https://doi.org/10.69639/arandu.v13i1.2009

Algoritmo de Forrajeo de Bacterias para el Alineamiento Múltiple de Secuencias Relacionadas con Alzheimer

Bacterial Foraging Algorithm for Multiple Alignment of Alzheimer's-Related Sequences

Erik Iván Toledo Galeana
toledo.e@uadec.edu.mx
Universidad Autónoma de Coahuila, Facultad de Sistemas
Saltillo Coahuila, México

María Magdalena Delabra-Salinas
https://orcid.org/0000-0002-4467-191X
magdalena-delabra-salinas@uadec.edu.mx
Universidad Autónoma de Coahuila, Facultad de Enfermería
Saltillo Coahuila, México

Ernesto Rios Willars
riose@uadec.edu.mx
https://orcid.org/0000-0002-7157-8884
Universidad Autónoma de Coahuila, Facultad de Sistemas
Saltillo Coahuila, México

Artículo recibido: 18 enero2026-Aceptado para publicación: 20 febrero 2026
Conflictos de intereses: Ninguno que declarar.

RESUMEN

El objetivo de este trabajo es desarrollar un algoritmo de forrajeo de bacterias para el alineamiento múltiple de secuencias genéticas. La metodología consiste en comparar el desempeño de dos versiones del algoritmo (bactOf y bactAtp) en el alineamiento de tres conjuntos de secuencias de genes (A,B y C) relacionados con el AlzHeimer. Dichos genes son TREM2, EPHA1 y ABCA7 de diferentes organismos como Homo sapiens, Mustela putorius furo, Pongo abelii y Cervus elaphus, entre otros. Los resultados muestran que en el set A, el promedio de fitness de bactOf comienza en 1097.86 y aumenta gradualmente hasta alcanzar 1431.87 en la iteración 30, mientras que bactAtp inicia en 1570.53 y aumenta gradualmente hasta estabilizarse en 1596.10 a partir de la iteración 20. En el set B, bactAtp comienza con un promedio de fitness de 5636.59, mientras que bactOf inicia en 3626.53, lo que indica que bactAtp tiene un valor de fitness inicial significativamente superior. Finalmente, en el set C, bactAtp comienza con un valor de fitness mucho más alto (3239.54) que bactOf (492.07). Esta tendencia de mayor fitness en bactAtp se mantiene constante a lo largo de las iteraciones. Por otro lado, la cuantificación del número de funciones evaluadas (NFE) indica que en los Sets A y B, ambos algoritmos presentan un NFE idéntico de 653. Conclusiones: aunque ambos algoritmos, bactOf y bactAtp, son comparables en eficiencia de evaluación, bactAtp muestra un mejor desempeño en términos de fitness y rapidez de convergencia; esto es más notorio en el Set B. Esto se debe a su distribución aleatoria de gaps

que optimiza la diversidad genética y evita patrones no deseados, lo que permite una mejor exploración y evita mínimos locales.

Palabras clave: BFOA, Alzheimer, Algoritmo, MSA

ABSTRACT

The aim of this work is to develop a bacterial foraging algorithm for multiple sequence alignment. The methodology consists of comparing the performance of two versions of the algorithm (bactOf and bactAtp) in the alignment of three sets of gene sequences (A, B and C) related to Alzheimer's. These genes are TREM2, EPHA1 and ABCA7 from different organisms such as Homo sapiens, Mustela putorius furo, Pongo abelii and Cervus elaphus, among others. The results show that in set A, the average fitness of bactOf starts at 1097.86 and gradually increases until it reaches 1431.87 in iteration 30, while bactAtp starts at 1570.53 and gradually increases until it stabilizes at 1596.10 from iteration 20. In set B, bactAtp starts with a fitness average of 5636.59, while bactOf starts at 3626.53, indicating that bactAtp has a significantly higher starting fitness value. Finally, in set C, bactAtp starts with a much higher fitness value (3239.54) than bactOf (492.07). This trend of increased fitness in bactAtp remains constant throughout the iterations. On the other hand, the quantification of the number of evaluated functions (NFE) indicates that in Sets A and B, both algorithms have an identical NFE of 653. Conclusions: Although both algorithms, bactOf and bactAtp, are comparable in assessment efficiency, bactAtp shows better performance in terms of fitness and speed of convergence; this is more noticeable in Set B. This is due to its random distribution of gaps, which optimizes genetic diversity and avoids unwanted patterns, allowing better exploration and avoiding local minima.

Keywords: BFOA, Alzheimer, Algorithm, MSA

INTRODUCCIÓN

El alineamiento de secuencias genéticas es un concepto fundamental en bioinformática que implica comparar secuencias de ADN, ARN o proteínas para identificar regiones relacionadas desde una perspectiva funcional, estructural o evolutiva. Los alineamientos múltiples son una base crucial para análisis posteriores, como la evolución molecular, la predicción de genes y la identificación de variantes genéticas (Amorim et al., 2021). Los algoritmos de optimización metaheurística desempeñan un papel importante en la mejora de la alineación de secuencias. El algoritmo tradicional de programación dinámica (Lipman et al., 1989) para dos secuencias es demasiado lento cuando se usa con más de tres secuencias. Por esto los algoritmos metaheurísticos han llegado a ser fundamentales para resolver múltiples problemas de alineación en bioinformática (Zambrano-Vega et al., 2017).

Algoritmo de Forrajeo de Bacterias (BFOA)

Es un enfoque bioinspirado basado en el comportamiento de forrajeo de la *bacteria Escherichia coli*. Este algoritmo optimiza funciones simulando el movimiento de las bacterias en busca de nutrientes (Guo et al., 2021). El movimiento de la E. coli tiene dos características principales: nado y tumbo. Estos movimientos son posibles gracias a los flagelos de la bacteria. Los pasos básicos del algoritmo se describen a continuación.

El nado*:* En el algoritmo BFOA, el nado se replica haciendo que las bacterias (es decir, las posibles soluciones) se muevan hacia los nutrientes (soluciones cada vez mejores) y evaluando si mejora en cada movimiento (Hernández-Ocaña et al., 2016).

El tumbo: Cuando *E. coli* detecta un ambiente desfavorable o busca cambiar de dirección, los flagelos giran, haciendo que la bacteria gire, cambiando su trayectoria. En BFOA, este comportamiento se simula con cambios aleatorios en la solución (Pang et al., 2018).

Quimiotaxis: La simulación del proceso de quimiotaxis del BFOA es una parte. Cada ciclo representa una iteración del algoritmo, en el que las bacterias se mueven a través del espacio de búsqueda para encontrar mejores soluciones (Xu & Chen, 2014).

Evaluación*:* En la naturaleza, la supervivencia del más apto define las características genéticas dignas de ser transmitidas a las nuevas generaciones. Esto se simula en el BFOA sometiendo las posibles soluciones a una evaluación en la que se ve la calificación (fitness) (Chen et al., 2017).

La función de interacción se calcula de acuerdo con la expresión *g()*.

$$g(cell_k) = \sum_{i=1}^{S} \left[-d_{attr} * e^{(-w_{attr} * \sum_{m=1}^{P} (cell_m^k - other_m^i)^2)} \right] + \sum_{i=1}^{S} \left[h_{repel} * e^{(-w_{repel} * \sum_{m=1}^{P} (cell_m^k - other_m^i)^2} \right]$$

La primera célula representa una bacteria, mientras que la segunda célula representa una célula vecina. "dattr" y "wattr" son coeficientes de atracción, y "hrepel" y "wrepel" son

coeficientes de repulsión. "S" representa el número de bacterias en la población, y "P" significa el número de dimensiones en el problema de optimización.

Reproducción: En BFOA, las soluciones mejor evaluadas se clonan, las soluciones menos efectivas se eliminan de la memoria y se repite el proceso de búsqueda (Biswas et al., 2010).

Eliminación y dispersión: Este proceso elimina resultados poco prometedores que, en lugar de converger en un resultado deseado, se alejan de él.

MATERIALES Y MÉTODOS

Para la conformación de los conjuntos de secuencias de prueba, se consideraron aquellos genes relacionados con la enfermedad de Alzheimer.

Variantes raras de TREM2 están asociadas con un mayor riesgo de desarrollar Alzheimer, afectando la respuesta de la microglía a las placas de amiloide-β y la senescencia microglial. Para integrar el set A, se obtuvieron secuencias de la base de datos NCBI correspondientes a este gen de los organismos Homo Sapiens, Mus musculus, Rattus norvegicus y Mandrillus leucophaeus.

Así mismo, el gen EPHA1, que codifica el receptor tipo A de la familia de las efrinas, ha sido objeto de diversos estudios debido a su implicación en esta enfermedad. Con secuencias de este gen correspondientes a organismos Homo sapiens, Mus musculs, Pan troglodytes se integró el conjunto B de secuencias. Finalmente, se integró el conjunto C, considerando el gen ABCA7, identificado como un factor de riesgo significativo para la enfermedad de Alzheimer. Los organismos que componen este conjunto son Homo sapiens, Mustela putorius furo, Pongo abelii y Cervus elaphus. En la tabla 1 se describen los conjuntos de A a C en este trabajo.

Tabla 1

Secuencias genéticas integradas en el conjunto A al C

Set	Sequence 1	Sequence 2	Sequence 3	Sequence 4
A	AAH32362.1	Q99NH8.1	NP_001418899.1	XP_011842060.1
B	EAL23789.1	NP_076069.2	XP_009452691.2	
C	NP_061985.2	XP_044943937.1	XP_063575886.1	OWK12339.1

Las secuencias genéticas y los algoritmos desarrollados en este trabajo están disponibles en el proyecto gitHub: https://github.com/Frederick161021/BFOA_Repository

Se han desarrollado dos algoritmos que a continuación se describen.

Algoritmo BactFo

Se inicializa el número de bacterias y parámetros de operación. Así como la matriz que contienen las secuencias, y los parámetros del puntuaje: *blsumscore*, *fitness* y NFE. Se rellenan con gaps las secuencias más cortas, se hace una copia de la matriz, se define aleatoriamente el numero (entero) de gaps que se insertarán, que pueden ir de 0 a el numero definido al inicio que limita el número máximo de gaps agregados por ciclo. La bacteria es evaluada en *autoEvalua*() en el que se asigna una calificación, para primero contar el número de gaps en la columna y luego

quitarlos y checar el número de pares, los pares al evaluador que crea la matriz BLOSUM62. Al finalizar el ciclo, aumenta en 1 el número de funciones evaluadas. Se agregan entre todas las bacterias la que mejor calificación obtuvo y se imprimen las características de la bacteria, su interacción, *fitness* y su NFE, después se eliminan la mitad de la población de bacterias con la menor calificación y se duplica el número de bacterias con mayor *fitness*, a su vez insertando una nueva bacteria completamente aleatoria.

Algoritmo BactAtp

Después de haber analizado el algoritmo de BactFo, se logró visualizar un punto de mejora con la modificación del método "*cuadra*()". Se observó que, al momento de cuadrar las longitudes de las secuencias genéticas se genera una gran acumulación de gaps al final de las secuencias que podrían ser aprovechados en otros lugares de la secuencia, en una posición aleatoria. Esto puede ofrecer las siguientes ventajas: Diversidad genética, al distribuir los gaps de manera aleatoria. Evitar Patrones, la acumulación de gaps al final podría inducir patrones no deseados en la matriz de secuencias. Mejora en la optimización, la modificación puede contribuir a una mejor exploración del espacio de búsqueda.

Uno de los principales problemas es que los gaps son elementos que afectan negativamente al puntaje (fitness). Al distribuir los gaps aleatoriamente dentro de las secuencias, haciéndolas menos representativas de soluciones de alta calidad. Esto provoca que las secuencias con muchos gaps tengan un impacto negativo en la evaluación de su fitness. Para abordar este problema se hicieron dos versiones del método cuadra: *cuadra1* (el método original que coloca los gaps al final de las secuencias) y *cuadra2* (el método que coloca los gaps de forma aleatoria). A continuación, se detalla cómo funciona esta solución y por qué es efectiva: En primer lugar, se utiliza *cuadra1* para cuadrar la matriz de secuencias, lo que implica que los gaps se añaden al final de las secuencias más cortas. Esto permite que la matriz mantenga la mayor cantidad posible de información relevante en las posiciones iniciales de las secuencias.

La combinación de *cuadra1* y *cuadra2* en el método de "tumbo nado" permite que el BFOA elimine rápidamente las columnas de gaps al tiempo que mantiene la diversidad necesaria para una exploración efectiva del espacio de soluciones. Con base en las observaciones, se realizan nuevos cambios en el algoritmo, como son el aumento de la población de bacterias (de 6 a 15): Incrementa la diversidad, permitiendo una mejor exploración del espacio de soluciones y reduciendo el riesgo de quedarse en mínimos locales. Más bacterias aleatorias (de 1 a 2): Mantiene la diversidad y ayuda a escapar de mínimos locales. Parámetros de atracción (*dAttr = 0.1, wAttr = 0.2*): Fomentan que las bacterias se agrupen en torno a buenas soluciones sin aglomerarse demasiado rápido. Mayor repulsión (*hRep = 0.2, wRep = 13*): Evita que las bacterias se concentren en un solo lugar, asegurando una mejor exploración del espacio. En el método Cuadra2, primero, se implementó un control de posiciones de gaps para asegurar que no se inserten consecutivamente, y evitar que empeore la calidad del alineamiento.

Experimento

Ambos algoritmos alinean los conjuntos de secuencias, y cada proceso de alineación se repite 30 veces para calcular el valor promedio. La primera métrica de rendimiento que estamos analizando es *NFE*, que es el número de evaluaciones de funciones. El objetivo es que el algoritmo disminuya el valor de *NFE* para mejorar su eficiencia computacional. Un presupuesto de evaluación más extenso tiene un impacto significativo en la comparación de las metaheurísticas, lo que afecta el rendimiento, la convergencia promedio y la diversidad de la población. La segunda métrica de rendimiento que nos interesa es el valor de aptitud de las mejores bacterias en las iteraciones del algoritmo. El factor de peso determina el valor de aptitud y se utiliza para resolver problemas complejos de optimización. Por otro lado, los parámetros con los que los algoritmos trabajaron son los indicados en la tabla 2.

Tabla 2

Parámetros de operación para ambos algoritmos en la comparativa de desempeño con los sets de prueba

Algoritmo	d_{attr}	w_{attr}	h_{repel}	w_{repel}	Número de Bacterias	Iteraciones	Bacterias eliminadas	Número de gaps a insertar por tumbo
BactOf	0.1	0.2	0.2	10	16	30	1	1
BactAtp	0.1	0.2	1	15	16	30	1	1

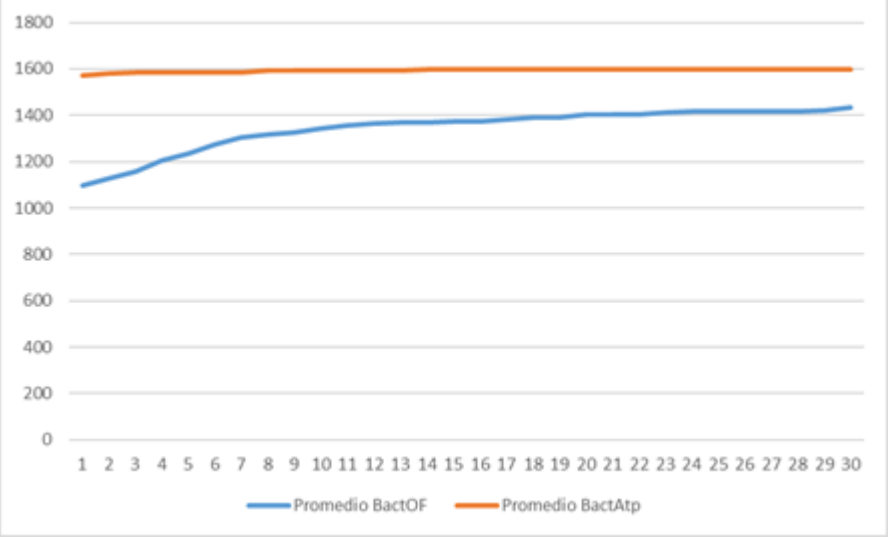
RESULTADOS Y DISCUSIÓN

En la figura 1 se observan los valores de fitness promedio para los algoritmos bactOf y bactAtp a lo largo de 30 iteraciones. Los resultados muestran que bactAtp tiene un valor inicial de fitness más alto que bactOf y se mantiene consistentemente superior en todas las iteraciones. El promedio de fitness para bactOf comienza en 1097.86 y aumenta gradualmente hasta alcanzar 1431.87 en la iteración 30, mientras que bactAtp inicia en 1570.53 y sube lentamente hasta estabilizarse en 1596.10 a partir de la iteración 20.

A lo largo de las iteraciones, el algoritmo bactAtp presenta valores de fitness que varían ligeramente y tienden a estabilizarse antes que bactOf. Esta estabilización y la menor variación en las iteraciones finales sugiere que bactAtp converge más rápido en términos de fitness en comparación con bactOf.

Figura 1

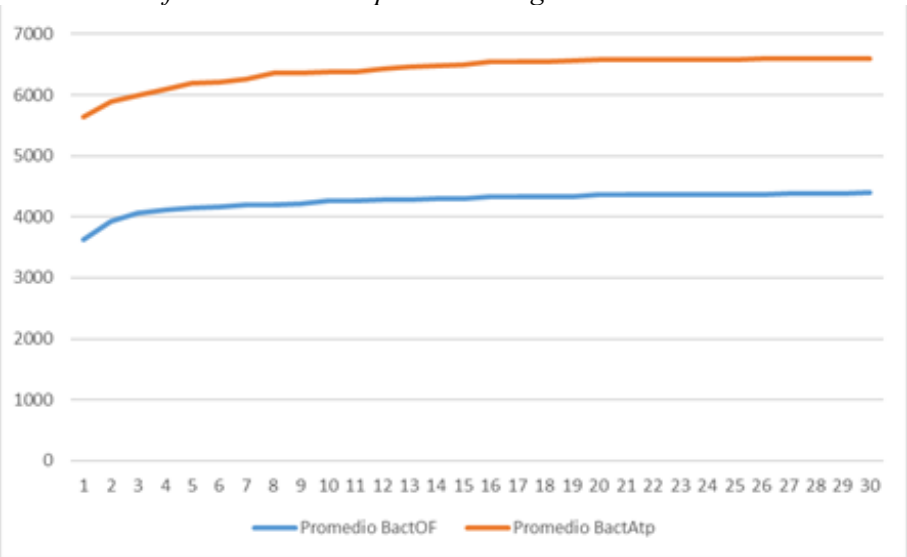
Promedios de fitness para ambos algoritmos en el set A



La figura 2 ilustra el promedio *Fitness* Set B, bactAtp comienza con un promedio de fitness de 5636.59, mientras que bactOf inicia en 3626.53, mostrando nuevamente que bactAtp tiene un valor de fitness inicial significativamente superior.A lo largo de las iteraciones, ambos algoritmos incrementan sus valores de fitness, pero bactAtp mantiene una ventaja constante sobre bactOf. En la iteración final, bactAtp alcanza un promedio de 6604.24 mientras que bactOf llega a 4398.12. Los incrementos de fitness en bactAtp son notoriamente más altos, estabilizándose alrededor de la iteración 18, donde los valores de fitness empiezan a oscilar levemente alrededor de 6573-6604. BactOf, aunque también presenta una tendencia de crecimiento, muestra una estabilización más lenta y en niveles inferiores comparado con bactAtp.

Figura 2

Promedio del fitness alcanzado por ambos algoritmos en el set B.

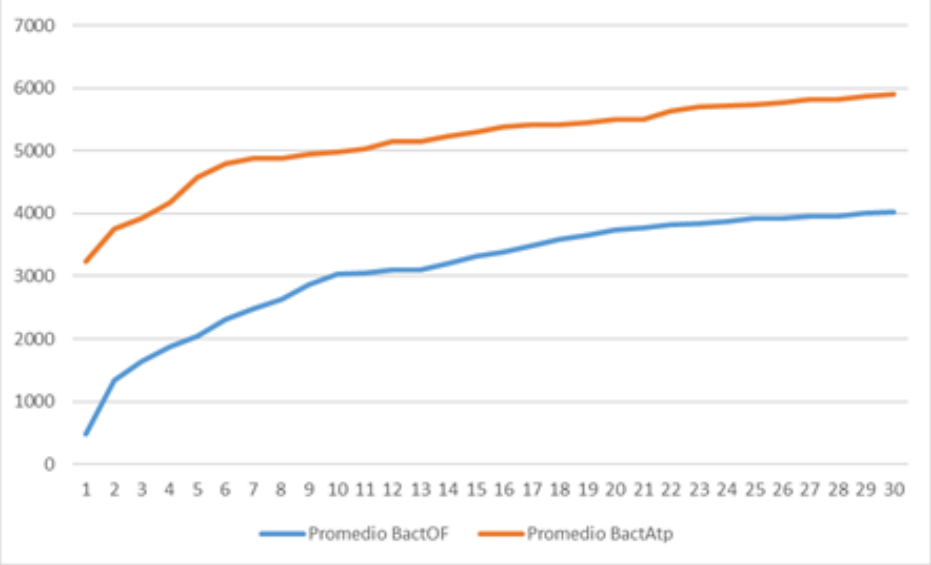


En la figura 3 se muestra el promedio Fitness par el Set C, bactAtp comienza con un valor de fitness mucho más alto (3239.54) en comparación con bactOf (492.07). Esta tendencia de mayor fitness en bactAtp se mantiene constante a lo largo de las iteraciones.

Ambos algoritmos muestran un crecimiento sostenido en sus valores de fitness. Sin embargo, bactAtp mantiene una ventaja notable y consistente en cada iteración, alcanzando un valor final de 5901.30, mientras que bactOf llega a 4027.77. La brecha entre ambos algoritmos se amplía de forma gradual, lo cual indica que nuevamente bactAtp converge a valores de fitness significativamente superiores y con más rapidez.

Figura 3

Promedio del fitness de ambos algoritmos para el set C



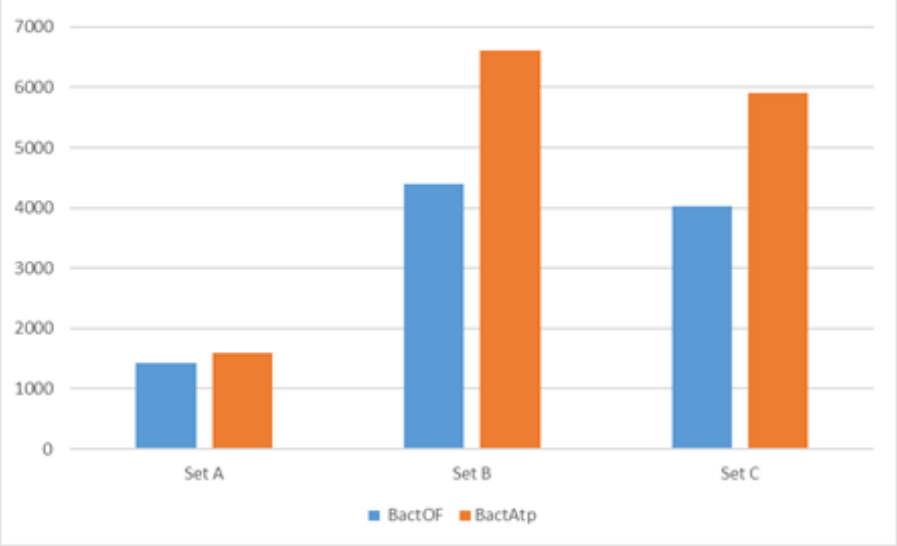
En la figura 4," se presentan los promedios finales de las calificaciones de fitness, esto con el objetivo de comparar los resultados obtenidos en los algoritmos BactOF y BactAtp evaluados en los tres sets (A, B y C). Los valores muestran que BactAtp supera a BactOF en todos los sets.

Para el **Set A**, BactAtp alcanzó un promedio de fitness ligeramente superior al de BactOF. Esto indica que, en este set, ambos algoritmos convergen a valores de fitness más cercanos, sugiriendo un rendimiento relativamente similar. En el **Set B**, la diferencia se hace mucho más notable pues BactAtp alcanzó una diferencia destacable a los resultados obtenidos de BactOF en los tres sets de prueba realizados. Esta amplia diferencia sugiere que BactAtp es significativamente más efectivo debido a las características de las secuencias genéticas del archivo FASTA correspondiente, las cuales parecen favorecer la estrategia de BactAtp.

Finalmente, en el **Set C**, BactAtp nuevamente supera a BactOF. Aunque BactAtp sigue demostrando superioridad, esta diferencia es intermedia entre las observadas en los sets A y B. Podríamos concluir con esto que BactAtp muestra consistentemente un mejor desempeño en términos de fitness en los tres sets, siendo especialmente notable en el Set B, lo cual puede sugerir que BactAtp es más eficiente para ciertos tipos de estructuras genéticas.

Figura 4

Comparativa de fitness entre algoritmos para cada set de secuencias.

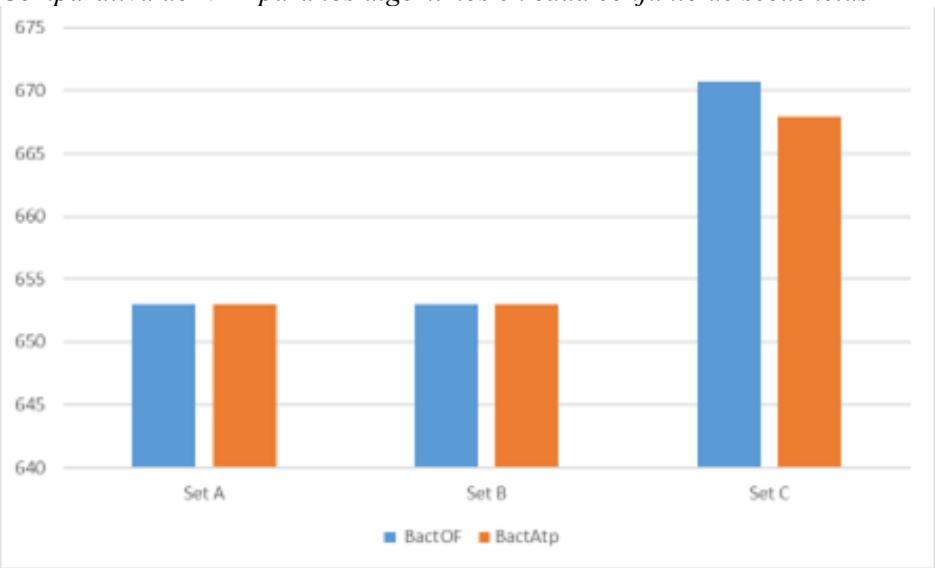


Número de Funciones Evaluadas

La figura 5 muestra el NFE para los algoritmos BactOF y BactAtp, en los Sets A y B, ambos algoritmos presentan un NFE idéntico de 653, lo que sugiere que, en estos conjuntos, tanto BactOF como BactAtp realizaron la misma cantidad de evaluaciones para alcanzar sus soluciones finales. Para el Set C, se observa una ligera diferencia en el NFE. Esta diferencia puede ser relevante en contextos donde se busca minimizar el uso de recursos computacionales.

Figura 5

Comparativa de NFE para los algoritmos en cada conjunto de secuencias



CONCLUSIONES

En conclusiones podemos observar que, aunque ambos algoritmos, *BactOf* y *BactAtp*, son comparables en eficiencia de evaluación (NFE), *BactAtp* muestra un mejor desempeño en términos de fitness y rapidez de convergencia, esto es más notorio en el Set B. Esto se debe a su distribución aleatoria de gaps que optimizan la diversidad genética y evitan patrones no deseados, lo que permite una mejor exploración y evita mínimos locales. También cabe mencionar que, aunque la diferencia en NFE es mínima, *BactAtp* alcanza resultados ligeramente mejores con menos evaluaciones en el Set C.

Es resaltable que el algoritmo aún puede mejorar con un control de inserción de gaps más preciso para evitar acumulaciones indeseadas, una adaptación dinámica de parámetros para mejorar la búsqueda en el espacio de soluciones, y un incremento gradual en la población de bacterias para explorar sin sobrecargar los recursos computacionales. En conclusión, *BactAtp* destaca como la opción más eficiente y adaptable para estudios genéticos complejos.

Agradecimientos

Los autores desean agradecer a la Facultad de Sistemas de la Uiversidad Autónoma de Coahuila por el apoyo en este proyecto

REFERENCIAS

Amorim, A. R., Zafalon, G. F. D., Contessoto, A. de G., Valêncio, C. R., & Sato, L. M. (2021). Metaheuristics for multiple sequence alignment: A systematic review. *Computational Biology and Chemistry*, *94*, 107563. https://doi.org/10.1016/j.compbiolchem.2021.107563

Biswas, A., Das, S., Abraham, A., & Dasgupta, S. (2010). Analysis of the reproduction operator in an artificial bacterial foraging system. *Applied Mathematics and Computation*, *215*(9), 3343–3355. https://doi.org/10.1016/j.amc.2009.10.023

Chen, Y.-P., Li, Y., Wang, G., Zheng, Y.-F., Xu, Q., Fan, J.-H., & Cui, X.-T. (2017). A novel bacterial foraging optimization algorithm for feature selection. *Expert Systems with Applications*, *83*, 1–17. https://doi.org/10.1016/j.eswa.2017.04.019

Guo, C., Tang, H., Niu, B., & Boon Patrick Lee, C. (2021). A survey of bacterial foraging optimization. *Neurocomputing*, *452*, 728–746. https://doi.org/10.1016/j.neucom.2020.06.142

Hernández-Ocaña, B., Pozos-Parra, Ma. D. P., Mezura-Montes, E., Portilla-Flores, E. A., Vega-Alvarado, E., & Calva-Yáñez, M. B. (2016). Two-Swim Operators in the Modified Bacterial Foraging Algorithm for the Optimal Synthesis of Four-Bar Mechanisms. *Computational Intelligence and Neuroscience*, *2016*, 1–18. https://doi.org/10.1155/2016/4525294

Lipman, D. J., Altschul, S. F., & Kececioglu, J. D. (1989). A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, *86*(12), 4412–4415. https://doi.org/10.1073/pnas.86.12.4412

Pang, B., Song, Y., Zhang, C., Wang, H., & Yang, R. (2018). An improved Bacterial Foraging Optimization algorithm using novel chemotaxis and swarming strategy. *2018 IEEE International Conference on Information and Automation (ICIA)*, 1107–1112. https://doi.org/10.1109/ICInfA.2018.8812479

Xu, X., & Chen, H. (2014). Adaptive computational chemotaxis based on field in bacterial foraging optimization. *Soft Computing*, *18*(4), 797–807. https://doi.org/10.1007/s00500-013-1089-4

Zambrano-Vega, C., Nebro, A. J., Durillo, J. J., García-Nieto, J., & Aldana-Montes, J. F. (2017). Multiple Sequence Alignment with Multiobjective Metaheuristics. A Comparative Study. *International Journal of Intelligent Systems*, *32*(8), 843–861. https://doi.org/10.1002/int.21892