

# Comparación del algoritmo centro estrella paralelo con uno basado en la colonia artificial de abejas (ABC) en el alineamiento múltiple de secuencias

Wilson César Callisaya Choquecota, Yessenia Deysi Yari Ramos

nosliwsys@gmail.com, ydyari@ucsp.edu.pe

Escuela de Posgrado de la Universidad Nacional Jorge Basadre Grohmann, Perú  
Esq. Av. Pinto con Av. Bolognesi (Local Central de la UNJBG – Tacna)  
Tacna-Perú  
Universidad Católica San Pablo, Perú  
Campus San Lázaro- Quinta Vivanco s/n, Urb. Campiña Paisajista, Arequipa  
Arequipa-Perú

**Resumen:** En la actualidad, se ha producido un considerable esfuerzo para desarrollar algoritmos que comparan las secuencias de macromoléculas biológicas, cuyo objetivo es detectar las relaciones evolutivas tanto estructurales como funcionales. El alineamiento múltiple es el que aporta mayor información biológica. El algoritmo centro estrella que en su proceso usa el alineamiento de pares de Needleman-Wunsch determina el alineamiento óptimo de varias secuencias. El uso de la programación paralela disminuye el tiempo de ejecución de este algoritmo. Los algoritmos de inteligencia de enjambre son ampliamente usados para resolver problemas de optimización en particular el algoritmo de la colonia artificial de abejas (ABC). Este trabajo presenta una comparación del algoritmo centro estrella paralelo con el algoritmo de colonia artificial de abejas en el alineamiento múltiple de secuencias comparando los tiempos de respuesta de ambos algoritmos y los puntajes de sus alineamientos. Se ha adaptado el algoritmo colonia artificial de abejas sin el uso de la programación paralela para realizar el alineamiento múltiple de secuencias. El software utilizado para ello fue el C# con la librería TPL (Task Parallel Library). Los resultados muestran que el algoritmo colonia artificial de abejas tiene un menor tiempo de respuesta mientras más secuencias se tengan que alinear, y si sus longitudes son grandes.

**Palabras clave:** Biología Computacional, Alineamiento múltiple de secuencias, Programación Paralela, Bioinformática, Artificial Bee Colony.

**Abstract:** At present there has been a considerable effort to develop algorithms that compare the sequences of biological macromolecules, which aims to detect evolutionary relationships both structural and functional. Multiple alignment is the most biologically that provides information, the algorithm center star in the process pairwise alignment using Needleman-Wunsch determines the optimal alignment of several sequences. The use of parallel programming time decreases execution of this algorithm. The swarm intelligence algorithms are widely used to solve optimization problems including the algorithm of artificial bee colony (ABC). This paper presents a comparison of parallel algorithm star center with the algorithm artificial bee colony in the multiple sequence alignment comparing the response times of both algorithms and their alignments scores. It has adapted the algorithm artificial bee colony without the use of parallel programming for multiple sequence alignment. The software used for this was the C # with TPL (Task Parallel Library). The results show that the artificial bee colony algorithm has a shorter response time while more sequences are to be aligned, and if their lengths are large.

**Keywords:** Computational Biology, multiple sequence alignment, Parallel Programming, Bioinformatics, Artificial Bee Colony.

## 1 Introducción

Uno de los principales problemas de la biología computacional es el de alineamiento de secuencias biomoleculares (ADN, ARN o secuencias de aminoácidos), ya que la similitud de secuencias implica similitud funcional o estructural significativa.[1]

Una extensión natural del alineamiento de pares es la alineación de secuencias múltiples, que es alinear múltiples secuencias relacionadas para lograr adaptación óptima de las secuencias. La ventaja del alineamiento de múltiples secuencias es que revela más información biológica que muchas alineaciones por parejas recién nos permitirían. La alineación de secuencias múltiples es también un prerrequisito esencial para llevar a cabo el análisis filogenético. Es posible utilizar la programación dinámica para alinear cualquier número de secuencias como para la alineación por parejas. Sin embargo, la

cantidad de tiempo y la memoria de computación que requiere aumentan exponencialmente a medida que el número de secuencias aumenta. En la práctica, se utilizan con mayor frecuencia los enfoques heurísticos. [2]

Varias investigaciones se han realizado para ayudar a resolver este problema eficientemente, pero poco se ha intentado usando el paradigma paralelo, esto debido a los costes que implicaba un computador paralelo y a su difícil implementación dado que había que darle importancia a la comunicación entre los procesadores, pero en la actualidad ya existen librerías que nos apoyan a desarrollar en paralelo, dando la oportunidad a enfocarse solo en el problema de fondo.

La inteligencia de enjambre se define brevemente como el comportamiento colectivo de los enjambres descentralizados y auto-organizados. Varios algoritmos se han desarrollado en función de los diferentes comportamientos inteligentes de enjambres de abejas. La

principal ventaja de ABC es que no es sensible a los valores de los parámetros iniciales y no se ve afectado por la creciente dimensión del problema.[3][4]

Este paper presenta una comparación del algoritmo centro estrella paralelo con uno basado en colonia artificial de abejas para el alineamiento de múltiples secuencias.

El resto de éste paper está organizado de la siguiente manera. La Sección 2 se muestra los trabajos previos. En la Sección 3 se explica cómo es el procedimiento para realizar el alineamiento de múltiples secuencias con el algoritmo centro estrella y dando referencia general al algoritmo de colonia artificial de abejas. La Sección 4 detalla el pseudocódigo de ambos algoritmos en el alineamiento múltiple de secuencias. La Sección 5 muestra los experimentos y resultados de la aplicación en la comparación de los algoritmos, midiendo sus tiempos cuando las secuencias cambian su longitud y tamaño. Finalmente se dan las conclusiones en la Sección 6.

## 2 Trabajos previos

Hay varios investigadores que refieren al uso de las colonia artificial de abejas mostrando la importancia de su estudio como algoritmo como en [5] que compara entre algoritmo de colonia de abejas estándar y el algoritmo de colonia de abejas adaptativo que propuso en el problema del vendedor viajero (Travelling Salesman Problem), en [6] su trabajo presenta un algoritmo de colonia de abejas artificial modificado (ABC) para la solución de costo mínimo en un SIG basado en raster, en [7] presenta un enfoque multiobjetivo paralelo basado en el algoritmo artificial Colonia de abejas para el cuidado de las solicitudes de tráfico de baja velocidad en los canales ópticos de alta capacidad, finalmente presentan un estudio comparativo con métodos tradicionales, en el que muestra que la inteligencia de enjambre supera a los resultados previos en trabajos similares, en [8] usa el algoritmo de las abejas, precisando que este algoritmo da una solución casi óptima para el problema de búsqueda.

Pero hay diversos problemas en la biología computacional en los que puede apoyarnos el algoritmo de enjambre y hay varios investigadores que exploran diversas soluciones en esta área trabajos como en [9] que indica que para resolver estos problemas es necesaria la combinación de computación bioinspirada y computación paralela para hacer frente a la complejidad necesaria para obtener soluciones óptimas en tiempos reducidos es por ello que en su trabajo presenta un estudio de rendimiento en máquinas multinúcleo de una adaptación multiobjetivo paralela del algoritmo artificial colonia de abejas para inferir filogenias de acuerdo con la máxima parsimonia y criterios de máxima verosimilitud, en [10] estudia la utilización del paralelismo para tratar de combinar el comportamiento de diferentes algoritmos para la resolución de problema real de búsqueda de patrones de ADN.

El uso de los múltiples núcleos del procesador ha apoyado a varios de los investigadores mencionados pero uno de los mayores problemas de la biología computacional es el alineamiento de secuencias particularmente alinear múltiples secuencias, es así que [11] detalla en su trabajo

que la alineación de unos pocos cientos de secuencias mediante herramientas populares de alineación progresiva requiere varias horas en ordenadores secuenciales. Por ello presenta un diferente enfoque para MSA en plataformas de hardware reconfigurable para obtener un alto rendimiento a bajo costo, en [12] indica que la alineación de secuencias múltiples es una extensa tarea informática por ello presenta un modelo de alineación de secuencias múltiples en paralelo efectivo capaz de resolver este problema.

Uno de los primeros intentos para resolver el problema del alineamiento múltiple de secuencias fue con algoritmo centro estrella y aún siguen mejorándolo esta vez con la paralelización como indica [13].

Dar un planteamiento para el algoritmo de la colonia artificial de abejas para resolver el problema del alineamiento múltiple de secuencias ha sido estudiado por investigadores como en [14] y [15] haciendo las comparaciones pertinentes con otros algoritmos de alineamiento múltiple.

## 3 Teoría del dominio

### 3.1. Alineamiento de secuencias

El alineamiento no es más que un acomodo de gaps en secuencias para poder lograr una mayor cantidad de coincidencias en diferentes secuencias. Si se tiene la secuencia A= gctgaacg y B= ctataatc los alineamientos podrían ser como se muestra en la Fig. 1. [16]

g c t g - a a - c g	Caso 1
- c t a t a a t c -	
g c t g a - a - - c g	Caso 2
- - c t - a t a a t c	
- - - - - g c t g a a c g	Caso 3
c t a t a a t c - - - - -	

Figura 1. Ejemplos de alineamiento.

Para determinar cuál es el mejor alineamiento para las secuencias se usan sistemas de puntuación en donde a las identidades (Match) se les dan puntuaciones positivas y las no coincidencias (Mismatch) y espacios añadidos (Gaps) valores negativos. Por ejemplo sean las secuencias a= ACTTG y b=AGATT dando a la coincidencia 1, a la no coincidencia 0 y al gap -1 un posible alineamiento sería como muestra la Fig. 2. [17]

a = A C - T T G	
b = A G A T T -	
score = 1 0 -1 1 1 -1	

Figura 2. Ejemplo de alineamiento con puntuación.

### 3.2. Programación dinámica para el alineamiento de secuencias (Algoritmo de Needleman-Wunsch)

Para alinear las secuencias con programación dinámica se debe definir un valor para el match (*coincidencia*), mismatch (*no coincidencia*) o seleccionar una matriz de sustitución y el gap (puntaje cuando ocurre un *Indel*), el

proceso se realiza en 3 pasos:

Paso 1 Inicialización: Ambas secuencias se ubican en una matriz F de  $m \times n$  ( $m$  y  $n$  longitudes de ambas secuencias), luego el valor de la posición  $F(0,0) = 0$  y se llena tanto la primera fila y la primera columna con múltiplos del valor del Gap como lo describe el ejemplo en la Fig.3.

Paso 2 llenado de la matriz de scores: Se procede a llenar todos los valores de la matriz según la función descrita en la Fig. 4 el ejemplo de la Fig. 6 describe el proceso del alineamiento.

Paso 3 Identificación del alineamiento – Traceback: Este paso inicia siempre en la posición  $(m,n)$  en el cual está el score del mejor alineamiento y se hace un recorrido hacia atrás para identificar el alineamiento como se describe en la Fig. 6.[18][19]

		Sequence 2								
		F	M	D	T	P	L	N	E	
Sequence 1	F	0	-2	-4	-6	-8	-10	-12	-14	-16
	K	-2								
	H	-4								
	M	-6								
	E	-8								
	D	-10								
	P	-12								
	L	-14								
	E	-16								

Figura 3. Inicialización con gap.

$$F(i, j) = \max \begin{cases} F(i - 1, j - 1) + s(x_i, y_j), \\ F(i - 1, j) - d, \\ F(i, j - 1) - d. \end{cases}$$

Figura 4. Función del alineamiento global

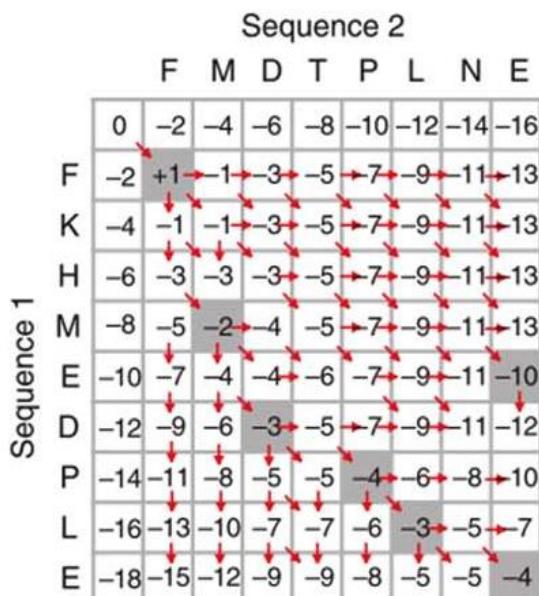


Figura 5. Llenado de la matriz de scores.

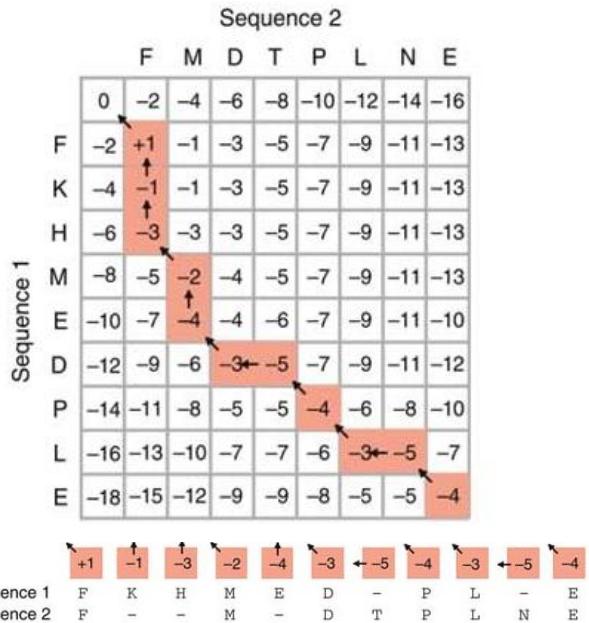


Figura 6. Traceback (recorrido hacia atrás).

### 3.3. El puntaje en el alineamiento múltiple de secuencias

La tarea de evaluar el puntaje se vuelve una tarea compleja, una solución para este problema es la evaluación por suma de pares. Para evaluar el puntaje para el siguiente alineamiento en la cuarta columna por suma de pares y considerando un puntaje para la coincidencia el valor de 1 para la no coincidencia y para el gap un valor de -2, se realiza como indica la Fig. 7. [20]

$$\begin{aligned} \text{MQPILLLV} & SP(I, -, I, V) = sc(I, -) + sc(I, I) + sc(I, V) - \\ \text{MLR-LL--} & sc(-, I) + sc(-, V) + sc(I, V) \\ \text{MK-ILLL-} & = -2 + 1 - 1 - 2 - 2 - 1 \\ \text{MPPVILV} & = -7 \end{aligned}$$

Figura 7. Ejemplo de evaluación de puntaje para la cuarta columna.

### 3.4. Algoritmo centro estrella

Consiste en encontrar una secuencia que sea la más similar con el resto aplicando alineamiento por pares, el proceso requiere encontrar todos los alineamientos por pares, encontrar el centro Sc (Secuencia central), alinear las demás secuencias con Sc respetando el principio de consistencia. Un ejemplo del uso del algoritmo de alineamiento centro estrella se muestra en la Fig. 8.

a)	S1	S2	S3	S4	S5		
	A T T G C C A T T	-	7	-2	0	-3	
	S2	A T G G C C A T T	7	-	-2	0	-4
	S3	A T C C A A T T T T	-2	-2	-	0	-7
	S4	A T C T T C T T T	0	0	0	-	-3
	S5	A C T G A C C	-3	-4	-7	-3	-17
b)	S1	S2	S3	S4	S5		
	S1	-	7	-2	0	-3	
	S2	7	-	-2	0	-4	
	S3	-2	-2	-	0	-7	
	S4	0	0	0	-	-3	
	S5	-3	-4	-7	-3	-17	
c)	S1	S2	S3	S4	S5		
	A T T G C C A T T	-	7	-2	0	-3	
	S2	A T G G C C A T T	7	-	-2	0	-4
	S3	A T C C A A T T T T	-2	-2	-	0	-7
	S4	A T C T T C T T T	0	0	0	-	-3
	S5	A C T G A C C	-3	-4	-7	-3	-17
d)	S1	S2	S3	S4	S5		
	A T T G C C A T T	-	7	-2	0	-3	
	S2	A T G G C C A T T	7	-	-2	0	-4
	S3	A T C C A A T T T T	-2	-2	-	0	-7
	S4	A T C T T C T T T	0	0	0	-	-3
	S5	A C T G A C C	-3	-4	-7	-3	-17

Figura 8. Ejemplo de uso del algoritmo centro estrella.

En la Fig. 08, se muestra en (a) Las secuencias iniciales para alinear las cuales serán alineadas por el algoritmo de Needleman Wunsch (b) Matriz de puntajes por pares resultados del alineamiento por el algoritmo de Needleman Wunsch el centro se designa a S1 al tener mejor puntuación (c) Las secuencias alineadas con S1 (d) Se construye el alineamiento múltiple aplicando la consistencia, cualquier gap que se introduce ya no es retirado. [20]

### 3.5. Algoritmo colonia artificial de abejas

Conocido como Artificial Bee Colony (ABC) es uno de los algoritmos más recientes en el dominio de la inteligencia colectiva. Creado por Dervis Karaboga en 2005, el cual fue motivado por el comportamiento inteligente observado en las abejas domésticas para llevar el proceso de forrajeo.

ABC es un algoritmo de optimización combinatoria basado en poblaciones, en el cual las soluciones del problema de optimización, llamadas fuentes de alimento, son modificadas por las abejas artificiales que fungen como operadores de variación. El objetivo de estas abejas es descubrir las fuentes de alimento con mayor néctar.

En el algoritmo ABC, las abejas artificiales se mueven en un espacio de búsqueda multidimensional eligiendo fuentes de néctar dependiendo de su experiencia pasada y la de sus compañeros de colmena o ajustando su posición. Algunas abejas (exploradoras) vuelan y eligen las fuentes de alimento aleatoriamente sin usar experiencia. Cuando encuentran una fuente de néctar mayor, memorizan su posición y olvidan la anterior. De este modo, ABC combina métodos de búsqueda local y búsqueda global, intentando equilibrar el proceso de la exploración y de la explotación del espacio de búsqueda. [21].

El modelo define los siguientes componentes principales:

**Fuente de alimento:** El valor de una fuente de alimento depende de muchos factores, como su proximidad a la colmena, riqueza o la concentración de la energía y la facilidad de extracción de esta energía.

**Abejas Empleadas:** Están asociadas a una fuente de alimento, actual o en explotación. Llevan con ellas información sobre esa fuente en particular, su distancia, ubicación y rentabilidad para compartirla, con una cierta probabilidad a sus demás compañeras.

**Abejas Exploradoras:** Están en constante búsqueda de una fuente de alimento. Hay dos tipos Scout y Observadora.

**Scout:** Se encargan de buscar en el entorno que rodea a la colmena nuevas fuentes de alimento.

**Observadora:** Buscan una fuente de alimentos a través de la información compartida por las empleadas o por otras exploradoras en el nido.

El intercambio de información entre las abejas es la más importante incidencia en la formación de un conocimiento colectivo, ya que mediante esta interacción las abejas decidirán el comportamiento que debe llevar la colmena.

Sus principales modos de comportamiento son:

La incorporación a una fuente de néctar: La comunicación

entre las abejas relacionadas con la calidad de fuentes de alimento se produce en la zona de baile (danza de las abejas). En donde con la información obtenida sobre todas las fuentes de alimento que están disponibles se determina cuál de todas las fuentes es la más rentable para así incorporarse a ella.

El abandono de una fuente: Se determina conforme al valor de la fuente y al número de visitas que se le hace, es decir mediante la danza se termina si una fuente ya no es rentable y por consiguiente debe ser abandonada [22].

## 4 Diseño de los algoritmos utilizados

### 4.1. Elaboración del algoritmo centro estrella paralelo

La implementación algorítmica descrita en 2.4 se observa en el algoritmo 1, las variables usadas son:

val= Que contendrá la matriz de puntajes.

Imax= Que contiene el valor del índice de la secuencia principal.

SP= Que contiene el valor de la secuencia principal.

af= Que contiene las secuencias alineadas finales.

---

**Algoritmo 1** Algoritmo centro estrella paralelo basado en algoritmo centro estrella [9].

---

**Requiere:**

La función *wunsch* (requiere sec1, sec2, devuelve puntaje, sec1alineada, sec2alineada).

La cantidad de secuencias p.

Un arreglo de secuencias asec()

La función IndiceSecPrincipal que nos devuelve el índice de la secuencia principal.

La función UNIRGAPS que junta los gaps de dos secuencias.

La función secfinal que nos devuelve la secuencia final añadiendo los gaps de la secuencia principal.

**Asegurar:**

val[0, 0] = 0

val[p - 1, p - 1] = 0

**Para i = 0 Hasta p – 2 Hacer**

**Para j=i+1 Hasta p-1 Hacer en Paralelo**

Wunsch(asec(i), asec(j), val[i,j], seq[i,j,1], seq[i,j,2])

Val[j,i]=val[i,j]

Seq[j, i, 2]=seq[i,j,1]

Seq[j, i, 1]=seq[i,j,2]

**Fin Para**

**Fin Para**

Imax= IndiceSecPrincipal(val, p)

/\* Hallando la secuencia principal \*/

**Si Imax = 0 entonces**

SP = seq[Imax, 1, 1]

**Si no**

SP = seq[Imax, 0, 1]

**Fin si**

**Para i = 0 Hasta p – 2 Hacer**

**Si i >= Imax entonces**

SP = UNIRGAPS(SP, seq[Imax, i + 1, 1])

**Si no**

SP = UNIRGAPS(SP, seq[Imax, i, 1])

**Fin Para**

/\* generando las secuencias alineadas finales \*/

**Para j=0 Hasta p-1 Hacer en Paralelo**

**Si i= Imax entonces**

Para su uso, ha sido necesario diseñar la función Secfinal

como se muestra en el algoritmo 2, teniendo como variables:

fs= Es el valor de la secuencia final.

g= variables auxiliar que indicara el número de gaps que se van añadiendo.

#### **Algoritmo 2 Secfinal (secp,secp2, secs)**

##### **Requiere:**

La secuencia principal secp

La secuencia principal secundaria

La secuencia secundaria

Longitud de la secuencia principal ls1

Longitud de la secuencia principal secundaria ls2

##### **Asegurar:**

g=0

**Para** i = 0 **Hasta** ls1-1 **Hacer**

**Si** i < ls2 + g **entonces**

        fs = fs + s3[i - g]

**Si no**

        fs = fs + "-"

**fin Si**

**Si no**

        fs = fs + "-"

**fin si**

Para la construcción de la secuencia principal ha sido necesario unir los gaps de las diferentes secuencias principales como se muestra en algoritmo 3.

#### **Algoritmo 3 UNIRGAPS(secp, secq)**

##### **Requiere:**

Secuencias a unir gaps secp (p) y secq (q)

La función longitud() que da la cantidad de una cadena

Inicializar los incrementos ip y iq en 0.

##### **Asegurar:**

**Si** longitud (p)< longitud(q) **entonces**

    T = p

    p = q

    q = T

**fin si**

MaxL = longitud(p)

MinL = longitud(q)

**Para** i = 0 **Hasta** MaxL + ip-1 **Hacer**

**Si** i < MinL + iq **entonces**

        Si p[i - ip] = q[i - iq] **entonces**

            C = C + p[i - ip]

**Si no**

            C = C + "-"

**Si** p[i - ip] = '-' **entonces**

            iq = iq + 1

**Si no**

            ip = ip + 1

**fin si**

**fin si**

**Si no**

En el algoritmo 3, usa las variables:

T= Auxiliar para intercambio y garantizar que p tenga la mayor longitud.

MaxL y MinL= Contienen la máxima y mínima longitud.

C= Contiene las secuencias unidas.

**Algoritmo 4 wunsch** (requiere sec1, sec2, devuelve puntaje, sec1alineada, sec2alineada) [9]

##### **Requiere:**

La función max (Máximo de 3 números), la función de similitud S y el puntaje del gap, la función Invertir\_texto que invierte el orden del texto

Las longitudes de las secuencias n y m

##### **Asegurar:**

**Para** i=0 **Hasta** n **Hacer**

    f(i,0)=i x gap

**Fin Para**

**Para** i=0 **Hasta** m **Hacer**

    f(0,i)=i x gap

**Fin Para**

**Para** i=1 **Hasta** n **Hacer**

**Para** j=1 **Hasta** m **Hacer**

        f(i,j)=max(f(i-1,j)+gap, f(i-1,j-1)+s(sec1, sec2, i,j), f(i,j-1)+gap)

**Fin Para**

**Fin Para**

i = n     /\*Inicio del traceback\*/

j = m

**Mientras** (i > 0) o (j > 0) **hacer**

**Si** (i > 0 y j=0) o (i>0 y j >0 y (f[i, j]=f[i - 1, j]+gap)) **entonces**

        Asec1 = Asec1 + sec1[i - 1]

        Asec2 = Asec2 + "-"

        i = i - 1

**Si no**

**Si** (i=0 y j>0) o (i > 0 y j >0 y f[i, j]=f[i, j - 1]+gap) **entonces**

            Asec1 = Asec1 + "-"

            Asec2 = Asec2 + sec2[j - 1]

            j = j - 1

**Si no**

            Asec1 = Asec1 + sec1[i - 1]

            Asec2 = Asec2 + sec2[j - 1]

            i = i - 1

            j = j - 1

**fin si**

**fin si**

El algoritmo 4 muestra el algoritmo de alineamiento de pares de Needleman Wunsch necesario para la construcción de la matriz de puntajes por pares según [9].

## 4.2. Elaboración del algoritmo basado en colonia artificial de abejas en el alineamiento múltiple de secuencias

El algoritmo de la colonia artificial de abejas en el alineamiento múltiple de secuencias se adaptó del algoritmo presentado por [23], y se generaron funciones adicionales para su uso. A continuación, se detalla paso a paso el algoritmo junto a las funciones adicionales.

El algoritmo 5 muestra el algoritmo adaptado de [23].

Siendo ne y nenjambre el valor de del enjambre actual y el número de enjambres, nscout y nobservadoras el número de abejas scout y de observadoras respectivamente.

---

**Algoritmo 5** Algoritmo ABC basado en [23]**Requiere:**

Función Generar Alineamiento Aleatorio  
 Función GenerarCadenaFavoritaYSupGap(sec)  
 Función InsertarySuprimirGap(GapT, sec)  
 Función de evaluación Seval(s1,s2, i)

**Asegurar**

Ingresamos las secuencias  
**Para** ne=1 **Hasta** nenjambre **Hacer**  
**Para** B=1 **Hasta** nscout **Hacer**  
 Generar alineamiento aleatorio  
 Construimos la secuencia favorita  
 Evaluamos las secuencias con la secuencia favorita  
 Guardamos el alineamiento en la lista de optimización.  
**Fin Para**  
**Para** B=1 **Hasta** nobservadoras **Hacer**  
 Seleccionar un alineamiento de manera aleatoria de la lista de optimización  
 Seleccionar la secuencia con el menor puntaje  
 Insertar y eliminar un gap de manera aleatoria  
 Evaluamos la nueva secuencia con la secuencia favorita  
**Si** es de mayor calidad **entonces**  
 La guardamos en la lista de optimización  
**Si no**  
 Descartamos el nuevo alineamiento  
**Fin si**  
**Fin Para**  
 Seleccionamos el alineamiento que tenga mayor calidad de la lista de optimización.  
 Guardamos este alineamiento la lista de alineamiento Elite  
**Fin Para**  
 Seleccionar el mejor alineamiento del alineamiento Elite

Como se aprecia, las abejas scout primero generan alineamientos aleatorios. Este proceso se observa en el algoritmo 6, las variables que se usan son IGAM que es el incremento gaps de forma aleatoria que se realiza a la secuencia que no sobrepasa a la secuencia mayor que se asume que estará en la posición 0 de secuencias, y sec2 almacenará el alineamiento aleatorio.

---

**Algoritmo 6** Generar\_aLINEAMIENTO\_aleatorio(secuencias[])**Requiere:**

Un arreglo de secuencias "secuencias"  
 El incremento de gaps a la secuencia mayor IGMAX  
 La función Rand(inicio,fin) que genera un número aleatorio entre el inicio y fin.  
 La función InsertarT() que inserta en un texto en una determinada posición una letra.  
 La función Longitud() que indica la cantidad de letras de un texto  
 El tamaño de la máxima secuencia MaxSec  
**Asegurar:**

IGAM = Rand(0, IGMax + 1)

**Para** i = 0 **Hasta** p-1 **Hacer**

seq2[i] = secuencias[i]

**Para** j = 1 **Hasta** MaxSec + IGAM – Longitud(seq2[i]) **Hacer**

InsertarT(seq2[i], Rand(0, Longitud(seq2[i]) + 1), "-")

**Fin Para**

El siguiente paso es la construcción de la secuencia favorita y para ello se escoge la letra que más se repite. Un ejemplo de ello se puede observar en la Fig. 9, de rojo la secuencia favorita y debajo las 3 secuencias generadas de un alineamiento aleatorio.

<b>A</b>	<b>G</b>	<b>T</b>	<b>C</b>	<b>A</b>	<b>A</b>	<b>T</b>
A	A	T	C	G	A	T
A	G	T	C	A	T	T
A	G	-	G	A	A	G

Figura 9. Ejemplo de generación de la secuencia favorita.

---

**Algoritmo 7** GenerarCadenaFavoritaYSupGap(secuencias[])**Requiere:**

Un arreglo de secuencias "secuencias"  
 La longitud de la mayor secuencia L  
 Un arreglo de letras(A, C, T, G)  
 La función Remover() que quita una letra de una posición específica

**Asegurar:**

**Para** i = 0 **Hasta** L-1 **Hacer**

**Para** k = 0 **Hasta** 3 **Hacer**

num[k] = 0

**Fin Para**

**Para** j = 0 **Hasta** p-1

**En caso de** seq[j][i] **Haga**

Caso 'A': num[0] = num[0] + 1

Caso 'C': num[1] = num[1] + 1

Caso 'T': num[2] = num[2] + 1

Caso 'G': num[3] = num[3] + 1

**Fin Caso**

**Fin Para**

mayor = num[0]

imayor = 0

**Para** j = 1 **Hasta** 3 **Hacer**

**Si** mayor < num[j] **entonces**

mayor = num[j]

imayor = j

**fin si**

**Fin Para**

**Si** mayor = 0 **entonces**

**Para** j = 0 **Hasta** p-1 **Hacer**

Remover(seq2[j], i)

**Fin para**

L = L - 1;

i = i - 1;

Hay que tener consideración que en la generación de alineamiento aleatorio podría existir una columna llena de gaps, por lo tanto, esa columna de ser retirada, por ello el algoritmo 7 genera la secuencia favorita y remueve una columna que contiene solamente gaps.

El siguiente paso que realizan los scout es evaluar las secuencias con la secuencia favorita. Para ello, será necesario usar una función de score, por ejemplo si consideramos una puntuación 1 para un match, -1 para un mismatch y 0 para un gap, tomando como datos el ejemplo de la Fig. 9 la matriz de puntuación sería la mostrada en la Fig. 10, pudiendo observar en el lado derecho en rojo el puntaje final de la secuencia evaluada con la secuencia favorita.

1	-1	1	1	-1	1	1	<b>3</b>
1	1	1	1	1	-1	1	<b>6</b>
1	1	0	-1	1	1	-1	<b>2</b>

Figura 10. Puntajes realizados de las secuencias con la secuencia favorita.

Para ello, se generó el algoritmo 8, el cual realiza la evaluación de dos secuencias carácter a carácter, según la posición deseada, fácilmente adaptable para comparar una secuencia con la secuencia favorita.

---

**Algoritmo 8** Seval(s1,s2, i)**Requiere:**

Las secuencias s1 y s2

La posición a comparar i

Valor del match, missmatch y Gap

**Asegurar:****Si**  $s1[i] = s2[i]$  **y**  $s1[i] \neq '-'$  **entonces**

P= Match

**Si no****Si**  $s1[i] = '-'$  **o**  $s2[i] = '-'$  **entonces**

P= Gap

**Si no**

P=Mismatch

**Fin si**

Luego se guarda el alineamiento en una lista para optimizar por las observadoras. Las observadoras escogen aleatoriamente uno de los alineamientos de la lista y proceden a intentar mejorar la calidad del alineamiento. Para ello, una vez seleccionado el alineamiento, insertan y eliminan un gap de manera aleatoria a la secuencia que tenga el menor puntaje conseguido al evaluarlo con su secuencia favorita. Un ejemplo se muestra en la Fig. 11.

- a) A-TGC-GGTA-CCGT-G
- b) A-TGC-GGTA-CCGT-G
- b) A-TG-C-GGTA-CCGTG
- c) A-TGC-GGTA-C CGT-G
- c) A-TGCGGTA-C-CGT-G

Figura 11. Ejemplo de inserción y eliminación de gaps aleatoriamente.

---

**Algoritmo 9** InsertarySuprimirGap(GapT, sec)**Requiere:**

Una secuencia sec

El total de Gaps de la secuencia sec (GapT)

La función Rand(inicio,fin) que genera un número aleatorio entre el inicio y fin.

La función InsertarT() que inserta en un texto en una determinada posición una letra.

La función Longitud() que indica la cantidad de letras de un texto

**Asegurar:****Si**  $GapT \neq 0$  **entonces**

Isec = Longitud(sec)

iig = Rand(0, Isec + 1)

InsertarT(sec, iig, "-")

eg = Rand(1, GapT + 1)

ge = 0

**Para** i = 0 **Hasta** Isec **Hacer****Si**  $sec[i] = '-'$  **entonces**

ge = ge + 1

**Si** i = iig **entonces**

ge = ge - 1

**Si no****Si** ge = eg **entonces**

sec = sec.Remove(i, 1)

i = Isec

**Fin si****Fin si**

En la Fig. 11 en a), se observa la secuencia con menor puntaje, en b), se aprecia la inserción 5 y la eliminación de gap de la posición 16, de la misma forma en c), la

eliminación de un gap en la posición 6 y la inserción de uno en la posición 13. Para este procedimiento, se elaboró el algoritmo 9, siendo las variables usadas:

iig= índice de la inserción del gap de forma aleatoria.

eg= número de gap a eliminar.

ge= contador de gaps transcurridos

Si su puntaje es mejor al anterior al evaluarlo con su secuencia favorita, lo guardamos sino lo descartamos, luego de este proceso se selecciona el alineamiento que tiene mejor puntuación total, y se guarda en la lista de alineamientos elites para cada enjambre. Finalmente, se escoge el mejor alineamiento elite.

## 5 Experimentos y resultados

La herramienta de software ha sido implementada en C# usando el Visual Studio y el Framework 4.0, ya que este Framework provee la librería TPL y esta librería nos da soporte para bucles paralelos [24].

Para el caso del algoritmo ABC, se consideró como parámetros 15 enjambres, 15 scout y 15 observadoras, para realizar la medición de tiempos se ha usado la clase Stopwatch disponible en la librería “System.Diagnostic” de C#, así podemos obtener un cronómetro de gran precisión.

El experimento fue realizado en un ordenador de procesador Intel (R) Core (TM) i5-3210M de 2.5 Ghz, el cual tiene 4 núcleos, con memoria de 8Gb. En la Fig. 12 se observa la interfaz de la aplicación final con los tiempos obtenidos usando Stopwatch para 20 secuencias de longitudes no mayores a 2700 caracteres.

En el estudio del rendimiento, se generó 900 grupos de secuencias para ser alineadas, teniendo grupos 3 a 32 secuencias para alinear (un total de 30 grupos), realizándose 30 alineaciones por grupo de longitudes aproximadas de 90 a 2700 pares de bases (pb). Los tiempos de ejecución (en milisegundos) resultados del experimento se observan en las Fig. 13, Fig. 14 y Fig. 15.

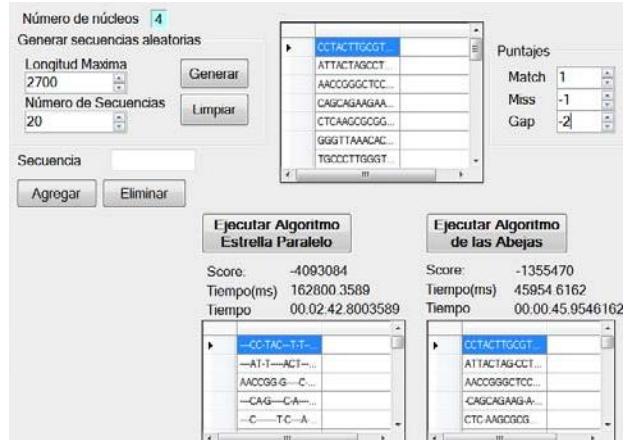


Figura 12. Interfaz de la aplicación de alineamiento de secuencias con los tiempos empleados.

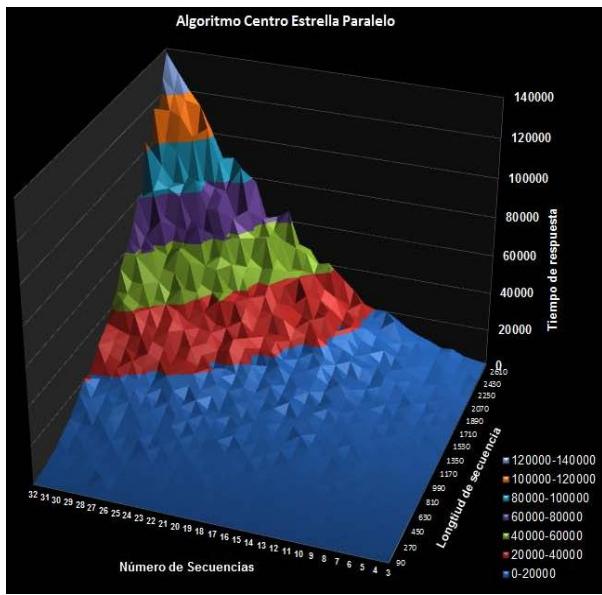


Figura 13. Tiempos de ejecución para el algoritmo centro estrella paralelo según la longitud y número de secuencias a alinear.

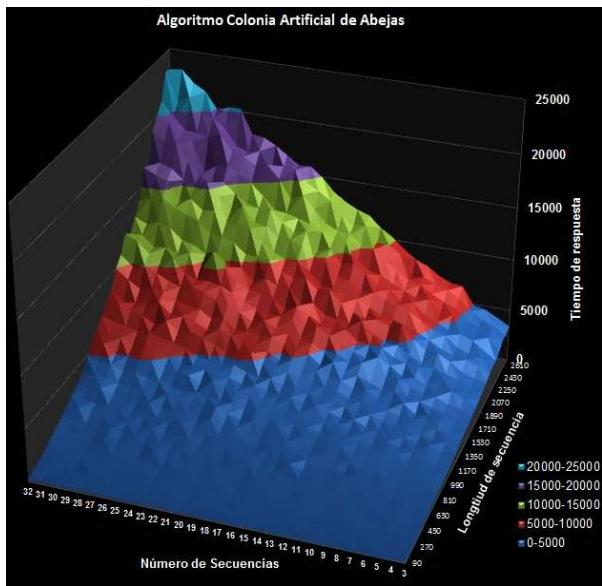


Figura 14. Tiempos de ejecución para el basado en colonia artificial de abejas según la longitud y número de secuencias a alinear.

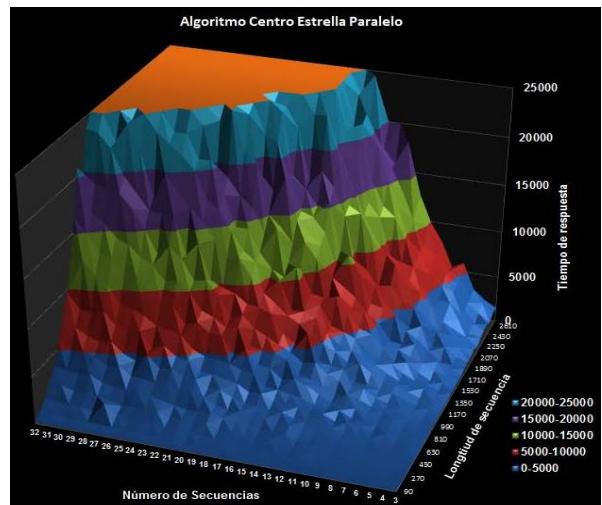


Figura 15. Tiempo de ejecución del algoritmo centro estrella paralelo con tiempos de ejecución menores a 25000 milisegundos según la longitud y número de secuencias a alinear

En la Fig. 13, observamos que el algoritmo centro estrella paralelo tiene tiempos de ejecución; llegan hasta 140000 milisegundos a medida que la longitud y número de secuencias se incrementan. En la Fig.14, observamos que el algoritmo basado en colonia artificial de abejas sus tiempos de ejecución llegan hasta 25000 milisegundos a medida que la longitud y número de secuencias se incrementan; en la Fig. 15, se puede observar los tiempos de ejecución del algoritmo centro estrella paralelo hasta un tope de 25000 milisegundos.

## 6 Conclusiones y trabajos futuros

En este paper, se ha realizado una comparación de los algoritmos centro estrella y del basado en colonia artificial de abejas en el alineamiento múltiple de secuencias, como muestran los resultados el algoritmo centro estrella tiene menores tiempos de ejecución para pocas secuencias y de longitudes pequeñas, pero a medida que el número de secuencias y longitudes se incrementan, el algoritmo basado en colonia artificial de abejas tiene mejores tiempos.

Los resultados indican que en promedio el algoritmo colonia artificial de abejas es más eficiente a pesar de no ser un algoritmo paralelo.

Siendo el algoritmo de la colonia artificial de abejas fácilmente paralelizable su desempeño se incrementaría de acuerdo con el número de núcleos. Pero al considerar paralelizar este algoritmo habrá que considerar el uso excesivo que se le puede dar al generador de números aleatorios para que cada núcleo pueda tener una semilla diferente y no reste eficiencia en el uso de los múltiples procesadores.

## Referencias bibliográficas

- [1] D. Gusfield, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology, 1st editio. New York, United States, 1997, p. 534.
- [2] J. Xiong, Essential Bioinformatics. New York, United States, 2006, p. 339.
- [3] J. Chand, H. Sharma, and S. Singh, Artificial bee colony algorithm: a survey, Int. J. Advanced Intelligence Paradigms, 2013.
- [4] D. Karaboga, B. Gorkemli, C. Ozturk, y N. Karaboga,, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif Intell Rev. Springer, 2012.
- [5] A. Rekaby, A. Youssif, and A. Sharaf,. Introducing Adaptive Artificial Bee Colony Algorithm and Using it in Solving Traveling Salesman Problem, Science and Information Conference, 2013.
- [6] K. Eldrandaly, M. Hassan and N. AbdelAziz, A Modified Artificial Bee Colony Algorithm for Solving Least-Cost Path Problem in Raster GIS, An International Journal of Applied Mathematics & Information Sciences, 2015.
- [7] Á. Rubio, M. Vega, J. Gómez, and J. Sánchez, A Parallel Multiobjective Artificial Bee Colony

- Algorithm for Dealing with the Traffic Grooming Problem, IEEE 14th International Conference on High Performance Computing and Communications, 2012.
- [8] G. Luo, S. Huang, Y. Chang and S. Yuan, A parallel Bees Algorithm implementation on GPU, Journal of Systems Architecture, 2013.
- [9] S. Santander, M. Vega, J. Gómez, and J. Sánchez., Evaluating the Performance of a Parallel Multiobjective Artificial Bee Colony Algorithm for Inferring Phylogenies on Multicore Architectures, IEEE International Symposium on Parallel and Distributed Processing with Applications, 2010.
- [10] D. González, Metaheurísticas, Optimización Multiobjetivo y Paralelismo para Descubrir Motifs en Secuencias de ADN, España, Universidad de Extremadura, 2013.
- [11] T. Oliver, B. Schmidt, D. Nathan, R. Clemens, and D. Maskell, Multiple Sequence Alignment on an FPGA, Proceedings of the 11th International Conference on Parallel and Distributed Systems, 2005.
- [12] K. Nguyen, Y. Pan, y G. Nong, Parallel Progressive Multiple Sequence Alignment on Reconfigurable Meshes, International Conference on Bioinformatics and Computational Biology. Las Vegas, 2010.
- [13] D. Sundfeld, G. Teodoro, A. Cristina, and M. Melo, Parallel A-Star Multiple Sequence Alignment with Locality-Sensitive Hash Functions, Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), 2015.
- [14] X. Lei, J. Sun, X. Xu and L. Guo, Artificial bee colony for solving multiple sequence alignment, IEEE Fifth International Conference Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010.
- [15] C. Ozturk and S. Aslan, A new artificial bee colony algorithm to solve the multiple sequence alignment problem, Int. J. Data Mining and Bioinformatics, Vol 14, 2016.
- [16] A. Lesk, Introduction to Bioinformatics. New York, United States: Oxford University Press, 2002, p. 283.
- [17] D. Higgins and W. Taylor, Bioinformatics: Sequence, Structure and Databanks, Briefings in Bioinformatics, New York, United States. (2000), Vol. 2, p. 249.
- [18] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids. New York, United States, 1998, p. 356.
- [19] J. Pevsner, Bioinformatics and Functional Genomics, 2nd Editio. New York, United States, 2009, p. 897.
- [20] J. Setubal and J. Meidanis, Introduction to Computational Molecular Biology. Boston, United States, 1997, p. 296.
- [21] J. García, E. Alba, "Algoritmos Basados en Inteligencia Colectiva para la Resolución de Problemas de Bioinformática y Telecomunicaciones", Universidad de Málaga, 2007.
- [22] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8: 687-697, 2008.
- [23] P. Borovska, V. Gancheva and N. Landzhev, "Massively parallel algorithm for multiple biological sequences alignment," *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, Rome, 2013, pp. 638-642.
- [24] A. Freeman, Pro .NET 4 Parallel Programming in C#. Berkeley, CA: Apress, 2010, p. 328.