DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 10 No. Especial 3 (2022) 22-29

Método de identificación de patrones con arreglos relacionales en secuencias de ADN
Pattern identification method using relational arrays in DNA sequences

V. I. Sobrevilla-Solís [a], A. Franco-Arcega [a, *], L. H. García-Islas [a], E. Rueda-Soriano [a], V. López-Morales [a]
J. Suárez-Cansino [a]

[a] *Área Académica de Computación y Electrónica, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*

Resumen

Las secuencias biológicas contienen información importante de los organismos vivos. El análisis de estas secuencias pueden proporcionar información que ayuda a los biólogos a un mejor entendimiento de estos organismos. El descubrimiento de patrones frecuentes en un grupo de secuencias de ADN se ha vuelto uno de los grandes retos en la aplicación de técnicas de minería de datos. Existe un considerable tiempo y esfuerzo empleado para obtener patrones frecuentes secuenciales cuando los métodos se basan en algoritmos Apriori, como GSP y KeySegment. Este trabajo propone el diseño de un método basado en mapeo de secuencias para aumentar la búsqueda de patrones frecuentes contiguos en un grupo de secuencias de ADN. El presente artículo muestra experimentos utilizando conjuntos de secuencias de ADN cuyas longitudes varían desde los 1000 hasta 5000 nucleótidos, obtenidas desde una base de datos biológica. Estos experimentos demostraron un algoritmo eficaz para la identificación de patrones frecuentes en secuencias de ADN comparado con otros algoritmos.

Palabras Clave: Minería de patrones secuenciales, patrones frecuentes contiguos, secuencias biológicas, secuencias de ADN, bioinformática

Abstract

Biological sequences contain a significant amount of genetic information from living organisms. The analysis of these sequences can provide information that might help biologists better understand them. The discovery of frequent patterns from a group of DNA sequences has become one of the greatest challenges in the application of data mining techniques. There is considerable time and effort expended in obtaining sequential frequent patterns when the methods utilized are based on Apriori algorithms such as GSP or Key-segment. This paper proposes a sequence mapping-based algorithm designed to augment the search for contiguous frequent patterns in a group of DNA sequences. The present paper presents experiments for several sets of DNA sequences whose whole length varies between 1000 and 5000 nucleotides obtained from a biological database. These experiments demonstrated a faster algorithm for frequent pattern mining on DNA sequences compared with other related algorithms

Keywords: Sequential pattern mining, frequent contiguous patterns, biological sequences, DNA sequences, bioinformatics

1. Introducción

El interés de emplear técnicas de Minería de Datos a información proveniente 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 (Han and Micheline, 2012). Para lograr esto, es indispensable tener los conocimientos necesarios para interpretar la información obtenida, requiriendo el apoyo de uno o varios especialistas en el tema. 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 (Rocha and Ferreira, 2018). Una de las tareas importantes dentro de la bioinformática es el análisis del ADN, debido a que es la

*Autor para la correspondencia: afranco@uaeh.edu.mx
Correo electrónico: afranco@uaeh.edu.mx (Anilu Franco-Arcega), so314246@uaeh.edu.mx (Víctor Ignacio Sobrevilla-Solís), luishg@uaeh.edu.mx (Luis Heriberto García-Islas), estebanrs@uaeh.edu.mx (Esteban Rueda-Soriano), virgilio@uaeh.edu.mx (Virgilio López-Morales), jsuarez@uaeh.edu.mx (Joel Suárez-Cansino).

Historial del manuscrito: recibido el 01/04/2022, última versión-revisada recibida el 29/05/2022, aceptado el 02/08/2022, publicado el 31/08/2022. **DOI:** https://doi.org/10.29057/icbi.v10iEspecial3.8928

plantilla molecular que contiene extensa información genética necesaria para la creación y funcionamiento de cualquier ser vivo (Starr et al., 2011). El análisis de secuencias de ADN, consiste en el descubrimiento de similitudes tanto funcionales como estructurales, así como las diferencias entre múltiples secuencias biológicas. Esto puede hacerse comparando las nuevas secuencias (desconocidas) con las bien estudiadas (conocidas) secuencias (Escobar, Murillo, y Soto, 2011). Las secuencias de ADN se almacenan en bases de datos biológicas, las cuales se pueden describir como una recopilación de información que ha sido obtenida a través de diversos medios, como pueden ser experimentos científicos, publicaciones y análisis computacional (Singh, 2006).

La base de datos más utilizada es la base de GenBank, dependiente del National Centre for Biotechnology Information (NCBI) - (Centro Nacional de Información Biotecnológica) perteneciente al National Institute of Health (NIH) (Instituto Nacional de Salud), la cual es parte del International Nucleotide Sequence Database Collaboration (INSDC) compuesto por el DNA DataBank of Japan (DDBJ), el European Molecular Biology Laboratory (EMBL) y el GenBank (Mizrachi,2016). 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 (D'Haeseleer, 2006).

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 (2014), 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 de las transacciones" (Aggarwal and Han, 2014). 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.

Para detectar un patrón dentro de un grupo de transacciones, se realizan conteos de aparición de cada elemento. De primera instancia, se define que un patrón se compone de sólo un elemento, y posteriormente se le van agregando más elementos para encontrar patrones más grandes. La aplicación de la MPF se puede extender a diferentes áreas del conocimiento, como mercadotecnia, medicina, química y biología, dado que sus datos pueden ser procesados por este tipo de métodos y pueden ser representados como patrones (Aggarwal and Han, 2014).

El hallazgo de patrones con métodos de MPF se realiza 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 (Srikant and Agrawal, 1996), PrefixSpan, FreeSpan (Han et al., 2000), AprioriAll (Agrawal and Srikant, 1996), 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.

En este trabajo, se presenta una estrategia que permite hallar patrones frecuentes dentro de un conjunto de secuencias de ADN, de una forma eficiente. Esta estrategia es aplicada a un conjunto de secuencias obtenidas de un importante repositorio de datos biológicos, GenBank (Mizrachi, 2016), con el fin de observar su uso y mejora sobre otras técnicas similares.

2. Método de Identificación de patrones basado en índices

Dentro del análisis de elementos biológicos, una de las aplicaciones más importantes es el descubrimiento de subsecuencias que resulten en patrones frecuentes dentro de secuencias de ADN (Bailey, 2008). A continuación, se presenta un método eficiente de búsqueda, obteniendo los patrones frecuentes de un conjunto de secuencias. Para poder describir este método, se presentan las siguientes definiciones.

Definición 1: Sea $\Sigma = A, C, G, T$ el conjunto conformado por cuatro letras que representan a los nucleótidos: Adenina (A), Citosina (C), Guanina (G), Tiamina (T). Siendo estos elementos los que conforman las secuencias de ADN.

Definición 2: Sea una secuencia de ADN $S = S_1, S_2, \dots S_n \; con \; S_i \; \epsilon \; \Sigma, (i = 1, \dots, l)$ 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.

Ejemplo 1: Sea $\Sigma = \{A, C, G, T\}$ y
S = <ACGTGTAAAACTCTTGTT >, S es una secuencia de 18 nucleótidos, por ende está constituido por 18 caracteres.

Definición 3: Sea $S = S_i \mid S_i \; \epsilon \; \Sigma$ una secuencia de ADN y $S_1 = S[j] \mid s[j] \; \epsilon \; S_i$ una subsecuencia de S. Así mismo, S es una súper secuencia de S_1.

Definición 4: Sea $S = s_i | s_i \; \epsilon \; \Sigma$ una secuencia de ADN y $s_1 = S[j] \mid s[j] \; \epsilon \; s_i$ 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.

Definición 5: Una secuencia candidata s_i será evaluada a partir de sus apariciones dentro de S, obteniéndolas de $f_{support}(s_i)$.
Definición 6: El parámetro umbral es un parámetro definido por el usuario utilizado para valorar la(s) aparición(es) de una subsecuencia s_i dentro de una secuencia *S*. Si una subsecuencia *si* tiene un valor de $f_{support}(S_i)$ igual o mayor al establecido en el umbral, será utilizado para generar nuevas subsecuencias candidatas en las siguientes iteraciones empleando la propiedad anti-monótona de soporte.

Ejemplo 2: Sea la secuencia
S =< CTAAGTCCGTAGCCGACT >, las subsecuencias
s_1 =< TAA > y s_2 =< CCG >, y un valor *umbral* = 2. Se calcula el $f_{support}(s_j)$ de cada subsecuencia obteniendo

$f_{support}(s_1)$=1 y $f_{support}(s_2)$=2, dejando solo a s_2 como una subsecuencia candidata que iguala o supera el umbral establecido.

Propiedad 1:
La propiedad anti-monótona de soporte considera dos condiciones:

1. Si una subsecuencia candidata es 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 se describe en la siguiente ecuación:

$$\forall X, Y \mid (X \subseteq Y) \Rightarrow f_{support}(X) \geq f_{support}(Y)$$

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: Sea la secuencia
S =<CTAAGTCCGTAGCCGACT >, las subsecuencias
s_1 =< TAA > y s_2 =< CCG >, y un valor umbral = 2. s_2 es una subsecuencia candidata frecuente, por ende todas sus subsecuencias (< CC > y < CG >) también son frecuentes. Por otro lado, s_1 es una subsecuencia candidata no frecuente, por lo tanto cualquier supersecuencia derivada de esta subsecuencia candidata (<ATAA>, <CTAA>, <GTAA>, <TTAA>, <TAAA>, <TAAC>, <TAAG>, <TAAT>, ...) no será frecuente.

Definición 7: Sea $S = s_i | s_i \epsilon \Sigma$ una secuencia de ADN y $C = \{s_1, s_2, s_3, \ldots, s_j \; \forall \; s_j \mid f_{support}(s_j) \geq umbral\}$ el conjunto de candidatos sobrevivientes. *C* será considerado el conjunto de patrones frecuentes contiguos.

Ejemplo 4: Sea S =< ACGTGTAAAACTCTTGTT > y el *umbral* = 2. C = {< A >, < C >, < T >, < G >, <AA>, < AC >, < CT >, < GT >< TG >, < TT >, <AAA>, < TGT>}.

Definición 8: sea κ = S_1, S_2, ..., S_n, un grupo de secuencias de ADN a analizar de *n* número de elementos del conjunto.

Ejemplo 5: Sea κ = {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 2. Los ejemplos 7 y 8 presentan estos casos.

Ejemplo 6: Sea κ = {<ACGTGTAAAACTCTTGTT>, <CTAAGTCCGTAGCCG> donde la *longitud*(S_1) = 18 y *longitud*(S_2) = 15.

Ejemplo 7: Sea κ = {<ACGTGTAAAACTCTTGTT>, <CTAAGTCCGTAGCCGACT>} donde la *longitud*(S_1) = 18 y *longitud*(S_2) = 18.

De acuerdo a la definición 3, se pueden encontrar subsecuencias s_i en una secuencia *S* de ADN. Para este caso, una subsecuencia s_i puede estar presente en *f* número de secuencias *S* pertenecientes al conjunto *κ*, donde *f* <= *n*. A su vez los *f* número de elementos del conjunto κ son supersecuencia del s_i.

Ejemplo 8: Sea *κ = {<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 aparece en la secuencia S_1; por último, s_3 sólo se presenta en la secuencia S_2.

Definición 9: Sea la función $f_{support}$ la cual calcula la cantidad de apariciones de una subsecuencia candidata dentro del conjunto de secuencias *κ*. Es decir, el número de secuencias del conjunto *κ* en las que existe dicha subsecuencia, sin importar las veces que se encuentre en una sola secuencia.

Definición 10: Sea la variable umbral, un valor es ingresado por el usuario, y empleado para determinar si una subsecuencia candidata es una subsecuencia sobreviviente o no, con base en las *f* apariciones dentro del conjunto *κ*. A su vez, *umbral* <= *n*.

Tomando la definición 7, una subsecuencia sobreviviente S_i es aquella subsecuencia candidata que $f_{support} \geq Umbral$. Con base en esto, se propone la siguiente definición para este método:

Definición 11: Sea la secuencia candidata s_i, aquella que cumple con la condición $f_{support} \geq Umbral \therefore f_{support} \leq n$. para formar parte del conjunto *C* y sea una subsecuencia sobreviviente.

Ejemplo 9: Sea *κ = {<ACGTGTAAAACTCTTGTT>, <CTAAGTCCGTAGCCGACT>}*, las subsecuencias s_1 =<*GTA*>, s_2 =<*AAC*> *y* s_3 =<*CCG*>, y el *umbral* = 2. Se calcula la frecuencia de cada subsecuencia candidata, dando los siguientes resultados: $f_{support(s_1)} = 2$, $f_{support(s_2)} = 1$, $f_{support(s_3)} = 1$ La subsecuencia candidata s_1 es la que iguala el umbral debido a que aparece en las secuencias S_1 y S_2 del conjunto κ. Por otro lado, la subsecuencia s_2 sólo aparece en S_1 y s_3 dos veces solo en S_2, ambas subsecuencias obteniendo un valor $f_{support(s_3)}$ igual a 1. Por lo tanto la subsecuencia candidata *s1* pasa a ser una subsecuencia sobreviviente, que será utilizada para generar nuevos candidatos, y las subsecuencias s_2 y s_3 son eliminadas.

2.1. Procedimiento

El procedimiento que sigue el método propuesto en este trabajo se basa en el flujo que se muestra en al Figura 1, donde se puede apreciar que existen tres etapas principales:

mapeo de las secuencias, generación de subsecuencias candidatas y evaluación de subsecuencias candidatas.

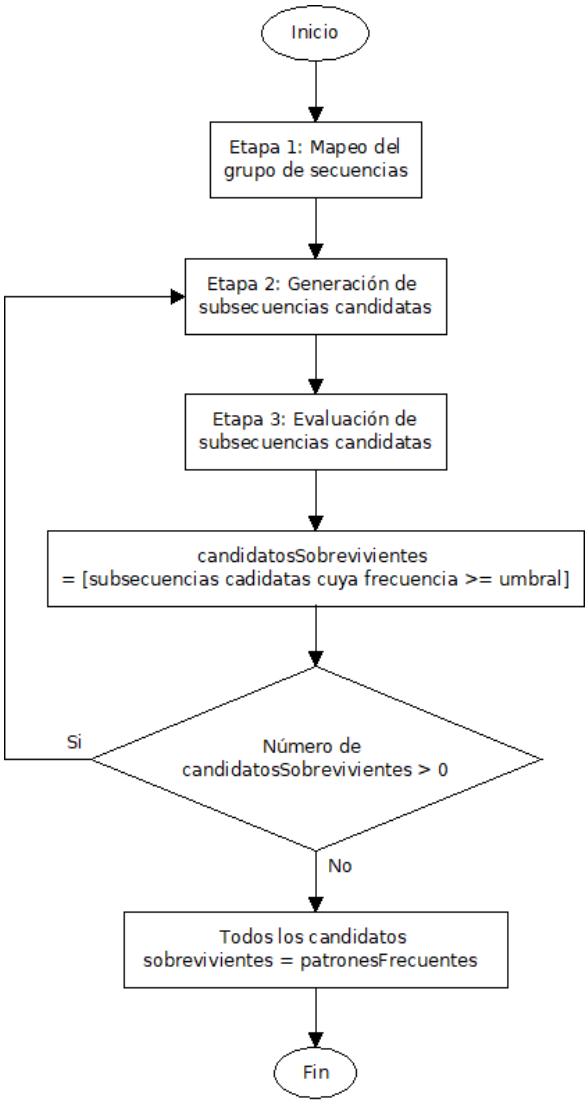

Figura 1: Flujo general de ejecución del método propuesto.

El primer paso es el 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 método; en seguida, se ejecutan procesos iterativos en los cuales se generan subsecuencias candidatas, para ser evaluadas con un umbral establecido por el usuario, que indica el número de secuencias de ADN en las que tiene que aparecer la subsecuencia aspirante a superviviente. Si y sólo si, el número de apariciones de una subsecuencia candidata es igual o mayor al umbral, pasa a ser una subsecuencia sobreviviente y servirá para generar más subsecuencias candidatas en posteriores iteraciones. Este método finaliza cuando ya no existen más subsecuencias candidatas que cumplan con el umbral, y las subsecuencias sobrevivientes pasan a ser patrones frecuentes del conjunto de secuencias de ADN.

2.1.1. Etapa 1: mapeo del grupo de secuencias

En esta etapa se obtienen todas las posiciones de los nucleótidos del conjunto de secuencias de ADN de entrada, formando un mapa con 4 columnas:

- (*elmtn*): contiene a los nucleótidos del ADN para distinguir las posiciones entre los nucleótidos.
- (*Seq*): la secuencia en la que se ubica.
- (*Pos*): la posición dentro de la secuencia.
- (*nextE*): el siguiente elemento de la posición

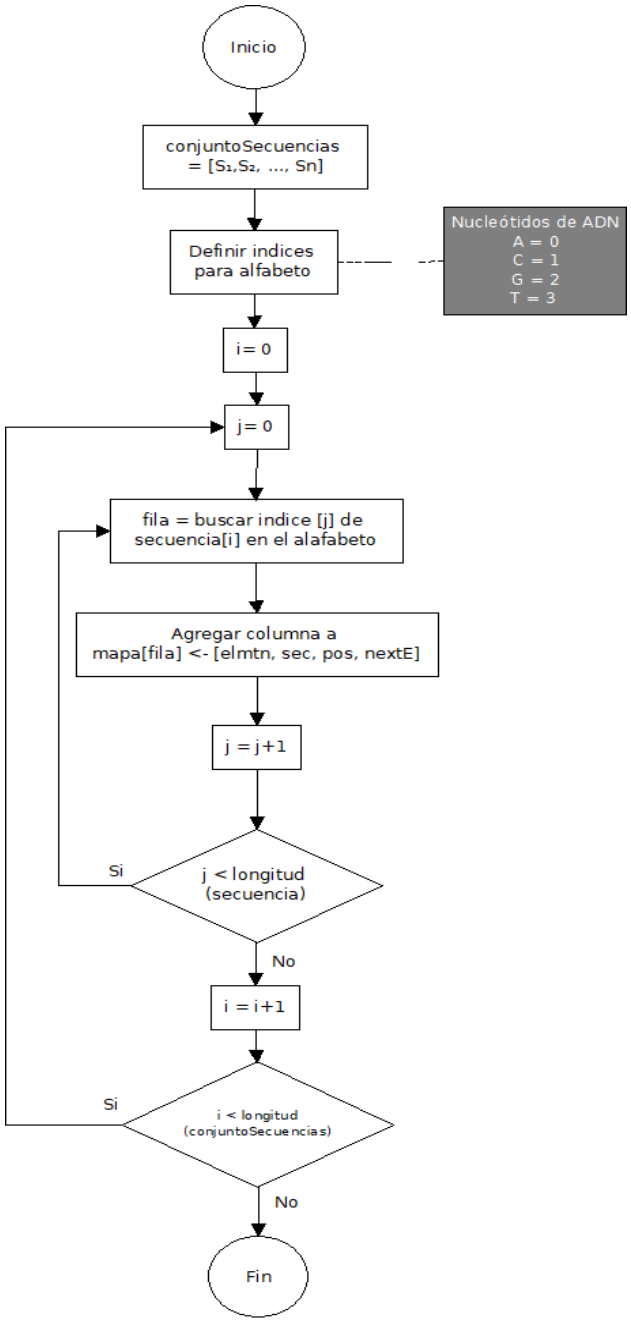

Figura 2: Generación del mapa de posiciones para un conjunto de secuencias

Por ejemplo, sea grupo de tres secuencias de ADN:
κ= { <*ACGTGTAAAACTCTTGTT*>,
<*CTAAGTCCGTAGCCGACT*>,
<*GGATCCAATCGCTAATCG*> },
donde longitud (S_1, S_2, S_3)} = 18 y cuyo min_{sup} asignado a este ejemplo es de 2.

La Figura *2* muestra el proceso de construcción del mapa para búsqueda.

Para comenzar el proceso de búsqueda de los patrones frecuentes en el ejemplo anterior, donde se tienen tres secuencias, se mapearán todos los nucleótidos de cada secuencia del grupo κ, tal como se muestra en la Tabla 1.

Tabla 1. Mapeo de nucleótidos dentro del grupo de secuencias κ.

Nucleótido	Secuencia	Posición	Nucleótido siguiente
A	S_1	0 6 7 8 9	C A A A C
A	S_2	2 3 10 15	A G G C
A	S_3	2 6 7 13 14	T A T A T
C	S_1	1 10 12	G T T
C	S_2	0 6 7 12 13 16	T C G C G T
C	S_3	4 5 9 11 16	C A G T G
G	S_1	2 4 15	T T T
G	S_2	4 8 11 14	T T C A
G	S_3	0 1 10 17	A C A N/A
T	S_1	3 5 11 13 14 16 17	A C T G T N/A
T	S_2	3 8 12 15	A C A N/A
T	S_3	3 8 12 15	C C A C

2.1.2. 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 2^4 subsecuencias candidatas iniciales de longitud 2, utilizando los cuatro nucleótidos del ADN (A, C, T, G).

Posteriormente, en las siguientes iteraciones *i*, este primer paso utilizará las subsecuencias candidatas de longitud *n* obtenidos en la iteración *i-1* (provenientes de la tercera etapa, descrita en la etapa 3) para crear nuevas subsecuencias candidatas. A partir de cada subsecuencia candidata se generarán cuatro nuevos candidatos de longitud *i+1* debido a que se les concatena un carácter más del alfabeto de nucleótidos.

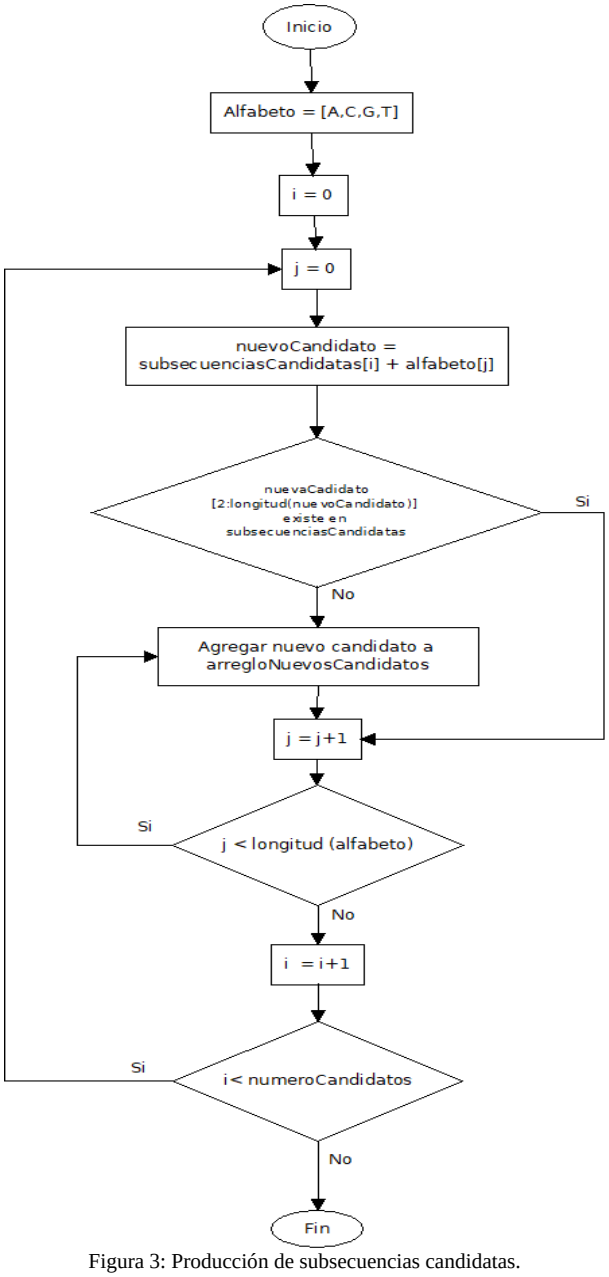

Figura 3: Producción de subsecuencias candidatas.

Por ejemplo, si «AA» es una candidata sobreviviente, se crearán las subsecuencias { «AAA» , «AAC» , «AAG» , «AAT» }. Después serán evaluadas por la propiedad anti-

monótona de soporte para evitar generar candidatos no dispensables [10]. En la Figura 3 se muestra el proceso de generación de subsecuencias candidatas.

2.1.3. Etapa 3: evaluación de subsecuencias candidatas

Dentro de los ciclos iterativos, la segunda etapa es la evaluación de subsecuencias candidatas generadas previamente.

El proceso de evaluación comienza identificando la posición exacta del último carácter de cada una de las subsecuencias candidatas (*nextChar*) a partir de su ubicación en el mapa: el primer nucleótido de la subsecuencia (*elmtn*), secuencia de ADN (*Seq*) y la posición (*Pos*) facilitando la búsqueda del siguiente elemento (*nextE*) para compararlo. En seguida, se calcula la cantidad de las apariciones de cada subsecuencia en el conjunto de secuencias de ADN, mediante la función $f_{Support}$, sin importar cuantas veces exista la subsecuencia en una sola secuencia. Esto da como resultado un valor de "frecuencia secuencial" del elemento, permitiendo identificar en cuantas secuencias de ADN del conjunto de entrada de datos, existe una subsecuencia específica.

El valor umbral que ingresa el usuario, se emplea para determinar si una secuencia candidata es una secuencia sobreviviente, mediante la operación $f_{Support(secuenciaCandidata)}$ umbral y las que superen o sean iguales a el umbral serán subsecuencias sobrevivientes y empleadas para generar nuevas subsecuencias candidatas.

La Figura 4 presenta el proceso que se desarrolla para evaluar subsecuencias candidatas, identificando del lado izquierdo el proceso cuando se genera el primer conjunto de patrones, y del lado derecho la búsqueda en los mapas tomando como base los patrones existentes.

3. Resultados

Para llevar a cabo el presente estudio, se utilizaron 122 secuencias de ADN de diferentes organismos obtenidas de la base de datos biológica del GenBank (Mizrachi, 2016), perteneciente al NCBI (National Center of Biotechnology Information - Centro Nacional de Información de Biotecnología). Se comparó el presente método propuesto contra el algoritmo GSP (Srikant and Agrawal, 1996), ya que es el más utilizado para la identificación de patrones frecuentes en secuencias de ADN. Ambos algoritmos fueron implementados en Python 3.0 y se consideró un umbral de 2 para obtener el conjunto completo de patrones frecuentes.

El algoritmo GSP basa su funcionamiento en la generación y poda de subsecuencias candidatas, así como el recorrido de bases de datos a través de búsquedas secuenciales para obtener como resultado el conjunto de subsecuencias frecuentes. Se compara únicamente con este algoritmo debido a que es el que presenta un comportamiento similar al propuesto, incluso dando el mismo resultado en cuanto a los patrones frecuentes encontrados.

La Tabla 2 muestra un resumen de los experimentos realizados. Como se puede apreciar, se aplicaron los métodos a 12 conjuntos de secuencias de ADN. En la tabla se muestra el número de secuencias por experimento y la longitud promedio de las secuencias en cada uno de ellos. Cabe hacer mención que en ambos algoritmos se obtuvo la misma cantidad de patrones en cada experimento. La diferencia entre estos métodos radica en el tiempo de procesamiento empleado para la identificación de patrones frecuentes. Este tiempo se muestra en formato minutos:segundos.DécimasSegundo.

Tabla 2: Resumen de tiempos de ejecución de experimentos.

No.	Secs.	Long. prom.	No. Patrones	Propuesto	GSP
1	10	1001	1191	00:00.390	00:15.309
2	10	1000	1309	00:00.410	00:16.769
3	10	1000	1349	00:00.330	00:16.983
4	10	1001	1292	00:00.390	00:17.072
5	10	1001	1380	00:00.420	00:18.004
6	10	1000	1463	00:00.410	00:18.230
7	11	1000	1379	00:00.350	00:19.399
8	10	1000	1908	00:00.430	00:24.474
9	13	1001	1979	00:00.450	00:33.003
10	13	1001	2037	00:00.450	00:35.457
11	5	4580	1417	00:00.660	00:45.600
12	10	1000	495961	00:52.630	21:06.100

En la Figura 5 se pueden apreciar los tiempos de ejecución para los dos métodos, para cada uno de ellos se uso una escala única, con la cual se puede apreciar . Se ha omitido el experimento 12 de esta gráfica debido a que la diferencia entre los tiempos es considerablemente superior a los otros.

En particular en este experimento, el método propuesto en este trabajo es 24 veces más rápido que GSP. Como se puede apreciar en la figura, al tener un mayor número de secuencias y longitud de éstas, el método propuesto mejora los tiempos de ejecución de manera considerable.

La Tabla 2 muestra los diferentes resultados obtenidos para cada experimento. El algoritmo propuesto demuestra su eficacia, ya que reduce los tiempos de ejecución entre 24 y 78 veces.

La razón de tan considerable diferencia en los tiempos de ejecución se debe al método de búsqueda propuesto en este trabajo, primero a través de la transformación de las secuencias de ADN en una serie de mapas y después, de la búsqueda y descarte de posiciones de los patrones candidatos, haciendo uso de dichos mapas en vez de buscar de manera secuencial en toda la secuencia cada que se busque el número de ocurrencias de un patrón candidato, esta es la principal

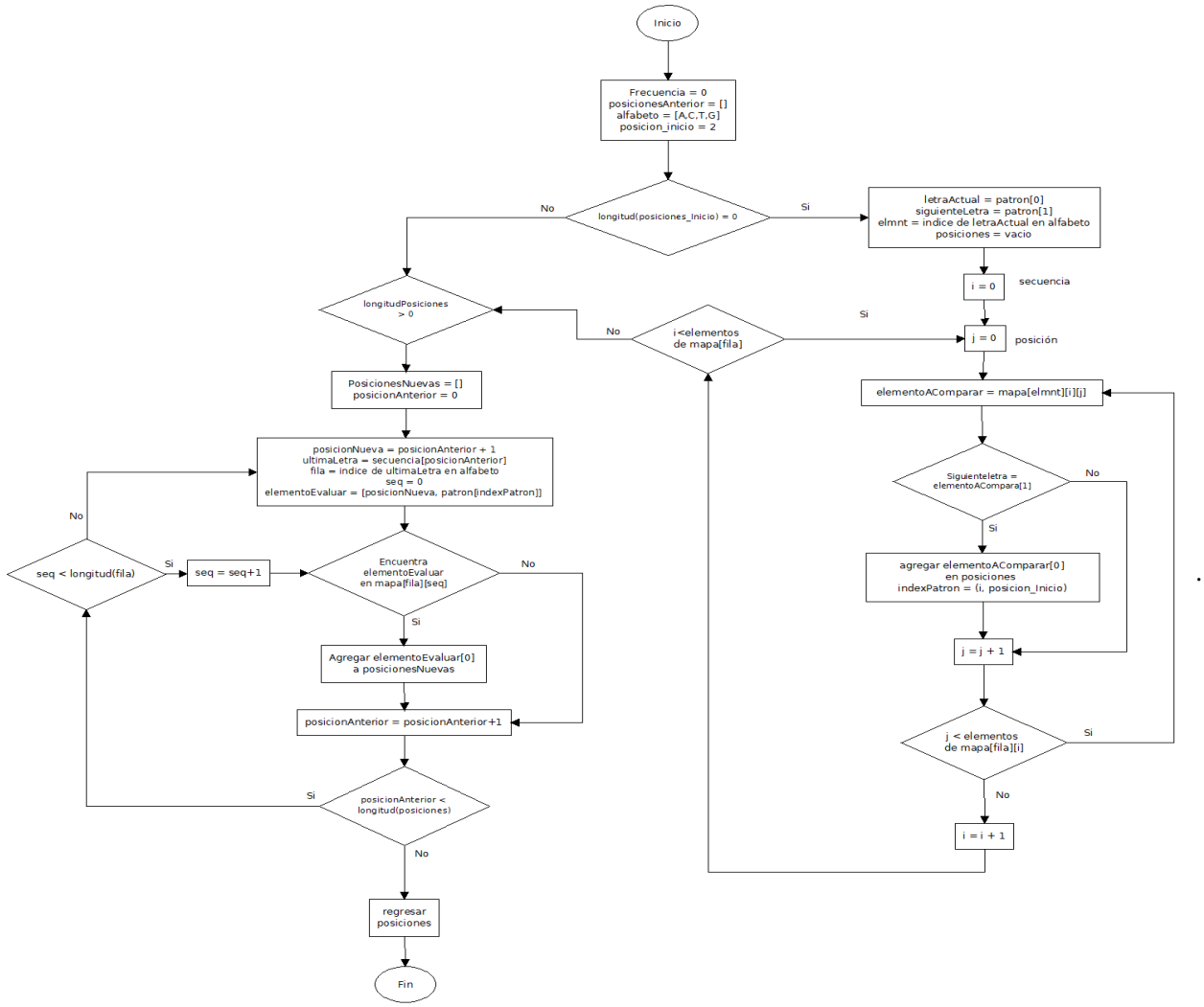

Figura 4: Extracción de frecuencias de las subsecuencias candidatas a través del mapa de posiciones.

diferencia entre esta propuesta y el algoritmo GSP, el cual hace búsquedas secuenciales por cada patrón candidato dentro de todas y cada una de las secuencias.

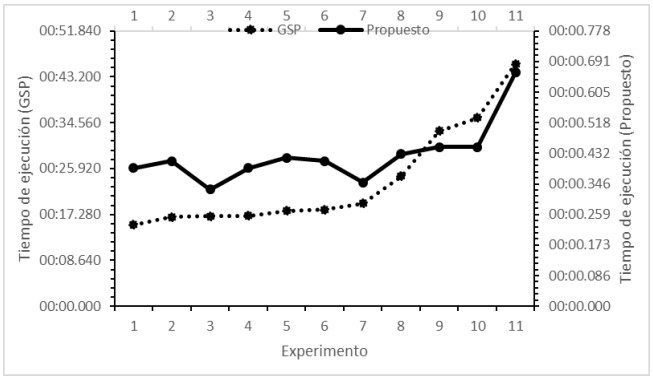

Figura 5: Comparación de tiempos de ejecución

Conclusiones

El presente trabajo presenta un nuevo método para la identificación de patrones frecuentes en grupos de secuencias de ADN, con el fin de poder obtener de ellas información relavante en el ámbito biológico.

El método propuesto consta de tres etapas que buscan hacer búsquedas más eficientes y en consecuencia, obtener un rendimiento mejor del método. Como puede verse en la sección de resultados, el método se ejecuta con notable eficacia al compararlo con un algoritmo clásico de identificación de patrones frecuentes, llamado GSP.

Los experimentos mostrados confirman que dichas mejoras son efectivas y eficaces para la identificación de estos elementos en grupos de secuencias de ADN.

Dentro de los trabajos futuros se contemplan incorporar búsquedas paralelizadas para mejorar aún más la eficacia de éstas, así como utilizar secuencias de ADN con longitudes mayores.

Referencias

Aggarwal, C. and Han, J., (2014). Frequent Pattern Mining. Springer International.

Agrawal, R., y Srikant, R. (1995)., Mining sequential patterns. En Proceedings of the eleventh International Conference on Data Engineering (pp. 3–14). Washington, DC, USA: IEEE Computer Society

Bailey, T.L., (2008) Discovering Sequence Motifs, page 231–251. Humana Press.

D'haeseleer, P., (2006). What are DNA sequence motifs? Nature Biotechnology, 24(4):423–425.

Escobar, C. A. M., Murillo, L. V. R., y Soto, J. F. (2011). Tecnologías bioinformáticas para el análisis de secuencias de adn. Scientia et technica, 3 (49), 116-121.

Han, J. and Micheline, K., (2012). Data mining. Morgan Kaufmann.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., y Hsu, M.-C., (2000). Freespan: frequent pattern-projected sequential pattern mining. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 355–359).

Mizrachi, I., (2016). Genbank: The nucleotide sequence database. consultado el 11-11-2021 desde http://www.ncbi.nlm.nih.gov/books/NBK21105/

Rocha, M., Ferreira, P.G., (2018). Bioinformatics Algorithms: Design and Implementation in Python. Elsevier.

Singh, G. (2006). Fundamentals of bioinformatics and computational biology. Springer International Publishing.

Srikant, R., y Agrawal, R., (1996). Mining sequential patterns: Generalizations and performance improvements. En P. Apers, M. Bouzeghoub, y G. Gardarin (Eds.), Advances in database technology (pp. 1–17). Berlin, Heidelberg: Springer Berlin Heidelberg.

Starr, C., Evers, C.A., Starr, L., (2011). Biología: Conceptos y Aplicaciones. Cengage Learning.