ISSN 1870-4069

Rendimiento del algoritmo basado en el forrajeo de bacterias con distribución uniforme, gaussiana y exponencial

Margarita Hernández-Hernández, Betania Hernández-Ocaña, José Adán Hernández-Nolasco, José Hernández-Torruco

Universidad Juárez Autónoma de Tabasco, México

192H13004@alumno.ujat.mx, {betania.hernandez, jose.nolasco, jose.hernandezt}@ujat.mx

Resumen. En este trabajo se analiza el comportamiento y rendimiento del algoritmo basado en el forrajeo de bacterias TS-MBFOA usando poblaciones de bacterias generadas con distribución gaussiana y exponencial. TS-MBFOA es una metaheurística de inteligencia colectiva que emula el comportamiento de las bacterias E.Coli, probado con éxito en problemas de optimización numérica con restricciones. Sin embargo, este algoritmo requiere de un costo computacional elevado debido a sus múltiples parámetros, por ende, en este trabajo se busca mejorar el rendimiento del algoritmo reinicializando la población de bacterias con distintas distribuciones. En este estudio preliminar el algoritmo fue probado en los problemas conocidos como la esfera y resorte de tensión/compresión, donde el algoritmo con distribución exponencial genera soluciones con mejor consistencia.

Palabras clave: Optimización, forrajeo de bacterias, metaheurística, distribución uniforme, distribución gaussiana, distribución exponencial.

Performance of the Algorithm based on the Foraging of Bacteria with Uniform, Gaussian and Exponential Distribution

Abstract. In this paper, the behavior and performance of the algorithm based on the bacterial foraging TS-MBFOA is analyzed using swarm of bacteria generated with gaussian and exponential distribution. TS-MBFOA is a swarm intelligence metaheuristic that emulates the behavior of E.Coli bacteria, successfully tested on constrained numerical optimization problems. However, this algorithm requires a high computational cost due to its multiple parameters, therefore, this work seeks to improve the performance of the algorithm by reinitializing the population of bacteria with different distributions. In this preliminary study the algorithm was tested in the problems known as the sphere and tension/compression spring, where the algorithm with exponential distribution generates solutions with better consistency.

Keywords: Optimization, bacterial foraging, metaheuristic, uniform distribution, gaussian distribution, exponential distribution.

1. Introducción

Los algoritmos bio-inspirados en la naturaleza son ampliamente usados para resolver problemas de optimización numérica con restricciones. La comunidad científica ha hecho uso de ellos para resolver problemas complejos de diversas áreas como medicina [5].

Estos algoritmos son conocidos como metaheurísticas, las cuales permiten resolver problemas de optimización con o sin restricciones, combinatorios o numéricos de manera aproximada, es decir generan una o más soluciones factibles cercanas al óptimo.

Un problema de optimización es también conocido como un problema general de programación no-lineal y se puede definir como: Minimizar ó Maximizar $f(\vec{x})$ sujeta a: $g_i(\vec{x}) \leq 0, i = 1, ..., m$ y/o $h_j(\vec{x}) = 0, j = 1, ..., p$.
donde $\vec{x} \in R^n$ tal que $n \geq 1$, es el vector de soluciones $\vec{x} = [x_1, x_2, ..., x_n]^T$, donde cada $x_i, i = 1, ..., n$ está delimitada por el límite inferior y superior $L_i \leq x_i \leq U_i$; m es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos, las restricciones podrían ser lineales o no lineales). Si denotamos con F a la región factible (donde se encuentran todas las soluciones que satisfacen al problema) y con S a todo el espacio de búsqueda, entonces $F \subseteq S$.

Los algoritmos bio-inspirados se dividen en dos grandes grupos, los Algoritmos Evolutivos (AEs), cuyo funcionamiento se basa en emular el proceso de evolución natural y la supervivencia del más apto y los algoritmos de Inteligencia Colectiva (IC) se basan en el comportamiento social y cooperativo de organismos simples e inteligentes como insectos y aves [6].

En el año 2002 K. Passino propone al algoritmo del forrajeo de bacterias (BFOA, por sus siglas en inglés), en el cual cada bacteria trata de maximizar su energía obtenida por unidad de tiempo empleada en el proceso de forrajeo, donde también evade sustancias nocivas. Más aun, las bacterias se pueden comunicar entre sí mediante la segregación de sustancias [10].

Su funcionamiento consiste en cuatro pasos: quimiotáxis (movimientos de nados-giros), agrupamiento, reproducción y eliminación-dispersión. Una mejora de BFOA es el algoritmo BFOA modificado (MBFOA), el cual hace una disminución de los parámetros y ciclos del algoritmo [9]. Este algoritmo aplica un mecanismo para el manejo de las restricciones usando las reglas de factibilidad de Deb [3].

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 [6].

Recientemente, el algoritmo TS-MBFOA, una propuesta basada en MBFOA, intercala dos nados en el proceso quimiotáxico, el primero es el nado original con tamaño de paso aleatorio y el segundo nado incluye el operador de mutación usado en los algoritmos evolutivos para mejorar la capacidad de exploración y explotación del algoritmo [7].

Uno de los inconvenientes de esta propuesta es la convergencia prematura donde algunas veces el algoritmo puede caer en óptimos locales, esto debido a la poca diversidad en el cúmulo de bacterias provocado por los procesos de agrupamiento y reproducción del algoritmo.

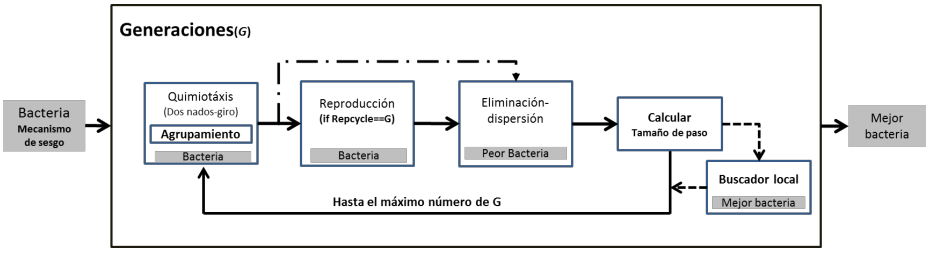

Fig. 1. Procesos generales de TS-MBFOA.

Las propuestas MBFOA, IMBFOA, TS-MBFOA hacen uso de la distribución uniforme para generar cada una de las bacterias de la población generacional del algoritmo.

Revisando el estado del arte, algunas propuestas de algoritmos bio-inspirados hacen uso de distintas distribuciones como gaussiana, gamma y exponencial para mejorar su rendimiento con la prevalencia de diversidad en la población lo que conlleva una mejor distribución de la población en el espacio de búsqueda.

En la propuesta realizada por Chen et al. [2], se hace uso de una mutación gaussiana para mejorar el rendimiento de la propuesta basada en BFOA con el objetivo de aumentar la diversidad de la población y evitar una convergencia prematura.

En esta propuesta los autores generan las nuevas posiciones de las bacterias usando una longitud de paso y la mutación gaussiana que consiste en generar un número aleatorio entre [0,1] con distribución gaussiana que escala la posición actual de la mejor bacteria de la población.

donde los autores obtuvieron competitivos resultados, al resolver problemas de pruebas clásicos, con mayor precisión y una mejor convergencia comparando contra la versión original de BFOA y otras metaheurísticas.

En las propuestas [14, 1, 13, 11], se hace uso de la distribución gaussiana y gamma para mejorar la diversidad de la población de algoritmos evolutivos, específicamente hacen uso de estas distribuciones para el operador de mutación.

En este artículo se propone hacer un reinicio a la mitad de la población de bacterias, cuando la mitad de la población de bacterias sean factibles, es decir, cuando no se violen restricciones del problema a resolver. Este reinicio se hará en una única ocasión durante las generaciones del algoritmo.

La generación de la nueva población de bacterias hace uso de la información de todas las bacterias factibles de la población, para la distribución gaussiana y exponencial el valor de la media es obtenido del conjunto de las bacterias factibles de la población.

Esto con el objetivo de que las nuevas bacterias estén dentro o cerca de la región factible del problema a resolver y evitar generar bacterias fuera del espacio de búsqueda del problema.

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

El TS-MBFOA es un algoritmo derivado de MBFOA propuesto para resolver PONR [4], 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.n 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:

$$\theta^i(j+1, G) = \theta^i(j, G) + (\beta)(\theta_1^r(j, G) - \theta_2^r(j, G)), \tag{1}$$

donde $\theta_1^r(j, G)$ y $\theta_2^r(j, G)$ 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 la dirección del paso $\phi(i)$ se calcula con el operador de giro original de BFOA definido en la Ecuación 3:

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

donde $\Delta(i)$ 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 [8].

Algoritmo 1: Pseudocódigo de TS-MBFOA.

```
Crear una población inicial de bacterias aleatorias θ^i(j, 0)∀i, i = 1, ..., S_b
Evaluar f(θ^i(j, 0))∀i, i = 1, ..., 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 θ^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, peores bacterias y duplicar el resto de
        bacterias S − S_r
    end
    Realizar el proceso de eliminación-dispersión eliminando a la peor bacteria
    θ^ω"(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
```

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 $\mathrm{RepCycle}$.

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. 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.

Cabe recalcar que el manejo de restricciones de este algoritmo se hace mediante las reglas de factibilidad de Deb [3], las cuales fueron incluidas desde la versión MBFOA. La estructura del TS-MBFOA se presenta en la Figura 1. El pseudocódigo de TS-MBFOA es presentado en el algoritmo 1.

3. Distribuciones

3.1. Distribución uniforme

La distribución uniforme es una distribución continua que modela un rango de valores con igual probabilidad y se especifica mediante cotas inferior y superior [a, b]. Si $x \sim u(a, b)$, su función de distribución viene dada por:

$$F(x) = \begin{cases} 0 & \text{para} \quad x < a, \\ \dfrac{x-a}{b-a} & \text{para} \quad a \leq x < b, \\ 1 & \text{para} \quad x \geq b. \end{cases} \tag{6}$$

Aplicando el método de inversión de la función de distribución se obtiene el siguiente esquema:

1. Generar un número aleatorio u.
2. Tomar $x = a + u(b - a)$.

3.2. Distribución gaussiana

La distribución Gaussiana o normal es, sin duda, la distribución de probabilidad más importante del Cálculo de probabilidades y de la Estadística.

Fue descubierta, como aproximación de la distribución binomial, por Abraham De Moivre (1667-1754) y publicada en 1733 en su libro The Doctrine of Chances; estos resultados fueron ampliados por Pierre-Simon Laplace (1749-1827), quién también realizó aportaciones importantes.

En 1809, Carl Friedrich Gauss (1777-1855) publicó un libro sobre el movimiento de los cuerpos celestes donde asumía errores normales, por este motivo esta distribución también es conocida como distribución Gaussiana.

La importancia de la distribución normal queda totalmente consolidada por ser la distribución límite de numerosas variables aleatorias, discretas y continuas, como se demuestra a través de los teoremas centrales del límite [12].

Teorema Central del límite especifica que: Sean $x_1, ..., x_n$ variables aleatorias independientes e idénticamente distribuidas con media μ y desviación típica σ. Entonces, la distribución de:

$$\frac{\sum_{i=1}^{n} x_i - n\mu}{\sigma\sqrt{n}}, \tag{7}$$

Es aproximadamente $N(0, 1)$ cuando el tamaño muestral n es suficientemente grande. La distribución normal queda totalmente definida mediante dos parámetros: la media (μ) y la desviación estándar o desviación típica (σ). Su función de densidad es simétrica respecto a la media y la desviación estándar nos indica el mayor o menor grado de apertura de la curva que, por su aspecto, se suele llamar campana de Gauss. Esta distribución se denota por N (μ, σ).

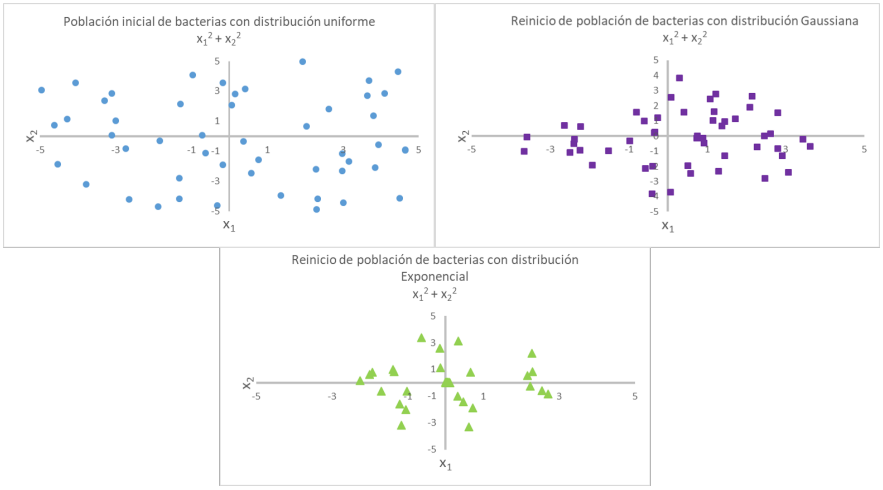

Fig. 2. Población de bacterias con distribución uniforme, gaussiana y exponencial.

La distribución $N(\mu, \sigma)$ se puede relacionar con la distribución $N(0, 1)$, mediante el siguiente proceso al que se denomina tipificación o estandarización:

$$X\ N(\mu, \sigma) \Rightarrow Z = \frac{X - \mu}{\sigma}. \tag{8}$$

A la distribución $N(0, 1)$ se le denomina Normal Estándar. En diversos lenguajes de programación como java, existen funciones programadas que permiten generar un número aleatorio con distribución normal con media y desviación estándar de cero. El siguiente esquema permite generar un número aleatorio con distribución normal con media y desviación estándar definida por el usuario final:

1. Generar un número aleatorio u con distribución $N(0, 1)$.
2. Definir los valores de σ y μ de un conjunto de valores.
3. Tomar $x = u \times \sigma + \mu$.

3.3. Distribución exponencial

La distribución exponencial es una distribución continua que se utiliza para modelar tiempos de espera para la ocurrencia de un cierto evento. Esta ley de distribución describe procesos en los que interesa saber el tiempo hasta que ocurre determinado evento; en particular, se utiliza para modelar tiempos de supervivencia [12].

Esta distribución se puede caracterizar como la distribución del tiempo entre sucesos consecutivos generados por un proceso de Poisson; por ejemplo, el tiempo que transcurre entre dos heridas graves sufridas por una persona. La media de la distribución de Poisson, λ, que representa la tasa de ocurrencia del evento por unidad de tiempo, es el parámetro de la distribución exponencial, y su inversa es el valor medio de la distribución.

Tabla 1. Mejor bacteria del algoritmo TS-MBFOA con diferentes distribuciones.

TS-MBFOA con distribución	x_1	x_2	$f(\vec{x}) = x_1^2 + x_2^2$
Uniforme	1.81824E-12	-3.86054E-12	1.82097E-23
Gaussiano	**4.02235E-15**	**-1.85888E-15**	**1.96347E-29**
Exponencial	3.80961E-14	4.19444E-14	3.21064E-27

Para generar un valor de una variable aleatoria X con distribución exponencial y parámetro λ a partir de un valor de una variable aleatoria $u = u(0,1)$ se utiliza el método de la transformada inversa para obtener la siguiente ecuación:

$$X = -\frac{1}{\lambda} \ln (1 - u) . \tag{9}$$

Utilizando el hecho de que si $u = u(0,1)$ entonces $1 - u = u(0,1)$. El esquema para generar un número aleatorio con distribución exponencial es:

1. Generar un número aleatorio u .
2. Tomar $x = -\frac{1}{\lambda} \ln(u)$.

3.4. TS-MBFOA con reinicio de población

En esta propuesta, el algoritmo TS-MBFOA es adaptado para reiniciar la mitad de la población, en este caso las peores bacterias, con una población aleatoria con distribución gaussiana o exponencial. El reinicio es único durante las generaciones del algoritmo y se lleva acabo cuando la mitad de las bacterias de la población es factible, es decir, no se violan restricciones.

Esto para garantizar que la nueva población de bacterias este cerca o dentro de la región factible del problema y evitar que se generen bacterias fuera del espacio de búsqueda determinado por el rango de las variables del problema a resolver. Inicialmente TS-MBFOA la población de bacterias es totalmente generada con distribución uniforme.

De igual manera, se incluye un validador y corrector de números aleatorios con el objetivo de generar números dentro del espacio de búsqueda del problema. Después de generar cada número aleatorio con distribuciones gaussiana o exponencial.

Para ello, cada número es validado verificando que se encuentre dentro del rango mínimo y máximo de cada variable de diseño del problema a resolver. Cuando un número aleatorio viola los rangos permitidos, un nuevo número aleatorio es generado con el mismo tipo de distribución usada. Para el caso de la distribución uniforme, una bacteria es generada con la Ecuación 10:

$$\theta^i(j, G) = L_i x_i + u * (U_i x_i - L_i x_i), \tag{10}$$

donde $L_i x_i$ es el límite inferior de la variable x_i, $U_i x_i$ es el límite superior de la variable x_i y u es un número aleatorio entre (0,1). Una bacteria con distribución gaussiana o normal es generada con la Ecuación 11:

$$\theta^i(j, G) = u * \sigma(x_i) + \mu(x_i), \tag{11}$$

Tabla 2. Estadísticas básicas de las 30 ejecuciones independientes de TS-MBFOA con diferentes distribuciones.

Criterios	TS-MBFOA uniforme	TS-MBFOA gaussiana	TS-MBFOA exponencial
Mejor	**0.012665672**	0.012665852	0.012665788
Media	0.012670502	0.012670288	**0.012669101**
Peor	0.012682834	0.012692327	**0.012680927**
Desv. Est.	4.25232E-06	5.1684E-06	**3.8474E-06**

donde u es un número aleatorio con distribución $N(0,1)$, en lenguaje java corresponde a la función nextGaussian(). μ es la media de la mitad de la población de bacterias factibles para cada variable x_i y σ es la desviación estándar de la mitad de la población de bacterias factibles para cada variable x_i. Una bacteria con distribución exponencial es generada con la Ecuación 12:

$$\theta^i(j,G) = \left(-\frac{1}{\lambda}\right)\ln(u), \tag{12}$$

donde u es un número aleatorio entre [0,1] y λ es la media de la mitad de la población de bacterias factibles para cada variable x_i. La media y desviación estándar utilizada en estas distribuciones hacen uso de la información obtenida por las bacterias hasta la generación donde la mitad de la población de bacterias es factible.

4. Resultados

El algoritmo TS-MBFOA con reinicio de población fue probado en dos problemas de pruebas y ejecutado una computadora con las características: Laptop de 4GB RAM, procesador de 2.3Ghz y un sistema operativo Windows de 64 bit. El lenguaje de programación fue java con el entorno de desarrollo integrado Netbeans IDE 8.0.2.

Dos experimentos fueron realizados para analizar el rendimiento del algoritmo con diferentes distribuciones. En el experimento 1, TS-MBFOA es probado con el problema conocido como la esfera con el objetivo de observar de manera gráfica la distribución de la población de bacterias con las diferentes distribuciones, cabe mencionar que el algoritmo TS-MBFOA inicia con una población con distribución uniforme.

El problema de la esfera se representa como $f(\vec{x}) = x_1^2 + x_2^2$, en este problema el rango de ambas variables es [-5, 5]. Los parámetros del algoritmo TS-MBFOA fueron calibrados con un conjunto de 50 ejecuciones independientes con cinco combinaciones de valores diferentes a los parámetros.

La mejor combinación de valores resultante fue: bacterias (S_b) = 50, tamaño de paso (R) = 0.0005, factor de escalamiento (β) = 1.8, ciclo quimiotáxico (N_c) = 12, bacterias a reproducir (S_r) = 1, frecuencia de reproducción ($\mathrm{RepCycle}$) = 100 y número de evaluaciones = 15,000.

En la Figura 3.2 se presenta la población inicial de bacterias del algoritmo TS-MBFOA en tres ejecuciones independientes. En el problema de la esfera es posible graficar sus variables debido a que solo cuenta con dos dimensiones.

Tabla 3. P-value de la prueba Wilcoxon signed rank test aplicada a los resultados de TS-MBFOA con diferentes distribuciones.

TS-MBFOA uniforme vs exponencial	**TS-MBFOA uniforme vs gaussiana**	**TS-MBFOA exponencial vs gaussiana**
0.05744	0.68916	0.28914

Este problema no tiene restricciones de igualdad o desigualdad, por lo tanto, el reinicio de población con distribución gaussiana y exponencial es llevada a cabo en la generación 10 del algoritmo para efectos de visualizar como quedan dispersas las bacterias en el espacio de búsqueda.

Como se puede observar en las gráficas, la población de bacterias con distribución exponencial se encuentra más agrupada hacia el origen, cabe mencionar, que el valor óptimo de la función de la esfera se encuentra en dicho lugar, cuando x_1 y x_2 toman el valor de cero.

La población de bacterias con distribución uniforme es la que abarca más lugares en el espacio de búsqueda, lo cual permite una mayor diversidad en la población, sin embargo, esto puede alentar al algoritmo a encontrar las zonas prometedoras donde se encuentre el óptimo global del problema.

Finalmente, la población de bacterias con distribución gaussiana hace el efecto de campana, generando pocas bacterias en los límites del espacio de búsqueda y concentra a muchas en la parte central de los límites de ambas variables de diseño.

En la Tabla 1 se presenta a la mejor bacteria de la población al finalizar la ejecución del algoritmo TS-MBFOA con diferentes distribuciones, donde el algoritmo TS-MBFOA con distribución gaussiana obtuvo el mejor resultado.

En el experimento 2, el algoritmo TS-MBFOA con diferentes distribuciones fue probado en el problema del resorte de tesión/compresión, donde se busca minimizar el peso de un resorte 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 procesadas 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 el número de rollos involucrados N(x_3). Formalmente, el problema puede expresarse de la siguiente manera:

$$(x_3 + 2)x_2\ x_1^2. \tag{13}$$

Sujeto a:

$$g_1(X) = 1 - ((x_2^3\ x_3)/(71785\ x_1^4)) \leq 0, \tag{14}$$

$$g_2(X) = ((4\ x_2^2 - x_1\ x_2)/12566(x_2\ x_1^3 - x_1^4))) + (1/5108\ x_1^2) - 1 \leq 0, \tag{15}$$

$$g_3(X) = 1 - (140.45\ x_1/x_2^2\ x_3) \leq 0, \tag{16}$$

$$g_4(X) = (x_2 + x_1/1.5) - 1 \leq 0, \tag{17}$$

donde:

$$\begin{aligned} 0.05 &\leq x_1 \leq 2, \\ 0.25 &\leq x_2 \leq 1.3, \\ 2 &\leq x_3 \leq 15. \end{aligned} \tag{18}$$

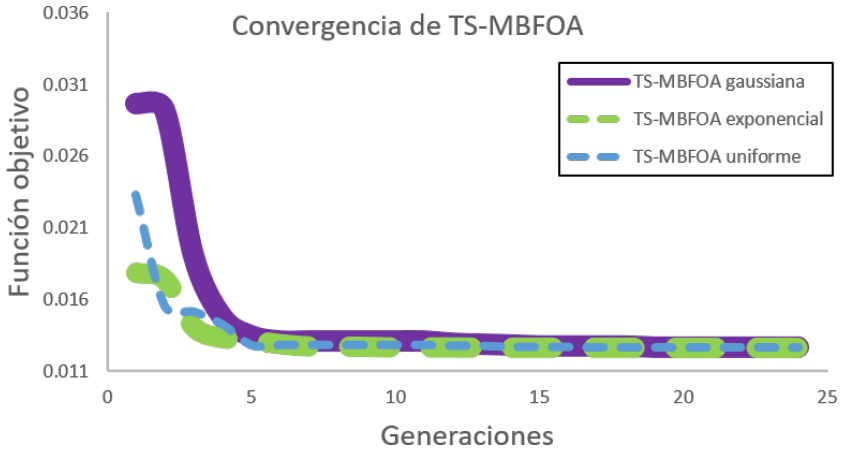

Fig. 3. Convergencia de TS-MBFOA con distribución uniforme, gaussiana y exponencial.

El algoritmo TS-MBFOA fue ejecutado en 30 corridas independientes con los mismos valores a los parámetros usados en el experimento 1. Los resultados estadísticos de las ejecuciones se presentan en la Tabla 2 donde la versión del TS-MBFOA con distribución uniforme obtuvo la mejor solución al problema, sin embargo, el algoritmo TS-MBFOA con reinicio de población exponencial obtuvo los mejores resultados en media, peor y desviación estándar.

La prueba no paramétrica de Wilcoxon signed rank test fue aplicada con un 95 % de certeza sobre el conjunto de las 30 mejores soluciones encontradas por cada versión del algoritmo TS-MBFOA. Los resultados indican que no hay una diferencia significativa al comparar las tres versiones del algoritmo TS-MBFOA como se muestra en la Tabla 3.

La gráfica de convergencia del algoritmo TS-MBFOA es presentada en la Figura 4. Los datos usados para generar las gráficas corresponden a la ejecución número 15, la cual es la mediana de las 30 ejecuciones independientes realizadas. En esta gráfica se puede observar que el algoritmo TS-MBFOA con distribución exponencial converge de manera más rápida antes de la generación número 5.

Los algoritmos con distribución uniforme y gaussiana convergen después de la generación número 5, aunque en la versión con distribución uniforme se logra apreciar como el algoritmo converge en algún óptimo local antes de la generación número 5, sin embargo, logra salir y converger en el óptimo global.

En general, el algoritmo TS-MBFOA obtiene resultados competitivos para el problema de prueba con las diferentes distribuciones. La versión con distribución exponencial obtiene resultados con menos desviación estándar, lo cual permite inferir que es más estable que las otras versiones.

5. Conclusión

El algoritmo basado en el forrajeo de bacterias E.Coli TS-MBFOA fue probado con distintas distribuciones. La versión original del algoritmo probado crea su población de bacterias con distribución uniforme.

En este artículo se propuso reiniciar la mitad de la población de bacterias (peores) con una población de bacterias aleatorias con distribuciones gaussiana y exponencial con el objetivo de observar el rendimiento del algoritmo en problemas de optimización numérica con restricciones.

El reinicio de la población se lleva acaba en una sola ocasión cuando la mitad de población de bacterias es factible debido a que se hace uso de la media y desviación estándar de esta mitad de bacterias para poder generar las nuevas bacterias con distribución gaussiana o exponencial.

Dos experimentos fueron realizados, en el primero el problema de la esfera fue implementado para observar de manera gráfica como se distribuyen las bacterias en el espacio de búsqueda. En el segundo experimento, TS-MBFOA fue probado en un problema de diseño ingenieril conocido como resorte de tensión/compresión.

30 ejecuciones independientes fueron realizadas a las tres versiones del algoritmo con distribución uniforme, gaussiana y exponencial. Donde los resultados presentan una mejor consistencia con la versión del TS-MBFOA con distribución exponencial.

Sin embargo, la prueba no paramétrica de Wilcoxon signed rank test con el 95 % de confianza determinó que no hay diferencia significativa entre los resultados de las tres versiones del algoritmo.

Como trabajo futuro, se espera probar cada una de las versiones del algoritmo TS-MBFOA con diferentes distribuciones en otros problemas de optimización numérica con restricciones y comprobar si el uso de otra distribución diferente de la uniforme permite una convergencia más rápida del algoritmo sin ser una convergencia prematura.

Referencias

1. Arabas, J., Opara, K.: Population diversity of nonelitist evolutionary algorithms in the exploration phase. IEEE Transactions on Evolutionary Computation, vol. 24, no. 6, pp. 1050–1062 (2020) doi: 10.1109/TEVC.2019.2917275
2. Chen, H., Zhang, Q., Luo, J., Xu, Y., Zhang, X.: An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine. Applied Soft Computing, vol. 86, pp. 105884 (2020) doi: 10.1016/j.asoc.2019.105884
3. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, vol. 186, no. 2-4, pp. 311–338 (2000) doi: 10.1016/s0045-7825(99)00389-8
4. Hernández-Ocaña, B., Pozos-Parra, M. D. 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, vol. 2016, pp. 1–18 (2016) doi: 10.1155/2016/4525294
5. Hernandez-Ocaña, B., Chavez-Bosquez, O., Hernandez-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
6. Hernandez-Ocaña, B., Pozos-Parra, M. D. 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
7. Hernández-Ocaña, B., Hernández-Torruco, J., Chávez-Bosquez, O., Calva-Yáñez, M., Portilla-Flores, E.: Bacterial foraging-based algorithm for optimizing the power generation

of an isolated microgrid. Applied Sciences, vol. 9, no. 6, pp. 1261 (2019) doi: 10.3390/app9061261

8. 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 (2013) doi: 10.1080/0305215x.2013.776550
9. Mezura-Montes, E., Hernández-Ocaña, B.: Bacterial foraging for engineering design problems: Preliminary results. In: Memorias del 4to Congreso Nacional de Computación Evolutiva (2008)
10. Passino, K. M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems, vol. 22, no. 3, pp. 52–67 (2002) doi: 10.1109/mcs.2002.1004010
11. dos Santos Coelho, L., Ayala, H. V. H., Mariani, V. C.: A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. Applied Mathematics and Computation, vol. 234, pp. 452–459 (2014) doi: 10.1016/j.amc.2014.01.159
12. de Galicia. bvón mantida e publicada en internet pola Consellería de Sanidade e o Servizo Galego de Saúde, X.: Ayuda de distribuciones de probabilidad. In: Distribuciones de Probabilidad. pp. 1–72 (2014)
13. Singh, P., Dwivedi, P., Kant, V.: A hybrid method based on neural network and improved environmental adaptation method using controlled Gaussian mutation with real parameter for short-term load forecasting. Energy, vol. 174, pp. 460–477 (2019) doi: 10.1016/j.energy.2019.02.141
14. Tirumala, S. S.: A quantum-inspired evolutionary algorithm using Gaussian distribution-based quantization. Arabian Journal for Science and Engineering, vol. 43, no. 2, pp. 471–482 (2017) doi: 10.1007/s13369-017-2641-9