

Optimización en Ingeniería Basada en el Forrajeo de Bacterias: Una Propuesta Preliminar.

Efrén Mezura-Montes* y Betania Hernández-Ocaña[‡]

* Laboratorio Nacional de Informática Avanzada
Rébsamen 80, Centro, Xalapa, Veracruz, 91000, MÉXICO
emezura@lania.mx

‡Universidad Juárez Autónoma de Tabasco
División Académica de Informática y Sistemas
Km. 1 Carretera Cunduacán-Jalpa de Méndez
042H4005@dais.ujat.mx

Resumen Este artículo presenta una propuesta preliminar de la adaptación del algoritmo basado en el comportamiento social de grupos de bacterias para solucionar problemas de diseño en ingeniería sin usar funciones de penalización. Tres mecanismos se agregan para dicho fin: (1) el uso de criterios simples de factibilidad para manejar las restricciones del problema, (2) una estrategia de control para los límites de las variables de diseño y (3) un tamaño de paso diferente para cada variable de diseño. El desempeño del algoritmo propuesto fue evaluado en dos problemas de benchmark en diseño en ingeniería. Los resultados obtenidos son competitivos al compararlo con otras estrategias representativas del estado del arte.

Palabras Clave: Inteligencia colectiva, optimización con restricciones, forrajeo de bacterias

1. Introducción

Los algoritmos evolutivos, (AEs) son ampliamente utilizados como técnicas alternativas a la programación matemática para resolver problemas de optimización [10]. Por otro lado, en los años 1990's surge un conjunto de algoritmos que ya no emulan la evolución de las especies, sino comportamientos colaborativos hallados en animales muy simples como insectos o aves. A este conjunto de nuevos algoritmos se les agrupó en el área de la Inteligencia Colectiva (Swarm Intelligence en inglés) [5]. Los dos algoritmos que dieron origen al área fueron la optimización mediante cúmulos de partículas (Particle Swarm Optimization, PSO) [7] y la Colonia de Hormigas [4]. PSO está basado en las coreografías que siguen algunas aves al buscar alimento o refugio y está diseñado para resolver problemas de optimización numérica. La Colonia de Hormigas modela diferentes comportamientos encontrados en los hormigueros y principalmente se ha utilizado para resolver problemas combinatorios.

La emulación de comportamientos de forrajeo a nivel de bacterias es considerado un algoritmo novedoso dentro de la inteligencia en cúmulos. Las ideas iniciales fueron propuestas por Bremermann [1] y luego aplicadas por Bremermann y Anderson en el entrenamiento de una red neuronal [2]. De este modelo inicial, llamado Bacteria Chemotaxis (BC), por emular la reacción de las bacterias a atractores químicos, se han reportado estudios de aplicación en problemas de optimización numérica no restringida [11]. Sin embargo, el modelo más popular en la literatura especializada es el propuesto por Passino [12] y llamado Bacterial Foraging Optimization Algorithm (BFO), que a diferencia del modelo BC, toma en cuenta el proceso completo de forrajeo de la bacteria *E. Coli*: movimientos de nado y giro (chemotaxis), reproducción y eliminación-dispersión. Este modelo ha sido utilizado para resolver problemas de optimización en diferentes áreas [8]. Sin embargo, su uso en la resolución de problemas en espacios restringidos no se ha reportado en la literatura especializada y de ahí nace la motivación del presente trabajo, donde se parte de la hipótesis de que el BFO puede ser una opción heurística que provea de resultados competitivos al resolver problemas de diseño en ingeniería, con respecto a otros algoritmos. En este trabajo se presenta una adaptación del modelo original del BFO para resolver problemas de diseño en ingeniería en presencia de restricciones. La intención es agregar tres mecanismos que le permitan a las bacterias encontrar primero soluciones factibles y después optimizar el valor de la función de costo de diseño de dos problemas de ingeniería. Este artículo está organizado de la siguiente manera: En la Sección 2 se presenta la definición formal del problema a resolver. En la Sección 3 se explican el algoritmo BFO así como las tres modificaciones hechas al algoritmo original. Posteriormente, en la Sección 4 se presentan los resultados obtenidos con nuestro BFO modificado para establecer las conclusiones y el trabajo futuro en la sección 5.

2. Definición del problema

Muchos problemas de ingeniería se pueden modelar como el problema general de programación no lineal en el cual queremos: encontrar \mathbf{x} que optimiza $f(\mathbf{x})$ sujeta a: $g_i(\mathbf{x}) \leq 0, i = 1, \dots, m$ $h_j(\mathbf{x}) = 0, j = 1, \dots, p$ donde $\mathbf{x} \in R$ es el vector de soluciones $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, donde cada $x_i, i = 1, \dots, n$ está definida por los límites 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 y con S a todo el espacio de búsqueda, entonces debe de ser claro que $F \subseteq S$. Es bien sabido que este problema no cuenta con un método que lo resuelva eficaz y eficientemente en todos los casos. Por ende existe una diversidad de métodos clásicos [14] y heurísticos [10] que lo atacan de diversas formas.

3. Nuestra propuesta

El comportamiento quimiotáctico de bacterias puede ser modelado como un proceso de optimización. Esta idea fue desarrollada por Hans Bremermann, [1] quien observó el siguiente comportamiento de bacterias al momento de buscar áreas con altos contenidos de nutrientes:

1. Las bacterias giran y salen corriendo en línea recta con una dirección aleatoria.
2. Si la dirección apunta hacia una concentración creciente, las bacterias continúan en esa dirección hasta que la concentración disminuye o se llega al máximo (o mínimo si el problema de optimización lo indica). Antes de que las bacterias sientan la disminución de concentración, estas giran y toman una nueva dirección aleatoria.
3. Si la dirección apunta hacia una concentración disminuida, entonces las bacterias detienen su nado o recorrido, giran y toman una nueva dirección aleatoria.
4. Las direcciones aleatorias son independientes y uniformemente distribuidas.

Estas ideas iniciales motivaron a Passino [12] a proponer un modelo que también contempla los pasos de reproducción y eliminación de bacterias y así lograr la cooperación entre ellas para llegar a áreas prometedoras en su ambiente. Estas ideas dan origen al siguiente algoritmo (ver Tabla 1):

<ol style="list-style-type: none">1 Inicialización de parámetros del algoritmo.2 Generar la población inicial de bacterias aleatoriamente.3 Repetir hasta que se cumpla un número máximo de ciclos de eliminación-dispersión:<ol style="list-style-type: none">3.1 Repetir hasta que se cumpla un número máximo de ciclos de reproducción:<ol style="list-style-type: none">3.1.1 Repetir hasta que se cumpla un número máximo de ciclos quimiotácticos:<ol style="list-style-type: none">3.1.1.1 Aplicar los operadores de nado y giro a cada bacteria3.1.2 Duplicar a las mejores bacterias de la población.3.2 Eliminar a la peor bacteria.
--

Tabla 1. Pseudocódigo del BFO

En este trabajo se propone adaptar el algoritmo de Passino [12] (Tabla 1) para resolver problemas de diseño en ingeniería como se mencionó en la Sección 2. Tres mecanismos son añadidos al BFO:

1. Se agrega un mecanismo para manejar las restricciones del problema basado en las reglas de factibilidad de Deb [3]
 - a) Entre dos soluciones factibles, aquella con el mejor valor de la función objetivo es seleccionada.
 - b) Entre una solución factible y otra no factible, la factible es seleccionada.

- c) Entre dos soluciones no factibles, aquella con la menor suma de violación de restricciones es seleccionada. La suma de violación de restricciones se calcula con la siguiente fórmula $\left(\sum_{i=1}^{m+p} \max(0, g_i(\mathbf{x}))\right)$.
2. Se agrega un mecanismo simple para asegurar que los elementos en \mathbf{x} (variables de diseño) no rebasen sus límites establecidos $L_i \leq x_i \leq U_i, \forall i = 1 \dots, n$. Si $x_i < L_i$ entonces $x_i = 2L_i - x_i$ y Si $x_i > U_i$ entonces $x_i = 2U_i - x_i$.
 3. Se utilizan tamaños de paso diferentes para cada variable del problema calculados de la siguiente manera: $\sigma_i(0) = 0.0021 * (\Delta x_i / \sqrt{n})$, donde Δx_i es determinado por $U_i - L_i$, donde $U_i - L_i$ son el límite superior e inferior de la variable de diseño i y n es el número de variables del problema.

El pseudocódigo para el algoritmo BFO modificado se presenta en la Tabla 2, donde los pasos quimiotácticos, de reproducción y de eliminación-dispersión se agrupan en un solo ciclo generacional.

<ol style="list-style-type: none"> 1.- Inicialización de parámetros del algoritmo. Los tamaños de paso se inicializan por cada variable 2.- Generar la población inicial de bacterias aleatoriamente. 3.- Repetir hasta que se cumpla un número máximo de generaciones: <ol style="list-style-type: none"> 3.1 Aplicar el ciclo quimiotáctico (nado y giro) N_c veces usando las reglas de Deb y el mecanismo de límites 3.2 Duplicar a las mejores bacterias de la población. 3.3 Eliminar a la peor bacteria.

Tabla 2. Pseudocódigo del BFO modificado

4. Experimentos y resultados

Para efectos de probar el desempeño de esta propuesta se utilizó el diseño experimental comúnmente reportado en la literatura especializada [6], donde el comportamiento de un algoritmo con elementos estocásticos se evalúa mediante un conjunto de corridas independientes con el mismo conjunto de valores para los parámetros del algoritmo y el cálculo de estadísticas para determinar la calidad y consistencia de los resultados. Los problemas de diseño en ingeniería comúnmente usados como benchmark[9] y utilizados en este trabajo se detallan a continuación:

Problema 1: Resorte de tensión/compresión.

Se busca minimizar el peso de un resorte de tensión/compresión. (Ver Figura 1) 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 las variables para ser procesadas en el algoritmo BFO son: Diámetro del rollo $D(x_2)$, diámetro del cable $d(x_1)$ y el número de rollos involucrados $N(x_3)$.

Formalmente, el problema puede expresarse de la siguiente manera:

Minimizar: $f(N, D, d) = (N + 2)Dd^2$

Sujeto a:

$$g_1(\mathbf{x}) = 1 - (D^3 N / 71785 d^4) \leq 0$$

$$g_2(\mathbf{x}) = (4D^2 - dD / 12566 (Dd^3 - d^4)) + (1 / 5108 d^2) - 1 \leq 0$$

$$g_3(\mathbf{x}) = 1 - (140.45 d / D^2 N) \leq 0 \quad g_4(\mathbf{x}) = (D + d / 1.5) - 1 \leq 0$$

Donde: $0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3$ y $2 \leq x_3 \leq 15$.

Problema 2: Viga soldada.

Se busca minimizar el costo de una viga soldada (ver Figura 1), sujeta a restricciones como la tensión de corte (τ), tensión de dobléz (σ), sujeción de la carga sobre la barra (P_c), desviación final de la viga (δ) y restricciones de equipo. Formalmente, el problema puede expresarse de la siguiente manera:

Minimizar: $f(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

Sujeto a:

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{max} \leq 0 \quad g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{max} \leq 0$$

$$g_3(\mathbf{x}) = x_1 - x_4 \leq 0 \quad g_4(\mathbf{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$$

$$g_5(\mathbf{x}) = 0.125 - x_1 \leq 0 \quad g_6(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{max} \leq 0 \quad g_7(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0$$

Donde: $\tau(\mathbf{x}) = \sqrt{(\tau)^2 + 2\tau'\tau''\frac{x_2^2}{2R} + (\tau'')^2}$ $\tau' = \frac{P}{\sqrt{2x_1x_2}}, \tau'' = \frac{MR}{J}$,

$$M = P \left(L + \frac{x_2}{2} \right), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2} \right)^2}$$

$$J = 2 \left\{ \frac{x_1x_2}{\sqrt{2}} \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2} \right)^2 \right] \right\} \sigma(\mathbf{x}) = \frac{6PL}{x_4x_3^2}, \delta(\mathbf{x}) = \frac{4PL^3}{Ex_3^3x_4}$$

$$P_c(\mathbf{x}) = \frac{4,013 \sqrt{\frac{EGx_3^2x_4^6}{36}}}{L^2} \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right)$$

$P = 6000\text{lb}, L = 14\text{in}, E = 30 \times 10^6\text{psi}, G = 12 \times 10^6\text{psi}, \tau_{max} = 13,600\text{psi},$
 $\sigma_{max} = 30,000\text{psi}, \delta_{max} = 0.25\text{in}$

Donde: $0.1 \leq x_1 \leq 2.0, 0.1 \leq x_2 \leq 10.0, 0.1 \leq x_3 \leq 10.0$ y $0.1 \leq x_4 \leq 2.0$.

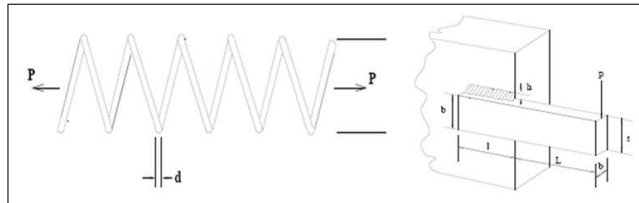


Figura 1. Resorte de tensión/compresión y viga soldada a optimizar.

Los parámetros utilizados para encontrar el valor óptimo de las variables y obtener el costo mínimo de la función son los siguientes: No. de bacterias= 50, $N_c = 18$, No. máximo de generaciones= 150, Bacterias a reproducirse= 25, Bacterias a eliminarse= 1.

Se ejecutaron 30 corridas diferentes a cada uno de los problemas presentados, con los parámetros mencionados anteriormente y se obtuvieron las siguientes estadísticas (mejor, media, desviación estándar) resumidas en la Tabla 3 en el renglón titulado “Propuesta”. En la Figura 2 se presenta la gráfica de convergencia para cada uno de los problemas en la corrida en el valor de la mediana de las 30 ejecuciones independientes. Se puede observar que en el problema 1 hacia la generación 20 el algoritmo logra converger a una solución competitiva, por otro lado, en el problema 2 el algoritmo converge una primera vez en la generación 14 y la convergencia final ocurre en la generación 129.

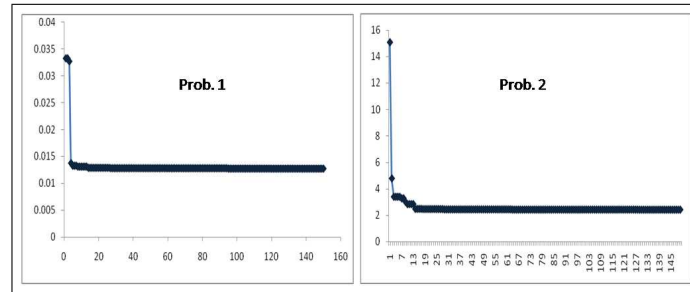


Figura 2. Gráficas de convergencia para los problemas de prueba.

Para comparar nuestros resultados se utilizaron 6 algoritmos del estado del arte en optimización bio-inspirada: COMOGA, CHVEGA, CHNPGA y CHMOGA, [9], todos algoritmos genéticos con manejo de restricciones usando conceptos multiobjetivo, la técnica de Ray & Liew [13] que realiza una simulación de una civilización y el PSO de He et al. [6]. En la Tabla 3 se presentan las estadísticas resultantes de las diferentes técnicas para los problemas de prueba.

Basado en los resultados obtenidos y resumidos en la tabla 3, discutiremos las siguientes características:

Calidad de los resultados (mejor solución obtenida): Para el problema 1, He et al. obtiene el mejor resultado, nuestra propuesta se coloca en la tercera posición. He et al. obtienen el mejor resultado para el problema 2, nuestra propuesta se coloca en la tercera posición.

Prob.	Algoritmos	Óptimo	Mejor	Media	Desv.Est.	Eval.
Prob.1	Ray & Liew	0.012681	0.012669	0.012923	5.96E-4	25167
	He et al.		0.012665	0.012702	4.1E-5	15000
	COMOGA		0.012929	0.014362	8.64E-4	80000
	CHVEGA		0.012688	0.012886	2.09E-4	80000
	CHNPGA		0.012683	0.012752	6.20E-5	80000
	CHMOGA		0.12680	0.012960	3.63E-4	80000
Propuesta	0.012671	0.012759	1.36E-4	135000		
Prob.2	Ray & Liew	2.380	2.385	3.255	9.6E-1	33000
	He et al.		2.380	2.381	5.2E-3	30000
	COMOGA		2.471	2.726	1.20E-1	80000
	CHVEGA		2.386	2.393	3.8E-3	80000
	CHNPGA		2.382	2.420	2.56E-2	80000
	CHMOGA		2.386	2.504	9.9E-2	80000
	Propuesta		2.383	2.400	1.2E-2	135000

Tabla 3. Resultados experimentales de los 2 problemas de prueba. El resultado en **negrita** significa el mejor resultado.

Consistencia de los resultados (mejor valor en la media y desviación estándar): Para el problema 1, He et al. muestra los mejores resultados, quedando nuestra propuesta en la tercera posición. Para el problema 2, He et al. muestran buenos resultados con respecto a la media, pero en cuestión de desviación estándar CHVEGA obtiene los mejores resultado, nuestra propuesta se coloca en la tercera posición.

Con respecto al número de evaluaciones requeridas por cada algoritmo para obtener los resultados antes discutidos (última columna en la Tabla 3, puede verse que el algoritmo de He et al. es el que menos evaluaciones requiere. Sin embargo, este algoritmo, para poder funcionar, requiere que se genere una población inicial de soluciones factibles (que cumplen con las restricciones del problema), lo cual puede ser muy difícil en problemas con zonas factibles muy pequeñas (generar aleatoriamente soluciones factibles es una tarea muy complicada que puede tomar mucho tiempo de cómputo). Por otro lado, nuestro algoritmo basado en bacterias, aunque requiere de muchas más evaluaciones, parte de una población inicial de soluciones aleatorias, normalmente no factibles, lo cual facilita su uso y disminuye el tiempo de cómputo en problemas altamente restringidos.

5. Conclusiones y trabajos futuros

En este artículo se presentó una propuesta preliminar de adaptación del algoritmo basado en el forrajeo de bacterias para resolver problemas de optimización en ingeniería en presencia de restricciones. El algoritmo propuesto logró encontrar soluciones factibles en las 30 corridas independientes, además de converger

rápidamente a una solución competitiva. Al ser comparado con algoritmos bioinspirados del estado del arte, nuestra propuesta mostró resultados de calidad y una consistencia moderada. El costo computacional del algoritmo propuesto es aún alto, pero parte del trabajo futuro consiste en disminuir ese alto número de evaluaciones y mejorar su consistencia en los resultados finales.

Agradecimientos: El primer autor agradece el apoyo de CONACyT mediante el proyecto 79809. El segundo autor agradece a la UJAT por el apoyo para realizar una estancia de verano en LANIA.

Referencias

1. H. Bremermann. Chemotaxis and optimization. *J. Franklin Inst.*, 297:397–404, 1974.
2. H. Bremermann and R. Anderson. An alternative to back-propagation: A simple rule of synaptic modification for neural net training and memory. Technical Report PAM-483, Center for Pure and Applied Mathematics, University of California, Berkeley, California, USA, 1997.
3. K. Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.
4. M. Dorigo, V. Maniezzo, and A. Coloni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions of Systems, Man and Cybernetics-Part B*, 26(1):29–41, 1996.
5. A. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.
6. S. He, E. Prempan, and Q.H.Wu. An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems. *Engineering Optimization*, 36(5):585–605, October 2004.
7. J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, UK, 2001.
8. R. Majhi, G. Panda, G. Sahoo, P. Dash, and D. Das. Stock market prediction of s&p 500 and djia using bacterial foraging optimization technique. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 2569–2575. Singapore, IEEE Service Center, September 2007.
9. E. Mezura-Montes and C. A. C. Coello. Constrained optimization via multiobjective evolutionary algorithms. In *Multiobjective Problems Solving from Nature*, pages 53–76. Springer, Heidelberg, Germany, 2008.
10. Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Germany, 2nd edition, 2004.
11. S. D. Muller, J. Marchetto, S. Airaghi, and P. Koumoutsakos. Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, 6(1):16–29, 2002.
12. K. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67, 2002.
13. T. Ray and K. Liew. Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior. *IEEE Transactions on Evolutionary Computation*, 7(4):386–396, August 2003.
14. G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. *Engineering Optimization. Methods and Applications*. John Wiley and Sons, 1983.