ISSN 1870-4069

Comparación del rendimiento del algoritmo basada en forrajeo de bacterias con Matlab y Octave

Adrian García-López, Betania Hernández-Ocaña, Oscar Chávez-Bosquez, José Hernández-Torruco, Juana Canul-Reich

Universidad Juárez Autónoma de Tabasco,
División Académica de Ciencias y Tecnologías de la Información.
México

192H13002@alumno.ujat.mx,
{betania.hernandez, oscar.chavez, jose.hernandezt,
juana.canul}@ujat.mx

Resumen. Existen metaheurísticas bio-inspiradas en la naturaleza basadas en bacterias que emulan su comportamiento de manera computacional, tal es el caso del Algoritmo de Optimización Basado en el Forrajeo de Bacterias Modificado con Dos Nados (TS-MBFOA, por sus siglas en ingles). Este algoritmo genera un conjunto de soluciones óptimas aproximadas a problemas de optimización complejos en un tiempo razonable. TS-MBFOA está codificado de manera modular en el lenguaje de programación M de Matlab, el cual es un software comercial cuyo alto costo no es accesible para todo usuario. Existen alternativas de uso para ejecutar código del lenguaje M, tal es el caso de GNU Octave, un proyecto de software libre de gran utilidad en el campo del cálculo numérico. En este artículo se hace una comparación del rendimiento del algoritmo TS-MFOA implementado y ejecutado en Matlab y Octave. Ambos entornos son usados para ejecutar a TS-MBFOA y dar solución al problema de prueba conocido como el resorte de tensión/compresión, con el objetivo de determinar las ventajas y desventajas de estas dos plataformas y comprobar si el lenguaje de programación influye en el rendimiento del algoritmo en el proceso de optimización.

Palabras clave: Forrajeo de bacterias, metaheurísticas, Matlab, Octave, optimización.

Performance Comparison of the Bacterial Foraging Algorithm Under Matlab and Octave

Abstract. There are bio-inspired metaheuristics in nature based on bacteria that computationally emulate their behavior; such is the case of the Modified Bacterial Foraging Optimization Algorithm with two swims (TS-MBFOA). This algorithm generates a set of approximately optimal solutions to complex optimization problems in a reasonable time. TS-MBFOA is implemented in a

modular way in the Matlab M programming language, commercial software with high cost not accessible to all users. There are alternatives to execute code of the M language; such is the case of GNU Octave, a free software project of great utility in numerical calculation. This article compares the performance of the TS-MFOA algorithm implemented and executed in Matlab and Octave. Both environments are used to run TS-MBFOA and solve the test problem known as the tension/compression spring to determine the advantages and disadvantages of these two platforms and check if the programming language influences performance.

Keywords: Bacterial foraging, metaheuristic, Matlab, Octave, optimization.

1. Introducción

Una gran clase de problemas lógicos que surgen de situaciones del mundo real pueden formularse como problemas de optimización, ya sea para maximizar o minimizar las variables de diseño de una función objetivo y, por lo tanto, se pueden describir como una búsqueda de la mejor solución. Existen una serie de problemas reales de optimización de difícil solución que requieren encontrar el camino más corto para su solución y una secuencia óptima de sus procesos de trabajo.

Estos problemas suelen requerir agrupamientos, ordenaciones o asignaciones de un conjunto discreto de objetos que satisfagan ciertas restricciones. Para resolver un problema de optimización se necesita asistirse por programación matemática, pero al momento de su implementación pueden ser complicado o confuso [8].

Existen técnicas que ofrecen una mejor solución que la programación matemática, estas se llaman metaheurísticas o algoritmos bio-inspirados y son estrategias para diseñar o mejorar los procedimientos matemáticos orientados a obtener un alto rendimiento [14]. Los algoritmos bio-inspirados tienen dos sub-grupos que se distinguen de acuerdo con el tipo de fenómeno natural en el que se basan: los Algoritmos Evolutivos (AE) que basan su funcionamiento en la teoría de la evolución de las especies y la supervivencia del más apto [4].

Por otro lado, los Algoritmos de Inteligencia Colectiva (AIC) que toman comportamientos de seres vivos como: parvadas de aves, bancos de peces, cúmulos de insectos, hormigas, bacterias, entre otros [2], y estas interactúan en su ambiente donde surgen comportamientos cooperativos que permiten a estos seres simples, de manera individual resolver problemáticas complejas de manera conjunta.

Uno de los AIC más novedoso es el Algoritmo de optimización basado en el forrajeo de bacterias (BFOA, por sus siglas en inglés), este simula de manera computacional el proceso completo de forrajeo de la bacteria E. Coli y fue propuesto por Passino [13]. Su funcionamiento consiste en cuatro pasos: quimiotáxis (movimientos de nados-giros), agrupamiento, reproducción y eliminación-dispersión. Durante estas etapas las bacterias se enfrentan a varias problemáticas en la búsqueda de sus alimentos [10].

El BFOA requiere de la calibración de muchos parámetros y su costo computacional es alto debido al grupo de ciclos anidados que permiten su funcionamiento.

En la propuesta MBFOA, los autores hacen una disminución de parámetros del algoritmo y se aplica un mecanismo para el manejo de las restricciones [11]. Otra propuesta del algoritmo es el Improved MBFOA donde la modificación implementa un mecanismo de sesgo para crear la población inicial, dos operadores de nado, tamaños de paso dinámico y el buscador local [7].

Una de las más recientes modificaciones y la que es utilizada en este artículo es el TS-MBFOA donde se intercalan dos nados en el proceso quimiotáxico. El primero es el nado original a excepción del tamaño de paso, el cual se propone sea aleatorio, y el segundo nado incluye el operador de mutación usado en los algoritmos evolutivos, la inclusión de ambos con el objetivo de mejorar la capacidad de exploración y explotación del algoritmo [5].

El TS-MBFOA es una metaheurística que permite tener resultados competitivos y favorables al solucionar Problemas de Optimización Numérica con Restricciones (PONR). Esta propuesta recientemente se ha utilizado con éxito para resolver problemas de diseño ingenieril, como el conocido Resorte de Tensión / Compresión [5] y la generación de menús nutritivos [6].

Dicho algoritmo posee sus propios parámetros que corresponden a cada uno de sus procesos, los cuales son independientes de las variables del problema a resolver y deben ser calibrados o ajustados por el usuario final de tal forma que permitan un mejor rendimiento del algoritmo.

Una de las características generales de las metaheurísticas es que algunas están codificadas en Matlab, un software comercial de alto costo con difícil acceso para algunos miembros de la comunidad científica. Existe una alternativa de solución para los algoritmos codificados en Matlab, este es GNU Octave, un entorno de programación equivalente pero de licencia libre.

Por esta razón, la implementación del TS-MBFOA se ejecuta en Matlab y GNU Octave para hacer la comparación de estos lenguajes de programación y determinar las ventajas de tiempos de ejecución y la calidad del conjunto de resultados que se obtienen. De esta manera, los resultados obtenidos se validan usando estadísticas básicas como mejor y peor valor, media, mediana y desviación estándar.

2. Two-Swim Modified Bacterial Foraging Optimization Algorithm (TS-MBFOA)

El TS-MBFOA es un algoritmo derivado de MBFOA propuesto para resolver PONR [5], en el cual una bacteria i representa una solución potencial y se denota como $\theta^i(j, G)$, donde j es el ciclo quimiotáxico y G es el ciclo generacional. Una generación consta de un proceso quimiotáxico, agrupamiento, reproducción y eliminación-dispersión.

En el proceso de quimiotáxis dos nados se intercalan, en cada ciclo solo un nado de explotación o exploración es realizado. El proceso comienza con el nado de explotación (nado clásico). Sin embargo, una bacteria no necesariamente intercalará exploración y explotación en los nados, ya que si la nueva posición de un nado dado, $\theta^i(j+1, G)$ tiene una mejor aptitud (basado en las reglas de factibilidad) que la posición original

$\theta^i(j, G)$, otro nado en la misma dirección se llevará a cabo en el siguiente ciclo. De lo contrario, un nuevo giro será calculado. El proceso se detiene después de N_c intentos.

El nado de exploración usa la mutación entre bacterias y es calculado con la Ecuación 1:

$$\theta^i(j+1, G) = \theta^i(j, G) + (\beta)(\theta^r_1(j, G) - \theta^r_2(j, G)), \tag{1}$$

donde $\theta^r_1(j, G)$ y $\theta^r_2(j, G)$ son dos bacterias diferentes seleccionadas aleatoriamente de la población. β es un parámetro definido por el usuario utilizado en el operador de agrupamiento el cual define la cercanía de la nueva posición de una bacteria con respecto a la posición de la mejor bacteria de la población, en este operador, β es un parámetro de control positivo para escalar los diferentes vectores en (0,1], es decir, escalas de la zona donde una bacteria puede moverse. El nado de explotación es calculado con el Ecuación 2:

$$\theta^i(j+1, G) = \theta^i(j, G) + C(i, G)\phi(i), \tag{2}$$

donde $\phi(i)$ se calcula con el operador de giro original de BFOA definido en la Ecuación 3:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}}, \tag{3}$$

$\Delta(i)^T$ es un vector aleatorio generado con elementos dentro de un intervalo $[-1, 1]$. $C(i, G)$ es el tamaño de paso aleatorio de cada bacteria actualizado con la Ecuación 4:

$$C(i, G) = R * \Theta(i), \tag{4}$$

donde $\Theta(i)$ es un vector generado de forma aleatoria de tamaño n con elementos dentro del rango de cada variable de decisión: $[U_k, L_k]$, $k = 1, ...n$, y R es un parámetro definido por el usuario para escalar el tamaño de paso, este valor debe ser cercana a cero (por ejemplo 5.00e-04). La inicial $C(i, 0)$ se genera utilizando $\theta(i)$. Este tamaño de paso aleatorio permite que las bacterias se puedan mover en diferentes direcciones dentro del espacio de búsqueda y evita la convergencia prematura como se sugiere en [9].

En el ciclo medio del proceso quimiotáxico es aplicado el operador de agrupamiento con la Ecuación 5, donde β es un parámetro positivo definido por el usuario entre (0,1):

$$\theta^i(j+1, G) = \theta^i(j, G) + \beta(\theta^B(G) - \theta^i(j, G)), \tag{5}$$

donde $\theta^i(j+1, G)$ es la nueva posición de la bacteria i, $\theta^B(G)$ es la actual posición de la mejor bacteria generacional y β es un parámetro llamado factor de escalamiento, el cual regula qué tan cerca estará la bacteria i de la mejor bacteria θ^B. Sin embargo, si una solución viola el límite de las variables de decisión, una nueva solución de x_i es generada aleatoriamente entre los límites inferior y superior $L_i \leq x_i \leq U_i$ de las variables de decisión.

En la reproducción se ordenan las bacterias con base en la técnica de manejo de restricciones, eliminando a las peores bacterias $S_b - S_r$ y duplicando a las mejores cada cierto número de ciclos, definido por el usuario con el parámetro $RepCycle$. En la eliminación-dispersión se elimina a la peor bacteria de la población $\theta^w(j, G)$ (basado en las reglas de factibilidad) y se genera una nueva aleatoriamente.

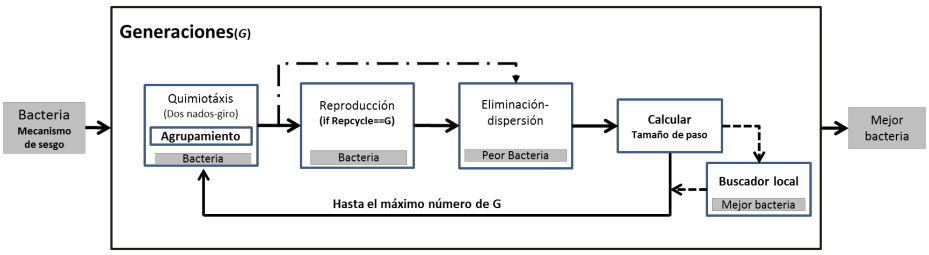

Fig. 1. Procesos generales de TS-MBFOA.

Algorithm 1: Pseudocódigo de TS-MBFOA.

Crear una población inicial de bacterias aleatorias $\theta^i(j,0)\ \forall i,\ i = 1,\ldots,S_b$
Evaluar $f(\theta^i(j,0))\ \forall i,\ i = 1,\ldots,S_b$
for *G=1 to GMAX* **do**
  for *i=1 to* S_b **do**
    for *j=1 to* N_c **do**
      En el proceso quimiotáxico intercalar los nados propuestos con las Ecuaciones 1 y 2. Aplicar el operador de agrupamiento con la Ecuación 5 usando β para la bacteria $\theta^i(j,G)$
    end
  end
  if $G\ mod\ RepCycle == 0$ **then**
    Realizar el proceso de reproducción ordenando la población de acuerdo a las reglas de factibilidad y eliminar a S_r peores bacterias y duplicar el resto de bacterias $S_b - S_r$.
  end
  Realizar el proceso de eliminación-dispersión eliminando a la peor bacteria $\theta^w(j,G)$ de la población actual considerando la técnica de manejo de restricciones.
  Actualizar el vector de tamaño de paso usando la Ecuación 4.
end

Fig. 2. Pseudocódigo del TS-MBFOA.

Aunque en su propuesta original de TS-MBFOA se utiliza un mecanismo de sesgo para generar la población inicial aleatoria y un buscador local, en este artículo no se hace uso de dicho mecanismo para consumir menos costo computacional.

La estructura del TS-MBFOA se presenta en la Figura 1. El pseudocódigo de TS-MBFOA es presentado en la Figura 2.

3. Materiales y método

Para la ejecución de la implementación del TS-MBFOA en Matlab y GNU Octave, a continuación, se describen cada uno de ellos, mostrando sus características principales como lenguajes de programación de alto nivel.

3.1. Matlab

Matlab no es sólo un entorno de desarrollo, también un lenguaje de programación. Matlab integra el cálculo, la visualización y la programación en un ambiente de desarrollo de alto nivel y básicamente permite resolver muchos problemas computacionales, específicamente aquellos que involucren vectores y matrices, en un

tiempo mucho menor al requerido para escribir un programa en un lenguaje escalar no interactivo tal como C o Fortran.

Matlab se utiliza ampliamente en: cálculos numéricos, desarrollo de algoritmos, modelado, simulación y prueba de prototipos, análisis de datos, exploración y visualización, graficación de datos con fines científicos o de ingeniería y desarrollo de aplicaciones que requieran de una interfaz gráfica de usuario (GUI, Graphical User Interface).

En el ámbito académico y de investigación, es la herramienta estándar para los cursos introductorios y avanzados de matemáticas, ingeniería e investigación. También es la herramienta usada para el análisis, investigación y desarrollo de nuevos productos tecnológicos [3].

Matlab está disponible en sistemas operativos como: Unix, Windows, macOS y GNU/Linux, pero también cuenta con desventajas como: software comercial de alto costo, no se puede ampliar, problemas eventuales de velocidad y poca distribución de ejecutables.

3.2. GNU Octave

GNU Octave es un entorno matemático para realizar cálculos numéricos, el cual es completamente gratuito basado en la filosofía GNU (Sistema operativo libre). Octave comparte características con Matlab donde se puede destacar que ambos ofrecen un intérprete permitiendo ejecutar órdenes en modo interactivo [1].

Entre las funcionalidades más importantes que incorpora Octave pueden mencionarse: elevada capacidad para resolver problemas del Álgebra Lineal, cálculo raíces de ecuaciones no lineales, integración numérica de funciones, manipulación de polinomios, integración de ecuaciones diferenciales, incorporación de cajas de herramientas, Se puede ejecutar en plataformas Unix y Windows, Puede cargar archivos con funciones de Matlab de extensión .m.

Octave soluciona algunos de los inconvenientes técnicos de Matlab, aunque no tiene módulos para el desarrollo de interfaz gráfica, algo visto como una desventaja. Pero cuenta con características especiales como: un buen analizador sintáctico y acepta estructuras complejas.

Aunque el intérprete tiene problemas de estabilidad como el de Matlab se solucionan más rápido puesto que el tiempo de publicación de nuevas versiones es más corto. Es además es muy fácil de extender en C++ y permite un acceso directo a su maquinaria interna y a sus bibliotecas para utilizarlas en cualquier programa en C++ [12]. A continuación, en la Tabla 1 se muestra un comparativo de características de Matlab y GNU Octave.

En la codificación existente se ha utilizado el paradigma de programación modular, con el fin de separar cada uno de los procesos del algoritmo, que básicamente son 4 procesos, además de tener una programación clara de cada uno de ellos usando métodos como funciones y procedimientos con y sin retorno de información.

En la plataforma Github se encuentra de manera gratuita la codificación del TS-MBFOA para resolver el problema del resorte de tensión/compresión en lenguaje M, la cual puede consultarse y descargarse a través del link https://github.com/garcialopez/TS-MBFOA-OCTAVEvsMATLAB.

Tabla 1. Características comparativas de Matlab y GNU Octave.

Característica	**GNU Octave**	**Matlab**
Ejecuta lenguaje M	Sí	Sí
Multiplataforma	Sí	Sí
Simulador	Sí	Sí
Soporte de programación	No	Sí
Entorno visual	Sí	Sí
Desarrollo de interfaces gráficas	No	Sí
Software libre	Sí	No

4. Resultados

El algoritmo TS-MBFOA es ejecutado en los entornos Matlab y Octave de manera independiente en 30 ejecuciones. El problema de prueba es el PONR conocido como Resorte de Tensión/Compresión donde se busca minimizar el peso de un resorte (ver Figura 3) sujeto a restricciones de desviación mínima, tensión de corte, frecuencia de oleada, límites sobre el diámetro exterior, esto sobre variables de diseño.

El diseño de la función para ser procesada en el algoritmo TS-MBFOA cuenta con las siguientes variables de diseño: Diámetro del cable d(x_1), Diámetro del rollo D(x_2), y Número de rollos involucrados N(x_3). Formalmente, el problema puede expresarse de la siguiente manera:

$$(x_3 + 2)x_2x_1^2. \tag{6}$$

Sujeto a:

$$\begin{aligned} &g_1(X) = 1 - ((x_2^3x_3)/(71785x_1^4)) \leq 0,\\ &g_2(X) = ((4x_2^2 - x_1x_2)/12566(x_2x_1^3 - x_1^4))) + (1/5108x_1^2) - 1 \leq 0,\\ &g_3(X) = 1 - (140.45x_1/x_2^2x_3) \leq 0,\\ &g_4(X) = (x_2 + x_1/1.5) - 1 \leq 0, \end{aligned} \tag{7}$$

donde:

$$\begin{aligned} &0.05 \leq x_1 \leq 2,\\ &0.25 \leq x_2 \leq 1.3,\\ &2 \leq x_3 \leq 15. \end{aligned} \tag{8}$$

El código en lenguaje M del TS-MBFOA fue ejecutado en una computadora con las características: 4GB RAM, procesador de 2.3Ghz y sistema operativo Windows de 64bit; el entorno Matlab R2018b bajo la licencia 40693035 y el GNU Octave versión 6.

El TS-MBFOA se configuró con 30 ejecuciones independientes y la calibración de los parámetros fue: bacterias (S_b) = 50, tamaño de paso (R) = 0.001, factor de escalamiento (β) = 1.9, ciclo quimiotáxico (N_c) = 40, bacterias a reproducir (S_r) = 1, frecuencias de reproducción ($RepCycle$) = 100 y número de evaluaciones = 30,000.

Los resultados obtenidos a la minimización del problema se presenta en las Tablas 2 y 3.

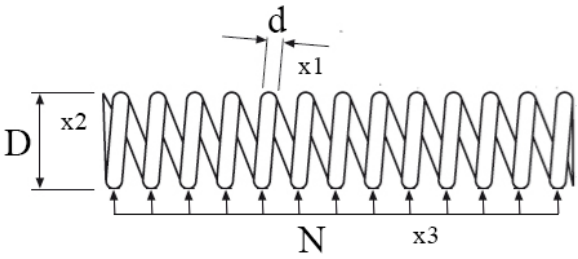

Fig. 3. Resorte de Tensión/Compresión.

Tabla 2. Resultados de TS-MBFOA ejecutado en Matlab para el problema del resorte.

Ejecución	x_1	x_2	x_3	**Función Objetivo**
1	0.051789696	0.359076172	11.1687731	0.012682902
2	0.051408726	0.350006599	11.69993827	0.012672682
3	0.051323133	0.347948886	11.83711678	0.012681989
4	0.05169103	0.356454463	11.31628262	0.012682867
5	0.051551231	0.352828141	11.54267337	0.012698303
6	0.050968603	0.339503268	12.38304812	0.012685288
7	0.051656821	0.355695836	11.35835672	0.012679063
8	0.051116893	0.343009175	12.15497521	0.012686556
9	0.052276436	0.370927952	10.52020387	0.012691499
10	0.051539769	0.352875798	11.53039952	0.012682867
11	0.052027796	0.364794473	10.83824442	0.012677241
12	0.052621549	0.379460676	10.07431931	0.012686934
13	0.051119373	0.343141925	12.14543596	0.012684144
14	0.051382235	0.349344232	11.74137683	0.012673886
15	0.051859118	0.360791058	11.06753926	0.012679433
16	0.051595774	0.354339953	11.44481574	0.012682452
17	0.051174813	0.344460045	12.0487119	0.012673247
18	0.051311125	0.347679802	11.84412994	0.012672672
19	0.05275138	0.382662497	9.923899948	0.012697022
20	0.051110004	0.342691795	12.17608431	0.012690298
21	0.051225268	0.345619307	11.97511887	0.012674242
22	0.052603802	0.379041853	10.0937464	0.012684761
23	0.051642013	0.355479518	11.37752278	0.012682259
24	0.05193514	0.362645388	10.97988345	0.012696253
25	0.051694988	0.356852268	11.28483768	0.012668978
26	0.052128581	0.367359078	10.69530642	0.012673185
27	0.051164621	0.344221262	12.0646512	0.01267378
28	0.052598746	0.378968032	10.1001054	0.01268652
29	0.051280044	0.346939455	11.89501968	0.0126768
30	0.05186574	0.36094871	11.06059164	0.012681467

La comparación de los resultados obtenidos de las 30 ejecuciones independientes del algoritmo determina que ambas ejecuciones encuentran resultados óptimos y

Tabla 3. Resultados de TS-MBFOA ejecutado en Octave para el problema del resorte.

Ejecución	x_1	x_2	x_3	**Función Objetivo**
1	0.051236884	0.345914002	11.95354752	0.012671213
2	0.05167156	0.356096572	11.34346759	0.012686436
3	0.051163989	0.343987178	12.08848084	0.012686307
4	0.051828296	0.360063954	11.1034556	0.012673581
5	0.051562802	0.353639641	11.48479737	0.012678807
6	0.05210742	0.366577403	10.7485808	0.012688979
7	0.051626443	0.355093061	11.3907663	0.012673367
8	0.051757885	0.358271685	11.2029732	0.01267177
9	0.052283211	0.370773217	10.52561293	0.012694975
10	0.051687167	0.356496828	11.31148198	0.012677906
11	0.05179902	0.359258975	11.14740871	0.012673334
12	0.051452182	0.350926475	11.64531196	0.012676729
13	0.051909759	0.361838903	11.00347399	0.01267864
14	0.051414867	0.349886947	11.71274694	0.012683223
15	0.051435782	0.35051317	11.66883685	0.012675544
16	0.051457705	0.351081024	11.63494457	0.012675397
17	0.05176511	0.358435112	11.20035225	0.012678572
18	0.051909904	0.361894529	11.00566901	0.012682801
19	0.051645052	0.355580688	11.37507666	0.012685041
20	0.050965892	0.339442726	12.38383121	0.012682368
21	0.051179829	0.344543074	12.04574631	0.01267611
22	0.05204823	0.365353585	10.80359713	0.012672354
23	0.052494637	0.376357715	10.23368654	0.01268785
24	0.05213442	0.36725143	10.71944589	0.012696405
25	0.050833354	0.336478217	12.58991746	0.012685492
26	0.051955975	0.362936516	10.95593073	0.012693175
27	0.051849943	0.360503843	11.07555731	0.012672627
28	0.05242227	0.374463276	10.33320982	0.012691618
29	0.051170897	0.344192341	12.07537644	0.012685491
30	0.050757896	0.33464182	12.71671054	0.012688146

Tabla 4. Resultados de ejecución de GNU Octave y Matlab.

Herramienta	**GNU Octave**	**Matlab**
Modificación del código	No	No
Interfaz gráfica	No	Sí
Ejecuciones	30	30
Tiempo en segundos	$9,120$	78

aproximados al mejor valor conocido dentro la literatura especializada para el problema de optimización utilizado, el cual es 0.0126.

Cabe mencionar que en ambos lenguajes de programación se obtiene las estadísticas básicas sin la modificación de código fuente del algoritmo codificado en scripts m. Sin embargo, una de las desventajas de ser ejecutada en Octave es el tiempo de ejecución (ver Tabla 4) y que no tiene la capacidad para desarrollar una interfaz de usuario.

En la Tabla 5 se presenta un comparativo de los resultados de las estadísticas básicas

Tabla 5. Comparación de estadísticas básicas del algoritmo TS-MBFOA en Matlab y Octave.

Criterios	TS-MBFOA en Matlab	TS-MBFOA en Octave
Mejor	0.012668978283364	0.0126712133127548
Media	0.012681986380528	0.01268147532895791
Mediana	0.012682355678653	0.01268058712340608
Desviación estándar	7.659412902293769e-06	7.427784107212482e-06
Peor	0.012698303467978	0.01269640528719985

del TS-MBFOA al ser ejecutado en los dos entornos de desarrollo. Los resultados de ambas ejecuciones son generados con los mismos parámetros.

De acuerdo a los valores de la Tabla 5, TS-MBFOA es un algoritmo que mejores resultados obtiene al ser ejecutado en ambas plataformas. Matlab tiene mejor racha de tiempo cumpliendo las 30 ejecuciones en 1 minuto con 18 segundos. Al ejecutar TS-MBFOA en GNU Octave el tiempo de ejecución de 30 ejecuciones independientes fue de 2 horas con 53 minutos, tiempo significativo que implica un atraso para la obtención de los resultados, pero con la ventaja de ser lenguaje de programación libre.

Los resultados se obtuvieron con la misma configuración sin la necesidad de hacer modificaciones en el código, además de obtener resultados óptimos aproximados como lo muestra la media y desviación estándar de la Tabla 5.

En general, los resultados de la ejecución de GNU Octave son competitivos, pero se requiere de un mayor tiempo de ejecución.

De acuerdo a la prueba no paramétrica de *Wilcoxon signed rank test* con un 95 % de confianza, no existe una diferencia significativa entre los resultados de TS-MBFOA ejecutado en Matlab y Octave con un p-value de 0.68916, tomando como valores para la comparación la mejor solución obtenida de cada una de las 30 ejecuciones independientes encontrada por el TS-MBFOA (ver Tablas 2 y 3).

5. Conclusiones

En este artículo el algoritmo basado en el forrajeo de bacterias TS-MBFOA fue implementado en lenguaje M y ejecutado en los entornos de desarrollo Matlab y Octave con el objetivo de comparar el rendimiento del algoritmo en un problema de optimización númerica con restricciones de diseño ingenieríl conocido como el Resorte de tensión/compresión. Además de dar a conocer a la comunidad científica las ventajas y desventajas de usar estos entornos en algoritmos bio-inspirados.

El TS-MBFOA se ejecutó en ambos entornos sin la necesidad de modificar el código fuente del algoritmo. 30 ejecuciones independientes fueron realizadas, donde la principal desventaja de Octave fue el tiempo de ejecución con una diferencia de 2 horas y 53 minutos en comparación con Matlab con tan solo uso 1 minuto con 18 segundos.

A diferencia de Matlab, Octave es un software libre y ejecuta cualquier código generado en scripts .m, excepto programación de interfaces gráficas. En los experimentos realizados, Matlab obtuvo soluciones mucho más rápido, lo cual es una gran ventaja para la toma de decisiones del usuario final ante la solución de problemas complejos de la vida real.

La calidad de los resultados, obtenidos por ambas implementaciones, son competitivas.

De acuerdo a lo presentado en las tablas de estadísticas básicas y la prueba no paramétrica de *Wilconxon Signed Rank Test*, no existe una diferencia significativa entre los resultados de TS-MBFOA ejecutado en Matlab y Octave.

La comunidad científica puede hacer uso de Matlab y Octave en la implementación y ejecución de algoritmos bio-inspirados para la solución de problemas complejos. La codificación en lenguaje M es aceptable en ambos entornos, los resultados obtenidos tienen la misma calidad, sin embargo, el tiempo de ejecución es más elevado en Octave, aunque, Matlab es un software comercial bajo licencia costosa.

Como trabajo futuro, se espera implementar a TS-MBFOA en lenguage Java, el cual es Open source, y analizar el rendimiento del algoritmo en el software que ejecute dicho lenguaje. Además, de probar el algoritmo en un conjunto de pruebas más amplio para obtener conjeturas o inferencias más profundas del rendimiento de este algoritmo en diversos lenguajes de programación.

Referencias

1. Companioni-Guerra, A., Cuesta-Llanes, E., Hernández-Blanco, Y., Orovio-Cobo, V., Días-Ramos, S.: Entorno integrado para el trabajo con GNU/Octave. Revista Cubana de Ciencias Informáticas, vol. 6, no. 4, pp. 1–8 (2012)
2. Engelbrecht, A. P.: Fundamentals of computational swarm intelligence. John Wiley & Sons (2005)
3. Esqueda-Elizondo, J. J.: Matlab e interfaces gráficas. CONATEC 2002: Instituto Tecnológico de Ciudad Madero (2002)
4. Fogel, D. B.: An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks, vol. 5, no. 1, pp. 3–14 (1994) doi: 10.1109/72.265956
5. Hernández-Ocaña, B., Pozos-Parra, P., Mezura-Montes, E., Portilla-Flores, E. A., Vega-Alvarado, E., Calva-Yáñez, M. B.: Two-swim operators in the modified bacterial foraging algorithm for the optimal synthesis of four-bar mechanisms. Computational Intelligence and Neuroscience, no. 17, pp. 1–18 (2016) doi: 10.1155/2016/4525294
6. Hernández-Ocaña, B., Chávez-Bosquez, O., Hernández-Torruco, J., Canul-Reich, J., Pozos-Parra, P.: Bacterial foraging optimization algorithm for menu planning. IEEE Access, vol. 6, pp. 8619–8629 (2018) doi: 10.1109/ACCESS.2018.2794198
7. Hernández-Ocana, B., Pozos-Parra, P., Mezura-Montes, E.: Improved modified bacterial foraging optimization algorithm to solve constrained numerical optimization problems. Applied Mathematics and Information Sciences, vol. 10, no. 2, pp. 607–622 (2016) doi: 10.18576/amis/100220
8. Herrera, F.: Introducción a los algoritmos metaheurísticos. Ciencias de la Computación e IA (2006)
9. Kasaiezadeh, A., Khajepour, A., Waslander, S. L.: Spiral bacterial foraging optimization method: Algorithm, evaluation and convergence analysis. Engineering Optimization, vol. 46, no. 4, pp. 439–464 (2014) doi: 10.1080/0305215X.2013.776550
10. Mezura-Montes, E., Cetina-Domínguez, O., Hernández-Ocaña, B.: Nuevas heurísticas inspiradas en la naturaleza para optimización numérica (2010)
11. Mezura-Montes, E., Hernández-Ocaña, B.: Bacterial foraging for engineering design problems: Preliminary results. In: Memorias del 4o Congreso Nacional de Computación Evolutiva (COMCEV'2008) (2008)
12. Nogueras, G. B.: Octave: Una alternativa real a Matlab a coste cero. Fluid Dynamics Group, UPM, pp. 1–7 (2007)

13. Passino, K. M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine, vol. 22, no. 3, pp. 52–67 (2002) doi: 10.1109/MCS.2002.1004010
14. Suarez, A. O.: Una aproximación a la heurística y metaheurísticas. INGE@ UAN-Tendencias en la Ingeniería, vol. 1, no. 2 (2013)