UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN - TACNA

Escuela de Posgrado

MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA
ADMINISTRACIÓN DE TECNOLOGÍAS DE INFORMACIÓN

COMPARACIÓN DEL ALGORITMO CENTRO ESTRELLA PARALELO CON UNO BASADO EN LA COLONIA ARTIFICIAL DE ABEJAS (ABC) EN EL ALINEAMIENTO MÚLTIPLE DE SECUENCIAS

TESIS

PRESENTADA POR:

WILSON CÉSAR CALLISAYA CHOQUECOTA

Para optar el Grado Académico de:

MAESTRO EN CIENCIAS (*MAGISTER SCIENTIAE*) CON MENCIÓN EN INGENIERÍA DE SISTEMAS E INFORMÁTICA – ADMINISTRACIÓN DE TECNOLOGÍAS DE INFORMACIÓN

TACNA - PERÚ

2018

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

Escuela de Posgrado

MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA
ADMINISTRACIÓN DE TECNOLOGÍAS DE INFORMACIÓN

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

Tesis sustentada y aprobada el 10 de Agosto del 2018; estando el jurado calificador integrado por:

PRESIDENTE : .....
Dr. Edwin Antonio Hinojosa Ramos

SECRETARIO : .....
M.Sc. Edgar Aurelio Taya Acosta

MIEMBRO : .....
Dra. Karin Yanet Supo Gavancho

ASESOR : .....
M.Sc. Hugo Manuel Barraza Vizcarra

Agradecimiento

A Dios

Por haber puesto en mi camino, a las personas que sin dudarlo me ayudaron para consolidar este logro importante

A mi familia

A mi madre Manuela y mi padre Isidro que han estado conmigo en todos los momentos de mi formación, mis hermanos por sus consejos y palabras de aliento

A mis profesores

Por darme siempre consejos e ideas para enrumbarme en el correcto camino para ser un profesional

Dedicatoria

A Dios, ese ser Todopoderoso que nos dio la vida y que siempre nos acompaña, a mis queridos padres Isidro y Manuela, por todo el sacrificio para hacer de mí una persona de bien

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Pág.

RESUMEN

Después de un considerable esfuerzo, en la actualidad, se ha desarrollado 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: Alineamiento múltiple de secuencias, bioinformática, biología computacional, colonia artificial de abejas, programación paralela.

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: Artificial Bee Colony, bioinformatics, computational biology, multiple sequence alignment, parallel programming.

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.

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.

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 costos que implicaba un computador paralelo y a su dificil implementación dado que se tenía que dar 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.

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

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. Descripción del problema

1.1.1. Antecedentes del problema

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:

Santander, S., Vega, M., Gómez, J. y Sánchez, J. (2010) en su trabajo titulado “Evaluating the Performance of a Parallel Multiobjective Artificial Bee Colony Algorithm for Inferring Phylogenies on Multicore Architectures”, indican que una amplia variedad de problemas de optimización requiere la combinación de la computación bioinspirada y la computación paralela para hacer frente a la complejidad necesaria para obtener soluciones óptimas en tiempos reducidos. La era de núcleos múltiples permite al investigador explotar arquitecturas modernas para resolver estos problemas NP-duros. La inferencia de árboles filogenéticos describen la hipótesis de evolución de las especies, este es un problema conocido en el área de biología computacional. Como el espacio de posibles topologías de árboles aumenta exponencialmente con el número de especies, las búsquedas exhaustivas no se pueden aplicar. Además, las dificultades adicionales surgen cuando hay que considerar simultáneamente múltiples medidas con el estado óptimo para resolver el problema. En este trabajo, se 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. Los resultados experimentales muestran que su propuesta se puede modificar a otros enfoques basados en técnicas avanzadas de computación de alto rendimiento en grandes conjuntos de datos.

González, D. (2013) en su tesis doctoral titulada “Metaheurísticas, Optimización Multiobjetivo y Paralelismo para Descubrir Motifs en Secuencias de ADN”, estudia la utilización del paralelismo para tratar de combinar el comportamiento de diferentes algoritmos y así mejorar el rendimiento global de la aplicación resultante. Para ello, tuvo que usar la programación paralela y distribuida OPENMP y MPI respectivamente, diseñando estrategias paralelas que aprovechan las capacidades de varios algoritmos multiobjetivo, mejorando considerablemente la calidad de los resultados obtenidos de una forma sencilla. Estas estrategias las utilizo para la resolución de problema real de búsqueda de patrones de ADN.

Callisaya, W., Callohuari, R., y Jimenez, J. (2013) en el trabajo de “Programación Paralela para el Alineamiento de Secuencias de ADN usando Task Parallel Library (TPL)”, se presenta una implementación paralela para el problema de alineamiento de pares de secuencias usando el C#, modificando el algoritmo original de Needleman Wunsch para su implementación, en el cual se demuestra que la propuesta algorítmica paralela supera a la secuencial.

El uso de los múltiples núcleos del procesador apoya a varios de los investigadores mencionados; pero, uno de los mayores problemas de la biología computacional es el alineamiento de secuencias, existen

diferentes investigaciones para intentar apoyar al problema de alinear múltiples secuencias.

Oliver, T., Schmidt, B., Nathan, D., Clemens, R. y Maskell, D. (2005) en su trabajo titulado “Multiple Sequence Aligment on an FPGA” indican que los biólogos moleculares con frecuencia calculan múltiples alineamientos de secuencias (MSA) para identificar regiones similares en las familias de proteínas, la alineación progresiva es un método ampliamente utilizado para calcular las MSA. Sin embargo, la alineación de unos pocos cientos de secuencias mediante herramientas populares de alineación progresiva requiere varias horas en ordenadores secuenciales. Debido al rápido crecimiento de la secuencia de bases de datos biológica biólogos tienen que calcular MSA en un tiempo mucho más corto. En su trabajo presentan un diferente enfoque para MSA en plataformas de hardware reconfigurable para obtener un alto rendimiento a bajo costo. El uso de este diseño como un bloque de construcción en el que construyeron una matriz sistólica lineal para llevar a cabo un cálculo de distancia de secuencia de pares usando programación dinámica. Sus resultados muestran una aplicación con un ahorro significativo de tiempo de ejecución en un estándar de la FPGA.

Nguyen, K., Pan, Y. y Nong, G. (2010) en su trabajo titulado “Parallel Progressive Multiple Sequence Alignment on Reconfigurable Meshes”, sostienen que una de las tareas más fundamentales y difíciles en la bioinformática es identificar secuencias relacionadas y su significado biológico escondido. El método de la mejor práctica más popular y probada para llevar a cabo esta tarea es alinear múltiples secuencias juntas. Sin embargo, la alineación de secuencias múltiples es una extensa tarea informática. Además, el avance en las técnicas de ADN / ARN y la

secuenciación de proteínas ha creado una gran cantidad de procesos a analizar que excede de la capacidad de los modelos informáticos tradicionales. Por lo tanto, un modelo de alineación de secuencias múltiples en paralelo efectivo capaz de resolver estos problemas está en una gran demanda. En sus resultados diseñan soluciones tanto para la programación dinámica por pares algoritmos de adaptación local y global sobre el modelo de computación reconfigurable malla de tiempo de gestión. Para alinear secuencias m con un máximo de longitud n, que la combinación de las soluciones de programación dinámica por pares paralelos con componentes paralelos de nuevo diseño. También precisan que ello reduce con éxito la complejidad de tiempo de ejecución de la secuencia del algoritmo del alineamiento múltiple progresiva de O (m x n) a O (m) utilizando O (m x n3) unidades para los sistemas de puntuación. La solución general de algoritmo de alineación de secuencias múltiples toma O (m x n4) unidades de procesamiento y completa en el tiempo O (m).

Lalwani, S., Kumar, R. y Gupta, N. (2013) en su trabajo titulado “A Review on Particle Swarm Optimization variants and their applications to Multiple Sequence Alignment”, precisan que la alineación de secuencias múltiples es una de las técnicas más utilizadas en bioinformática para descubrir información funcional, estructural y evolutiva de secuencias biológicas. La alineación de secuencias múltiples es un problema NP-completo y una zona difícil de la bioinformática. Los enfoques clásicos no son eficientes para este tipo de problema debido a la mayor complejidad de tiempo y espacio. En este trabajo también precisan que la optimización de enjambre de partículas es una de las mejores metaheurísticas, analizan las variantes de optimización de enjambre de partículas y sus aplicaciones para la alineación de secuencias múltiples.

Wang, L., Leebens, J., Kerr, P., Beckmann, K., Pamphilis, C. y Warnow, T. (2011), en su obra titulada “The Impact of Multiple Protein Sequence Alignment on Phylogenetic Estimation”, expresan que la alineación de secuencias múltiples es normalmente el primer paso en la estimación de los árboles filogenéticos, con la suposición de que si mejoran las alineaciones mejoran, también lo harán reconstrucciones filogenéticas. Durante la última década, se han desarrollado nuevos métodos de alineación de secuencias múltiples para mejorar los análisis comparativos de estructura de proteínas, pero estos nuevos métodos no han sido utilizados normalmente en los análisis filogenéticos. En el trabajo, nos informan sobre un estudio de simulación que han realizado para evaluar las consecuencias del uso de estos nuevos métodos de alineación de secuencias múltiples en términos de la reconstrucción filogenética resultante. Descubrieron que mientras la precisión de alineación se correlaciona positivamente con exactitud filogenética, la cantidad de mejora en la estimación filogenética resulta de una mejor alineación y que puede variar desde muy pequeño a sustancial. Observaron que la precisión filogenética es altamente correlacionada con la alineación de precisión, cuando las secuencias son más difíciles de alinear, y que la variación en la alineación de precisión puede tener poco impacto en filogenética precisión, cuando las tasas de error de alineación son generalmente bajos. Discuten estas observaciones y las implicaciones para sus futuros trabajos.

Mokaddeml, A y Elloumi, M. (2013) en su trabajo titulado “A Multiple Sequence Alignment Algorithm Based on a New Distance and a New Score Function”, presentan un nuevo algoritmo de alineamiento progresivo, llamado Motalign, para múltiples secuencias de alineación (MSA). Su algoritmo adopta una distancia basada nuevo perfil para comparar dos secuencias biológicas y una nueva función de puntuación, llamada GPSP,

para alinear los perfiles. Ellos comparan los resultados obtenidos por su algoritmo a los obtenidos por otros, como CLUSTALW2, muscular y MAFFT, mediante el uso de la puntuación de la columna (CS) y la suma de parejas consigan (MSF), obteniendo resultados relevantes.

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 se indica en la siguiente publicación.

Sundfeld, D., Teodoro, G., Cristina, A. y Melo, M. (2015) en su trabajo titulado “Parallel A-Star Multiple Sequence Alignment with Locality-Sensitive Hash Functions”, proponen y evalúan una solución paralela para el problema de alineamiento múltiple de secuencias basada en el algoritmo A-Star. En la solución en paralelo, se utiliza una estructura de datos multi-índice, plantillas y una función hash localidad. Los resultados lo recogieron en dos máquinas (4 núcleos y 32 núcleos), con conjuntos de secuencias reales y sintéticas que van desde 3 a 14 secuencias, demostrando que la solución paralela ejecuta 2,89 × 4,77 × y más rápido que una herramienta MSA paralela el estado de la técnica, con un aumento proporcional en el uso de memoria.

La dotación de un planteamiento para el algoritmo de la colonia artificial de abejas para resolver el problema del alineamiento múltiple de secuencias es también ampliamente explorado.

Lei, X., Sun, J., Xu, X. y Guo, L. (2010) en su publicación titulada “Artificial bee colony for solving multiple sequence alignment”, proponen al

algoritmo colonia artificial de abejas (ABC) para resolver el problema de alineación de secuencias múltiples (MSA). Indican que el algoritmo ABC es un enfoque novedoso optimización inspirado por un comportamiento inteligente particular de enjambres de abejas. Tomado el carácter discreto del problema MSA en consideración, se introduce un nuevo método de algoritmo de ABC para determinar una fuente de alimento en el barrio. El rendimiento de su enfoque ABC se compara con otros algoritmos utilizados comúnmente para MSA, los resultados computacionales demuestran la superioridad del nuevo algoritmo de ABC sobre algoritmo genético (GA) y la optimización de enjambre de partículas (PSO) para muchas secuencias con diferente longitud y la identidad. El nuevo enfoque indica que es más sólido y obtiene una mejor calidad matemática y biológica.

Ozturk, C. y Aslan, S. (2016) en su trabajo titulado “A new artificial bee colony algorithm to solve the multiple sequence alignment problem”, señalan que la alineación de tres o más secuencias al mismo tiempo es uno de los problemas más difíciles en la bioinformática. En el trabajo, proponen un nuevo algoritmo de la colonia artificial de abejas (ABC-alineador) para resolver alineación de secuencias múltiples. Múltiples alineamientos obtenidos de ABC-alineador se comparan en términos de la SPS, COFFEE y el SP estándar de las puntuaciones con optimización por enjambre de partículas (PSO), Algoritmos Genéticos (GA) y el algoritmo básico artificial Colonia de abejas (ABC); con la alineación de secuencias de Algoritmos Genéticos (SAGA) y paquetes de software ClustalX; y con nueve herramientas de alineación muy conocidos que incluyen CLUSTALW, CLUSTAL OMEGA, DIALIGN-TX, MAFFT, músculo, POA, Probalign, ProbCons y T-COFFEE, sobre las secuencias extraídas de la BAliBASE 1.0, 3D_ali y BAliBASE 3.0 bancos de datos de referencia, respectivamente. A partir de los resultados de la simulación, concluyeron

que el algoritmo propuesto ABC-alineador supera a los otros meta-heurísticos basados en la población y obtienen mejores puntajes que los paquetes de software utilizados en los experimentos sin requerir ninguna información adicional, o aplicación de procedimientos complejos.

Borovska, P., Gancheva, V., Markov, S., Georgiev, I. y Asenov, E. (2011) en su trabajo titulado “Parallel Performance Evaluation and Profiling of Multiple Sequence Nucleotide Alignment on the Supercomputer BlueGene/P”, dan a conocer que la alineación de secuencias múltiples es un método importante en el análisis de ADN y las proteínas. ClustalW se ha convertido en la herramienta más popular e implementa un método progresivo para la alineación de secuencias múltiples. El objetivo del estudio fue proponer la evaluación del desempeño de la eficiencia de múltiples alineación paralela en el superordenador Blue Gene / P para el estudio de caso de la investigación de las secuencias de nucleótidos virales y encontrar el consenso motivos y dominios variables en los diferentes segmentos del genoma del virus de la gripe. La evaluación del desempeño paralelo y perfiles de alineación de secuencias múltiples fueron realizados sobre la base de la aplicación del programa paralelo que se basa en el método ClustalW y una base de datos de réplica local de todos los aislamientos disponibles de los ocho segmentos del virus de la gripe A. Se realizaron pruebas paralelas con parámetros de rendimiento, tiempo de ejecución, el tiempo de aceleración y perfiles que se estimaron experimentalmente. La estimación de rendimiento y de perfiles de análisis demostró que el sistema paralelo está bien equilibrado debido a la actuación de la distribución de datos a todos los demás procesadores así como la comunicación y la sincronización.

1.1.2. Problemática de la investigación

En el mundo de la bioinformática el alineamiento de secuencias es usado para la identificación de las diferentes secuencias genómicas. Este es el principal problema de la biología computacional, el tiempo de respuesta para el alineamiento de secuencias ha sido el mayor problema.

En el Perú la bioinformática está en sus comienzos y en lo que respecta a biología computacional poco se hace, hay un desconocimiento de los profesionales de esta área de la utilidad de aprovechar las técnicas algorítmicas como inteligencia de enjambre o algoritmos paralelos.

En la actualidad, en el campo de la bioinformática se ha iniciado un esfuerzo para desarrollar algoritmos que comparan secuencias biomoleculares. El alineamiento múltiple es el proceso que aporta mayor información biológica. Uno de los algoritmos de alineamiento múltiple es el centro estrella, que en su proceso usa el alineamiento de pares de Needleman-Wunsch que determina el alineamiento óptimo de varias secuencias, sabemos que el uso de la programación paralela disminuye el tiempo de ejecución de este algoritmo, conocemos también que los algoritmos de inteligencia de enjambre son ampliamente usados para resolver problemas de optimización en particular el algoritmo de la colonia artificial de abejas. Siendo el alineamiento múltiple un problema de optimización, es posible emplear este algoritmo bioinspirado en este campo.

1.2. Formulación del problema

En el momento de decidir qué algoritmo usar para realizar un alineamiento, se seleccionará el que ofrezca el menor tiempo de respuesta, pero para resolver este problema el número de secuencias y la longitud de las mismas puede variar los tiempos de respuesta. A su vez, los diferentes autores ofrecen una pobre información sobre su funcionamiento que permita implementarlo. Siendo el algoritmo de colonia artificial de abejas una propuesta usada en diferentes ámbitos y considerando el algoritmo centro estrella como uno de los primeros algoritmos que se usaron para resolver el problema del alineamiento múltiple de secuencias. Lo señalado, plantea las siguientes interrogantes.

Problema general:

- ¿Qué algoritmo tiene un menor tiempo de respuesta al comparar el algoritmo centro estrella paralelo y uno basado en la colonia artificial de abejas (ABC) en el alineamiento múltiple de secuencias?

Problemas específicos:

- ¿Qué tiempos de respuesta tiene un algoritmo centro estrella paralelo en el alineamiento múltiple de secuencias?
- ¿Cómo paralelizar el algoritmo centro estrella para el alineamiento múltiple de secuencias?
- ¿Cómo es el funcionamiento del algoritmo centro estrella para el alineamiento múltiple de secuencias?
- ¿Qué tiempos de respuesta tiene un algoritmo basado en la colonia artificial de abejas (ABC) en el alineamiento múltiple de secuencias?
- ¿Cómo usar el algoritmo basado en colonia artificial de abejas (ABC) para el alineamiento múltiple de secuencias?

1.3. Justificación e importancia de la investigación

La alineación de secuencias para la bioingeniería es requerida para el desarrollo de nuevos fármacos así como para estudios genéticos.

El genoma humano tiene 3 200 000 000 millones de pares de bases, actualmente es imposible un alineamiento en un tiempo aceptable. Es por ello, que se requiere de nuevos métodos que puedan incrementar el tiempo de respuesta, existen varias alternativas computacionales que se pueden optar existiendo técnicas bioinspiradas como las de inteligencia de enjambre; en este caso particular, el de la colonia artificial de abejas, también existe la alternativa de programación paralela ambos pueden reducir el tiempo de respuesta de este problema.

Para poder realizar este proyecto será necesario usar tecnologías ya diseñadas previamente como la librería TPL, con la cual se desarrollará el problema de adaptar el algoritmo estrella en su versión paralela, también se implementará el algoritmo bioinspirado, de la colonia artificial de abejas para resolver el problema del alineamiento múltiple de secuencias y que servirán de base para resolver otros problemas similares.

La selección del algoritmo centro estrella para esta comparación, fue realizada porque es el primer algoritmo que resuelve el problema del alineamiento múltiple y en el caso del algoritmo de inteligencia de enjambre *artificial bee colony* planteado por Karaboga el 2005, ha sido porque es muy usado para resolver problemas de optimización siendo este último más actual; pero, poco se ha utilizado este algoritmo para resolver el problema del alineamiento múltiple. En el presente trabajo se presenta una adaptación detallada del algoritmo *artificial bee colony* en el alineamiento múltiple de secuencias.

Existe la necesidad de documentación de biología computacional, programación paralela, y de algoritmos bioinspirados de inteligencia

colectiva como de la colonia artificial de abejas. La mayoría de investigaciones no dan detalles de una implementación para dichos algoritmos y una descripción en pseudocódigo ayudaría a ese fin, siendo este proyecto de gran valor académico para futuras investigaciones en estas áreas, a fin de plantear nuevas propuestas de trabajo.

1.4. Alcances y limitaciones

Alcances

La presente tesis se enfocará en la medición de algoritmos según su tiempo de ejecución y puntaje obtenido en el escenario de un computador personal que posea 4 núcleos.

Limitaciones

- Bibliográfica: Existe escasa bibliografía en las bibliotecas locales. Se superó dicha limitación recurriendo a bibliografía de otros países.
- Área geográfica: No existen centros de investigación de bioinformática en la localidad, lo que impide realizar consultas a investigadores. Se superó dicha limitación contactando con personas entendidas en el área y asistiendo a congresos.
- Financiamiento: Es necesario contar con computadores de alto rendimiento para efectuar las pruebas necesarias. Se tuvo que autofinanciar dicho computador para realizar el experimento.
- Tiempo disponible: El tiempo es reducido al estar laborando.

1.5. Objetivos

1.5.1. Objetivo general

Comparar los tiempos de respuesta del algoritmo centro estrella paralelo con uno basado en la colonia artificial de abejas (ABC) en el alineamiento múltiple de secuencias.

1.5.2. Objetivos específicos

- Determinar los tiempos de respuesta del algoritmo centro estrella paralelo en el alineamiento múltiple de secuencias.
- Paralelizar el algoritmo centro estrella para el alineamiento múltiple de secuencias.
- Conocer el funcionamiento del algoritmo centro estrella para el alineamiento múltiple de secuencias.
- Determinar los tiempos de respuesta de un algoritmo basado en colonia artificial de abejas (ABC) en el alineamiento múltiple de secuencias.
- Usar el algoritmo basado en colonia artificial de abejas (ABC) para el alineamiento múltiple de secuencias.

1.6. Hipótesis

- Ho: El tiempo de respuesta del algoritmo centro estrella paralelo es menor comparado con uno basado en colonia artificial de abejas (ABC) en el alineamiento múltiples de secuencias.
- H1: El tiempo de respuesta del algoritmo estrella paralelo es mayor comparado con uno basado en colonia artificial de abejas (ABC) en el alineamiento múltiples de secuencias.

CAPÍTULO II

MARCO TEÓRICO

2.1. Antecedentes del estudio

Varios investigadores se refieren al uso de la colonia artificial de abejas mostrando la importancia de su estudio como algoritmo bioinspirado, entre ellos, se puede mencionar:

Karaboga, D., Gorkemli, B., Ozturk, C. y Karaboga, N. (2012) en su trabajo titulado: “A comprehensive survey: artificial bee colony (ABC) algorithm and apllications”, indican que la inteligencia de enjambre se define brevemente como el comportamiento colectivo de los enjambres descentralizados y auto-organizados. Los ejemplos bien conocidos de estos enjambres son bandadas de aves, los bancos de pesca y la colonia de insectos sociales tales como termitas, hormigas y las abejas. Aunque las características de auto-organización son requeridas por la inteligencia de enjambre son vistos con fuerza y claridad en las colonias de abejas de miel, los investigadores han comenzado recientemente a interesarse en el comportamiento de estos sistemas de enjambre para describir nuevos enfoques inteligentes, sobre todo desde el comienzo de los años 2000.

Durante una década, varios algoritmos se han desarrollado en función de los diferentes comportamientos inteligentes de enjambres de abejas. También precisan que la colonia de abejas artificial (ABC) es el que ha sido más ampliamente estudiado y aplicado para resolver los problemas del mundo real, hasta el momento. Últimamente, el número de investigadores que se interesan en ABC algoritmo está aumentando. El trabajo presenta un estudio exhaustivo de los avances con ABC y sus aplicaciones hasta esa fecha. Los autores esperan que su trabajo sea muy beneficioso para los investigadores que estudian la inteligencia de enjambre en particular algoritmo ABC.

Chand, J., Sharma, H. y Singh, S. (2013) en su trabajo titulado “Artificial bee colony algorithm: a survey”, precisan que con la creciente complejidad de los problemas de optimización del mundo real, la demanda de los optimizadores robustos, rápidos y precisos va en aumento entre los investigadores de diversos campos. El algoritmo de optimización ABC es relativamente un enfoque probabilístico basado en la población reciente y sencillo para la optimización global sobre los espacios continuos y discretos. Según los informes que ha superado a varias heurísticas de búsqueda cuando se ensaya sobre ambos problemas y de referencia en el mundo real a pesar de que utiliza menos parámetros de control y que puede ser utilizado de manera eficiente para resolver problemas de optimización

multimodales y multidimensionales. 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. ABC, al igual que otros algoritmos de optimización probabilísticos, tiene inconveniente inherente de convergencia prematura o estancamiento que lleva a la pérdida de la capacidad de exploración y explotación de ABC.

Rekaby, A., Youssif, A. y Sharaf, A. (2013) en su trabajo titulado “Introducing Adaptive Artificial Bee Colony Algorithm and Using it in Solving Traveling Salesman Problem”, detallan que el algoritmo de colonia artificial de abejas es un moderno algoritmo de inteligencia de enjambre. El documento propone una versión modificada del algoritmo de colonia de abejas artificial llamado "Colonia de abejas artificial adaptativa" (AABC). Comparan entre el algoritmo de colonia de abejas estándar y el algoritmo de colonia de abejas adaptativo propuesto a través problema del vendedor ambulante viajero (*Travelling Salesman Problem*). Problema del vendedor viajero es uno de los problemas más comunes en la evaluación de técnicas de búsqueda, por lo que el documento lo considera como un caso experimental para la discriminación el rendimiento de los algoritmos, estos experimentos fueron repetidos en diferentes puntos de referencia. El algoritmo de colonia de abejas artificial adaptativa propuesto presentó mayor eficiencia que el algoritmo estándar de la colonia de abejas artificial.

El valor final de la aptitud solución se ve reforzada por alrededor de 8% en el algoritmo de colonia de abejas artificial adaptativa en comparación con la solución de algoritmo de colonia de abejas artificial estándar.

Eldrandaly, K., Hassan, M. y AbdelAziz, N. (2015) en su trabajo titulado “A Modified Artificial Bee Colony Algorithm for Solving Least-Cost Path Problem in Raster GIS”, nos refieren sobre la aplicación del algoritmo de la colonia artificial de abejas para un problema de los sistemas de información geográfica (SIG raster), que consiste en encontrar la ruta con el menor coste acumulado entre dos ubicaciones en un modelo de trama de una superficie de coste. En este trabajo se presenta un algoritmo de colonia de abejas artificial modificado (ABC) para la solución de costo mínimo problema del camino en un SIG basado en raster. Esta modificación incluye la incorporación de una característica distintiva que no está presente en el ABC clásica. Un nuevo componente, el método de búsqueda de orientación de dirección, se utiliza para guiar una abeja a caminar hacia el destino final de manera más eficiente. Además, este artículo examina cómo la calidad de los caminos basados en la trama puede ser mejorada mediante el uso de patrones de conectividad de mayor tamaño. Los resultados experimentales muestran que el rendimiento del modelo ABC modificado es bastante cerca de algoritmo de Dijkstra, mientras que su complejidad y la solución computacional tiempo es mucho menor que el

algoritmo de Dijkstra. Los resultados también indican que las rutas a base de trama se pueden mejorar mediante el uso de patrones de conectividad de mayor tamaño.

Rubio, Á., Vega, M., Gómez, J. y Sánchez, J. (2012) en su trabajo titulado “A Parallel Multiobjective Artificial Bee Colony Algorithm for Dealing with the Traffic Grooming Problem”, presentan un enfoque multiobjetivo paralelo nuevo 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. El problema del tráfico en el cuidado de redes ópticas de malla es un problema NP-duro, por lo que el uso de metaheurísticas y el paralelismo de forma conjunta para aumentar el rendimiento de la red es una gran opción con el fin de reducir los tiempos de ejecución. El enfoque multiobjetivo paralelo se implementa mediante el uso de OpenMP. Ellos midieron el aumento de velocidad y la eficiencia obtenida para su enfoque paralelo con 2, 4, 8, y 16 núcleos. Teniendo resultados numéricos eficientes que son reportados en la fase experimental que lo llevaron a cabo en dos redes ópticas. Por último, presentan un estudio comparativo con métodos tradicionales; en el que muestran que el uso de la inteligencia de enjambre supera a los resultados previos en trabajos similares.

Aguilar, A. (2014) en su tesis titulada “Un algoritmo basado en la colonia artificial de abejas con búsqueda local para resolver problemas de optimización con restricciones”, describe una propuesta de algoritmo memético basado ABC para problemas de optimización con restricciones, el algoritmo propuesto fue probado con dos métodos de búsqueda local, Hooke-Jeeves y Complex, estos son incorporados dentro del ciclo evolutivo de ABC, particularmente después de la fase de abejas observadoras. El autor utilizó una métrica basada en la aptitud de la población actual para determinar el momento para aplicar la búsqueda local. Los resultados mostraron que la propuesta es competitiva resolviendo problemas de ese tipo.

Luo, G., Huang, S., Chang, Y. y Yuan, S. (2013) en su trabajo titulado “A parallel Bees Algorithm implementation on GPU” señalan sobre el uso de la GPU (Unidad de procesamiento de gráficos) para utilizar el algoritmo de las abejas, precisando que este algoritmo ofrece una solución casi óptima para el problema de búsqueda. En su trabajo deciden aplicar este algoritmo en CUDA (Compute Unified Device Architecture) y generando CUBA (CUDA basado en el Bee Algorithm), evalúan el rendimiento de CUBA mediante la realización de algunos experimentos sobre la base de numerosos problemas de optimización famosos. Los resultados muestran que CUBA supera significativamente al algoritmo de las abejas estándar en

numerosos y diferentes problemas de optimización. El mecanismo de comunicación entre las colonias en el mismo bloque es también punto importante para disminuir el tiempo en su algoritmo, y se eligió la comunicación de dos fases para un mejor resultado.

2.2. Bases teóricas

2.2.1. La importancia de la comparación de secuencias

El primer hecho de análisis de secuencias biológicas en las secuencias biomoleculares (ADN, ARN, o secuencias de aminoácidos), es la alta similitud de secuencias que por lo general implica similitud funcional o estructural significativa. Con el desarrollo de métodos rápidos para la comparación de secuencias, tanto con algoritmos heurísticos y potentes ordenadores paralelos, el descubrimiento basado únicamente en la homología de secuencia se ha convertido en rutina para los biólogos. La comparación de secuencias a gran escala, por lo general organizada como búsqueda de base de datos, es una herramienta muy poderosa para la inferencia biológica en la biología molecular moderna. Y esa herramienta es usada casi universalmente por los biólogos moleculares. Actualmente a nadie se le ocurriría publicar la secuencia de un gen clonado nuevo sin hacer una búsqueda en las de bases de datos (Gusfield, 1997, pp. 213–214).

Hoy en día, el procedimiento de rutina para el análisis de una nueva secuencia de la proteína casi siempre comienza con una comparación de la secuencia de la mano con las secuencias en una o más de las principales

bases de datos de secuencias. Una nueva secuencia se analiza mediante la extrapolación de las propiedades de sus 'Vecinos' en una búsqueda de base de datos. Estos métodos se han aplicado durante las últimas tres décadas con mucho éxito y han ayudado a identificar la función biológica de muchas secuencias de proteínas, así como para revelar muchas relaciones distantes e interesantes entre las familias de proteínas. En realidad, más secuencias han sido supuestamente caracterizadas por la búsqueda de bases de datos que los de cualquier otra tecnología. (Higgins y Taylor, 2000, pp. 167–168)

> Las cuestiones clave a tomar en cuenta al alinear son:
> (1) ¿Qué tipo de alineación se deben considerar?
> (2) El sistema de puntuación utilizado para clasificar las alineaciones
> (3) El algoritmo usado para encontrar una alineación con puntuación óptima.
> (4) Los métodos estadísticos utilizado para evaluar la importancia de una puntuación de alineamiento.
> (Durbin, Eddy, Krogh y Mitchison, 1998, p. 12)

2.2.2. Alineamiento de secuencias:

El análisis comparativo de secuencias es el primer paso para estudiar la relación de secuencia estructura-función de proteínas y secuencias de nucleótidos. La comparación de la secuencia de una determinada proteína con los de las bases de datos de proteínas anotadas a menudo genera pistas importantes sobre la estructura 3D de la proteína y su posible función. Cuando se trata de la búsqueda de secuencias relacionadas funcionalmente a menudo es beneficioso para buscar pequeñas partes de las secuencias que son similares entre sí. Esta

búsqueda consiste en hacer un alineamiento por pares de la secuencia de consulta a todas las secuencias en la base de datos, y ordenar las alineaciones resultantes de la puntuación de alineamiento. Los éxitos son algunas de las mejores secuencias de la base de datos que tienen una alineación de puntuación con la secuencia de consulta.

Si se logra generar un algoritmo eficiente, se obtendrá una buena posición para ir a pescar en los bancos de datos de secuencias relacionadas. Este análisis se complica al considerar los GAPS (huecos). En la figura 1, se aprecia ejemplos de alineamiento (Lesk, 2002, p. 154).

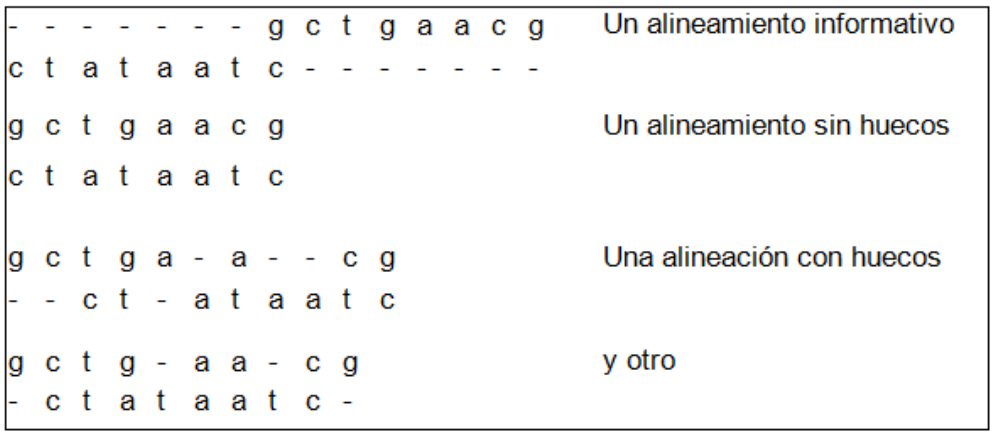

Figura 1: Ejemplos de alineamiento

Fuente: Lesk (2002)

2.2.3. Alineando con sistema de puntuación

Supongamos que una secuencia de ADN a, ha evolucionado a la secuencia de b a través de sustituciones, inserciones y deleciones. Por ejemplo, dicen que a = ACTTGA y b se obtiene mediante la sustitución de la segunda de base de C a G, la inserción de una A, entre la segunda y la

tercera base, y mediante la supresión de la quinta base G. La alineación correspondiente se muestra en la figura 2.

```
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

Fuente: Higgins y Taylor (2000)

Por ejemplo, en la alineación anterior, una puntuación de 1 se le dio a cada identidad, una puntuación de 0 se le dio a cada sustitución, y una puntuación negativa de -1 fue dada para cada espacio. En general, las calificaciones de las identidades y las sustituciones que se utilizan para marcar la alineación se llaman la matriz de puntuación, y las calificaciones de los huecos (Gaps) se llaman puntuación de sanciones. En conjunto se les llama el sistema de puntuación, con altas calificaciones (positivo) para las identidades, y las puntuaciones bajas o negativas para las sustituciones y los huecos, la estrategia básica para trazar la alineación correcta busca la alineación que obtuvo el mejor resultado (Higgins & Taylor, 2000, pp. 167–168).

2.2.4. Métodos de alineamiento de pares de secuencias

La alineación de dos secuencias se realiza utilizando los siguientes métodos:

a) Análisis de matriz de puntos

b) El algoritmo de programación dinámica (o DP)

c) Word o k-tupla métodos, tales como el utilizado por los programas FASTA y BLAST.

El método de programación dinámica, se utiliza para alineación global de secuencias de Needleman y Wunsch (1970) y garantiza en un sentido matemático proporcionar la alineación óptima (mejor o más alta puntuación), incluyendo la elección de la matriz de puntuación y la brecha de sanciones. El método de programación dinámica también puede ser lento, debido a la cantidad de etapas de cálculo, que aumentan aproximadamente como el cuadrado o el cubo de las longitudes de secuencia. El requisito de memoria de la computadora también aumenta con el cuadrado de la longitud de la secuencia. Por lo tanto, es difícil utilizar el método para secuencias muy largas (Mount, 2001, pp. 53–57).

2.2.5. Programación dinámica para el alineamiento de secuencias

Los algoritmos de programación dinámica nos ayudan a resolver problemas de optimización, los problemas en los que hay un gran número de posibles soluciones, pero sólo uno o un pequeño número de las mejores soluciones. Un algoritmo de programación dinámica encuentra la mejor solución porque divide el problema original en subproblemas más pequeños. El algoritmo de Needleman Wunsch utiliza una matriz de puntuaciones (mxn) en la que m y n son las longitudes de las secuencias que están siendo alineadas. A partir de la alineación de un gap contra sí misma (que se le otorga la puntuación inicial cero), el algoritmo se llena en la matriz de una fila a la vez. En cada posición en la matriz el algoritmo calcula las puntuaciones que dan como resultado para cada una de sus tres opciones, selecciona la que proporciona la puntuación más alta, a continuación, almacena un puntero en la posición actual a la posición

anterior que se utilizó para llegar a la alta puntuación. Cuando todas las posiciones en la matriz ha sido rellenado, se realiza una etapa de rastreo (Gibas y Reilly, 2001, pp. 157–158).

Ejemplo de un alineamiento usando la función de la ecuación 1 se muestra en la figura 3, y resultado de la alineación en la figura 4.

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

		R	H	E	E	I	I	I	K	V	F	F	I
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80	-88	-96
H	-8	0	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
H	-16	-8	8	0	-8	-16	-24	-32	-40	-48	-56	-64	-72
Q	-24	-15	0	10	2	-6	-14	-22	-30	-38	-46	-54	-62
K	-32	-22	-8	2	11	3	-5	-13	-17	-25	-33	-41	-49
L	-40	-30	-16	-6	3	13	5	-3	-11	-16	-24	-32	-39
V	-48	-38	-24	-14	-5	6	16	8	0	-7	-15	-23	-29
F	-56	-46	-32	-22	-13	-2	8	16	8	0	-1	-9	-17
F	-64	-54	-40	-30	-21	-10	0	8	13	7	6	5	-3

Figura 3: Alineamiento global de dos secuencias (RHEEIIIKVFI, HHQKLVFF) y su respectivo Traceback mostrado en oscuro

Fuente: Venkatarajan y Pandjassarame (2009)

RHEEIIIKVFFI

HHQKL---VFF-

Figura 4**:** Resultado del alineamiento de dos secuencias (RHEEIIIKVFI, HHQKLVFF)

Fuente: Venkatarajan y Pandjassarame (2009)

Por lo tanto, los datos de entrada y salida para el algoritmo corresponden a:

Entrada: Cadenas v y w, y una matriz de puntuación δ

Salida: Un alineamiento de v y w cuyo puntaje es definido por la matriz de puntuación δ, es el máximo de todos los posibles alineamientos de v y w (Jones y Pevzner, 2004, p. 177).

El alineamiento se realiza en 3 pasos como indica Pevsner: paso 1 inicialización de la matriz, paso 2 llenado de la matriz de Scores, paso 3 identificación del Alineamiento (Traceback).

Las figuras 5 y 6 detallan todo este proceso paso a paso (Pevsner, 2009, pp. 76–80).

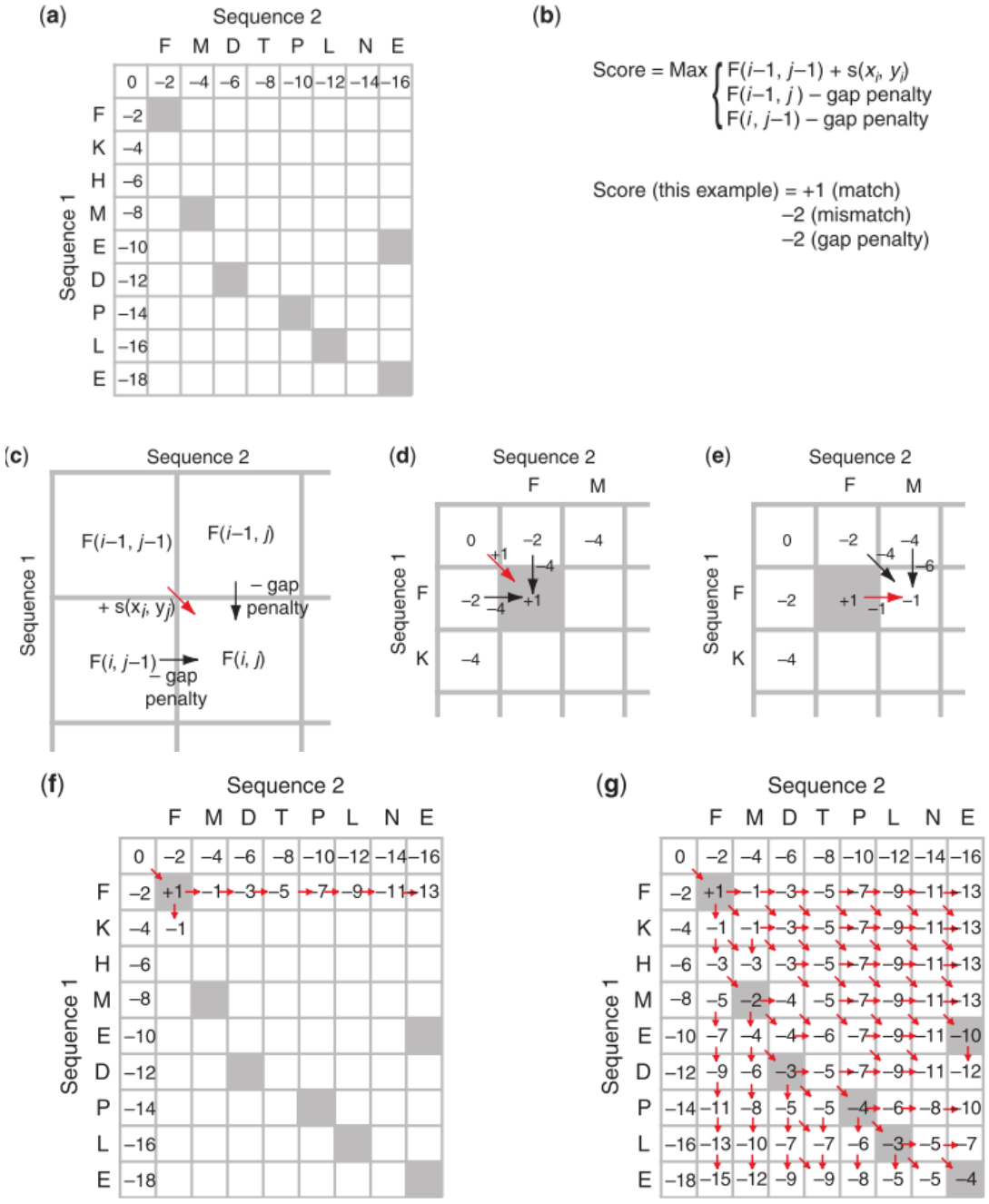

Figura 5: Procedimiento para los pasos de inicialización y llenado de la matriz

Fuente: Pevsner (2009)

En la figura 5 se muestra en (a) inicialización en (b) función a usar en la matriz y el puntaje respectivo de su funcion de similitud en (c) el llenado de cada celda depende de las posiciones superior, izquierda y diagonal, en (d) cálculo de la celda de la segunda fila y segunda columna en (e) cálculo de la celda de segunda fila y tercera columna en (f) llenado de la segunda fila y en (g) matriz de scores llenas con la función de similitud respectiva.

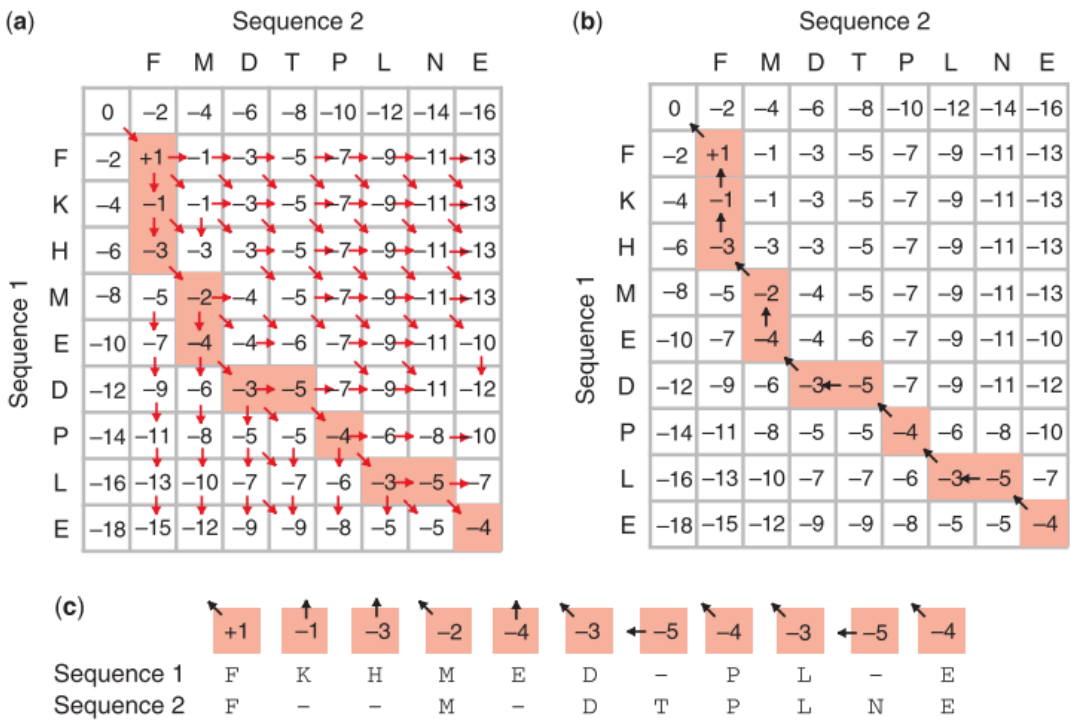

Figura 6: Paso número tres identificación del alineamiento (Traceback)

Fuente: Pevsner (2009)

La figura 6 (a) nos muestra el origen de los óptimos puntajes de cada celda, (b) el recorrido del óptimo puntaje para el alineamiento global, (c) las secuencias alineadas.

El pseudocódigo respectivo para los pasos 1 y 2 del alineamiento global se observa en seguidamente, y a continuación el paso 3 de Traceback.

```
Algorithm Similarity
    input: sequences s and t
    output: similarity between s and t
    m ← |s|
    n ← |t|
    for i←0 m do
    a[0,j] ← j x g
    for i ← 1 to m do
    for j ← 1 to n do
        a[i,j] ← max(a[i –I, j] + g,
                     a[i-I,j-1]+p(i,j),
                     a[i, j-1] + g)
    return a[m,n]
```

Pseudocódigo básico de la programación dinámica para la comparación de dos secuencias

Fuente: (Setubal y Meidanis, 1997, p. 52)

```
Algorithm Align
        input: indices I, j, array a given by algorithm Similarity
        output: aligment in align-s, align-t, and length in len
        if i=0 and j=0 then
                len ← 0
        else if > 0 and a[i,j] =a[i-1,j] + g then
                Align(i-1, j, len)
                len ← len +1
                align-s[len] ←s[i]
                align-t[len] ←  -
        else if > 0 and j > 0 and a[i, j] = a[i-1, j-1] + p(i,j) then
                Align(i-1,j-1,len)
                len ← len +1
                align-s[len] ← s[i]
                align-t[len] ← t[j]
        else // has to be j >0 and a[i,j] = a[I, j-1] + g
                Align(i, j-1, len)
                len ← len +1
                align-s[len] ← -
                align-t[len] ← r[j]
```

Pseudocódigo del algoritmo de traceback (Identificación del alineamiento).
Fuente: (Setubal y Meidanis, 1997, p. 53)

2.2.6. Alineamiento múltiple de secuencias

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. Por ejemplo, permite la identificación de patrones de secuencias conservadas y motivos de toda la secuencia, que no son obvias para detectar mediante la comparación de sólo dos secuencias. Muchos residuos de aminoácidos

conservados y funcionalmente críticos pueden ser identificados en un alineamiento múltiple de proteínas. La alineación de secuencias múltiples es también un prerrequisito esencial para llevar a cabo el análisis filogenético de las familias de secuencias y predicción de estructuras secundarias y terciarias de proteínas. En teoría, 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. Como consecuencia de ello, la programación dinámica completa no se puede aplicar para conjuntos de datos de más de diez secuencias. En la práctica, se utilizan con mayor frecuencia los enfoques heurísticos (Xiong, 2006, p. 63).

2.2.7. El puntaje en alineamientos múltiples 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 ecuación 2 (Setubal & Meidanis, 1997, p. 70).

```
MQPILLLV
MLR-LL--
MK-ILLL-
MPPVLILV
```

$$\begin{aligned} SP(I,\text{-},I,V) &= sc(I,\text{-}) + sc(I,I) + sc(I,V) + \\ &\quad sc(\text{-},I) + sc(\text{-},V) + sc(I,V) \\ &= -2 + 1 - 1 - 2 - 2 - 1 \\ &= -7 \end{aligned}$$

[2]

2.2.8. 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 figura 7 (Setubal & Meidanis, 1997, p. 70).

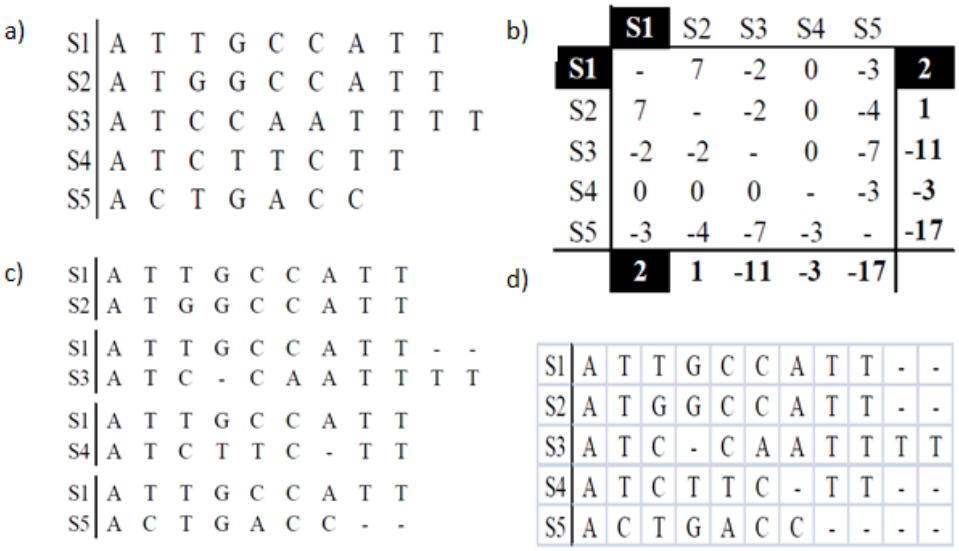

Figura 7: Ejemplo de uso para alinear con el algoritmo centro estrella

Fuente: (Setubal & Meidanis, 1997, p. 78)

En la figura 7 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 (Setubal y Meidanis, 1997, p. 78).

2.2.9. El procesamiento paralelo

El propósito principal de procesamiento en paralelo es realizar cálculos más rápidos que se puedan hacer con un único procesador mediante el uso de un número de procesadores al mismo tiempo. Un computador paralelo es simplemente una colección de procesadores,

normalmente del mismo tipo, interconectados de cierta manera para que la coordinación de sus actividades y el intercambio de datos. ¿Cómo se debe evaluar un algoritmo para su conveniencia para el procesamiento en paralelo? Al igual que en el caso de los algoritmos secuenciales, hay varios criterios importantes, como el rendimiento del tiempo, la utilización del espacio, y capacidad de programación. La situación de los algoritmos paralelos es más complicada debido a la presencia de parámetros adicionales, tales como el número de procesadores, las capacidades de las memorias locales, el esquema de comunicación, y los protocolos de sincronización (JáJá, 1997, pp. 2–3).

Actualmente, es rutinario tener procesadores con muchos núcleos y, posiblemente, la capacidad de ejecutar múltiples flujos de instrucciones dentro de cada núcleo. En otras palabras, la tecnología multinúcleo será la corriente principal. Es de vital importancia que las futuras aplicaciones puedan hacer uso efectivo del paralelismo que está presente en nuestro hardware. Pero, a pesar de importantes avances en la tecnología de compilación, tendrá el programador que ayudar, mediante la descripción de la concurrencia que se encuentra en los códigos de la aplicación (Chapman, Jost, & Van der Pas, 2008, pp. 1–3).

2.2.10. Niveles de paralelismo

Algoritmos y arquitecturas multiprocesador están estrechamente ligados. No podemos pensar en un algoritmo paralelo sin considerar el hardware paralelo que lo va a apoyar. Por el contrario, no podemos pensar en hardware paralelo sin pensar en el software paralelo que lo conduce. El paralelismo se puede implementar en diferentes niveles en un sistema informático usando técnicas de hardware y software:

- Paralelismo a nivel de datos, en los que operamos simultáneamente en varios trozos de un dato o en varios datos. Ejemplos, son la adición de bits en paralelo, multiplicación y división de números binarios, procesadores de vectores y matrices sistólicas para hacer frente a varias muestras de datos.
- Paralelismo a nivel de instrucción (ILP), en la que al mismo tiempo ejecutar más de una instrucción por parte del procesador. Un ejemplo, es el uso de la canalización de instrucciones.
- Paralelismo a nivel de hilo (TLP), un hilo es una parte de un programa que comparte recursos de procesador con otros hilos. Un hilo a veces se llama un proceso ligero. En TLP, múltiples subprocesos de software se ejecutan simultáneamente en un procesador o varios procesadores.
- Paralelismo a nivel de proceso, un proceso es un programa que se ejecuta en el equipo. Un proceso de reserva de sus propios recursos de la computadora, como espacio de memoria y registros. En este nivel varios programas se están ejecutando de forma simultánea sobre uno máquina o en varias máquinas.

La más famosa taxonomía procesador fue propuesto por Flynn sobre la base de los datos y las operaciones realizadas en estos datos:

- Único flujo de datos simple instrucción (SISD), este es el caso del procesador único.
- Instrucción simple flujo de datos múltiples (SIMD), todos los procesadores ejecutan la misma instrucción en diferentes datos. Cada procesador tiene sus propios datos en una memoria local, y los datos de cambio procesadores entre sí a través de esquemas de comunicación típicamente simples. Muchas aplicaciones científicas y de ingeniería se prestan para el procesamiento en paralelo con este esquema. Ejemplos de tales aplicaciones incluyen el procesamiento de

gráficos, compresión de vídeo, análisis de imágenes médicas, y así sucesivamente.

- Único flujo de datos múltiple instrucción (MISD), se podría argumentar que las redes neuronales y máquinas de flujo de datos son ejemplos de este tipo de procesadores en paralelo.
- Instrucción de flujo de datos múltiple Múltiple (MIMD), cada procesador está ejecutando sus propias instrucciones sobre sus datos locales. Ejemplos de tales procesadores paralelos son procesadores multinúcleo y multihilo multiprocesadores en general (Smith, 1993, pp. 14–16).

2.2.11. La clase Parallel

El “for y bucles foreach" son piedras angulares del desarrollo C #, y no toma mucho tiempo para un nuevo programador llegar al punto donde se convierten en una herramienta de uso frecuente.

El TPL contiene soporte para bucles paralelos, es decir, bucles en la que las iteraciones se realizan con un cierto grado de concurrencia. Se puede ver un ejemplo sencillo en el que se puede calcular los cuadrados de 100 valores enteros. Comenzando con bucles básicos y trabajar hasta más complejas permutaciones. Una vez que se haya dominado, podemos encontrar los bucles paralelos convenientes, fáciles de usar y a menudo más simple que las tareas para algunos tipos de problemas de programación.

La clase Parallel que nos brinda los bucles paralelos son posibles gracias a la clase System.Threading.Tasks.Parallel y esta clase proporciona métodos que se ocupan de crear administrar y coordinar tareas (Freeman, 2010, pp. 173–175).

2.3. Definición de términos

ALGORITMO: Conjunto de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

ALINEAMIENTO DE SECUENCIAS: Proceso mediante el cual se buscan las similitudes de dos secuencias de caracteres.

ALINEAMIENTO GLOBAL: Proceso por el cual, se realiza un alineamiento de todos los caracteres de dos secuencias.

MSA (Multiple Sequence Aligment): Alineamiento de múltiples secuencias, es un método de análisis fundamental usado en la bioinformática, el tiempo de cómputo crece exponencialmente con respecto al número de secuencias.

BIOINFORMÁTICA: Usa de herramientas computacionales para ayudar a resolver problemas biológicos.

BIOLOGÍA COMPUTACIONAL: Es el uso de algoritmos y ordenadores para facilitar el entendimiento de problemas biológicos. La biología computacional abarca varios campos ya establecidos: química, bioquímica, matemáticas, ingeniería de sistemas, física, estadísticas, etc.

PROGRAMACIÓN DINÁMICA: Es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas.

PROGRAMACIÓN PARALELA: Es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente (en paralelo).

SIMD: Simple instrucción y múltiples datos, refiere que una orden es aplicada a muchos datos en forma simultánea.

SPEEDUP: Incremento de la aceleración medida con la que se ve el rendimiento de la programación paralela con la secuencial.

TIEMPO DE EJECUCIÓN: Lapso en el cual el computador demora en resolver un problema.

TPL: Librería de programación de tareas. *Task Parallel Library*, difundido por Microsoft.

ALGORITMO BIOINSPIRADO: Se basa en emplear analogías con sistemas naturales. Tienen como características ser auto - adaptativos, auto-organizados y ser capaces de auto-aprender. Un sistema es adaptativo si mejora su desempeño en el tiempo. Un sistema es auto–organizativo si aumenta su organización con el tiempo. Debido a su capacidad de resolver problemas complejos se utiliza actualmente en muchos campos como logística, diseño en ingeniería, etc.

ALGORITMO DE ENJAMBRE: Es una rama de la inteligencia artificial que estudia el comportamiento colectivo de los sistemas descentralizados, auto-organizados, naturales o artificiales. El concepto se emplea en los trabajos sobre inteligencia artificial.

CAPÍTULO III

MARCO METODOLÓGICO

3.1. Tipo y diseño de la investigación

Según el método en el manejo de los datos se establece que la investigación es de tipo cuantitativo porque se brindarán resultados finales estadísticos, según el objetivo que persigue la investigación es de tipo aplicada o tecnológica ya que la investigación busca resolver un problema como el de alineamiento de secuencias.

De acuerdo con el grado de control de variables se establece el diseño de investigación experimental (Namakforoosh, 2005, p. 94).

Se realizó un diseño experimental con post-prueba únicamente y un grupo de control. Al no ser el proceso de alineamiento de secuencias influyente en los datos iniciales de las secuencias el grupo de control y el grupo experimental son iguales, para así medir exclusivamente los tiempos en ambos escenarios.

Grupos	Proceso experimental	Prueba de salida
G.C	X1	O1
G.E	X2	O2

Donde:

GC = Grupo de control

GE = Grupo experimental

O1; O2 = Prueba de salida

X1 = Algoritmo centro estrella paralelo

X2 = Algoritmo basado en colonia artificial de abejas

3.2. Población y muestra

3.2.1. Población

Se han considerado grupos de secuencias a alinear que fueron ser extraídas del GENBANK (Base de secuencias biomoleculares), teniendo en cuenta que esta población es desconocida en número de secuencias actualmente almacenadas, y de diferentes tamaños de 20 hasta 20 millones de pb (pares de bases), pero considerando que son secuencias de cantidad de letras de diferentes y de diferentes longitudes se puede generar en forma aleatoria para fines del experimento.

Se considerará a subgrupos para cada medición para longitudes de secuencias que no se diferencien en X pb a diferentes tamaños referenciales que estarán en longitudes de X pb a Y pb (siendo X y Y un número de nucleótidos predeterminado) que se usarán para observar los tiempos de ejecución de los algoritmos centro estrella paralelo y basado en colonia artificial de abejas a medida que las longitudes de secuencia cambian y el número de secuencias se incrementa.

3.2.2. Muestra

Para el cálculo del tamaño de muestra se hará en base a la fórmula clásica de Freeman: n=10*(k+1), donde k es el número de variables dependientes, para usar la regresión lineal sin problemas (Freeman, 1987).

Dado que el número de variables independientes son 2 mi k=2, entonces reemplazando en la formula n=10*(2+1), un total de n=30, dado que en el estudio se variara tanto la longitud de las secuencias y el número de secuencias a alinear, para nuestro estudio será de 30x30, un total de 900 alineamientos.

3.3. Operacionalización de variables

Operacionalización de las variables

Variable	Definición conceptual	Definición operacional	Indicadores
Variable independiente Algoritmo centro estrella paralelo	Algoritmo centro estrella diseñado con la técnica de programación que utiliza a los múltiples procesadores del computador para realizar una o varias tareas.		
Variable independiente Algoritmo basado en colonia artificial de abejas	Algoritmo bioinspirado de inteligencia de en enjambre que emula el comportamiento de las abejas para resolver problemas.		
Variable dependiente Alineamiento múltiple de secuencias	Secuencias biomoleculares de ADN, ARN o aminoácidos que alineadas por un algoritmo.	Tiempo de respuesta Score obtenido	Tiempo de ejecución Puntaje del alineamiento

Fuente: Elaboración propia

3.4. Técnicas e instrumentos para recolección de datos

Como técnica de recolección de datos se usará la observación, medición directa de los datos y como instrumento de medición guía de observación en una hoja de anotaciones donde se apuntará los datos obtenidos por el cronómetro digital de gran precisión que encontramos en la clase StopWatch disponible en la librería “System.Diagnostic” de C#, dicha herramienta brindada por la Microsoft en el Visual Studio 2010, el cual medirá el tiempo en que se demoran en realizar el alineamiento múltiple de secuencias ambos algoritmos estrella paralelo y del basado en colonia artificial de abejas.

3.5. Procesamiento y análisis de datos

Para el análisis del indicador tiempo de ejecución, se usarán gráficos comparativos y se calcularán los tiempos de respuesta del algoritmo estrella paralelo y del algoritmo basado en colonia artificial de abejas para las diferentes longitudes de secuencias y número de secuencias, en la cual, se realizará una regresión lineal múltiple para identificar cómo se comportan ambos algoritmos al variar la longitud y cantidad de secuencias.

Para el análisis del indicador puntaje del alineamiento se realizará una comparación de medias, para lo cual, será necesario realizar una prueba de normalidad y una prueba T o U según corresponda.

Se desarrollará también una aplicación para generar las secuencias en forma aleatoria un tamaño y longitud definido para el experimento.

CAPÍTULO IV

PROPUESTA DE DISEÑO DE ALGORITMOS

En este capítulo se detalla la paralelización del algoritmo estrella, y la adaptación del algoritmo colonia artificial de abejas para el alineamiento múltiple de secuencias.

4.1. Elaboración del algoritmos

4.1.1. Elaboración del algoritmo estrella paralelo

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

val= contendrá la matriz de puntajes

Imax= contiene el valor del índice de la secuencia principal

SP= contiene el valor de la secuencia principal

af= contiene las secuencias alineadas finales

Para su implementación es necesario diseñar la función *Secfinal* la cual requiere a *SP* que es la secuencia con los *gaps* añadidos finales, el cual según la secuencia principal secundaria *seq[Imax,j, 1]* en la cual se realiza una comparación para añadir gaps si se requiere en la secundaria *seq[Imax, j, 2].*

Algoritmo 1 Algoritmo centro estrella paralelo basado en algoritmo centro estrella .

Requiere:
La función *wunsch* (requiere sec1, sec2, devuelve puntaje, sec1alineada, sec2alineada).
La cantidad de secuencias a alinear 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 j= Imax entonces
             af[j] = SP
        Si no
            af[j] = secfinal(SP, , seq[Imax, j, 2]);
                                Fin Para
 /* Las secuencias alienadas se encuentran en af */
```

Para la construcción de la secuencia principal ha sido necesario unir los *gaps* de las diferentes secuencias principales como se muestra en algoritmo 2.

En el algoritmo 2 usa las variables:

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

MaxL y MinL= contienen la máxima y minima longitud.

C= contiene las secuencias unidas.

Algoritmo 2 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
      C = C + p[i - ip]
   Fin si
Fin Para    /* C contiene la cadena con los gaps unidos */
```

El algoritmo 3 muestra el algoritmo de alineamiento de pares de Needleman Wunsch, necesario para la construcción de la matriz de puntajes por pares según Setubal y Meidanis, 1997, p. 52.

Algoritmo 3 *wunsch* (requiere sec1, sec2, devuelve puntaje, sec1alineada, sec2alineada)

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
Fin mientras
Asec1 = Invertir_texto(Asec1)
Asec2 = Invertir_texto (Asec2)
Val=f(n,m)
/*El puntaje es Val y los alineamientos en Asec1 y en Asec2 */
```

4.1.2. Elaboración del algoritmo basado en colonia artificial de abejas para el alineamiento múltiple de secuencias

El algoritmo 4 de la colonia artificial de abejas para el alineamiento múltiple de secuencias es el algoritmo presentado en el COMTEL 2016 detallado en el Anexo 6, se generaron funciones adicionales para su uso, a continuación se detalla paso a paso el algoritmo junto a las funciones adicionales.

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.

Como se aprecia en el algoritmo 4 las abejas scout primeramente generan alineamientos aleatorios este proceso no es más que generar gaps aleatoriamente entre las secuencias hasta que el tamaño de las secuencias sea uniforme, el resultado se considerara un alineamiento aleatorio.

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 figura 8, de rojo la secuencia favorita y debajo las 3 secuencias generadas de un alineamiento aleatorio.

Algoritmo 4 Algoritmo de la colonia artificial de abejas

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
Mostrar el mejor alineamiento Elite /* Informar a la abeja madre */

A	G	T	C	A	A	T
A	A	T	C	G	A	T
A	G	T	C	A	T	T
A	G	-	G	A	A	G

Figura 8. Ejemplo de generación de la secuencia favorita

Fuente: Anexo 6

Se debe tener en 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 5 genera la secuencia favorita y remueve una columna que contiene solamente gaps.

Algoritmo 5 GenerarCadenaFavoritaYSupGap(secuencias[])

```
Requiere:
Un arreglo de secuencias “secuencias”
La longitud de la mayor secuencia L
Un arreglo de letras letras(A, C, T, G)
La función Remover() que quita una letra de una posición
especifica
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;
   Si no
      seq3 = seq3 + letras[imayor]
   Fin si
Fin para
```

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 missmatch y 0 para un *gap*, tomando como datos el ejemplo de la figura 8 la matriz de puntuación sería la mostrada en la figura 9, 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	**3**
1	1	1	1	1	-1	1	**6**
1	1	0	-1	1	1	-1	**2**

Figura 9: Puntajes realizados de las secuencias con la secuencia favorita
Fuente: Anexo 6

Luego se guarda el alineamiento en una lista para optimizar por las observadoras, estas 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 figura 10.

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

Figura 10 Ejemplo de inserción y eliminación de gaps aleatoriamente
Fuente: Anexo 6

En la figura 10 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 6, 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

Algoritmo 6 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 != 0 entonces
   lsec = Longitud(sec)
   iig = Rand(0, lsec + 1)
   InsertarT(sec, iig, "-")
   eg = Rand(1, Gapt + 1)
   ge = 0
   Para i = 0 Hasta lsec 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 = lsec
           Fin si
         Fin si
     Fin si
   Fin para
Fin si
/* sec contendrá la nueva secuencia */
```

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 élites para cada enjambre, finalmente se escoge el mejor alineamiento élite.

4.2. Análisis de complejidad de algoritmos

Para el análisis de complejidad de algoritmos se tiene que tener en cuenta el número de secuencias a alinear p y la longitud máxima de las secuencias a alinear n.

4.2.1. Análisis de complejidad del algoritmo estrella paralelo

Considerando la elaboración del algoritmo estrella paralelo descrito en 4.1.1 se observa el uso de las funciones UNIRGAPS que es de complejidad O(n) y de la función wunsch que tiene una complejidad $O(n^2)$.

El algoritmo estrella paralelo inicia con un bucle que usa la función wunsch y se realiza (p-1)*(p-2) veces, la complejidad de esta parte es de $O(p^2*n^2)$, posteriormente se usa también las función IndiceSecPrincipal que es de complejidad $O(p^2)$, después se hace uso (p-2) veces la función UNIRGAPS en un nuevo bucle, siendo la complejidad en esta nueva sección O(p*n) y finalmente cuando se generan las secuencias alineadas finales su complejidad es O(p).

Finalmente la complejidad del algoritmo estrella paralelo estará dada por:

$$O(p^2*n^2) + O(p^2) + O(p*n) + O(p) = O(p^2*n^2)$$

4.2.2. Análisis de complejidad del algoritmo basado en colonia artificial de abejas para el alineamiento múltiple de secuencias

Considerando la elaboración del algoritmo de la colonia artificial de abejas descrito en 4.1.2 se observa el uso de las constantes nenjambre, nscout y nobservadoras, también se hace uso de las funciones generar

alineamiento aleatorio, el cual su complejidad es de O(p), GenerarCadenaFavoritaYSupGap el cual su complejidad es de O(n*p), InsertarySuprimirGap que tiene una complejidad de O(n), la evaluación de las secuencias con la secuencia favorita que es de complejidad O(n*p) e Insertar y eliminar un gap de manera aleatoria que es de complejidad O(n).

Finalmente la complejidad del algoritmo colonia artificial de abejas estará dada por:

$$O(p) + O(n*p) + O(n) + O(n*p) + O(n) = O(n*p)$$

CAPÍTULO V

RESULTADOS

5.1. Descripción del experimento realizado

Se usó el C# en el Visual Studio y el Framework 4.0 la cual provee la librería TPL que nos da soporte para bucles paralelos Freeman, 2010.

Para el algoritmo ABC se consideró como parámetros 15 enjambres, 15 scout y 15 observadoras, para la medición de tiempos se usó la clase *StopWatch* disponible en la librería "*System.Diagnostic"* de C#, que cuenta con un cronómetro de gran precisión. El experimento fue realizado en un computador de procesador Intel (R) Core (TM) i5-3210M de 2.5 Ghz, el cual tiene 4 núcleos, con memoria de 8Gb. En la figura 11 se observa la interfaz de la aplicación final con los tiempos obtenidos usando *StopWatch* para 20 secuencias de longitudes no mayores a 2 700 caracteres.

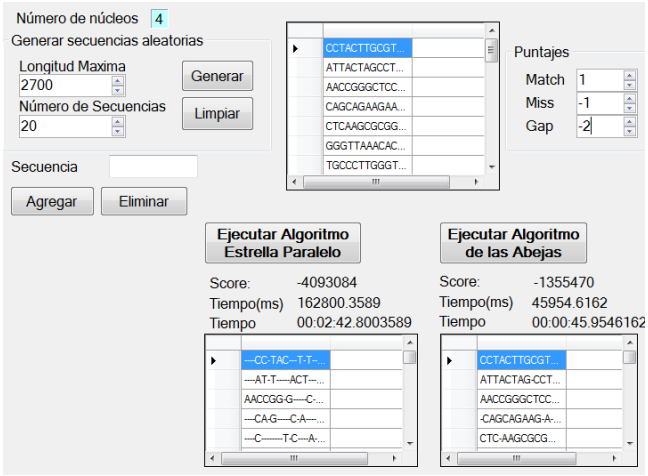

Figura 11. Interfaz de la aplicación de alineamiento de secuencias con los tiempos empleados

Fuente: Elaboración propia

5.2. Plan de pruebas

Antes de realizar el experimento se inició el computador en modo seguro, ya que esta opción limita las operaciones del sistema operativo a funciones básicas y utiliza un modo de gráficos reducido para la pantalla, posterior a ello se ejecutó el Visual Studio iniciando el código del algoritmo, después se ejecutó el administrador de tareas para verificar el rendimiento del computador antes de empezar el experimento, según la figura 12.

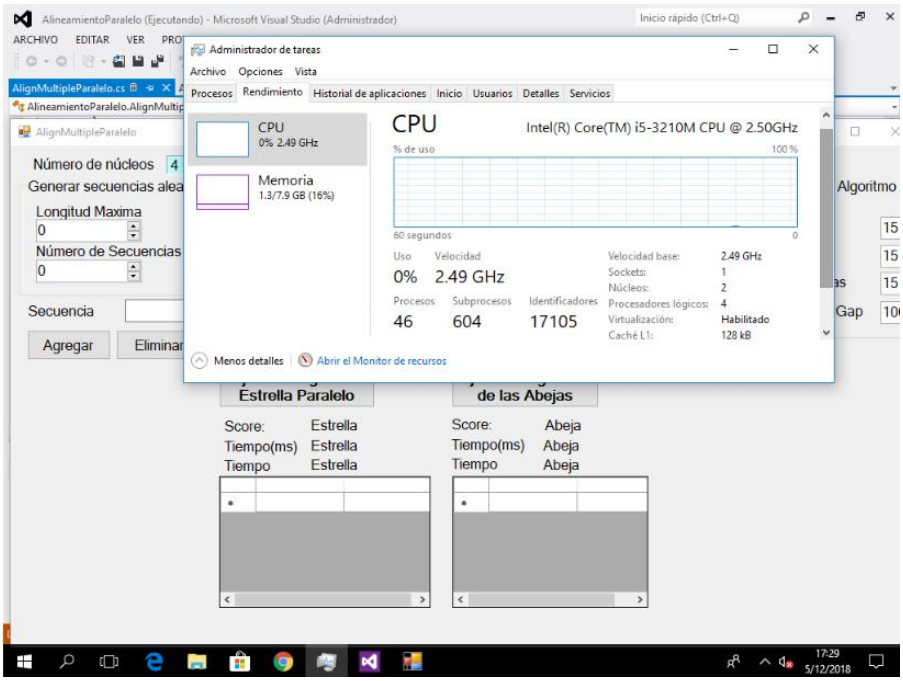

Figura 12. Interfaz de la aplicación de alineamiento de secuencias con los tiempos empleados

Fuente: Elaboración propia

5.3. Análisis del indicador Tiempo de Ejecución

En el estudio del tiempo de ejecución 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 2 700 pares de bases (pb). Los tiempos de ejecución (en milisegundos) resultados del experimento se observan en las figura 13, figura 14 y figura 15.

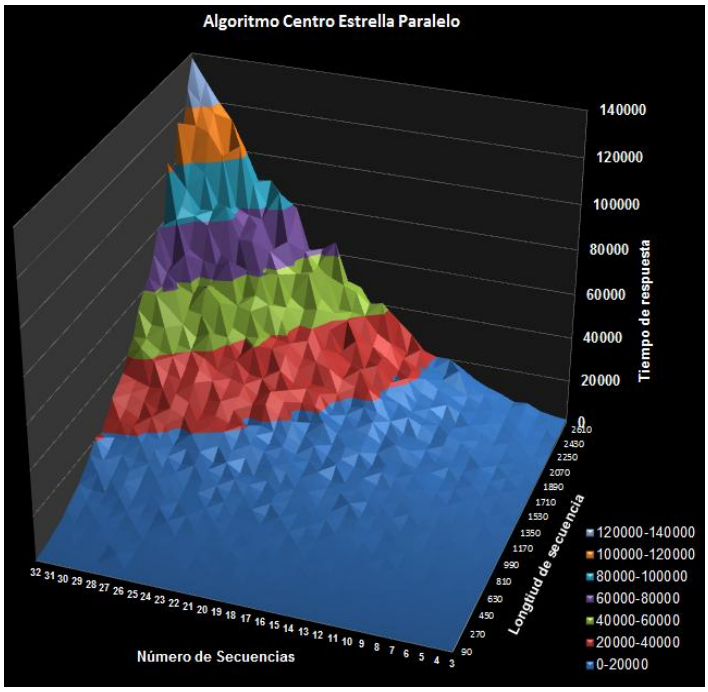

Figura 13. Tiempos de ejecución para el algoritmo centro estrella paralelo según la longitud y número de secuencias a alinear

Fuente: Elaboración propia

En la figura 13 observamos que el algoritmo centro estrella paralelo tiene tiempos de ejecución llegan hasta 140 000 milisegundos a medida que la longitud y número de secuencias se incrementan, en la figura 14 observamos que el algoritmo basado en colonia artificial de abejas sus tiempos de ejecución llegan hasta 25 000 milisegundos a medida que la longitud y número de secuencias se incrementan, en la figura 15 se puede observar los tiempos de ejecución del algoritmo centro estrella paralelo hasta un tope de 25 000 milisegundos.

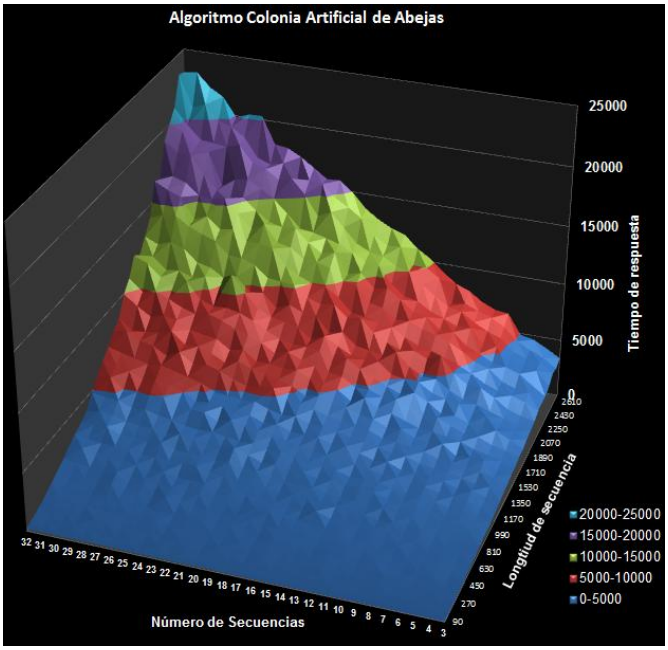

Figura 14. Tiempos de ejecución para el basado en colonia artificial de abejas según la longitud y número de secuencias a alinear

Fuente: Elaboración propia

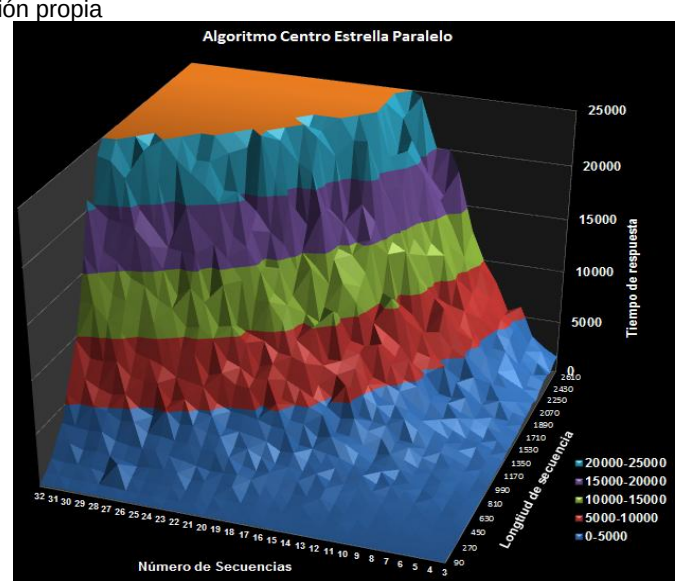

Figura 15. Tiempo de ejecución del algoritmo centro estrella paralelo con tiempos de ejecución menores a 25000 milisegundos según la longitud y número de secuencias a alinear

Fuente: Elaboración propia

Se realizó la regresión lineal múltiple para los tiempos de ejecución de ambos algoritmos según la ecuación 3:

$$Y= a_0+a_1X_1+a_2X_2 , \quad [3]$$

En donde :

Y= el tiempo de ejecución en milisegundos.

X_1= la longitud de la secuencia.

X_2= el número de secuencias a alinear

a_0= valor del interceptor.

a_1= coeficiente del X_1.

a_2= coeficiente del X_2.

Los resultados de la regresión lineal para el algoritmo de la colonia artificial de abejas indican:

a_0= -6 677,9304.

a_1= 6,5477.

a_2= 267,5531.

Con el coeficiente de determinación R^2 = 0,8365.

Siendo el coeficiente de determinación más próximo a 1 se acepta que la Ecuación 2 representa a estos datos.

$$Y= -6\ 677,9304+6,5477X_1+267,5531X_2, \quad [4]$$

Los resultados de la regresión lineal para el algoritmo centro estrella paralelo indican:

a_0= -36 903,1656.

a_1= 24,2725.

a_2= 1 590,2548.

Con el coeficiente de determinación R^2 = 0,6878

Siendo el coeficiente de determinación más próximo a 1 se acepta que la ecuación 5 representa a estos datos.

$$Y= -36\ 903{,}1656+24{,}2725X_1+1\ 590{,}2548X_2, \qquad [5]$$

Comparando los coeficientes de X_1 y X_2 en las ecuaciones 4 y 5 se verifica que la variable X_2 (número de secuencias a alinear) es la que mayor incremento genera en el tiempo de respuesta, por lo tanto es la más influyente.

Comparando el coeficiente a_1 de las ecuaciones 4 y 5 se verifica que el coeficiente a_1 de la ecuación 4 es aproximadamente 4 veces menor que el coeficiente a_1 de la ecuación 5, lo cual indica que el tiempo de respuesta total del algoritmo de la colonia artificial de abejas recibe un aumento de una cuarta parte del incremento que recibe el algoritmo estrella paralelo por cada incremento en la longitud promedio de las secuencias a alinear.

Comparando el coeficiente a_2 de las ecuaciones 4 y 5 se verifica que el coeficiente a_2 de la ecuación 4 es aproximadamente 6 veces menor que el coeficiente de a_2 de la ecuación 5, lo cual indica que el tiempo de respuesta total del algoritmo de la colonia artificial de abejas recibe un aumento de una sexta parte del incremento que recibe el algoritmo estrella paralelo por cada secuencia a alinear adicional.

5.4. Análisis del indicador puntaje del alineamiento

Se realizó una comparación de medias con los datos de los puntajes obtenidos en los alineamientos realizados por el algoritmo de la colonia artificial de abejas y el del centro estrella paralelo según los datos del Anexo 3 y 4, realizándose una prueba de normalidad y por último la prueba de Mann-Whitney como se describe en los siguientes apartados.

Análisis de normalidad de datos

Las hipótesis a probar fueron:

Para los puntajes obtenidos del algoritmo centro estrella paralelo

Ho: Los puntajes obtenidos del algoritmo centro estrella paralelo siguen una distribución normal

H1: Los puntajes obtenidos del algoritmo centro estrella paralelo no siguen una distribución normal

Para los puntajes obtenidos del algoritmo colonia artificial de abejas

Ho: Los puntajes obtenidos del algoritmo colonia artificial de abejas siguen una distribución normal

H1: Los puntajes obtenidos del algoritmo colonia artificial de abejas no siguen una distribución normal

Para ello, se usó el software estadístico IBM SPSS STATICS obteniendo el resultado para la prueba de normalidad el mostrado en la tabla 1.

Tabla 1

Resultados de la prueba de normalidad con SPSS

Pruebas de normalidad

		Kolmogorov-Smirnov			Shapiro-Wilk		
	Algoritmo	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Puntaje	Estrella	0,207	900	0,000	0,787	900	0,000
	Abejas	0,184	900	0,000	0,820	900	0,000

Fuente: Elaboración propia

Al ser el valor de p< α se rechaza Ho para ambos algoritmos. Es decir los puntajes obtenidos por el algoritmo centro estrella paralelo y el colonia artificial de abejas no siguen una distribución normal requisito necesario para realizar una prueba T según Limache y Limache (2017), dado que no se cumplió el supuesto de normalidad se aplicó la prueba U de Mann Whitney.

Aplicación de la prueba U de Mann Whitney

Según Moreno (2008), esta prueba es el equivalente no paramétrico de la prueba T sobre diferencia de medias para muestras independientes, y se emplea cuando no se cumplen los supuestos de la prueba T.

La hipótesis a probar con la prueba U de Mann-Whitney fue:

Ho: El puntaje promedio de los alineamientos realizados por el algoritmo de colonia artificial de abejas es igual al puntaje promedio de los alineamientos realizados por el algoritmo centro estrella paralelo.

Ha: El puntaje promedio de los alineamientos realizados por el algoritmo de colonia artificial de abejas es diferente al puntaje promedio de los alineamientos realizados por el algoritmo centro estrella paralelo.

Aplicando la prueba U de Mann-Whitney, con un nivel de significancia del α= 5 % y un nivel de confianza del 95 % se obtiene lo que se indica en las tablas 2 y 3.

Tabla 2

Rangos de la prueba de Mann-Whitney

Rangos

	Algoritmo	N	Rango Promedio	Suma de rangos
Puntaje	1	900	799,35	719417,00
	2	900	1001,65	901483,00
	Total	1800		

Fuente: Elaboración propia

Tabla 3

Estadísticos de prueba de Mann-Whitney

	Puntaje
U de Mann-Whitney	313967,000
W de Wilcoxon	719417,000
Z	-8256,000
Sig. asintótica (bilateral)	0,000

Fuente: Elaboración propia

Al ser el valor $p < \alpha$ se rechaza Ho. Es decir el puntaje promedio de los alineamientos realizados por el algoritmo de colonia artificial de abejas es diferente al puntaje promedio de los alineamientos realizados por el algoritmo centro estrella paralelo.

Verificando el promedio según la tabla 4 de los puntajes del algoritmo centro estrella paralelo es de -660 524 y el puntaje promedio del algoritmo centro estrella paralelo es de -288 588 siendo este el ultimo el que tiene un mayor puntaje promedio y que según la prueba U de Mann Whitney no ha sido debido al azar.

Tabla 4

Valores descriptivos de los algoritmos estudiados

	Algoritmo			Estadístico	Error estándar
Puntaje	Estrella	Media		-660 524,4178	26 957,9658
		95% de intervalo de confianza para la media	Límite inferior	-713 432,2904	
			Límite superior	-607 616,5451	
		Media recortada al 5 %		-567 323,8531	
		Mediana		-322 511,0000	
		Varianza		6,541E+11	
		Desviación estándar		808 738,9735	
		Mínimo		-4 390 696,0000	
		Máximo		-151,0000	
		Rango		4 390 545,0000	
		Rango intercuartil		901 109,7500	
		Asimetría		-1,6290	0,0820
		Curtosis		2,2820	0,1630
	Abejas	Media		-288 588,0489	10 679,0151
		95% de intervalo de confianza para la media	Límite inferior	-309 546,7509	
			Límite superior	-267 629,3469	
		Media recortada al 5 %		-254 744,5951	
		Mediana		-163 483,0000	
		Varianza		1,026E+11	
		Desviación estándar		320 370,4527	
		Mínimo		-1 482 819,0000	
		Máximo		-150,0000	
		Rango		1 482 669,0000	
		Rango intercuartil		382 324,7500	
		Asimetría		-1,4400	0,0820
		Curtosis		-1,5040	0,1630

Fuente: Elaboración propia

CAPÍTULO VI

DISCUSIÓN

Según los resultados obtenidos se pudo observar que el algoritmo basado en la colonia artificial de abejas tuvo menor razón de cambio en los tiempos de respuesta esto es similar a lo que indica Chand, J., Sharma, H. y Singh, S. (2013) sobre que el algoritmo ABC no se ve afectado por la creciente dimensión del problema.

Observando los resultados de los puntajes y de los tiempos obtenidos en el algoritmo de la colonia artificial de abejas se puede observar que se han conseguido resultados más óptimos como indica también Lei, X., Sun, J., Xu, X. y Guo, L. (2010)

CONCLUSIONES

Se ha estudiado e implementado los algoritmos centro estrella paralelo y el algoritmo basado en colonia artificial de abejas usando para ello pseudocódigos descriptivos de dichos procesos como se indica en el capítulo IV.

Se determinó los tiempos de los algoritmos centro estrella paralelo y el algoritmo basado en colonia artificial de abejas según los resultados de la regresión lineal múltiple se observa que el algoritmo de la colonia artificial de abejas muestra un tiempo de respuesta menor para un gran número de secuencias, así como en secuencias de gran longitud, y que el algoritmo basado en la colonia artificial tiene una menor razón cambio en los tiempos de respuesta, tanto si la longitud de la secuencia o el número de secuencias se incrementan.

Según los resultados en 5.3 siendo el a_2 mayor que a_1 en las ecuaciones 1 y 2 se concluye que el número de secuencias a alinear es más influyente que la longitud de estas secuencias. Los resultados indican que el algoritmo basado en la colonia artificial de abejas es más eficiente a pesar de no ser un algoritmo paralelo.

Finalmente se pudo observar que los puntajes obtenidos por el algoritmo basado en la colonia artificial de abejas son mayores a los del centro estrella paralelo.

RECOMENDACIONES

Se recomienda ampliar la investigación para compararlo con otras técnicas de alineamiento múltiple de secuencias, tanto realizar su estudio en computadores científicos o altamente paralelos.

Se recomienda paralelizar el algoritmo basado en la colonia artificial de abejas considerando al 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

Aguilar, A.(2014) *Un algoritmo basado en la colonia artificial de abejas con búsqueda local para resolver problemas de optimización con restricciones*. Universidad Veracruzana. México.

Borovska, P., Gancheva, V., Markov, S., Georgiev, I. y Asenov, E. (2011). *Parallel Performance Evaluation and Profiling of Multiple Sequence Nucleotide Alignment on the Supercomputer BlueGene/P.* The 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications.

Callisaya, W., Callohuari, R., y Jimenez, J (2013). *Programación Paralela para el Alineamiento de Secuencias de ADN usando Task Parallel Library (TPL).* Proceedings of the 20th Intercon, pp. 156-162.

Callisaya, W. y Yari, Y (2016). *Comparación del algoritmo centro estrella paralelo con uno basado en la colonia artificial de abejas (ABC) en el alineamiento múltiple de secuencias.* Memoria COMTEL 2016, pp. 85-93

Chand, J., Sharma, H. y Singh, S.(2013). *Artificial bee colony algorithm: a survey*. Int. J. Advanced Intelligence Paradigms, Vol 5.

Chapman, B., Jost, G., y Van der Pas, R. (2008). *Using OpenMP: portable shared memory parallel programming.* (M. Press, Ed.) (p. 353). London, England.

Durbin, R., Eddy, S., Krogh, A., y Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids.* (C. U. Press, Ed.) (p. 356). New York, United States.

Eldrandaly, K., Hassan, M. y AbdelAziz, N. (2015). *A Modified Artificial Bee Colony Algorithm for Solving Least-Cost Path Problem in Raster GIS.* An International Journal of Applied Mathematics & Information Sciences.

Freeman, A. (2010). Pro .NET 4 *Parallel Programming in C#.* (Apress, Ed.) (p. 328). Berkeley, CA: Apress. doi:10.1007/978-1-4302-2968-1

Freeman, DH (1987) *Applied categorical data analysis*. New York. Marcel Dekker Inc.

Gibas, C., y Reilly, P. O. (2001). *Developing Bioinformatics Computer Skills.* (O'Reilly, Ed.) (1ra Editio., p. 446). United States Of America.

González, D. (2013) *Metaheurísticas, Optimización Multiobjetivo y Paralelismo para Descubrir Motifs en Secuencias de ADN.* España. Universidad de Extremadura.

Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* (C. U. Press, Ed.) (1st editio., p. 534). New York, United States.

Higgins, D., y Taylor, W. (2000). *Bioinformatics: Sequence, Structure and Databanks.* (O. U. Press, Ed.)Briefings in Bioinformatics (Vol. 2, p. 249). New York, United States. doi:10.1093/bib/2.2.202

JáJá, J. (1997). *An Introduction to Parallel Algorithms*. (A. Wesley, Ed.) (p. 566). United States of America.

Jones, N. C., y Pevzner, P. A. (2004). *An Introduction to bioinformatics algorithms*. (M. Press, Ed.) (p. 435). London, England.

Karaboga, D., Gorkemli, B., Ozturk, C. y Karaboga, N. (2012). *A comprehensive survey: artificial bee colony (ABC) algorithm and apllications*, Artif Intell Rev. Springer.

Lalwani, S., Kumar, R. y Gupta, N. (2013). *A Review on Particle Swarm Optimization variants and their applications to Multiple Sequence Alignment*. Journal of Applied Mathematics & Bioinformatics, vol3, no. 2.

Lesk, A. M. (2002). *Introduction to Bioinformatics*. (O. U. Press, Ed.) (p. 283). New York, United States: Oxford University Press.

Lei, X., Sun, J., Xu, X. y Guo, L. (2010). *Artificial bee colony for solving multiple sequence alignment.* IEEE Fifth International Conference Bio-Inspired Computing: Theories and Applications (BIC-TA).

Limache, E. y Limache, W., (2017). *Métodos estadísticos aplicados a la investigación en ciencias sociales*, Perú, Tacna, Universidad privada de Tacna.

Luo, G., Huang, S., Chang, Y. y Yuan, S. (2013). *A parallel Bees Algorithm implementation on GPU*. Journal of Systems Architecture.

Mokaddeml, A y Elloumi, M. (2013). *Motalign: A Multiple Sequence Alignment Algorithm Based on a New Distance and a New Score Function*. 24th International Workshop on Database and Expert Systems Applications.

Moreno, E. (2008). *Manual de Uso de SPSS*. Universidad Nacional de Educación a Distancia. España, Madrid.

Mount, D. (2001). *Bioinformatics: Sequence and Genome Analysis*. (C. S. H. L. Press, Ed.) (p. 564). New York, United States.

Namakforoosh, Mohamman Naghi (2005), *Metodología de la Investigación*, México: Editorial Limusa, S.A. de C.V.

Nguyen, K., Pan, Y. y Nong, G. (2010). *Parallel Progressive Multiple Sequence Alignment on Reconfigurable Meshes*. International Conference on Bioinformatics and Computational Biology. Las Vegas.

Oliver, T., Schmidt, B., Nathan, D., Clemens, R. y Maskell, D. (2005). *Multiple Sequence Aligment on an FPGA.* Proceedings of the 11th International Conference on Parallel and Distributed Systems.

Ozturk, C. y Aslan, S. (2016). *A new artificial bee colony algorithm to solve the multiple sequence alignment problem*. Int. J. Data Mining and Bioinformatics, Vol 14.

Pevsner, J. (2009). *Bioinformatics and Functional Genomics*. (r J. W. & Sons, Ed.) (2nd Editio., p. 897). New York, United States.

Rekaby, A., Youssif, A. y Sharaf, A. (2013). *Introducing Adaptive Artificial Bee Colony Algorithm and Using it in Solving Traveling Salesman Problem*. Science and Information Conference.

Rubio, Á., Vega, M., Gómez, J. y Sánchez, J. (2012). *A Parallel Multiobjective Artificial Bee Colony Algorithm for Dealing with the Traffic Grooming Problem*. IEEE 14th International Conference on High Performance Computing and Communications.

Santander, S., Vega, M., Gómez, J. y Sánchez, J. (2010). *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.

Setubal, J., y Meidanis, J. (1997*). Introduction to Computational Molecular Biology*. (P.-K. P. Company, Ed.) (p. 296). Boston, United States.

Smith, J. R. (1993). *The Design and Analysis of Parallel Algorithms*. (O. U. Press, Ed.) (p. 510). New York, United States.

Sundfeld, D., Teodoro, G., Cristina, A. y Melo, M. (2015). *Parallel A-Star Multiple Sequence Alignment with Locality-Sensitive Hash Functions.* Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS).

Venkatarajan, S. M., y Pandjassarame, K. (2009). *Bioinformatics: A Concept-Based Introduction*. (Springer, Ed.) (p. 184). New York, United States.

Wang, L., Leebens, J., Kerr, P., Beckmann, K., Pamphilis, C. y Warnow, T. (2011). *The Impact of Multiple Protein Sequence Alignment on Phylogenetic Estimation*. IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol.8, no. 4, pp. 1108-1119.

Xiong, J. (2006). *Essential Bioinformatics.* (C. U. Press, Ed.) (p. 339). New York, United States.

ANEXOS

Anexo 1: Tiempos del algoritmo Centro estrella paralelo

Tiempos del algoritmo centro estrella																
	Número de secuencias															
Longitud de secuncias	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
90	11.8694	3.6778	5.0859	27.9755	9.3135	12.5266	18.1278	19.5125	21.6082	26.1572	33.6132	34.6957	36.0467	94.5101	49.5476
180	6.4271	11.1637	21.0434	20.5922	38.6675	39.3379	56.4768	59.9733	95.3415	96.8756	103.8943	117.7609	149.4841	187.6303	220.863
270	18.1274	16.9299	52.6118	60.2758	73.6301	81.7774	97.8432	126.1701	245.808	241.1174	303.6786	305.2694	309.4489	334.2581	416.9821
360	26.9085	43.7724	69.2901	117.7187	157.2225	199.7892	176.4588	289.7754	356.4325	328.163	430.7326	561.6065	501.8976	612.1037	722.5389
450	42.2038	60.1334	66.7703	110.7164	190.2893	277.8432	291.2726	380.437	418.8008	522.0912	802.9042	769.2429	717.1077	903.6038	1124.9106
540	50.4602	125.7887	145.7861	220.1068	229.236	334.2601	453.2482	576.8763	657.141	812.4529	1096.3013	1078.2095	1319.2017	1457.7315	1669.7358
630	56.1907	197.8692	159.14	262.9059	418.1887	539.3931	616.4204	849.6968	989.8666	1084.19	1171.2295	1619.5999	1554.1161	1774.5933	1839.3442
720	75.6286	189.4682	244.3038	526.8426	403.7013	767.6209	971.3212	899.5799	1498.978	1560.494	1730.6451	1815.6423	1918.78	2488.7403	2415.2965
810	109.8264	248.6344	297.3529	490.5745	562.1468	814.6283	826.4797	1456.559	1374.9076	1569.1725	2029.9357	2249.7847	2491.6255	2633.2826	3850.1701
900	159.7866	247.1319	390.8347	696.3311	674.7331	944.5547	1238.636	1335.6185	1335.9839	2073.8342	2802.7194	3245.7362	2945.6348	3974.8119	4144.1453
990	241.6005	253.0036	477.9116	680.7859	817.0853	972.2416	1514.534	1805.8751	1546.0732	2244.324	2858.5308	3040.4775	4543.0267	5030.4423	5042.5141
1080	216.3472	327.2422	640.5981	915.7684	956.9615	1333.833	2093.534	2263.1111	2553.9605	2982.9953	3958.4361	3740.0818	4676.0874	5261.4957	5953.7402
1170	252.9454	477.9649	490.8943	997.0778	1411.818	1569.881	1850.972	1929.3145	3136.7267	3300.2142	3559.2846	6263.2573	4954.9528	5904.3875	7748.6597
1260	263.9749	601.2331	668.7883	1195.606	1643.544	1846.698	2539.778	3335.0278	3213.7433	4175.9489	4858.3219	5470.8867	6504.0992	7181.7364	8291.0965
1350	250.0471	573.1426	1123.244	1306.231	1453.515	2066.184	2773.09	3545.535	3909.7268	4299.0303	5385.789	6772.7894	7268.9233	7206.1519	10152.12
1440	463.5699	480.2836	907.3502	1095.031	1711.856	2512.99	2734.464	3557.5686	4581.5694	5118.8587	5565.8361	7585.3836	7828.8722	11180.845	9961.5988
1530	360.0914	727.778	704.6734	1376.048	2255.457	2073.331	3426.43	4678.7825	5139.5482	6459.5214	8072.2816	7772.7184	8046.167	10440.206	10840.818
1620	468.0356	881.3235	1180.772	1778.742	2197.721	2886.653	3474.535	5160.5842	6422.3013	6561.8627	7568.0493	6567.3481	11166.678	12746.543	12440.836
1710	486.9229	908.494	1535.893	1636.973	3044.596	2782.008	3037.129	4695.3422	5547.682	7072.9318	8024.1658	8854.1785	12094.168	13173.319	14256.485
1800	667.2899	800.1754	846.4052	1889.403	2984.828	3709.503	5195.638	5746.9794	6081.9227	9053.5547	10820.876	9649.7967	12885.713	14321.695	14751.05
1890	760.9577	1052.914	1965.052	1804.378	3855.305	3427.917	4270.23	5520.0464	7419.467	8966.5785	10028.23	10635.478	15427.85	14965.428	16554.128
1980	776.5872	1472.558	1971.427	2884.604	3585.666	5031.838	5671.198	7088.2062	6657.6623	8328.6195	9232.1569	13749.382	16653.665	18043.637	17079.335
2070	992.308	1523.844	2017.385	2142.178	3270.113	4507.489	6899.012	8838.8541	8140.0815	11174.538	13714.957	14398.297	19522.948	23237.463	23164.834
2160	595.4751	1499.047	2114.049	2813.498	4319.182	5585.751	6746.114	8673.5274	9077.9648	10683.91	12937.814	15918.329	18578.746	21795.101	28192.373
2250	1006.901	1300.164	2711.551	3244.377	3372.867	5581.143	7682.82	7690.3051	8551.8389	12467.424	13877.976	18563.444	21122.2	28123.821	25010.333
2340	1299.624	1771.682	2670.055	3688.232	3847.87	6528.592	7806.361	9839.421	12150.418	13155.746	15672.465	21374.724	18959.808	28290.98	26606.562
2430	1387.697	1184.559	2409.25	4340.169	4780.61	6880.695	8325.164	9291.8137	12689.578	15412.855	18910.327	21919.057	20825.763	25734.674	29692.299
2520	1009.619	1822.782	2755.425	4676.826	5596.419	7463.27	8450.645	11056.497	13825.732	16225.231	18550.572	20362.243	23309.692	29207.992	32792.761
2610	1396.816	2423.944	2937.077	4701.924	5458.453	7248.235	9798.465	12015.111	18656.291	16086.643	17563.85	25328.473	22789.141	25894.354	44247.446
2700	1509.224	2195.311	3199.373	5959.281	5389.556	8026.467	9886.285	12603.517	16569.594	20028.249	19963.21	21866.073	28593.868	33819.553	37136.262

Tiempos del algoritmo centro estrella																
		Número de secuencias														
Longitud de secuncias		18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	90	56.8828	58.8813	72.2853	65.2711	81.738	82.9909	89.3495	94.1103	93.3147	112.0616	120.3862	150.542	149.2164	159.1946	166.5307
	180	209.37	248.9982	211.6968	306.3564	307.4739	299.6313	384.9798	342.856	376.1507	466.9682	484.7221	556.062	549.6784	650.7019	698.3653
	270	449.8835	503.4818	602.8702	726.4208	693.4899	830.9691	864.0988	845.3149	952.8185	2041.8938	970.1291	1203.2051	1278.8974	1286.0967	1415.5186
	360	759.5213	1139.117	1041.222	1173.439	1053.882	1483.682	1540.674	1530.791	1388.8646	1754.712	1956.0403	2169.1395	2039.8745	2506.0706	2575.5035
	450	1367.488	1656.888	1573.652	1516.09	1757.761	2019.967	2259.263	2485.5744	2500.6767	2568.2562	2798.0813	3420.6021	3323.199	3446.9425	3593.8355
	540	1570.932	1823.912	2412.37	2693.128	2746.594	3078.943	3059.346	3273.0212	3332.219	4207.523	4055.1426	4736.7512	5174.5724	4911.9354	5787.0814
	630	2222.054	2824.803	2763.926	3517.415	3650.125	4132.777	4014.829	5240.9133	4356.0654	4912.9313	6028.2903	5745.5245	6144.6933	6808.9992	7753.5802
	720	3129	3394.953	3622.396	3743.463	4671.226	4872.614	5842.778	5377.6175	6906.7919	6873.134	7154.2059	8356.939	8126.6159	8196.8786	9701.188
	810	3972.29	4381.004	4616.66	4738.939	5277.876	6594.095	6868.838	7077.0966	7963.7348	8415.5568	8758.9729	10535.163	11270.242	12741.748	12680.773
	900	5461.721	4687.975	6529.92	6622.349	7535.847	7114.834	8857.639	9018.12	10147.853	11248.949	11374.351	12832.393	13186.476	14879.679	16449.63
	990	5338.389	7004.927	6821.191	7688.492	8227.437	9911.4	10897.48	10664.514	12125.795	12800.76	13873.063	14420.142	15737.234	16562.045	20950.456
	1080	6567.345	7874.879	7384.942	9443.396	10839.96	10571.88	12622.98	13713.298	12793.244	15106.394	16990.224	18365.949	18718.285	18955.359	19958.887
	1170	8215.222	8379.696	9265.464	9704.754	10288.51	13860.36	12561.6	16339.841	16370.845	16525.732	18086.916	19606.541	22207.893	23535.559	23866.656
	1260	9200.436	10632.73	12389.01	12341.22	13979.6	15678.19	16919.37	18340.08	19859.166	17392.846	21812.311	24331.91	22431.114	25574.72	29298.307
	1350	9512.259	11691.42	12671.84	14439.38	16514.67	17375.9	20109.45	21451.938	22382.437	23388.244	26139.439	24286.858	27030.114	29522.794	32040.221
	1440	10580.54	12104.31	14429.94	14382.16	17473.89	20784.92	20777.36	23134.023	23105.082	26872.747	29544.995	30891.865	30462.398	32300.399	34527.786
	1530	14285.71	13696.48	16108.67	17058.7	18764.66	17822.16	23603.96	26401.291	25825.215	32644.905	32799.638	34469.95	39142.527	44074.546	41133.439
	1620	13948.24	15056.95	19632.16	21262.95	25130.34	23810.94	26753.23	28221.247	31498.942	35584.618	35743.05	39372.363	40880.426	45436.666	45048.279
	1710	18355.78	17058.5	15949.38	24310.07	27934.11	25183.36	28963.04	30556.095	32924.714	37702.142	40209.404	41568.742	48477.67	48581.008	54553.209
	1800	18231.14	21327.17	22054.03	26612.82	25771.5	29712.93	30715.97	35927.838	36874.802	46246.449	43073.744	47814.703	53790.46	59371.976	59635.022
	1890	17708.95	24011.36	25374.75	26677.63	31437.33	29154.85	40644.77	35633.514	42209.836	43051.88	50658.535	51805.857	61228.373	59914.772	63378.604
	1980	22032.07	24100.13	24182.62	29843.15	35556.08	37457.94	41423.95	40730.34	43912.722	54227.652	60755.123	56087.156	58666.66	63653.859	70928.127
	2070	25561.03	29558.86	31364.44	31957.16	37791.76	38629.76	41512.48	49438.728	50799.938	54381.872	58053.933	57002.631	70153.071	64793.71	80529.173
	2160	25469.81	30343.04	34466.53	37115.47	36370.93	44399.15	50550.16	53849.326	51393.086	67626.178	64589.693	65683.593	69664.745	82324.436	78144.395
	2250	26249.09	30298.27	31981.03	40374.31	39752.98	47391.36	48023.88	61618.313	61427.459	56795.066	66721.108	78831.141	82079.14	85287.971	104079.12
	2340	34279.13	27778.26	37444.24	45900.79	44630.91	49136.86	58702.01	58137.343	71796.219	71686.594	85720.355	82788.962	82494.513	90521.843	86920.095
	2430	33499.96	39738.13	42165.22	48015.92	58051.45	57282.23	61457.93	65587.845	76337.882	74657.749	81792.949	83843.763	96007.149	100900.65	116512.63
	2520	39918.48	40840.97	40827.57	56136.34	56383.89	54020.72	68237.18	62420.832	76557.049	77451.494	80324.655	102274.05	111483.07	113807.33	111295.71
	2610	42494.34	47092.39	49384.59	47125.65	56466.16	61899.22	71376.16	86774.678	80892.117	89403.228	104820.35	94874.924	117857.81	121167.36	127208.73
	2700	37858.23	46608.06	48433.55	65423.59	61567.36	59690.67	78204.74	81695.602	88818.951	87902.718	101332.31	113180.92	120480.5	128989.67	137881.59

Anexo 2: Tiempos del algoritmo basado en colonia artificial de abejas

Tiempos del algoritmo basado en colonia artificial de abejas																
		Número de secuencias														
Longitud de secuncias		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	90	28.629	31.4312	37.7956	45.2934	47.6424	66.8758	57.9362	65.0539	67.703	72.9269	84.0972	96.3546	93.4021	93.3939	102.9283
	180	51.6495	63.6495	74.4405	111.2817	100.9024	116.7839	119.472	133.8711	173.3958	167.2577	185.6241	183.7599	193.812	202.5955	227.1739
	270	82.3792	106.7421	120.58	143.4326	168.566	223.198	214.889	234.1397	231.5592	289.9745	276.1904	295.6521	323.9798	389.2111	355.6648
	360	124.9094	149.4394	184.0744	200.0601	254.8674	321.5015	307.8134	324.222	368.4743	437.8667	441.6993	450.0087	485.8892	520.3071	507.4007
	450	178.7753	195.3206	304.7176	310.3829	301.3961	349.6962	471.6696	453.0753	532.2853	564.3275	531.6975	594.3942	680.5962	742.7186	668.4611
	540	221.1142	272.3089	358.7462	414.2953	486.2226	486.285	533.2833	589.4158	669.5909	717.5741	746.6522	835.5141	843.0837	893.3038	891.8908
	630	294.1746	333.7453	448.9097	481.5841	530.4425	667.2571	671.4814	764.9271	758.6937	896.7645	909.4156	950.7093	1022.52	1157.6199	1317.8215
	720	375.679	443.1082	499.0601	532.6183	667.9944	701.1293	807.5656	910.0659	924.802	972.4058	1031.1188	1431.4765	1295.1676	1305.9431	1425.3958
	810	474.8729	505.6079	578.1958	689.8695	831.8263	849.6233	989.7586	930.8642	1120.7147	1217.9655	1356.1985	1436.8297	1547.6487	1649.8267	1561.5064
	900	528.2852	659.1776	723.3292	796.8604	891.282	1085.891	1165.005	1366.8872	1364.0185	1461.2538	1449.3486	1607.9993	1827.8553	1857.8284	1873.9049
	990	528.4839	727.0584	860.9275	920.5436	1031.134	1198.891	1294.123	1395.4239	1746.899	1850.3573	1826.9826	2021.7351	1982.2769	2265.298	2306.7058
	1080	652.679	739.9353	909.2432	974.0639	1254.297	1430.648	1413.265	1601.6181	1836.3265	1928.8629	1911.4871	2238.8529	2340.56	2516.9071	2675.9626
	1170	733.3685	862.3097	1167.298	1269.267	1324.736	1633.988	1639.714	1925.9063	1945.3737	2195.1121	2337.1945	2512.9086	2692.1368	2760.1104	2971.4017
	1260	867.5114	979.8864	1336.251	1249.198	1644.42	1804.868	1843.284	1917.4249	2380.5823	2268.6474	2678.046	3028.6516	2978.7882	3354.3667	3339.5193
	1350	1002.99	1247.471	1252.51	1492.073	1764.274	1829.909	2054.854	2451.2448	2517.7278	2791.7351	3010.4153	3148.0087	3398.9384	3649.417	3671.1201
	1440	1102.758	1382.523	1640.277	1738.478	1944.768	2058.435	2433.886	2644.6295	2837.7173	2885.3121	3404.9546	3526.9243	3662.8838	3943.6331	4311.4744
	1530	1245.955	1408.865	1974.15	1982.441	1865.84	2719.808	2565.775	2727.0366	3331.6455	3221.4073	3721.6674	3865.5175	4320.2046	4520.1823	4888.7843
	1620	1314.992	1507.673	1791.12	2057.267	2475.809	2686.134	2942.025	2965.9553	3374.1597	3814.1595	4040.1163	4704.3815	4198.283	5061.5768	5351.9405
	1710	1478.495	1734.755	2023.701	2513.221	2537.659	3035.078	3499.951	3536.1459	3963.387	4467.4235	4413.3281	4853.784	4814.7921	5541.6728	5711.3756
	1800	1503.331	2062.527	2637.376	2660.123	2798.453	3283.983	3221.689	4061.252	4612.2803	4288.3234	4738.585	5288.4757	5818.2257	6526.7884	6344.5121
	1890	1719.184	2097.708	2369.073	3034.47	3011.571	3677.311	3858.724	4227.4268	4936.4272	4971.8167	5190.8948	6094.6924	6029.1331	6211.884	6797.5587
	1980	1861.273	2086.041	2463.985	2832.416	3234.227	3462.844	4037.599	4828.2527	4837.4534	5473.0419	6569.4906	6023.8973	6274.4469	6479.4268	7454.0971
	2070	1991.1	2260.906	2641.484	3450.246	4061.85	4152.361	4931.402	6278.9733	5781.9367	6302.1215	6153.1361	6575.7117	7257.996	8317.9303	8133.8846
	2160	2328.297	2851.598	3241.619	3430.673	3859.573	4525.01	5099.215	5460.2956	5891.0894	6174.0835	6697.8788	7324.5594	7577.3989	7850.0084	7985.5586
	2250	2400.014	3107.732	3268.164	3668.142	4574.866	4930.783	5685.626	6293.8913	6730.781	7031.6948	7543.1478	7440.9895	8309.3496	8512.2974	9863.8422
	2340	2364.044	3121.912	3944.866	4343.515	4527.291	5091.582	5611.308	6609.5482	6627.936	7508.1856	7774.1326	8655.5576	9479.4949	9225.7617	10426.71
	2430	2650.909	3834.68	3984.216	4340.561	5386.759	5415.61	6582.699	6542.2692	7018.6763	7449.9143	8155.7146	8385.883	9271.4764	10714.284	10842.656
	2520	3244.971	3981.46	4147.661	4475.513	5532.547	5927.675	6544.685	7257.1725	7523.0325	8241.0841	9084.8156	9978.5641	10311.025	10972.112	11171.564
	2610	3029.919	3793.33	4289.607	4873.78	5935.042	6564.386	6944.345	7426.8072	7198.9094	9229.1711	10022.195	9941.358	11472.994	11381.263	12025.111
	2700	3388.646	4040.379	4638.096	4869.481	6579.605	6705.946	7221.296	7991.3851	8425.8232	9274.1788	10264.769	10957.337	12005.854	12364.319	12949.695

Tiempos del algoritmo basado en colonia artificial de abejas																
		Número de secuencias														
Longitud de secuncias		18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	90	106.846	138.6952	117.5844	123.1154	134.6363	139.7662	150.1615	154.2835	173.5962	195.8051	173.3934	174.5408	191.7852	181.8005	190.1045
	180	221.4114	239.6538	261.9728	300.963	284.1693	333.7182	298.8357	350.4351	339.7513	342.3938	376.5723	425.0789	384.9552	393.3069	450.8794
	270	408.2216	466.0047	418.1566	475.8031	511.1417	477.8968	515.8521	530.2454	545.5978	635.2539	687.2775	601.8246	630.2718	668.9542	689.1318
	360	583.8294	572.6882	619.5042	640.9631	661.8345	667.1528	802.2871	795.6683	820.1962	832.3444	835.1121	895.4102	934.1525	984.8143	1021.7815
	450	701.2968	799.9061	813.4821	939.6301	961.3648	933.1918	976.6941	999.7528	1064.84	1158.7188	1204.5582	1173.1988	1237.5158	1309.474	1419.0947
	540	1000.427	1049.269	1030.947	1098.695	1122.321	1258.711	1468.482	1295.2801	1407.6185	1441.7088	1652.711	1546.0715	1610.8048	1662.1136	1783.289
	630	1351.031	1276.858	1360.797	1410.606	1558.035	1527.23	1675.785	1676.9791	1820.1309	1787.4845	2040.0169	1937.55	1956.2797	2099.395	2206.8042
	720	1476.491	1555.198	1669.726	1672.073	1860.926	1848.242	1828.756	2112.5506	2044.46	2242.7467	2230.5497	2322.6472	2404.2112	2768.5331	2638.7475
	810	1735.13	1857.461	1940.209	2068.36	2185.607	2448.137	2420.216	2443.1928	2512.822	2507.1942	2702.5916	2801.8581	2778.8849	2904.0615	3083.2696
	900	2009.631	2254.45	2322.55	2634.164	2488.702	2651.068	2712.48	2737.5485	3093.3837	2982.9723	3008.5327	3120.4995	3574.8217	3507.6429	3399.5657
	990	2508.278	2388.51	2806.134	3177.08	3005.242	3076.322	3275.78	3483.9898	3240.4014	3462.5728	3504.1362	3633.8619	3930.4372	4098.5255	3836.8848
	1080	2897.854	2887.937	3198.04	3108.91	3269.508	3469.543	3432.586	3537.7654	3816.0319	3992.6376	4294.9226	4212.5433	4632.8918	4696.7991	4861.7814
	1170	3161.262	3386.059	3637.649	3773.754	4118.123	3769.38	4304.445	4345.7285	4437.1688	4903.082	5129.2507	4980.1799	5005.1245	5806.995	5677.1404
	1260	3359.691	3821.521	3684.337	4135.663	4344.353	4461.939	4595.556	4435.071	5055.3487	5397.1732	5681.4402	5476.6886	5973.2113	6221.5736	6393.3606
	1350	4108.273	3965.982	4274.497	4689.877	4643.517	4755.262	4872.598	5408.8714	5729.8799	6030.3064	6302.3378	6199.4862	6485.7718	6755.38	7063.2677
	1440	4574.359	4736.874	5452.238	5397.066	5242.746	5581.913	6093.675	6449.7378	6309.327	7092.0593	6555.9039	7194.6211	7439.4357	8046.2872	7901.6941
	1530	4619.831	5268.874	5622.509	5846.298	6131.543	6553.559	6236.616	6819.337	7210.3881	7144.1453	8111.9241	7797.5509	7632.5014	7567.6104	8404.7855
	1620	5509.016	6043.046	5971.225	6189.866	6151.883	6968.452	6749.801	7491.0898	7665.7813	7855.9889	8523.3922	8710.8912	8399.0337	9101.6023	9943.9943
	1710	6040.976	6355.796	6841.756	6499.442	7204.759	7683.432	8002.824	8542.8559	8881.8691	8684.6197	9046.1423	9578.9688	10119.318	9795.8629	10775.563
	1800	6791.393	6934.365	6896.68	7189.645	7695.733	8335.94	8818.958	9061.4097	8988.2311	9396.8981	10728.441	10287.322	10743.126	11726.342	12219.177
	1890	7479.365	6956.141	7830.047	8081.112	8298.523	8855.221	10048.67	11367.843	9778.7046	10808.874	10559.352	10967.396	10824.796	11804.809	11747.806
	1980	7179.365	8449.814	8550.694	9015.376	8540.465	9667.099	9573.85	10651.317	10386.313	11068.05	11330.524	12317.74	12714.494	12316.456	12806.684
	2070	9123.492	8916.886	9287.482	10192.4	10286.89	10419.55	11327.96	11758.819	11788.29	12839.371	13246.916	13742.314	14456.649	14359.119	13881.664
	2160	9148.169	9919.963	9968.854	10457.99	11124.78	12413.5	12709.49	12602.965	13513.371	13344.957	14279.705	13818.145	15556.515	16015.486	16561.042
	2250	9972.618	10985.51	11284.5	11387.9	12185.87	12422.64	13867.28	13188.823	14388.818	15484.122	15237.526	15400.115	16296.878	16685.239	16298.165
	2340	10092.32	11807.5	11527.62	12278.6	13314.36	13254.71	13505.85	14299.995	15250.628	14941.749	15536.382	16690.67	18182.511	18256.192	18997.128
	2430	11413.82	11996.37	12314.44	12639.17	12787.45	13918.52	14980.88	15698.294	16334.341	16096.707	15399.085	18129.898	18334.363	19582.202	20012.606
	2520	11697.19	12909.3	14085.65	13529.01	15237.58	16430.21	16144.51	16896.482	17223.722	18606.79	18280.355	18423.548	18526.54	19610.777	20937.795
	2610	13384.99	14137.22	14788.38	15824	15976.96	15399.52	17094.96	16803.995	17207.937	18777.278	19500.079	19899.786	21099.13	20660.34	23272.912
	2700	13715.64	14928.75	15823.38	15751.73	16540.39	17304.39	17897.01	18053.029	20308.929	20280.169	19867.441	21446.236	22088.017	23202.639	23066.36

Anexo 3: Puntajes obtenidos del alineamiento múltiple con el algoritmo centro estrella paralelo

Puntajes del alineamiento multiple con el algoritmo centro estrella																
		Número de secuencias														
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Longitud de secuncias	90	-151	-622	-988	-1474	-2941	-2898	-4043	-5536	-7626	-13311	-13330	-16487	-17559	-25034	-25665
	180	-342	-846	-1443	-3197	-5379	-7468	-8879	-13414	-14425	-23565	-32033	-35160	-37638	-48532	-49128
	270	-520	-2148	-2551	-5413	-10182	-9327	-14535	-21351	-27387	-32132	-46248	-52143	-50531	-56057	-75866
	360	-777	-1797	-3581	-5373	-7269	-15733	-23127	-26560	-38802	-45071	-59087	-61346	-75158	-102973	-120835
	450	-658	-2758	-4552	-6541	-10996	-13904	-20962	-29077	-44586	-53052	-70830	-91459	-97396	-113339	-155015
	540	-880	-2294	-6318	-10410	-18533	-22492	-33344	-40771	-46335	-62052	-81974	-118068	-138113	-150549	-182429
	630	-1412	-2485	-5681	-10144	-17389	-28135	-33064	-41697	-77982	-80974	-84624	-121664	-154119	-178990	-179038
	720	-1425	-3812	-6383	-10569	-22810	-23808	-33554	-50807	-59878	-97727	-97746	-137457	-139162	-190069	-236694
	810	-1388	-2970	-7850	-14708	-19251	-26867	-39555	-56294	-68721	-82884	-125533	-131969	-169429	-199209	-258988
	900	-1681	-3560	-7137	-15372	-21142	-36517	-45054	-76667	-73830	-96292	-123791	-153120	-208792	-228696	-302899
	990	-1303	-4373	-8340	-15225	-24403	-42214	-48170	-65592	-88780	-128932	-147245	-178683	-226690	-255084	-287412
	1080	-1573	-5297	-8979	-16730	-25844	-41729	-56502	-73894	-108435	-130638	-162501	-201745	-208928	-307394	-347998
	1170	-1910	-6342	-14870	-18498	-29766	-61408	-63564	-76242	-98589	-163589	-195660	-221383	-281440	-260544	-340568
	1260	-2337	-6915	-11226	-19449	-30561	-46584	-71622	-89670	-118019	-171207	-221898	-231758	-250358	-344778	-423845
	1350	-2985	-6898	-11687	-26424	-45241	-55098	-74013	-85440	-113818	-186936	-187661	-246921	-284833	-337693	-437872
	1440	-2158	-9239	-17274	-25078	-33476	-62228	-84273	-109080	-149153	-187362	-206731	-289162	-306548	-377499	-412049
	1530	-2786	-9769	-14622	-22663	-34763	-55801	-77114	-91631	-130726	-186403	-236772	-263586	-302742	-397991	-501802
	1620	-2665	-7777	-18117	-28886	-51398	-72928	-95012	-97350	-174234	-209128	-235296	-246079	-444922	-373492	-520077
	1710	-2976	-8164	-14056	-32232	-47736	-65050	-96369	-135456	-155914	-214637	-261679	-331520	-394088	-451945	-526563
	1800	-2528	-9060	-18464	-30361	-51232	-73590	-84698	-128594	-190128	-185321	-316305	-322911	-391572	-442630	-483743
	1890	-2550	-14362	-19183	-40022	-48614	-72327	-127817	-166711	-169960	-280856	-360311	-353224	-411465	-494994	-517492
	1980	-3802	-7181	-16364	-29483	-50392	-67277	-122246	-167262	-163682	-241794	-269773	-405658	-396586	-604793	-597292
	2070	-2582	-8247	-19013	-31579	-48146	-96942	-100700	-140276	-198492	-308284	-321423	-379665	-472745	-526051	-682484
	2160	-5402	-10118	-22837	-33566	-58800	-82591	-143647	-149026	-215638	-220232	-323406	-349855	-434622	-522488	-610645
	2250	-3030	-9592	-17149	-38015	-57990	-84876	-126106	-157863	-191411	-256050	-352523	-424074	-462213	-635502	-718511
	2340	-2852	-11013	-21719	-40825	-60186	-88401	-127325	-175128	-214250	-344799	-320162	-399643	-491985	-597871	-727273
	2430	-3114	-12556	-21353	-43582	-66045	-83427	-106206	-174159	-231374	-279066	-343434	-512771	-553055	-731548	-729457
	2520	-5057	-14911	-25474	-36419	-56514	-101550	-146568	-162106	-261765	-320820	-335541	-403244	-559983	-593462	-755706
	2610	-3614	-13153	-21114	-48584	-67584	-104433	-137155	-193345	-236501	-282406	-388110	-475109	-531255	-674813	-717439
	2700	-3673	-14618	-23715	-41491	-83161	-97734	-149077	-214465	-227560	-313920	-369570	-442652	-559088	-642369	-797091

Puntajes del alineamiento multiple con el algoritmo centro estrella																
		Número de secuencias														
		18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Longitud de secuncias	90	-34668	-37342	-42337	-48572	-60419	-63636	-60879	-81704	-88149	-98241	-98781	-96804	-112921	-135071	-132201
	180	-67435	-69763	-68314	-93777	-126067	-124353	-139715	-155422	-147759	-159207	-206412	-200342	-229766	-243228	-255905
	270	-93625	-122467	-121676	-123225	-169738	-177054	-168007	-234137	-218263	-247330	-268678	-306016	-368256	-361807	-433927
	360	-109110	-145467	-192369	-198284	-219249	-263627	-273761	-277811	-331527	-342635	-407339	-405842	-461685	-547834	-570065
	450	-167198	-168571	-217652	-219138	-276058	-309397	-346726	-389206	-360767	-465296	-502154	-488002	-594031	-531542	-703869
	540	-145274	-210420	-228544	-270579	-294413	-355493	-336444	-382759	-487669	-494581	-536980	-707111	-704888	-866706	-790440
	630	-260823	-255700	-297142	-314286	-401833	-454851	-436672	-474315	-554789	-623754	-640111	-737073	-811133	-955705	-947701
	720	-215885	-283469	-322111	-352717	-439921	-510160	-468584	-623294	-598578	-660062	-730123	-834940	-831067	-927512	-1061274
	810	-328274	-349223	-363646	-429419	-532902	-500841	-516881	-659375	-720901	-722749	-904609	-1005748	-1038492	-1142296	-1238534
	900	-313381	-342744	-350110	-568188	-523284	-601997	-636528	-746964	-842989	-803094	-927654	-1024972	-1133436	-1179285	-1447541
	990	-371470	-380632	-485506	-509197	-599529	-606795	-677319	-744393	-914208	-1031186	-1102066	-1125745	-1422602	-1414370	-1363779
	1080	-394384	-452240	-459940	-519368	-617462	-688338	-768364	-853434	-967030	-882025	-1186512	-1154487	-1478101	-1371995	-1739935
	1170	-387897	-477808	-538142	-585723	-647163	-757200	-886054	-1007798	-1023775	-1261601	-1165159	-1419293	-1477568	-1703325	-1619578
	1260	-435121	-478524	-646383	-561660	-835706	-792361	-898739	-968640	-1055792	-1128045	-1327223	-1528100	-1541204	-1548868	-1713874
	1350	-400403	-559497	-639628	-691283	-772291	-867651	-922861	-1043303	-1188477	-1397400	-1448345	-1786209	-1737915	-1907461	-1812031
	1440	-604359	-597833	-651131	-615011	-855898	-954256	-983155	-1118926	-1165910	-1298796	-1745034	-1644371	-1720936	-1907650	-2243898
	1530	-591518	-570389	-666300	-775952	-872629	-1004599	-1171758	-1326428	-1519261	-1578046	-1632414	-1770128	-2036352	-2074213	-2278535
	1620	-637074	-695605	-639675	-797095	-902838	-1034043	-1279094	-1313316	-1710300	-1448346	-1723645	-1958665	-1905201	-2129820	-2406346
	1710	-539264	-639849	-730305	-832613	-971704	-1100242	-1261289	-1441764	-1489030	-1624263	-1610596	-2092768	-1974501	-2502041	-2568994
	1800	-613614	-701950	-860700	-860136	-967607	-1188041	-1294376	-1689989	-1490498	-1882114	-1972721	-1995802	-2108517	-2405001	-2738084
	1890	-723185	-745039	-848176	-1060855	-1022935	-1153288	-1358321	-1435422	-1725914	-1768364	-1912677	-2368996	-2339570	-2595757	-2893910
	1980	-640680	-679564	-838949	-956535	-1114846	-1333713	-1389226	-1591199	-1775778	-1962879	-2000385	-2250992	-2607511	-2584105	-2864293
	2070	-782316	-793246	-1058157	-1107368	-1189149	-1439395	-1596688	-1598305	-1964327	-2066163	-2229306	-2213119	-2876817	-2711237	-2933905
	2160	-795729	-938396	-880571	-1161530	-1272309	-1327751	-1563984	-1771060	-1966885	-2083472	-2401962	-2451712	-2721853	-3071716	-3015350
	2250	-792912	-880502	-1055502	-1161250	-1172022	-1412474	-1654369	-1852451	-2219134	-2087472	-2424558	-2435746	-2738405	-2913279	-3203736
	2340	-871983	-972458	-1034941	-1169801	-1468505	-1433890	-1800073	-1873423	-2108425	-2305409	-2540718	-2399604	-2905288	-2879845	-3088144
	2430	-839978	-1060649	-1154931	-1175840	-1422925	-1703765	-1779970	-1840627	-1902627	-2402628	-2677059	-2836702	-3034546	-3395852	-4008744
	2520	-875395	-897115	-1131051	-1393801	-1460847	-1505652	-1807571	-1971154	-2132599	-2251907	-2538651	-2919390	-3118096	-3519950	-3658653
	2610	-923080	-1080469	-1207876	-1367934	-1637004	-1808256	-1937843	-2167730	-2287390	-2309297	-2917356	-2754308	-3291027	-3759398	-3805688
	2700	-900087	-1095991	-1306398	-1249385	-1497412	-1649666	-1976619	-2215557	-2522784	-2940814	-2837827	-3108517	-3280159	-3684767	-4390696

Anexo 4: Puntajes obtenidos del alineamiento múltiple con el algoritmo basado en colonia artificial de abejas

Puntajes del alineamiento multiple con el algoritmo basado en colonia artificial de abejas																
		Número de secuencias														
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Longitud de secuncias	90	-150	-528	-943	-1619	-2267	-2746	-3073	-4125	-5676	-6444	-6814	-8995	-11184	-10848	-13659
	180	-531	-1199	-1665	-3097	-3747	-5687	-6527	-9620	-8734	-12355	-16564	-18761	-21165	-24975	-29021
	270	-716	-1909	-2751	-4176	-6645	-8995	-12638	-14808	-15260	-19195	-23099	-27381	-31584	-38860	-41624
	360	-1121	-2436	-3821	-4496	-6076	-9530	-15588	-16112	-18733	-29507	-31762	-35366	-43637	-52394	-54991
	450	-1215	-2908	-5902	-8134	-9892	-12232	-21004	-22385	-31057	-35209	-33870	-47118	-61191	-63580	-66620
	540	-1547	-3362	-5951	-8502	-14108	-17383	-21958	-28659	-34168	-40117	-44531	-57754	-62419	-68984	-76715
	630	-2414	-2559	-7658	-9853	-14708	-18958	-26431	-27623	-35281	-45477	-57187	-58717	-78730	-88245	-107312
	720	-2457	-4195	-7191	-7898	-18588	-17733	-24491	-38236	-37560	-47296	-59451	-74080	-90481	-99035	-122052
	810	-2468	-4292	-8649	-12435	-17915	-23127	-38622	-32423	-51188	-62650	-68256	-85370	-105318	-120112	-110772
	900	-2897	-5449	-9143	-11723	-20989	-28421	-36448	-49831	-66895	-67118	-70417	-80560	-116334	-115646	-142199
	990	-1980	-6271	-10005	-15072	-24298	-33929	-36312	-47950	-74182	-79593	-83703	-107480	-101037	-124921	-151038
	1080	-2792	-6749	-8306	-13538	-27808	-33113	-38986	-51173	-65570	-79225	-84069	-118038	-127211	-153234	-165575
	1170	-3117	-6492	-14842	-18022	-23974	-38785	-47960	-67374	-64721	-90368	-114360	-93753	-145225	-162057	-166977
	1260	-4033	-6147	-14926	-16741	-26146	-39799	-45251	-51705	-78817	-90975	-107304	-131418	-143949	-168184	-189395
	1350	-5038	-7349	-10004	-20719	-33505	-42150	-51535	-63582	-78953	-105713	-120704	-128335	-158808	-198134	-191787
	1440	-4102	-10019	-16213	-26647	-34133	-44584	-62936	-76414	-85927	-111391	-138480	-137225	-182131	-168097	-226855
	1530	-5086	-9221	-20970	-26415	-30718	-55009	-60446	-66436	-88243	-105508	-115818	-157232	-198499	-210623	-252603
	1620	-4763	-8574	-16447	-24106	-38646	-49991	-67345	-71559	-87332	-118903	-141419	-201628	-182358	-206105	-257115
	1710	-5471	-10296	-13996	-30609	-29781	-58419	-83167	-89866	-111563	-129376	-154907	-184892	-195148	-231540	-276741
	1800	-4746	-12540	-25220	-31414	-38006	-56153	-64991	-91485	-121991	-116597	-140565	-203845	-215166	-246527	-293071
	1890	-4591	-12275	-16470	-35450	-34409	-65758	-82975	-103579	-116488	-146739	-171576	-213745	-205867	-269275	-312160
	1980	-5676	-9212	-18255	-26501	-41298	-56781	-75838	-95886	-138253	-160962	-199490	-199435	-219465	-259788	-336230
	2070	-4313	-9473	-21097	-38673	-50398	-67303	-74484	-101399	-138418	-150922	-173549	-210375	-210481	-265326	-306850
	2160	-8395	-12201	-22969	-35108	-45500	-61980	-87057	-103405	-139912	-166761	-200860	-216615	-252352	-286038	-292166
	2250	-5723	-15007	-17089	-35773	-61896	-71664	-80839	-126775	-156421	-167320	-208559	-216641	-253095	-243538	-353200
	2340	-4673	-12900	-21069	-36676	-60765	-69478	-96665	-112663	-141730	-183068	-209148	-216258	-318060	-278662	-367967
	2430	-5608	-19048	-27772	-33395	-59497	-73243	-98774	-133205	-147093	-176117	-196876	-239047	-314703	-343508	-379729
	2520	-9037	-15753	-27316	-34889	-58272	-71872	-103293	-130939	-150021	-181540	-223820	-275311	-313354	-338512	-388946
	2610	-6173	-14412	-27080	-41029	-62523	-83522	-104259	-129416	-121723	-210176	-248011	-255304	-345387	-399318	-318249
	2700	-6949	-16334	-30345	-29420	-69604	-82147	-110939	-136574	-160293	-180311	-251885	-302721	-324255	-368910	-409582

Puntajes del alineamiento multiple con el algoritmo basado en colonia artificial de abejas																
		Número de secuencias														
		18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Longitud de secuncias	90	-15431	-18125	-18350	-22741	-22634	-26009	-28670	-33016	-38428	-40093	-38184	-41558	-42689	-43200	-46727
	180	-29810	-31650	-43625	-37874	-45207	-55131	-57723	-66280	-71169	-75248	-84165	-80750	-88446	-93108	-101224
	270	-48083	-53465	-59597	-59872	-70095	-72133	-81781	-93247	-95789	-107049	-125669	-117609	-129736	-145695	-153884
	360	-64193	-54569	-74974	-82776	-102679	-95367	-108562	-123109	-147773	-143109	-153272	-166952	-185335	-180580	-199194
	450	-71312	-70956	-94657	-114936	-121881	-127316	-136969	-148108	-167582	-189617	-200489	-191999	-226339	-252952	-271737
	540	-103254	-110484	-101642	-120459	-134587	-143820	-176509	-191891	-211029	-203643	-239669	-236168	-265055	-296126	-293323
	630	-115651	-112052	-139148	-138144	-156852	-164909	-207983	-200014	-258239	-269655	-267096	-309885	-329439	-346269	-355317
	720	-127101	-137025	-158204	-182787	-186824	-211707	-211429	-265938	-254307	-285743	-313399	-317069	-358565	-410188	-397558
	810	-138457	-160519	-175326	-208186	-228367	-222764	-252635	-292100	-310196	-330016	-365064	-359721	-384827	-395310	-439809
	900	-137565	-189771	-171858	-213071	-224195	-271734	-268438	-310801	-332433	-343034	-393999	-402825	-448800	-455719	-481036
	990	-178158	-177607	-216422	-241943	-265942	-275506	-288798	-345584	-354715	-400862	-416944	-466926	-491071	-529388	-481013
	1080	-189579	-198113	-248835	-248077	-273444	-322013	-324859	-355354	-428650	-435916	-452350	-482601	-539205	-594099	-661234
	1170	-194922	-241230	-263923	-294894	-337338	-308818	-395272	-380380	-433605	-495860	-543597	-583406	-581325	-616771	-687773
	1260	-214225	-229608	-248317	-303298	-329421	-347621	-380541	-419374	-450764	-563298	-542232	-572832	-677420	-701476	-699751
	1350	-253525	-263337	-296208	-320683	-346516	-380082	-398825	-439085	-493776	-547917	-570705	-684111	-715171	-745655	-809125
	1440	-275431	-296591	-314188	-372152	-377950	-388931	-451674	-489132	-558802	-566243	-604171	-668857	-764957	-824546	-868908
	1530	-247648	-311922	-335264	-379593	-417400	-501992	-489310	-503998	-598877	-557162	-648740	-710864	-742386	-741673	-894610
	1620	-294696	-333538	-328124	-368160	-382616	-472091	-505819	-561385	-600323	-620203	-707786	-746335	-813034	-841938	-956960
	1710	-277175	-348512	-425199	-370347	-408084	-516972	-546109	-610647	-649670	-680143	-732570	-804866	-819489	-928968	-945046
	1800	-316747	-329628	-388390	-416722	-493647	-525142	-590603	-605309	-678087	-667035	-810690	-838071	-882277	-912542	-1022573
	1890	-366250	-347262	-409456	-467402	-487021	-585046	-526361	-694501	-697966	-774978	-786927	-881215	-859739	-999801	-1068215
	1980	-345689	-391746	-459424	-472938	-486996	-547258	-595927	-692731	-761308	-741715	-781416	-956668	-1013745	-1096124	-1094325
	2070	-335868	-381146	-434986	-502236	-532804	-606188	-676332	-680833	-765804	-829899	-897294	-1031986	-996924	-1211735	-1130686
	2160	-391926	-415273	-458368	-499378	-614312	-609083	-634089	-716257	-851491	-795297	-942129	-1043260	-1104504	-1098711	-1262102
	2250	-422841	-456298	-522749	-525563	-615383	-638949	-740671	-709338	-819696	-976915	-1007570	-1019093	-1129421	-1223518	-1170913
	2340	-371799	-527095	-504715	-514987	-640281	-687033	-725068	-832764	-823210	-928090	-937568	-1076420	-1199895	-1271312	-1482315
	2430	-429365	-453124	-516514	-570573	-578977	-679477	-758544	-855319	-854947	-992024	-1064681	-1163791	-1211129	-1291201	-1303141
	2520	-402541	-498076	-583907	-538309	-641183	-779331	-767346	-956653	-906688	-1020523	-1164402	-1086936	-1167549	-1311365	-1481070
	2610	-421768	-474113	-566375	-686187	-720423	-783933	-814704	-796869	-971487	-1035863	-1037066	-1275293	-1287761	-1348130	-1449697
	2700	-497397	-514119	-620416	-581461	-716557	-859731	-818431	-896456	-965066	-1149999	-1186501	-1215173	-1333987	-1427708	-1482819

Anexo 5: Código de la implementación de los algoritmos

```
private void btnejecutarstar_Click(object sender, EventArgs e)
        {
            tiempo = Stopwatch.StartNew();
            long i, j;
            long[,] val;
            string[, ,] seq;
           // string s1, s2;
            val = null;
            val = new long[p + 1, p + 1];
            seq = null;
            seq = new string[p + 1, p + 1, 3];
            //evaluando secuencias con Wunsh
            //paralelizable
            val[0, 0] = 0;
            for (i = 0; i < p - 1; i++)
            {

                Parallel.For(i+1,p, pl =>
                  {

                    wunsch(this.dgvseq.Rows[(int)i].Cells[0].Value.ToString
                    (),
                    this.dgvseq.Rows[(int)pl].Cells[0].Value.ToString(),
                    out val[i, pl], out seq[i, pl, 1], out seq[i, pl, 2],
                    out val[pl, i], out seq[pl, i, 2], out seq[pl, i, 1]);
                  });
            }
            val[p - 1, p - 1] = 0;
            //fin evaluación
            //Determinando indice de la secuencia principal
            long[] Sum;
            int c2;
            long MaxI;
            long Imax;
            Sum = null;
            Sum = new long[p + 1];
            c2 = 0;
            MaxI = 0;
            Imax = 0;
            //seccion no paralelizable
            for (i = 0; i <= p - 1; i++)
            {
                //
                for (j = 0; j <= p - 1; j++)
                {
                    Sum[i] = Sum[i] + val[i, j];

                }
                if (c2 == 0)
                {
                    MaxI = Sum[i];
                    Imax = i;
```

```
                    c2 = 1;
                }
                else
                {
                    if (MaxI < Sum[i])
                    {
                        MaxI = Sum[i];
                        Imax = i;
                    }
                }
            }
          //FIN de determinación de indice de secuencia principal

            //No paralelizable tiene dependencias
            string SP; //Secuencia principal
            if (Imax == 0) SP = seq[Imax, 1, 1];
            else SP = seq[Imax, 0, 1];

            for (i = 0; i < p - 1; i++)
            {
                if (i >= Imax) SP = UNIRGAPS(SP, seq[Imax, i + 1, 1]);
                else SP = UNIRGAPS(SP, seq[Imax, i, 1]);

            }

            // generando las secuencias alineadas finales
            string[] af;
            af = null;
            af = new string[p + 1];

            // combinable
           Parallel.For(0, p, pl =>
                  {
                      if (pl == Imax) af[pl] = SP;
                      else af[pl] = secf(SP, seq[Imax,pl, 1], seq[Imax, pl,
2]);
                  });
            // fin de la generación
            // mostrando alineamientos finales
            this.dgvalinearparallel.DataSource = null;
            this.dgvalinearparallel.Rows.Clear();
            for (i = 0; i < p; i++)
            {
                this.dgvalinearparallel.Rows.Add();
                this.dgvalinearparallel.Rows[(int)i].Cells[0].Value =
af[i];
            }

            // fin del programa
            tiempo.Stop();
            dt = tiempo.Elapsed;
```

```
            lblstarscoreparallel.Text = "El score obtenido es: " +
fscorestar().ToString();
            this.lbltstarmiliparallel.Text =
dt.TotalMilliseconds.ToString();
            this.lbltiempostarparallel.Text = dt.ToString();
        }
        private void wunsch(string sec1, string sec2, out long val, out
string Asec1, out string Asec2,out long vala, out String Asec1a, out string
Asec2a)
        {
            long n, m;
            long[,] f;
            long Gap;
            n = sec1.Length;
            m = sec2.Length;
            f = null;
            f = new long[n + 1, m + 1];
            Asec1 = "";
            Asec2 = "";
            //Matriz de puntaje
            long i, j;
            Gap = (long)this.NumGap.Value;
            for (i = 0; i <= n; i++)
                f[i, 0] = i * Gap;
            for (i = 0; i <= m; i++)
                f[0, i] = i * Gap;
            for (i = 1; i <= n; i++)
                for (j = 1; j <= m; j++)
                    f[i, j] = max(f[i - 1, j] + Gap, f[i - 1, j - 1] +
s(sec1, sec2, i, j), f[i, j - 1] + Gap);

            val = f[n, m];
            // fin de matriz
            //traceback
            i = n;
            j = m;
            while (i > 0 || j > 0)
            {
                if ((i > 0 && j == 0) || (i > 0 && j > 0 && (f[i, j] == f[i
- 1, j] + this.NumGap.Value)))
                {
                    Asec1 = Asec1 + sec1[(int)i - 1];
                    Asec2 = Asec2 + "-";
                    i = i - 1;
                }
                else
                {
                    if ((i == 0 && j > 0) || (i > 0 && j > 0 && f[i, j] ==
f[i, j - 1] + this.NumGap.Value))
                    {
                        Asec1 = Asec1 + "-";
                        Asec2 = Asec2 + sec2[(int)j - 1];
```

```
                    j = j - 1;
                }
                else
                {
                    Asec1 = Asec1 + sec1[(int)i - 1];
                    Asec2 = Asec2 + sec2[(int)j - 1];
                    i = i - 1;
                    j = j - 1;
                }
            }

        }
        Asec1 = strreverse(Asec1);
        Asec2 = strreverse(Asec2);

        // fin de traceback
        vala = val;
        Asec1a = Asec1;
        Asec2a = Asec2;

    }
    //Invierte el orden de la secuencia necesario para el traceback
para el Wunsch
    public static string strreverse(string s)
    {
        char[] arr = s.ToCharArray();
        Array.Reverse(arr);
        return new string(arr);
    }
    //funcion de similitud que nos indica el puntaje de dos caracteres
de acuerdo al valor del match, miss y gap
    private long s(string s1, string s2, long x, long y)
    {
        char a, b;
        a = s1[(int)x - 1];
        b = s2[(int)y - 1];
        if (a == b)
            return (long)this.NumMatch.Value;
        else
            return (long)this.NumMiss.Value;
    }
```

```
        private void btnbee_Click(object sender, EventArgs e)
        {

            tiempo = Stopwatch.StartNew();
            int i, j, nscout, nonlooker, nswarm;
            nscout = (int)NumScout.Value;
            nonlooker = (int)NumOnlookers.Value;
            nswarm = (int)NumSwarm.Value;
            string[] secuencias;//almacenamos las secuencias en secuencias
            string[] secaligrandom;//secuencias alineadas aleatoriamente
            secuencias = null;
            secaligrandom = null;
            secuencias = new string[p + 1];
            secaligrandom = new string[p + 1];
            string[,] WSiOL;
            long[, ,] evalgap;

            evalgap = null;
            evalgap = new long[nscout, p + 1, 2];
            EliteHiveevalgap = null;
            EliteHiveevalgap = new long[nswarm, p];
            WSiOL = null;
            WSiOL = new string[nscout, p + 1];
            EliteAliHive = null;
            EliteAliHive = new string[nswarm, p];
            int iw, im;//indice a trabajar e indice mayor
            long mineval, eval, gap; //Menor score de una secuencia de un
posible alineamiento, eval nueva evaluacion, gap cantidad de gaps de la
secuencia
            string newsec; //nueva secuencia generada por abejas
observadoras
            long Msol, sol; //score de un optimal list y mayor score de los
optimal list
            int iMsol;//indice del mayor score de la optimal list
            int ns;//numero de enjambre actual
            for (i = 0; i < p; i++)
            {
                secuencias[i] =
this.dgvseq.Rows[i].Cells[0].Value.ToString();
            }
            for (ns = 0; ns < nswarm; ns++)
            {
                //Abejas Exploradoras
                for (i = 0; i < nscout; i++)
                {
                    //Generar un alineamiento aleatorio
                    secaligrandom = null;
                    secaligrandom = new string[p + 1];
                    RandomAlineamiento(ref secuencias,out
secaligrandom,ns); //guardando el alieneamiento aleatorio
                    //Eliminar Gaps sobrantes, y construir la secuencia
favorita
```

```
                    string seqfav; //secuencia favorita
                    Cadfandsupgap(secaligrandom, out secaligrandom, out
seqfav);
                    //Evaluar las secuencias contra la secuencia favorita y
realizar conteo de gaps por fila
                    for (j = 0; j < p; j++)
                    {
                        evalseq(secaligrandom[j], seqfav, out evalgap[i, j,
0], out evalgap[i, j, 1]);
                        //Guardando el alineamiento en Working set in
optimization list
                        WSiOL[i, j] = secaligrandom[j];
                    }
                    WSiOL[i, p] = seqfav;
                }
                //Abejas Observadoras

                for (i = 0; i < nonlooker; i++)
                {
                    //Seleccion del indice aleatorio de un alineamiento
multiple de la optimazion list Wsiol
                    iw = Rand(0, nscout,ns);
                    //seleccion de la secuencia con el menor puntaje
                    mineval = evalgap[iw, 0, 0];
                    im = 0;
                    for (j = 1; j < p; j++)
                    {
                        if (mineval > evalgap[iw, j, 0])
                        {
                            mineval = evalgap[iw, j, 0];
                            im = j;
                        }
                    }
                    // Agregar y eliminar un gap de forma aleatoria
                    newsec = insasupagap((int)evalgap[iw, im, 1], WSiOL[iw,
im],ns);
                    //Evaluar la nueva secuencia contra la favorita
                    evalseq(newsec, WSiOL[iw, p], out eval, out gap);
                    //si es mayor la guardamos sino la desechamos
                    if (eval > evalgap[iw, im, 0])
                    {
                        evalgap[iw, im, 0] = eval;
                        WSiOL[iw, im] = newsec;
                    }
                }
                //Buscar el mayor score de los diferentes posibles
alineamientos multiples de un enjambre

                sol = 0;
                for (i = 0; i < p; i++)
                {
                    sol = sol + evalgap[0, i, 0];
```

```
                }
                Msol = sol;
                iMsol = 0;
                for (i = 1; i < nscout; i++)
                {
                    sol = 0;
                    for (j = 0; j < p; j++)
                    {
                        sol = sol + evalgap[i, j, 0];
                    }
                    if (sol > Msol)
                    {
                        Msol = sol;
                        iMsol = i;
                    }
                }
                for (i = 0; i < p; i++)
                {
                    EliteHiveevalgap[ns, i] = evalgap[iMsol, i, 0];
                    EliteAliHive[ns, i] = WSiOL[iMsol, i];
                }

            }
            //Buscar el mayor score de los diferentes posibles
alineamientos multiples de un panal
            sol = 0;
            for (i = 0; i < p; i++)
            {
                sol = sol + EliteHiveevalgap[0, i];
            }
            Msol = sol;
            iMsol = 0;
            for (i = 1; i < nswarm; i++)
            {
                sol = 0;
                for (j = 0; j < p; j++)
                {
                    sol = sol + EliteHiveevalgap[i, j];
                }
                if (sol > Msol)
                {
                    Msol = sol;
                    iMsol = i;
                }
            }

            //Mostrar la mejor solucion del panal
            this.dgvbee.DataSource = null;
            this.dgvbee.Rows.Clear();
            for (i = 0; i < p; i++)
            {
                this.dgvbee.Rows.Add();
```

```
                this.dgvbee.Rows[(int)i].Cells[0].Value =
EliteAliHive[iMsol, i];
            }
            tiempo.Stop();
            dt = tiempo.Elapsed;
            lblbeescore.Text = "El score obtenido es:" +
fscorebee().ToString();
            lbltbeemili.Text = dt.Milliseconds.ToString();
            this.lbltiempobee.Text = dt.ToString();
        }
        private void RandomAlineamiento(ref string[] seq, out string[]
seq2,int ns)
        {
            double IGMax; //Incremento de gaps maximo
            IGMax = Math.Round(MaxSec * (float)this.NumpGap.Value / 100,
0);
            seq2 = new string[p];
            int IGAM = Rand(0, (int)IGMax + 1,ns); // Incremento de gaps
aleatorio Maximo
            for (int i = 0; i < p; i++)
            {
                seq2[i] = seq[i];
                seq2[i] = ADDGAP(MaxSec + IGAM - seq2[i].Length,
seq2[i],ns);
            }

        }
        private string ADDGAP(int numgap, string sec,int ns)
        {
            int i;


            for (i = 1; i <= numgap; i++)
            {
                sec = sec.Insert(Rand(0, sec.Length + 1,ns), "-");
            }
            return sec;
        }
        private void Cadfandsupgap(string[] seq, out string[] seq2, out
string seq3) //Cadena favorita y sup gaps en toda la columna
        {
            int i, j, L, mayor, imayor;
            int[] num = new int[5];
            char[] letras = new char[] { 'A', 'C', 'T', 'G' };
            L = seq[0].Length;
            seq3 = "";
            seq2 = seq;
            for (i = 0; i < L; i++)
            {
                for (int k = 0; k < 4; k++)
                    num[k] = 0;
                for (j = 0; j < p; j++)
```

```
                {
                    switch (seq[j][i])
                    {
                        case 'A': num[0] = num[0] + 1; break;

                        case 'C': num[1] = num[1] + 1; break;
                        case 'T': num[2] = num[2] + 1; break;
                        case 'G': num[3] = num[3] + 1; break;

                    }

                }
                mayor = num[0];
                imayor = 0;

                for (j = 1; j < 4; j++)
                {
                    if (mayor < num[j])
                    {
                        mayor = num[j];
                        imayor = j;
                    }

                }
                if (mayor == 0)
                {
                    for (j = 0; j < p; j++)
                    {
                        seq2[j] = seq2[j].Remove(i, 1);
                    }
                    L = L - 1;
                    i = i - 1;
                }
                else seq3 = seq3 + letras[imayor];

            }
        }
        // FIN CADENA FAVORITA
        //evaluar score por columnas y evaluar columna de counter
        private void evalseq(string s1, string sf, out long eval, out long
gap)
        {
            int i, L;
            L = s1.Length;
            eval = 0;
            gap = 0;
            for (i = 0; i < L; i++)
            {
                eval = eval + seval(s1, sf, i);
                if (s1[i] == '-') gap = gap + 1;

            }
```

```
        }
        private string insasupagap(int Gapt, string sec,int ns) // insertar
aleatoriamente y eliminar  aleatoriamente un gap
        {
            int iig, eg, ge, lsec;//indice de insercion de gap, eliminar
gap, gap eliminar contador

            if (Gapt != 0)
            {
                lsec = sec.Length;
                iig = Rand(0, lsec + 1,ns);
                sec = sec.Insert(iig, "-");
                eg = Rand(1, Gapt + 1,ns);
                ge = 0;
                for (int i = 0; i <= lsec; i++)
                {
                    if (sec[i] == '-')
                    {
                        ge = ge + 1;
                        if (i == iig) ge = ge - 1;
                        else if (ge == eg)
                        {
                            sec = sec.Remove(i, 1);
                            i = lsec;
                        }

                    }
                }
            }
            return sec;
        }
```

Anexo 6: Paper presentado al COMTEL 2016

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 habia que darle importancia a la comunicación entre los procesadores, pero en la actualidad ya existen librerias 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]

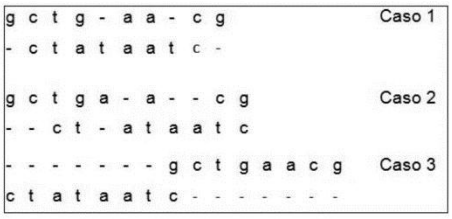

Figura 1. Ejemplos de alineamiento.

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 (Missmatch) 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*), missmatch (*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 x 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]

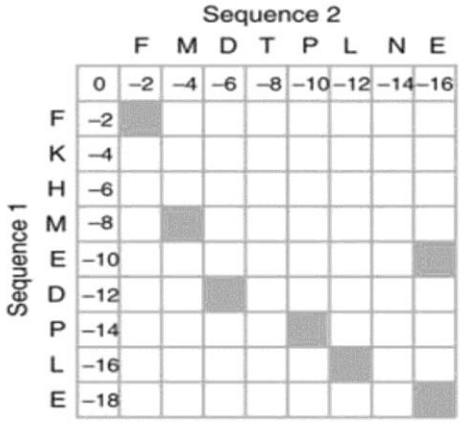

Figura 3. Inicialización con gap.

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

Figura 4. Función del alineamiento global

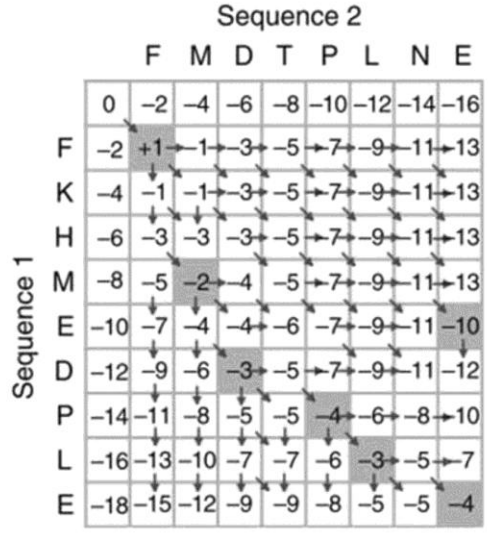

Figura 5. Llenado de la matriz de scores.

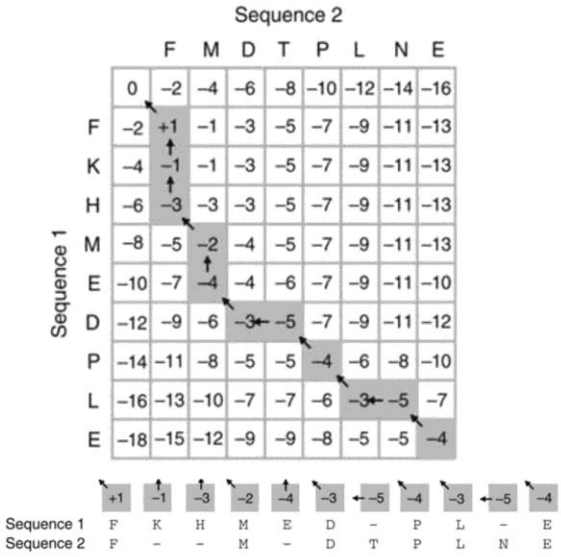

Figura 6. Traceback (recorrido hacia atrás).

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]

```
MQPILLLV
MLR-LL--
MK-ILLL-
MPPVLILV
```

$$\begin{aligned} SP(I,\text{-},I,V) &= sc(I,\text{-}) + sc(I,I) + sc(I,V) + \\ &\quad sc(\text{-},I) + sc(\text{-},V) + sc(I,V) \\ &= -2+1-1-2-2-1 \\ &= -7 \end{aligned}$$

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.

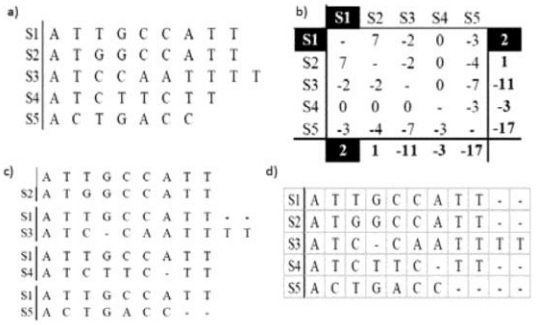

Figura 8. Ejemplo de uso del algoritmo centro estrella.

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 algoritmica 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 necsario 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
      Si s1[i] = s2[i - g] entonces
        fs = fs + s3[i - g]
      Si no
        fs = fs + "-"
        g = g + 1
      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 minima 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 adapto 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.

A	G	T	C	A	A	T
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 letras(A, C, T, G)
La función Remover() que quita una letra de una posición
especifica
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 missmatch 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	3
1	1	1	1	1	-1	1	6
1	1	0	-1	1	1	-1	2

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 caracter a caracter, 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] != '-' **entonces**
P= Match
Si no
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
A-TG-C-GGTA-CCGTG
c) A-TGC-GGTA-CCGT-G
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 != 0 **entonces**
lsec = Longitud(sec)
iig = Rand(0, lsec + 1)
InsertarT(sec, iig, "-")
eg = Rand(1, Gapt + 1)
ge = 0
Para i = 0 **Hasta** lsec **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 = lsec
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.

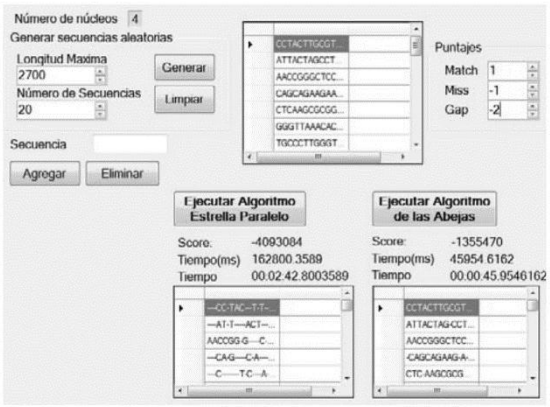

Figura 12. Interfaz de la aplicación de alineamiento de secuencias con los tiempos empleados.

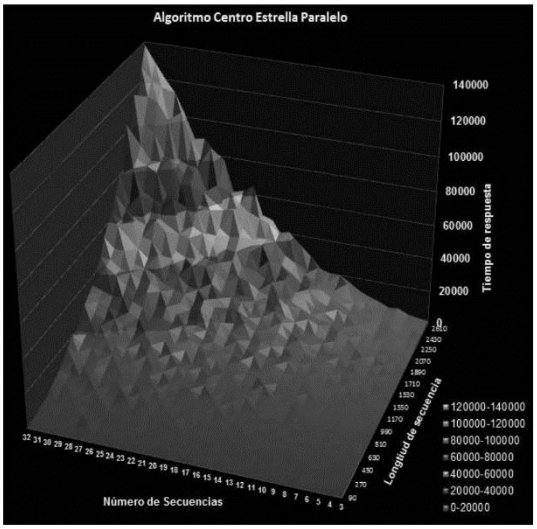

Figura 13. Tiempos de ejecución para el algoritmo centro estrella paralelo según la longitud y número de secuencias a alinear.

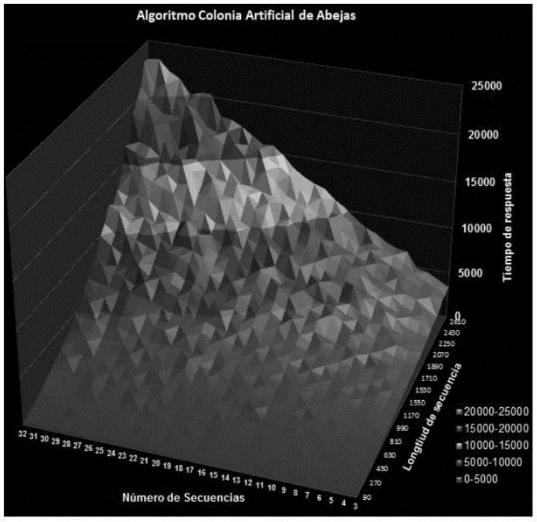

Figura 14. Tiempos de ejecución para el basado en colonia artificial de abejas según la longitud y número de secuencias a alinear.

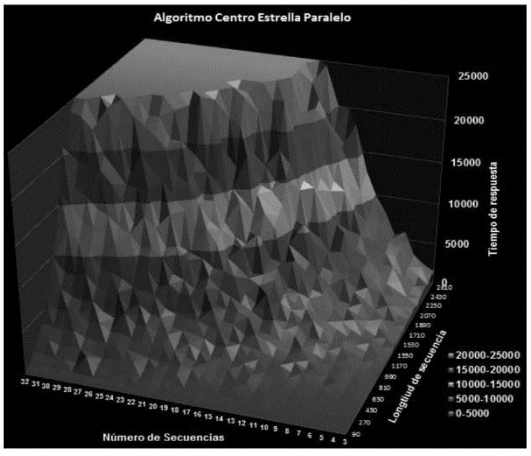

Figura 15. Tiempo de ejecución del algoritmo centro estrella paralelo con tiempos de ejecución menores a 25000 milisegundos según la longitud y número de secuencias a alinear

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 apllications, 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 Aligment 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.