá consultar los patrones frecuentes y los motifs que se obtuvieron del análisis de una o varias secuencias de ADN. Además estos ficheros permitirán un despliegue de resultados más organizado y eficiente. Para eso se necesitan programar las funciones esenciales que se desarrollaran en el siguiente sprint.

5.2.6. Sprint 5: despliegue de resultados

Se diseñaron los escenarios en los cuales se presentara la información obtenida durante una ejecución y en experimentos anteriores. El primero mostrara una tabla con la información de los resultados obtenidos en un análisis como son patrón o motif, la longitud, la ocurrencia y las posiciones en la que se encuentran los elementos. Esta información aplica para los motifs y los patrones frecuentes, tal como se expone en la Figura 5.11.

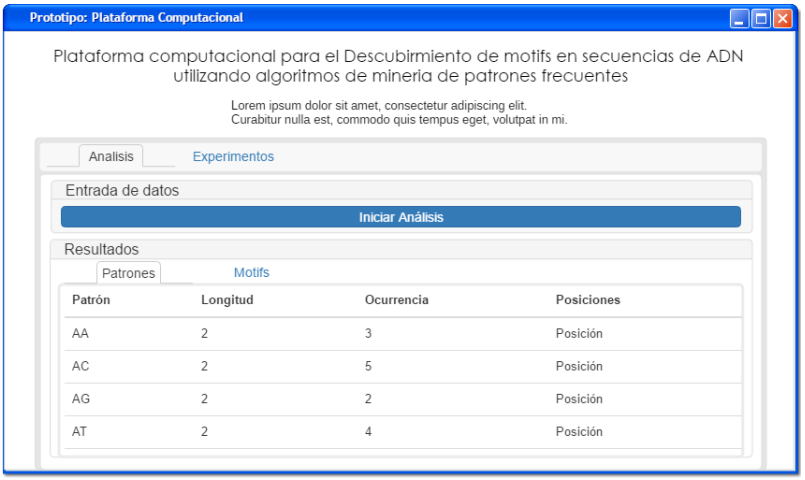


FIGURA 5.11: Prototipo de la visualización de los resultados obtenidos de una ejecución realizada

En la columna de posiciones tendrá un botón en el cual se mostrara las posiciones del patrón o motif de su fila correspondiente. Al momento de dar un click desplegara una ventana emergente que tendrán una tabla con las ubicaciones especificas(secuencia, posición) de cada elemento hallado, como lo muestra la Figura 5.12.

FIGURA 5.12: Prototipo de la visualización de las posiciones especificas de un elemento resultante (motif o patrón frecuente)

Para el caso de los experimentos realizados, se tendrá un apartado de consulta que incluye una tabla con la información especifica de cada experimento como el nombre del archivo de entrada (En el caso que de sea la entrada manual, este apartado lo etiquetará como texto), el número de las secuencias de ADN que se analizaron, el tipo de algoritmo que se empleo en el experimento, el valor de umbral utilizada, la longitud mínima empleada, la tolerancia entre elementos, y una columna de archivo en la que se podrá descargar los archivos resultantes de una ejecución en especificó. La Figura 5.13 muestra un prototipo con las características descritas.

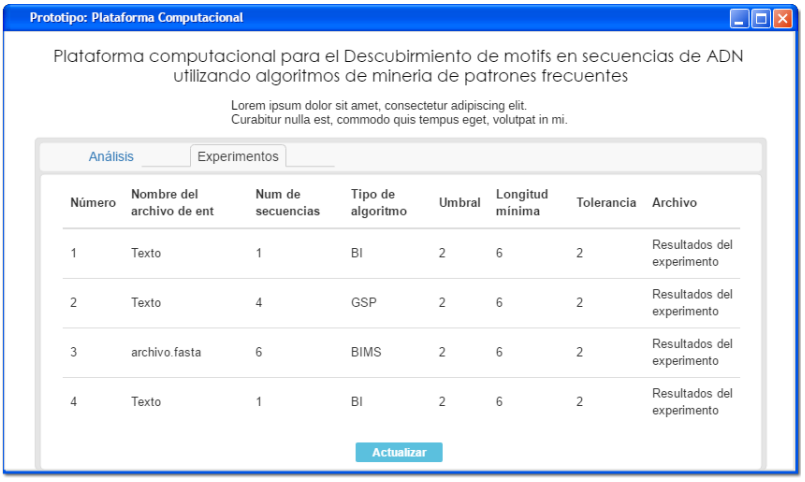
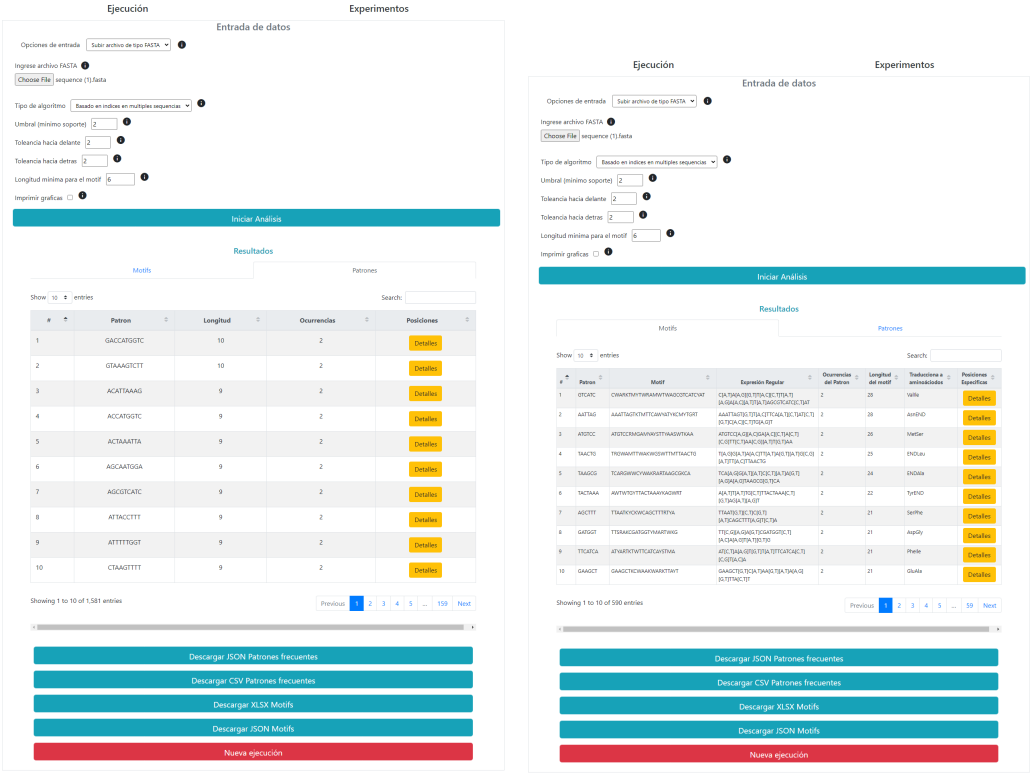


FIGURA 5.13: Prototipo de consulta de experimentos anteriores

Tomando en cuenta los prototipos anteriores y la información que pueden arrojar los experimentos, se construyeron los aparados gráficos junto con los respectivos mecanismos de visualización de datos. Cabe mencionar se realizaron algunos cambios en la presentación de los resultados como adición de los botones de descarga de cada archivo en especifico y un botón para iniciar una nueva ejecución al final de la tabla de resultados dentro de la sección de análisis, como lo muestra la Figura 5.14.



((A)) Tabla de los patrones frecuentes resultantes

((B)) Tabla de los motifs resultantes

FIGURA 5.14: Resultados desplegados dentro de la plataforma

En el caso de la tablas de las Figuras 5.14(a) y 5.14(b) se capturan los datos en formato JSON provenientes del servidor, y mediante JavaScript, Bootstraap y HTML se despliegan los resultados que se obtuvieron en la ejecución. La Figura 5.14 expone un despliegue de resultados en dos subsecciones: patrones frecuentes y motifs. Para la tabla de patrones de la Figura 5.14(a) esta formada por las cabeceras *Patrón, Longitud, Ocurrencia y Posiciones*; En el caso de la tabla de los motifs de la Figura 5.14(b) esta confirmada por los aspectos de *Patrón frecuente de origen, Motif, Expresión Regular, Ocurrencias del Patrón, Longitud del motif, Traducción a aminoácidos y Posiciones Especificas*. Las dos ultimas columnas de ambas tablas contienen un botón que muestra las posiciones en especifico del patrón o motif dento del grupo de secuencias ADN. Esta acción que se presenta en la Figura 5.15.

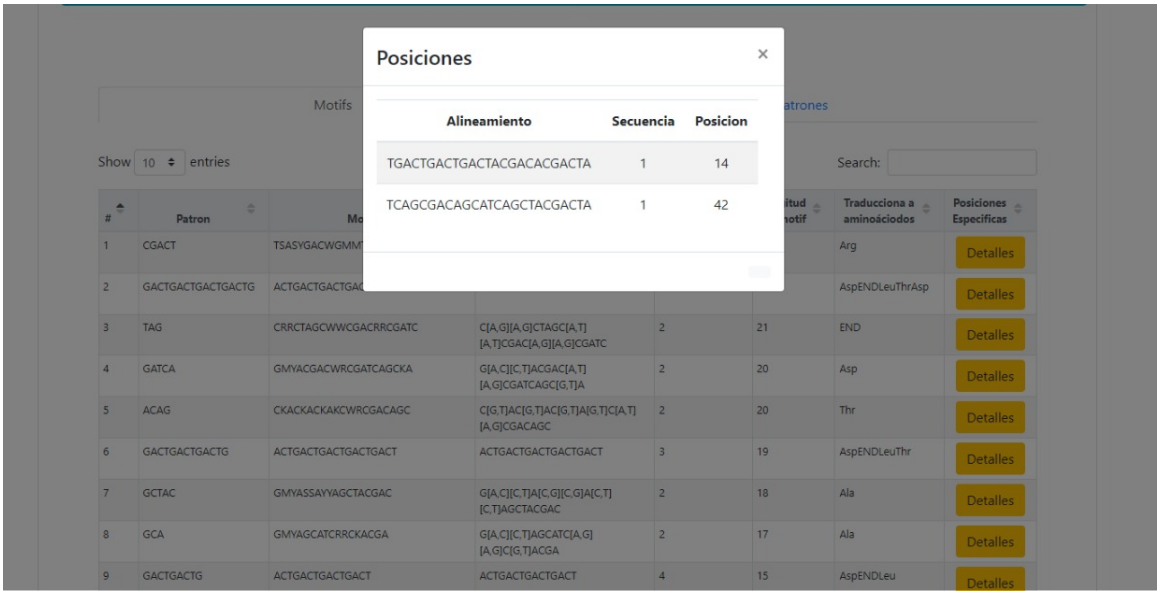


FIGURA 5.15: Despliegue de las posiciones exactas dentro de las secuencias de ADN de un patrón en especifico

Por otro lado, el aparado de *"experimentos"* contiene los archivos que se generaron en las ejecuciones pasadas (JSON, CSV, XLS) organizados en una tabla con la finalidad de consultarlos, tal como se puede ver en al Figura 5.16.

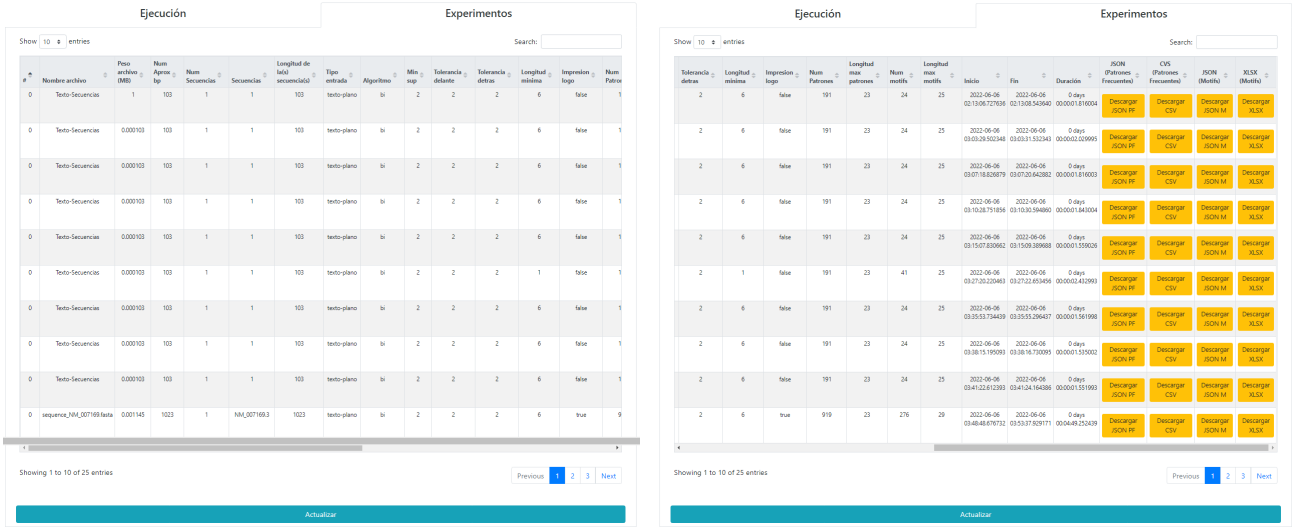


FIGURA 5.16: Apartado de experimentos en la plataforma.

La tabla de la Figura 5.16 tiene 24 columnas, de las cuales tienen diferentes condiciones dependiendo de las características de cada experimento realizado. Estas columnas son:

- Nombre archivo: Es el nombre del archivo fasta de entrada. En el caso del modo de entrada fuese “manual” tendrá la etiqueta “Texto-Secuencias”.
- Peso archivo (MB): Es el peso de archivo en megabytes. En caso de se sea una “entrada manual” se obtendrá el peso en megabytes de las secuencias de ADN ingresadas.
- Num Aprox bp: Es la cantidad aproximada de nucleótidos de todas las secuencias de ADN ingresadas.
- Num Secuencias: Es la cantidad de la secuencias de ADN ingresadas.
- Secuencias: Son los códigos de las secuencias de ADN obtenidas de un archivo FASTA. En el caso de la “entrada manual” muestra la enumeración de las secuencias de ADN. En el ambos casos están ordenadas conforme fueron ingresadas.
- Longitud de la(s) secuencia(s): Es una lista que contiene la longitud de todas las secuencias de ADN. De igual forma, esta lista esta ordenada conforme fueron ingresadas.
- Tipo entrada: Es el modo de entreda utilizado en el experimento (“Entrada manual” o “Subir archivo de tipo FASTA”).
- Algoritmo: Es el método utilizado para analizar las secuencias de ADN.
- Min sup: El valor de umbral o mínimo soporte empleado.
- Tolerancia delante: Es el valor de tolerancia hacia delante empleado.
- Tolerancia detrás: Es el valor de tolerancia hacia detrás empleado.
- Longitud mínima: Es el valor de longitud mínima utilizado para comprar la extensión de los motifs.
- Impresión logo: Es el valor binario que indica si se generaron los gráficos LOGO de las secuencias
- Num Patrones: Es la cantidad de patrones frecuentes obtenidos.

- Longitud max patrones: Es la longitud máxima hallada dentro de los patrones frecuentes.
- Num motifs: Es la cantidad de motifs hallados en el experimento.
- Longitud max motifs: Es la longitud máxima hallada dentro de los motifs.
- Inicio: Indica el inicio exacto del experimento en el formato: año-mes-día Hora:minuto:segundos.milisegundos.
- Fin: Es la finalización del experimento en el formato: año-mes-día Hora:minuto:segundos.milisegundos.
- Duración: Es el lapso del experimento en el formato: número de días Hora:minuto:segundos.milisegundos.
- JSON (Patrones Frecuentes): Es el botón para descargar el archivo JSON que contiene los patrones frecuentes.
- CVS (Patrones Frecuentes): Es el botón para descargar el archivo CSV con la información de los patrones frecuentes hallados.
- JSON (Motifs): Es el botón para descargar el archivo JSON que contiene los motifs.
- XLSX (Motifs): Es el botón para descargar el archivo XLS con la información de los motifs, y puede contiene los gráficos LOGO de secuencias.

Además cuenta con un botón de actualizar con la finalidad de renovar el registró con nuevos experimentos. Con esta ultima sección relacionada con el despliegue de los resultados, se finaliza el desarrollo de los cinco módulos que conforman a la plataforma, dando como resultado un software de tipo web que implementa algoritmos de minería de patrones frecuentes y mecanismos para la obtención de motifs en secuencias de ADN en formatos adecuados para su interpretación. A continuación se detallan las características de la plataforma.

5.3. Producto Resultante

El trabajo llevado a cabo dentro de los sprints en la metodología, presenta una plataforma que permite analizar una o varias secuencias de ADN con algoritmos de minería de patrones frecuentes tales como BI, BIMS o GSP. Así mismo, se añadió proceso que permite unir elementos recurrentes llamado EOM con el objetivo de crear motifs con mayor información a través de la configuración que se desee en

los parámetros de entrada. Además, la plataforma tiene capacidad de desplegar los resultados extraídos del análisis de forma organizada y descargarlos en diferentes formatos tales como JSON (para patrones frecuentes y motifs), CSV (para patrones frecuentes), XLS (para motifs). Junto con estas características, se tiene la posibilidad de consultar experimentos anteriores en los formatos antes mencionados. Asimismo, la implantación de tecnologías web de la plataforma permite que pueda ser utilizado sin la necesidad de emplear herramientas extra.

5.4. Comparativa de software

Al igual que la plataforma que se muestra a lo largo del capitulo, existe más software que está enfocado a la tarea de hallazgo de motifs en secuencias de ADN empleando distintas operaciones, ya sean probabilistas o de enumeración, los cuales pueden entregar datos útiles al usuario. Cada software tiene distintos parámetros para configurar, algoritmos implementados, formas de entrada de datos biológicos, así como distintas presentaciones de resultados, e incluso están implementados en distintos entornos como el web o stand alone. La Tabla 5.2 muestra la comparativa entre diferentes plataformas orientadas a la tarea de identificación de motifs en secuencias de ADN.

Plataformas Computacionales						
Nombre de la Plataforma	**Tipo de software**	**Algoritmo(s)**	**Tipo de algoritmo de generación de motifs**	**Entrada de datos**	**Salida de datos**	**Extras**
MEME suite	Web	MEME y DREME	Probabilístico y Conteo de palabras	Archivos FASTA	Un archivo HTML, un archivo de texto plano, un archivo XML	Enviar los resultados vía email
DAMBE	Escritorio	Gibbs Sample	Probabilístico	Archivos FASTA y GenBank, Conexión con base de datos biológicos	Resultados visibles en la interfaz, con opción de guardar en texto	multiplataforma y generación de un solo motif
YMF software	Web	YMF	Conteo de palabras	Archivos Fasta	Muestra en pantalla a los motifs con mayor puntaje	Envia los resultados vía email. Actualmente no está disponible al público
Plataforma computacional desarrollada en este proyecto	Web	Técnica de crecimiento propuesta	Conteo de palabras y basado en patrones frecuentes	Archivos FASTA y manual	Archivos JSON, CSV, XLSX y despliegue de resultados en la interfaz	Emplea algoritmos de minería de datos, multiplataforma

TABLA 5.2: Comparación entre plataformas enfocadas al descubrimiento de motifs en secuencias de ADN

Con la información presentada en la Tabla 5.2 se realizó un análisis, que va aspecto por aspecto, para conocer las peculiaridades en las que se destaca cada plataforma. Dentro del tipo de software, la mayoría se ejecutan en entornos web menos DAMBE que es de escritorio [41]; la plataforma que se desarrolla en este proyecto emplea una nueva estrategia para la búsqueda de motifs en secuencias de ADN, por su parte MEME suite ofrece dos formas de encontrar motifs, y el resto solo una; A su vez se emplean algoritmos de tipo probabilístico en MEME suite [2] y DAMBE [42], y los de conteo de palabras se utilizan en la plataforma que se desarrolla, en YMF software y también en MEME suite, otra diferencia importante entre la plataforma desarrollada es que se basa en Patrones Frecuentes para iniciar la búsqueda de los motifs; Por otro lado, todas las plataformas admiten de entrada de datos archivos FASTA, a su vez DAMBE incluye archivos Genbank y la conexión a bases de datos biológicas, y la plataforma de este proyecto permite una entrada manual de las secuencias de ADN; Cada una de las plataformas presenta los resultados de maneras diferentes:

- YMF solo presenta los resultados que obtuvo en pantalla [34].
- La plataforma de este proyecto presenta los resultados que obtuvo en pantalla, además de archivos dos archivos JSON (uno para patrones frecuentes y otro para motifs), un CSV para patrones frecuentes y un XLSX para motifs. Siendo este software el que más archivos resultantes le presenta al usuario.
- DAMBE por su parte, divide sus resultados en tres: los motifs hallados, la matriz peso-posición y la verificación de los resultados [41].
- MEME suite genera 3 archivos: un archivo HTML, un archivo de texto plano, un XML [2].

Por último, cada software tiene características adicionales como MEME suite e YMF software que envían los resultados por correo electrónico , o DAMBE y el propuesto en este trabajo que son programas multiplataforma, también la plataforma que se desarrolla en este proyecto presenta la particularidad de emplear algoritmos de minería de patrones frecuentes. [2][34][41]

En comparación con los otros trabajos, la plataforma que se está creando en este proyecto, presenta características similares como el tipo de software, los tipos de algoritmos que se emplean, o en el tipo de entrada de datos. A su vez tiene diferencias como las maneras que presentan los resultados, o la mas notoria, la utilización de algoritmos de minería de patrones frecuentes para emplear sus resultados en la identificación de motifs. Esta comparativa se realiza con plataformas que son de uso libre y de fácil acceso.

Capítulo 6

Conclusiones y Trabajos Futuros

El estudio de los seres vivos ha evolucionado de modo tal que puede llevarse a cabo con técnicas diferentes al estudio directo de los organismos, gracias a la información recopilada a lo largo de los años. En esta información se puede identificar la evolución de las especies, descubriendo funciones desconocidas y la formulación de nuevo conocimiento biológico. Dicha recopilación de información seguirá creciendo de manera importante. De ahí la relevancia del presente trabajo, el cual propone una herramienta que sea de utilidad para los estudiosos del tema en cuestión.

Con base en los objetivos planteados, no solo se conocieron sino que se interpretaron y utilizaron los conceptos de Motifs en secuencias de ADN para lograr implementarlos en una plataforma computacional. Además, se alcanzó la propuesta de un nuevo algoritmo de descubrimiento de motifs, llamado EOM, el cual toma como entrada resultados de algoritmos de identificación de patrones frecuentes para una secuencia o para grupos de secuencias de ADN.

La idea principal del algoritmo propuesto es identificar motifs a través de patrones frecuentes obtenidos de las secuencias de ADN. Para esto se introdujo una mejora de un algoritmo de descubrimiento de patrones frecuentes, que supera al algoritmo original en dos aspectos principales: (i) las secuencias de ADN se representan en un mapa de posiciones, el cual se recorre en una forma más eficiente, (ii) el algoritmo de búsqueda de candidatos a patrones frecuentes contiguos es más eficaz. Este algoritmo se le dio el nombre de BIMS.

El nuevo algoritmo de identificación de Motifs toma como base esos patrones frecuentes contiguos para tratar de identificar de cada uno de ellos un motif, enriqueciendo la información que puede proporcionar la concatenación de diversos patrones frecuentes, considerando ciertas diferencias de nucleótidos entre ellos, ya que favorece la longitud que pudiera tener esta subsecuencia resultante. Finalmente, este tipo de motifs podrá proporcionar información más interesante para los biólogos y estudiosos del ADN.

Dichos algoritmos propuestos fueron medidos contra algoritmos de la literatura y se demostró su eficiencia y eficacia de forma numérica, encontrando resultados prometedores. Por último, estos algoritmos propuestos, junto con interfaces de entrada y salida, fueron integrados para implementar una plataforma computacional que permite la identificación de Motifs en una o en un grupo de secuencias de ADN, la cual puede ser utilizada por investigadores afines a este tipo de información.

La plataforma resultante muestra una serie de interfaces amigables basada en términos y conceptos que los biólogos conocen y están familiarizados. Dicho desarrollo permite ingresar secuencias de ADN en un formato estándar de cualquier base de datos biológica y cuyo resultado lo despliega en un formato que puede ser entendible por los expertos del área y que muestra información potencialmente interesante, que el experto podrá evaluar si le es de utilidad en sus estudios correspondientes.

6.1. Aportación del trabajo de investigación

La aportación de este proyecto es el análisis e implementación de algoritmos de minería de patrones frecuentes para el procesamiento de información biológica. Estas acciones permitieron el desarrollo de una plataforma computacional apropiada para la búsqueda y hallazgo de motifs en secuencias de ADN. Adicionalmente, se identificaron qué algoritmos de minería de patrones son aptos para dicha tarea.

Por otra parte, este trabajo amplia las aplicaciones de los métodos de minería de patrones frecuentes en distintos escenarios que presentan inconvenientes en la exploración de elementos recurrentes en grandes volúmenes de información.

6.2. Trabajos Futuros

Los resultados del presente trabajo muestran un avance en la intención de proporcionar herramientas que sean de utilidad a los expertos en el área relacionada a la biología, y que hagan uso de las secuencias de ADN como su fuente de datos. En consecuencia, después de obtener los resultados mostrados, surgen algunos trabajos futuros entre los que destacan:

- Implementar la carga de datos biológicos con diferentes formatos, además del FASTA.
- Implementar métodos de búsqueda haciendo uso de algoritmos paralelos.

- Desarrollar nuevos algoritmos para la identificación de Motifs en secuencias de ADN.
- Validar los resultados encontrados en esta investigación por un experto en el área de interés.
- Realizar diversos análisis con casos de estudio específicos.

Bibliografía

[1] CHARU C. AGGARWAL and Jiawei Han. *FREQUENT PATTERN MINING*. SPRINGER INTERNATIONAL PU, 2014.

[2] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble. Meme suite: tools for motif discovery and searching. *Nucleic Acids Research*, 37(Web Server):202–208, May 2009.

[3] Timothy L. Bailey. Discovering novel sequence motifs with meme. *Current Protocols in Bioinformatics*, 00(1), Nov 2002.

[4] Timothy L. Bailey. *Discovering Sequence Motifs*, page 231–251. Humana Press, 2008.

[5] Timothy L. Bailey. Dreme: motif discovery in transcription factor chip-seq data. *Bioinformatics*, 27(12):1653–1659, May 2011.

[6] Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. Genbank. *Nucleic Acids Research*, 41(D1):D36–D42, Nov 2012.

[7] Modan K Das and Ho-Kwok Dai. A survey of dna motif finding algorithms. *BMC Bioinformatics*, 8(S7), Nov 2007.

[8] Patrik D'haeseleer. What are dna sequence motifs? *Nature Biotechnology*, 24(4):423–425, 2006.

[9] Roberth Figueroa-Diaz, Camilo Sólis, and Armando Cabrera-Silva. Metodologías tradicionales vs. metodologÍas Ágiles. 02 2007.

[10] Olga Filipova and Rui Vilão. *Software Development From A to Z: A Deep Dive into all the Roles Involved in the Creation of Software*. Apress, Oct 2018.

[11] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Roger Nkambou, Bay Vo, and Vincent S. Tseng. *High-Utility Pattern Mining: Theory, Algorithms and Applications*. Springer, 2021.

[12] Luis Heriberto García Islas. *Modelo de análisis de datos para la identificación de patrones en secuencias de ADN por medio del desarrollo de técnicas de minería de datos*. UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO, 2019.

[13] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining*. Morgan Kaufmann, 2012.

[14] Catherine A Hardy and Susan P. Williams. *Managing Information Risks and Protecting Information Assets in a Web 2.0 era*, pages 959–961. 23rd Bled eConference eTrust, 2010.

[15] Rashida Hasan and Jainal Uddin. Data mining techniques for informative motif discovery. *International Journal of Computer Applications*, 88(12):21–24, Feb 2014.

[16] Rashida Hasan and Jainal Uddin. Data mining techniques for informative motif discovery. *International Journal of Computer Applications*, 88(12):21–24, Feb 2014.

[17] Fatma A. Hashim, Mai S. Mabrouk, and Walid Al Atabany. Review of different sequence motif finding algorithms. *Avicenna Journal of Medical Biotechnology*, 11(2):130–148, 2019.

[18] David A. Hendrix. *Sequence Motifs*. 2019.

[19] Rodolfo Hernández Gutiérrez. *Estructura molecular, función y métodos de estudio de las proteínas*. McGRAW-HILL, 2009.

[20] Ángel Herráez. *BIOLOGÍA MOLECULAR E INGENIERÍA GENÉTICA*. Elsevier Health Sciences, 2012.

[21] Luis Joyanes Aguilar and Ignacio Zahonero Martínez. *Programación en C/C++, Java y UML*. Mc Graw Hill Education, 2014.

[22] Mehmed Kantardzic. *DATA MINING Concepts, Models, Methods, and Algorithms*. John Wiley Sons, Inc, 2011.

[23] Hajime Kitakami, Tomoki Kanbara, Yasuma Mori, Susumu Kuroki, and Yukiko Yamazaki. *Modified PrefixSpan Method for Motif Discovery in Sequence Databases*, page 482–491. Springer Berlin Heidelberg, 2002.

[24] Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, Oct 1993.

[25] Patricio Letelier and Mª Penadés. Métodologías ágiles para el desarrollo de software: extreme programming (xp). *Técnica Administrativa*, 01 2006.

[26] Corrado Loglisci, Eliana Salvemini, Antonio Turi, Giorgio Grillo, Donato Malerba, and Domenica D'Elia. Mining frequent patterns of biological spaced motifs. pages 117–124, 01 2009.

[27] Guillermo Pantaleo and Ludmila Rinaudo. *Ingenieria de software*. Alfaomega Grupo Editor Argentino, 2015.

[28] Juan R. Riesgo. Qué es el genoma humano. *Ciencia*, 53(1):6–11, Jan-Mar 2002.

[29] Miguel Rocha and Pedro G Ferreira. *Bioinformatics Algorithms: Design and Implementation in Python*. ELSEVIER, 2018.

[30] Eduardo A. Rodríguez Tello. *Introducción a la Bioinformática*. CINVESTAV-Tamaulipas, 2021.

[31] Thomas D. Schneider and R.Michael Stephens. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research*, 18(20):6097–6100, 10 1990.

[32] Poonam Sharma and Gudla. Balakrishna. Prefixspan: Mining sequential patterns by prefix-projected pattern. *International Journal of Computer Science 38;amp; Engineering Survey*, 2(4):111–122, Nov 2011.

[33] S. Sinha. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 30(24):5549–5560, Dec 2002.

[34] S. Sinha. Ymf: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 31(13):3586–3588, Jul 2003.

[35] Ian Sommerville. *Software engineering*. Pearson Education, 10 edition, 2016.

[36] Cecie Starr, Christine A Evers, and Lisa Starr. *Biología: Conceptos y Aplicaciones*. Cengage Learning, 2011.

[37] Paul Stothard. Iupac codes, 2000.

[38] Benjamin Jean-Marie Tremblay. Introduction to sequence motifs. *Bioconductor*, Oct 2021.

[39] Sebastian Ventura and José María Luna. *Pattern mining with evolutionary algorithms*. SPRINGER, 2016.

[40] Xuhua Xia. Position weight matrix, gibbs sampler, and the associated significance tests in motif characterization and prediction. *Scientifica*, 2012(2022):1–15, 2012.

[41] Xuhua Xia. *The DAMBE Manual*. 2014.

[42] Xuhua Xia. *Gibbs sampler for de novo motif discovery*, page 99–111. Springer, Jul 2018.