Un Nuevo Algoritmo Memético Basado en un Kernel Reproductor

1st Carlos O. Flor-Sánchez
División de Estudios de Posgrado e Investigación
Tecnológico Nacional de México/IT de Saltillo
Saltillo, México
cosvaldoflor@gmail.com

2nd Edgar O. Reséndiz-Flores
División de Estudios de Posgrado e Investigación
Tecnológico Nacional de México/IT de Saltillo
Saltillo, México
edgar.rf@saltillo.tecnm.mx

***Resumen*—En este estudio se presenta un nuevo método metaheurístico derivado del método de evolución del gradiente, en el cual se introduce por primera vez el concepto de kernel reproductor para estimar correctamente el gradiente numérico que se utiliza como regla de actualización. El método propuesto se denomina evolución del gradiente basado en kernel, explora el espacio de búsqueda utilizando un conjunto de vectores e incluye tres operadores principales: actualización, salto y regeneración. La dirección de búsqueda se determina utilizando la estimación del gradiente mediante el kernel reproductor con una expansión de la serie de Taylor local. El salto y la actualización de vectores permiten que este método evite los óptimos locales. Para evaluar el rendimiento del método presentado, se probaron catorce funciones de prueba. Después, fue realizada una comparación entre algunos de métodos metaheurísticos existentes en la literatura. Los resultados experimentales muestran que el método propuesto funciona mejor o igual de bien que otros métodos, como los algoritmos enjambre de partículas, evolución diferencial, colonia de abejas artificial y genético continuo, para la mayoría de los problemas de referencia probados.**

***Palabras clave*—Método de Optimización Metaheurística, Aproximación del Gradiente Basada en Kernel, Evolución del Gradiente, Algoritmos Meméticos.**

I. INTRODUCCIÓN

En los últimos años se han producido notables avances en los métodos y herramientas numéricas para modelar y optimizar los procesos tecnológicos en diferentes areas de la ingenieria. El modelado numérico y la optimización están presentes tanto en la investigación como en el sector industrial. Las técnicas de optimización numérica son ampliamente utilizadas en diversos campos como la ingenieria mecánica, química, civil, eléctrica o incluso en la física y la economía; Por mencionar sólo algunos. El objetivo de la optimización es encontrar una solución que maximice o minimice algunos parámetros operativos específicos involucrados en problemas del mundo real, ya sea un producto tecnológico, un servicio o incluso un proceso industrial.

Las técnicas de optimización se pueden clasificar en dos categorías: aproximadas (heurísticas y metaheurísticas) y exactas (basadas en gradientes). Los métodos aproximados como el algoritmo evolución diferencial [1], el algoritmo genético [2], el recocido simulado [3] y la optimización de enjambre de partículas [4] son algunos de los más populares y estudiados hasta ahora. Este tipo de algoritmos imitan comportamientos naturales o sociales. Aunque estos métodos no garantizan la mejor solución, tienen un buen equilibrio entre búsqueda local (explotación) y búsqueda global (exploración) para buscar efectivamente en el espacio dado [5].

Sin embargo, en la última década se han propuesto varios algoritmos novedosos y más eficientes. El principal objetivo es mejorar su capacidad de búsqueda global y la velocidad de convergencia. Por lo tanto, la optimización de ballenas (Whale Optimization Algorithm-WOA) [6], la búsqueda de polillas (Moth Search Algorithm - MS) [7], la optimización de mariposa monarca (Monarch Butter y Optimization-MBO) [8], el algoritmo de moho mucilaginoso (Slime Mould Algorithm-SMA) [9], la búsqueda de juegos del hambre (Hunger Games Search-HGS) [10], el optimizador RUNge Kutta (RUN) [11], el algoritmo de depredación de colonias (Colony Predation Algorithm-CPA) [12] y el optimizador de Harris Hawks (Harris Hawks Optimizer-HHO) [13] son otros buenos ejemplos de algoritmos innovadores que tienen un equilibrio entre exploración y explotación.

En el contexto de métodos exactos como el descenso de gradiente [14], el método de Newton [15], el gradiente conjugado [16], el método Quasi-Newton [17] y el algoritmo de Levenberg-Marquardt [18] , por citar a los más representativos, se garantiza la convergencia a la mejor solución. Sin embargo, requieren información de gradiente, lo que implica un gran desafío para atacar la mayoría de los complejos problemas de optimización del mundo real y, por lo general, quedan estancados en un óptimo local. Este hecho ha llevado a los investigadores de optimización a buscar alternativas para diseñar nuevos algoritmos capaces de enfrentar los problemas tecnológicos que se han vuelto cada vez más desafiantes. Para ello, se han incorporado los mejores atributos de diversas técnicas (operadores, filosofías, etc.) para sugerir nuevos algoritmos que comúnmente se denominan como meméticos (MAs) o híbridos [19]. A modo de ejemplo, J. Kuo y col. [20], presentaron un método de búsqueda basado en gradientes llamado gradient evolution (GE), que explora el espacio de búsqueda utilizando inteligencia de enjambre y su dirección de búsqueda es determinada por el método de Newton Raphson. A. Iman y col. [21], propusieron un algoritmo de optimización metaheurístico novedoso llamado optimizador basado en gradientes (GBO), este método se desarrolló a partir de las ideas del

método de Newton y también utiliza un conjunto de vectores para explorar el espacio de búsqueda. G. Saeed y col. [22], también utilizaron una idea similar, introdujeron un algoritmo metaheurístico que utiliza el método de Newton como su regla de actualización en un marco basado en la población, por lo que fue nombrado como newton metaheurístico (NMA). Layth R. Khale y col. [23], desarrollaron otro trabajo en esta dirección, combinaron el algoritmo de gradiente conjugado con el algoritmo de optimización de ballenas (WOA), este método (WOA-MCG) mostró una mejora significativa del algoritmo original. Finalmente, Huda I. Ahmed y col. [24], utilizaron las características de un nuevo algoritmo de gradiente conjugado espectral y lo aplicaron en el algoritmo de murciélagos para mejorar su rendimiento, el cual se denominó como algoritmo de murciélagos direccional (CG-BAT). Estos algoritmos mencionados previamente, exhibieron la importancia de incluir información del gradiente en su regla de actualización, ya que se obtuvieron resultados sobresalientes. Otra ventaja de los algoritmos híbridos es que no es necesario derivar la función objetivo, en lugar de eso, para un punto dado, $\mathbf{x}_i$, usan los dos vectores vecinos más cercanos para aproximar una dirección de descenso. Sin embargo, esta estrategia tiene tres inconvenientes principales (Fig. 1): primero, los vectores no siempre están a la misma distancia entre sí (1a), el segundo es que estos vectores no están necesariamente orientados en la dirección correcta (1b) y el último es que dos vectores vecinos no son suficientes para proporcionar información amplia sobre la curvatura de la función (1c), esto da como resultado una dirección de gradiente que no es confiable del todo. Por lo tanto, para hacer frente a los inconvenientes antes mencionados de los métodos híbridos reportados hasta ahora, la presente investigación busca desarrollar un algoritmo metaheurístico innovador basado en gradientes que calcule una aproximación de dirección de descenso a partir de más de 2 vecinos, presentando por primera vez el concepto de reproducción kernel para estimar correctamente el vector gradiente que se utiliza como parte de la ley de movimiento.

II. El algoritmo de optimización mono-objetivo propuesto

II-A. Aproximación de gradiente basada en funciones kernel

En esta sección se describe en detalle la idea central para el cálculo de gradientes basado en una función kernel reproductora. Suponga que un conjunto de puntos $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ alrededor de un $\mathbf{x}$ centrado están distribuidos con los valores de función correspondientes $f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_m)$, $\mathbf{x}_i, \mathbf{x} \in \Omega \subset \mathbb{R}^n$, $i = 1, ..., m$. El problema es encontrar un valor aproximado de ∇f en alguna ubicación centrada arbitrariamente $\mathbf{x}$. Por lo tanto, se puede aplicar el siguiente procedimiento:
Considere la expansión de Taylor de orden uno de $f(\mathbf{x}_i)$ alrededor de una partícula centrada $\mathbf{x}$,

$$f(\mathbf{x}_i) = f(\mathbf{x}) + \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(x_{k,i} - x_k) + e_i, \quad i = 1, ..., m \quad (1)$$

para $i = 1, ..., m$. Si esta expansión de Taylor se aplica a los m puntos se obtiene un sistema lineal de ecuaciones que se puede expresar en forma matricial como

$$\mathbf{e} = \mathbf{b} - M\mathbf{a}_f \quad (2)$$

donde

$$M = \begin{pmatrix} 1 & h_{1,1} & h_{2,1} & \cdots & h_{n,1} \\ 1 & h_{1,2} & h_{2,2} & \cdots & h_{n,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & h_{1,m} & h_{2,m} & \cdots & h_{n,m} \end{pmatrix} \quad (3)$$

con $h_{i,j} = (x_{i,j} - x_i)$ para $i = 1, ..., n$ y $j = 1, ..., m$.

$$\mathbf{a}_f = \left(f(\mathbf{x}), \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right)^t = (f(\mathbf{x}), \nabla f(\mathbf{x}))^t \quad (4)$$

$$\mathbf{b} = (f_1, f_2, ..., f_m)^t, \quad \text{dónde} \quad f_j = f(\mathbf{x}_j),\ j = 1, ..., m. \quad (5)$$

$$\mathbf{e} = (e_1, e_2, ..., e_m)^t \quad (6)$$

Si se minimiza la siguiente forma cuadrática

$$J = \sum_{j=1}^{n} w(\mathbf{x}, \mathbf{x}_j) e_j^2 \quad (7)$$

se pueden obtener los coeficientes óptimos. Por lo tanto, después de una minimización de $\mathbf{e}$ mediante el método de mínimos cuadrados ponderados, los estimadores de $\mathbf{a}_f$ serian:

$$\mathbf{a}_f = (M^t W M)^{-1} (M^t W)\mathbf{b}, \quad (8)$$

donde W se define como:

$$W = \begin{bmatrix} w(\mathbf{x} - \mathbf{x}_1) & 0 & \cdots & 0 \\ 0 & w(\mathbf{x} - \mathbf{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w(\mathbf{x} - \mathbf{x}_n) \end{bmatrix} \quad (9)$$

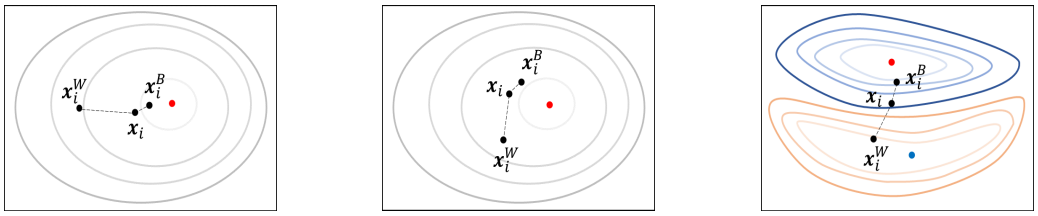
y w es una función kernel reproductora. En particular, en este trabajo se ha considerado una función de peso gaussiana que se define como

$$w(\mathbf{x} - \mathbf{x}_i) = \begin{cases} e^{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2 / h^2}, & \text{si } \frac{\|\mathbf{x} - \mathbf{x}_i\|}{h} \leq 1 \\ 0 & \text{de otra manera} \end{cases} \quad (10)$$

Esta función define la longitud de interacción entre nodos. Además, se pueden utilizar diferentes funciones de peso existentes en la literatura. Un esquema gráfico se presenta en la Fig. 2.

II-B. Método metaheurístico del gradiente basado en aproximación kernel

El método propuesto es llamado evolución del gradiente basado en kernel (KGE - Kernel Gradient Evolution) y se explica en detalle en esta sección. Dado a que utiliza un conjunto de vectores para explorar el espacio de búsqueda, pertenece a la familia de los algoritmos de inteligencia de enjambre. En cada iteración, tres operadores principales se utilizan para cada partícula: actualización, salto y regeneración.



(a) (b) (c)

Figura 1: Desventajas de una aproximación del gradiente basada en solo dos vectores vecinos.

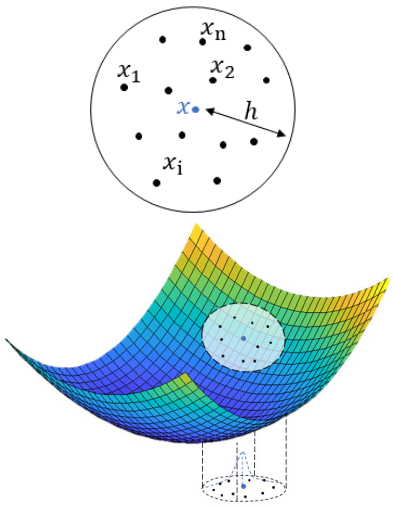


Figura 2: Concepto de aproximación basado en una función kernel

Paso 1: Inicialización

Primeramente, algunos hiperparámetros tienen que ser definidos: el número de iteraciones (T), el tamaño de la población (P), el tamaño del paso (α), la tasa de salto (J_r), la tasa de regeneración (S_r) y la tasa de reducción (ε). El número de iteraciones y el tamaño de la población depende de la complejidad del problema. Cuanto más complejo sea el problema, mayor número de iteraciones y vectores se requerirá. Los parámetros J_r, S_r y ε son números aleatorios con distribución uniforme en el intervalo [0, 1].

Subsecuentemente, la población inicial se distribuye aleatoriamente en el espacio de búsqueda, donde cada partícula representa un posible vector solución.

Paso 2: Actualización

La regla de actualización gobierna la dirección de búsqueda e implica dos partes: la regla de actualización basada en el gradiente y un factor de aceleración. El algoritmo KGE no deriva directamente la función objetivo; en su lugar, hace una estimación del vector gradiente tal como se describe en la Sección II-A.

Subsecuentemente, una vez estimado el gradiente, se renombra como $Kgrad$.

Entonces, la regla de actualización se transforma a un GradientMove, dado en (11):

$$GradientMove = -\alpha\,(kGrad) \tag{11}$$

Para mejorar la velocidad de convergencia de cada vector, también se agrega un factor de aceleración Acc. Este proceso utiliza el mejor vector de la población. El método KGE asume que el mejor vector es el más cercano a la solución óptima. Por lo tanto, considerando la mejor posición alcanzada hasta el momento $(\mathbf{y})$, los otros vectores se moverán en una mejor dirección. Adicionalmente, para asegurar que cada vector tenga un tamaño de paso diferente, el factor de aceleración (12) también se multiplica por un número aleatorio, $ra \sim [0, 1]$.

$$Acc = ra\left(\mathbf{y} - \mathbf{x}_i^t\right) \tag{12}$$

Finalmente, la ley de movimiento para el método KGE se establece de la siguiente manera:

$$\mathbf{u}_i^t = \mathbf{x}_i^t + GradientMove + Acc \tag{13}$$

donde $\mathbf{u}_i^t$ es el vector de transición obtenido al actualizar $\mathbf{x}_i^t$.

Paso 3: Salto

Este operador proporciona al método una habilidad de exploración, es decir, de buscar ampliamente a lo largo del espacio de búsqueda. El uso de este operador cambia significativamente la dirección del movimiento del vector. Para determinar si un vector dado puede saltar o no a una dirección diferente, el método KGE establece una tasa de salto (J_r). Por lo tanto, el salto vectorial se aplica a el vector de transición, $\mathbf{u}_i^t$, si $r_j \leq J_r$, donde $r_j \;\; N \sim (0, 1)$ y se actualiza en base a (14):

$$\mathbf{u}_i^t = \; -\mathbf{u}_i^t + \; rm \cdot (\mathbf{u}_i^t - \mathbf{nv}^t) \tag{14}$$

donde $\mathbf{nv}$ es cualquier vector vecino aleatorio en la población diferente al vector actual y $r_m \sim N\ (0, 1)$.

Paso 4: Regeneración

Este es un operador utilizado para proporcionar al método de elitismo, esto asegura que cada partícula se mueva siempre a una mejor posición. En la iteración t-ésima, el vector de transición, $\mathbf{u}_i^t$, almacena el resultado del vector actualizado y saltado, x_i^t. Por lo tanto, la siguiente posición del vector i, $\mathbf{x}_i^{t+1}$, se reemplaza por $\mathbf{u}_i^t$ solo si $f(\mathbf{u}_i^t)$ es mejor que $f(\mathbf{x}_i^t)$; si no, permanece en la siguiente iteración, $\mathbf{x}_i^{t+1}$=$\mathbf{x}_i^t$. Sin embargo, en algunos casos es difícil moverse a una mejor posición, generalmente ocurre cuando una partícula se estanca en un mínimo local. Por lo tanto, para solucionar este problema, la regeneración de vectores se realiza en un vector que se estanca después de varias iteraciones. Para identificar un vector estancado, el método registra el historial de actualización del vector que se denota como (s_i), el cual se inicializa en uno, este factor se reduce usando (ε) siempre que un vector no se mueva a una mejor posición. Si s_i es menor que s_r, el vector tiene que ser regenerado de acuerdo a (15):

$$s_i = s_i - \varepsilon \cdot s_i \tag{15}$$

donde ε es la tasa de reducción y es un número dentro de [0, 1].

Finalmente, el pseudocódigo para el algoritmo propuesto es presentado en el Algoritmo 1.

Algorithm 1 Método metaheurístico del gradiente basado en aproximación kernel

1: Inicialización, definir T, P, S_r, J_r, ε, α u_b, α l_b.
2: Crear una población aleatoria inicial
3: Establecer el historial de actualización inicial s_i=1 para cada vector
4: Evaluar vectores de acuerdo con la función objetivo
5: Almacenar el mejor vector $\mathbf{y}$
6: **for** cada iteración, t=1 to t =T **do**
7: **Actualización**
8: **for** cada vector, i=1 to i =P **do**
9: KGrad= Aproximación del gradiente basada en kernel
10: GradientMove= -α (Kgrad)
11: $\mathbf{u}_i^t = \mathbf{x}_i^t + GradientMove + Acc$
12: **Salto**
13: Generar $r_j \sim$ (0,1)
14: Seleccionar algún vecino (nv)
15: **if** $r_j <= J_r$ **then**
16: $\mathbf{u}_i^t = -\mathbf{u}_i^t + r_m \cdot (\mathbf{u}_i^t - nv^t)$
17: **end if**
18: Evaluar la función $f(\mathbf{u}_i^t)$
19: **Regeneración**
20: **if** $f(\mathbf{u}_i^t) < f(\mathbf{x}_i^t)$ **then**
21: $\mathbf{x}_i^{t+1}$=$\mathbf{u}_i^t$
22: **else**
23: $\mathbf{x}_i^{t+1}$, =$\mathbf{x}_i^t$ y actualizar s_i=s_i−ε·s_i
24: **end if**
25: **if** $s_i \leq s_r$ **then**
26: regenerar $\mathbf{x}_i^t$, calcular $f(\mathbf{x}_i^t)$ y definir s_i=1
27: **end if**
28: **end for**
29: actualizar $\mathbf{y}$
30: **end for**
31: Vector $\mathbf{y}$ es la solución final.

III. DESARROLLO EXPERIMENTAL

III-A. Validación

Para evaluar el rendimiento del método propuesto, este se puso a prueba con 14 funciones de referencia de optimización [25]. La Tabla I resume estas funciones y sus respectivas propiedades. El mínimo global para las funciones de prueba es $f(\mathbf{x}) = 0$, omitiendo la función Easom, que tiene su mínimo en $f(\mathbf{x}) = -1$. Con el fin de recopilar una buena cantidad de resultados estadísticos, se llevan a cabo 30 corridas de optimización independientes.

III-B. Configuración experimental

El método KGE utiliza una regla de tamaño de paso dinámico que disminuye a medida que aumenta el número de iteraciones y se da como

$$\kappa(i) = (u_b - l_b)\left(\frac{N_{iter} - i}{N_{iter} - 1}\right) + l_b, \tag{16}$$

donde l_b y u_b denotan los límites inferior y superior, respectivamente, N_{iter} es el número total de iteraciones y i se refiere a la iteración actual. Usando esta regla, el tamaño del paso disminuye gradualmente en el intervalo (l_b, u_b). Además, estos valores deben ser definidos cuidadosamente por el usuario.

Finalmente, la Tabla II resume la mejor combinación de ajustes de parámetros utilizados para comparar el algoritmo propuesto con otros métodos metaheurísticos con funciones en $\mathbb{R}^{10}$, $\mathbb{R}^{20}$ y $\mathbb{R}^{30}$.

Los resultados obtenidos se comparan con algunos algoritmos ya bien conocidos: PSO (Particle Swarm Optimization), DE (Differential Evolution), ABC (Artificial Bee Colony) y CGA (Continuous Genetic Algorithm).

IV. RESULTADOS

IV-A. Evaluación de la velocidad de convergencia

La velocidad de convergencia hacia la solución final es una característica importante de los algoritmos de optimización. La Fig. 3 muestra las gráficas de convergencia promedio de los algoritmos, los resultados experimentales indican que KGE supera a la mayoría de las funciones de prueba, más precisamente, tiene la mayor eficiencia en las funciones f_1, f_2, f_3, f_4,f_5, f_6, f_8, f_9, f_{10}, f_{11} y f_{12}.

Las otras funciones de prueba Rosenbrock f_7 y Withley (f_{14}) tienen el óptmimo global dentro de un valle largo y plano, lo que parece ser un problema para los métodos basados en gradientes como KGE, GE y DE. Sin embargo, el algoritmo ABC parece resolver esta tarea y alcanzó el mejor resultado en comparación con el resto de los métodos.

Según los resultados, los operadores de salto y regeneración juegan un papel importante en las funciones Ackley (f_9), Griewank (f_{10}) y Rastrigin (f_{11}), que son funciones con

Tabla I: Funciones de prueba

Nombre	Problema y rango
Sphere (f_1)	$f(x) = \sum_{i=1}^{D} x_i^2$, -100 $\leq x_i \leq$ 100
Weighted sphere (f_2)	$f(x) = \sum_{i=1}^{D} i x_i^2$, -100 $\leq x_i \leq$ 100
Sum of different power (f_3)	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$, -1 $\leq x_i \leq$ 1
Step function (f_4)	$f(x) = \sum_{i=1}^{D} \lfloor x_i + 0{,}5 \rfloor^2$, -100 $\leq x_i \leq$ 100
Schwefel 1.2 (f_5)	$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$, -100 $\leq x_i \leq$ 100
Schwefel 2.2 (f_6)	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$, -10 $\leq x_i \leq$ 10
Rosenbrock (f_7)	$f(x) = \sum_{i=1}^{D-1} ((x_i - 1)^2 + 100(x_{i+1} - x_i^2))$, -30 $\leq x_i \leq$ 30
Easom (f_8)	$f(x) = -(-1)(\prod_{i=1}^{D} cos^2(x_i) exp[-\sum_{i=1}^{D}(x_i - \pi)^2])$, -2$\pi \leq x_i \leq 2\pi$
Ackley (f_9)	$f(x) = -20exp(-0{,}2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)) + 20 + exp(1)$, -32.768 $\leq x_i \leq$ 32.768
Griewank (f_{10})	$f(x) = \frac{1}{400} \cdot \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}})$, -600 $\leq x_i \leq$ 600
Rastrigin (f_{11})	$f(x) = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10)$, -5.12 $\leq x_i \leq$ 5.12
Salomon (f_{12})	$f(x) = -cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}) + 0{,}1\sqrt{\sum_{i=1}^{D} x_i^2} + 1$, -100 $\leq x_i \leq$ 100
Schwefel 2.3 (f_{13})	$f(x) = 418{,}9829D - \sum_{i=1}^{D} x_i sin(\sqrt{\lvert x_i \rvert})$, -500 $\leq x_i \leq$ 500
Whitley (f_{14})	$f(x) = \sum_{i=1}^{D}\sum_{j=1}^{D} (\frac{y_{ij}^2}{4000} - cos(y_{ij}) + 1)$, $y_{ij} = (100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2$, -100 $\leq x_i \leq$ 100

Tabla II: La mejor configuración de parámetros

Función	Tasa de regeneración (s_r)	Tasa de salto (J_r)	Tasa de reducción (ε)	Rango de tamaño de paso
f_{13}	0.1	0.5	0.005	$(1, 1e^{-4})$
f_9, f_{10}, f_{11}	0.1	0.5	0.005	$(1e^{-32}, 1e^{-36})$
f_{12}	0.1	0.5	0.005	$(1e^{-64}, 1e^{-68})$
$f_1,\ f_2\ ,\ f_3, f_4,\ f_5, f_6$	0.1	0.5	0.005	$(1e^{-100}, 1e^{-104})$
$f_8,\ \ f_{14}$	0.1	0.5	0.005	Backtracking linesearch
f_7	0.5	0.1	0.005	Backtracking linesearch

muchos óptimos locales, un algoritmo que no puede resolver este problema puede quedar atrapado en estas zonas. Los resultados de convergencia muestran que el método KGE supera este desafío mejor que los otros algoritmos y también ofrece la solución óptima para las funciones Griewank y Rastrigin al igual que el algoritmo GE. Además, es notable que para las funciones de Salomon, Easom, Ackley y Griewank, el método KGE converge en mucho menos de 500 iteraciones.

Por otro lado, la función Schwefel 2.3 f_{13} es engañosa en el sentido de que el mínimo global está geométricamente distante, sobre el espacio de búsqueda, del resto de los mínimos locales. En consecuencia, la mayoría de los algoritmos de optimización tienden a converger en la dirección incorrecta. Esto explica por qué el método KGE requiere más iteraciones para encontrar el mínimo, pero profundiza más que el algoritmo DE cuando encuentra el valle correcto.

IV-B. Análisis de estabilidad y resultados estadísticos

Los resultados estadísticos (promedio y desviación estándar) obtenidos para los problemas $\mathbb{R}^{10}$,$\mathbb{R}^{20}$, y $\mathbb{R}^{30}$ son reportados en las Tablas III, IV y V respectivamente. Los resultados confirman que el método KGE produce resultados competitivos para los problemas de 3 dimensiones diferentes. Cabe señalar que las bajas desviaciones estándar del método KGE en casi todos los casos son evidencia de su estabilidad.

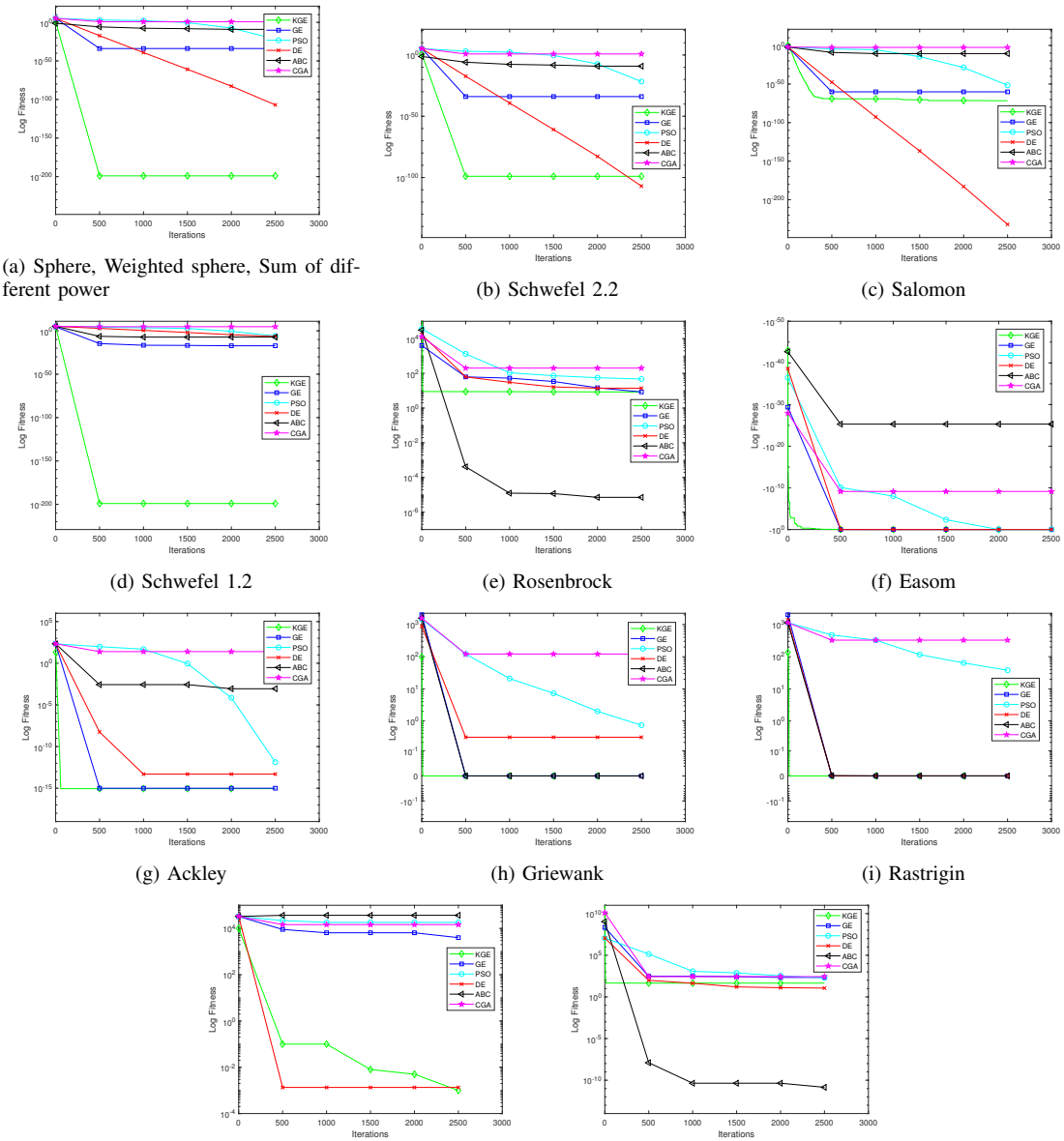
Además, se puede observar que el método KGE dió mejores resultados que la mayoría de los otros métodos para problemas $\mathbb{R}^{10}$,$\mathbb{R}^{20}$, y $\mathbb{R}^{30}$. El algoritmo GE logra resultados tan buenos como el método KGE en cuatro y seis problemas para $\mathbb{R}^{10}$,$\mathbb{R}^{20}$ y $\mathbb{R}^{30}$, correspondientemente. Esto hace que el algoritmo GE sea aquel cuyos resultados estén más cerca del método KGE en términos de resultados estadísticos. Sin embargo, en términos de velocidad de convergencia, el método KGE es mucho mejor. Finalmente, el método propuesto solo falla en 2 funciones por las razones antes mencionadas.

V. CONCLUSIONES

En este trabajo fue presentado un método nuevo y original, que se denomina evolución del gradiente basado en kernel (KGE - Kernel Gradient Evolution). El objetivo principal de este estudio fue proponer una nueva forma de aproximar el vector de gradiente mediante el uso de una función kernel, la idea central es aprovechar la información de tantos vectores vecinos en el enjambre como sea posible para obtener un gradiente más preciso.

El método reveló un gran rendimiento porque converge al mínimo en mucho menos de 500 iteraciones para la mayoría de las funciones y también alcanzó un mínimo más exacto en comparación con los otros algoritmos. Además, una desviación estándar notablemente baja demostró su gran estabilidad.

Además, de los resultados mencionados anteriormente, se puede concluir que:



(a) Sphere, Weighted sphere, Sum of different power

(b) Schwefel 2.2

(c) Salomon

(d) Schwefel 1.2

(e) Rosenbrock

(f) Easom

(g) Ackley

(h) Griewank

(i) Rastrigin

(j) Schwefel 2.3

(k) Withley

Figura 3: Gráficas de convergencia promedio para las funciones de prueba.

1. Una función kernel reproductora se puede utilizar eficazmente para aproximar el gradiente numérico de una función multidimensional.
2. El uso de más de dos partículas para construir una aproximación de gradiente proporciona información más confiable y amplia sobre la curvatura de la función en ese punto, lo que resulta en un gradiente más preciso. Entonces, de esta manera, el método KGE supera la desventaja de los métodos híbridos que aproximan el gradiente a través de solo dos partículas.
3. Refuerza la idea de que incluir información del gradiente en la regla de actualización dirige los algoritmos hacia el óptimo más rápido.
4. El método KGE propuesto tiene una gran combinación entre exploración y explotación.
5. El hecho de no derivar la función objetivo, hace que el método KGE sea un buen candidato para resolver problemas complejos que involucran funciones no dife-

Tabla III: Comparación de los resultados estadísticos, D=10; N=20; T=3000.

	KGE	GE	PSO	DE	ABC	CGA
f_1	2.0445e-206*	4.42e-29	1.08e-42	1.03e-128	2.39e-09	1.24e+00
	(0.00e+00)	(1.19e-28)	(5.60e-42)	(2.89e128)	(3.46e-09)	(0.23e+00)
f_2	2.4135e-206*	4.39e-31	1.20e-44	1.96e129	1.47e-08	4.40e+00
	(0.00e+00)	(1.73e-02)	(2.41e-02)	(1.25e-02)	(4.09e-02)	(1.23e+00)
f_3	5.6161e-212*	6.91e-64	1.18e-81	8.03e-273	2.41e-10	8.78e-02
	(0.00e+00)	(2.62e-63)	(6.10e-81)	(0.00e+00)	(5.24e-10)	(5.91e-02)
f_4	0.00E+00*	0.00e+00*	0.00e+00*	0.00e+00*	0.00e+00*	0.00e+00*
	(0.00E+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)
f_5	1.6856e-206*	7.26e-19	6.96e-12	8.19e-10	6.90e-08	5.14e+03
	(0.00e+00)	(7.04e-19)	(1.80e-11)	(1.31e-09)	(1.22e-07)	(2.70e+03)
f_6	1.6867e-101*	1.36e-16	9.60e-26	3.30e-72	1.11e-04	2.47e+00
	(1.1347e-101)	(3.53e-16)	(1.55e-25)	(6.86e-72)	(1.13e-04)	(4.46e-01)
f_7	8.33e+00	2.90e-01	3.05e+00	3.13e+00	5.82e-07*	3.98e+01
	(0.0639)	(4.24e-02)	(5.60e-01)	(1.43e+00)	(7.52e-07)	(1.18e+01)
f_8	-1.00e+00*	-1.00e+00*	-1.00e+00*	9.67e-01	5.80e-26	2.94e-05
	(0.00e+00)	(1.06e-16)	(3.39e-16)	(1.82e-01)	(1.33e-29)	(7.60e-05)
f_9	8.8818e-16*	3.49e-15*	4.68e-15*	4.44e-15*	4.35e-05	2.32e+00
	(0.00e+00)	(1.85e-15)	(9.01e-16)	(1.60e-30)	(4.39e-05)	(5.83e-01)
f_{10}	0.00e+00*	0.00e+00*	8.25e-02	1.48e-03	1.82e-10	9.94e+00
	(0.00e+00)	(0.00e+00)	(3.91e-02)	(3.97e-03)	(4.36e-10)	(6.44e+00)
f_{11}	0.00e+00*	0.00e+00*	2.96e+00	3.32e-02	4.04e-07	3.41e+01
	(0.00e+00)	(0.00e+00)	(1.21e+00)	(1.82e-01)	(5.62e-07)	(1.44e+01)
f_{12}	2.8015e-72*	9.99e-02	1.33e-01	9.99e-02	3.10e-06	2.04e-01
	(3.5716e-72)	(8.66e-09)	(4.80e-01)	(7.10e-09)	(2.92e-06)	(1.83e-02)
f_{13}	5.10e+00	2.54e+02	1.42e+03	7.90e+00*	4.15e+03	1.84e+03
	(250.5889)	(1.38e+02)	(1.95e+02)	(3.00e+01)	(1.67e+02)	(2.95e+02)
f_{14}	4.51 e+01	6.66e+00	1.64e+01	5.63e-01	1.38e-07*	1.29e+01
	(7.7540e-04)	(3.12e+00)	(9.42e+00)	(5.74e-01)	(5.29e-07)	(1.09e+01)

*Mejor resultado

Tabla IV: Comparación de los resultados estadísticos, D=20; N=30; T=6000.

	KGE	GE	PSO	DE	ABC	CGA
f_1	1.1328e-211*	1.13e_28	4.91e-32	2.80e-136	6.07e-10	2.37e+00
	(0.00e+00)	(1.30e-28)	(2.03e-31)	(7.19e-136)	(9.11e-10)	(3.25e-01)
f_2	2.6694e-209*	2.37e_30	5.27e-34	5.79e-138	1.18e-08	1.81e+01
	(0.00e+00)	(7.44e-02)	(2.95e-01)	(1.77e-02)	(2.59e-02)	(3.91e+00)
f_3	7.7674e-218*	7.48e-59	1.70e-63	0.00e+00a	5.15e-11	1.25e-01
	(0.00e+00)	(3.72e-58)	(6.39e63)	(0.00e+00)	(8.00e-11)	(8.08e-02)
f_4	0.00e+00*	0.00e+00*	0.00e+00*	0.00e+00*	0.00e+00*	0.00e+00*
	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)
f_5	1.0160e-209*	3.76e-17	8.24e-02	1.53e+00	1.52e-07	1.31e+04
	(0.00e+00)	(3.06e-17)	(9.45e-02)	(1.08e+00)	(3.77e-07)	(4.15e+03)
f_6	6.4048e-103*	2.12e-16	3.96e-22	6.65e-79a	1.15e-04	5.08e+00
	(3.1348e-103)	(1.77e-16)	(9.22e-22)	(1.96e-78)	(1.12e-04)	(6.05e-01)
f_7	18.3271	8.87e+00	1.24e+01	1.31e+01	1.69e-07*	9.50e+01
	(3.91 e-02)	(1.72e-01)	(1.94e+00)	(2.50e+00)	(1.94-07)	(1.05e+01)
f_8	-1.00e+00*	-1.00e+00*	-1.00e+00*	-1.00e+00*	-3.37e-51	-4.96e-16
	(0.00e+00)	(1.05e-16)	(6.26e-16)	(4.52e-16)	(5.66e-55)	(2.67e-15)
f_9	8.8818e-16*	5.63e-15*	9.30e-15*	4.91e-15*	2.16e-05	2.44e+00
	(0.00e+00)	(3.00e-16)	(2.55e-15)	(1.23e-15)	(1.84e-05)	(3.26e-01)
f_{10}	0.00e+00*	0.00e+00*	3.05e-02	0.00e+00*	9.10e-11	3.91e+00
	(0.00e+00)	(0.00e+00)	(2.39e-02)	(0.00e+00)	(2.16e-10)	(2.04e+00)
f_{11}	0.00e+00*	0.00e+00*	1.03e+01	3.65e-01	2.21e-07	8.58e+01
	(0.00e+00)	(0.00e+00)	(3.10e+00)	(6.12e-01)	(5.23e-07)	(1.80e+01)
f_{12}	3.3611e-72*	9.99e_02	2.50e-01	1.75e-01	4.29e-06	2.00e-01
	(3.2900e-72)	(4.81e-10)	(5.09e-02)	(4.33e-02)	(4.03e-06)	(3.74e-11)
f_{13}	4.02e+02	4.73e+02	4.39e+03	3.95e+00*	8.31e+03	4.14e+03
	(2.89e+02)	(1.78e+02)	(3.20e+02)	(2.16e+01)	(3.06e+02)	(5.12e+02)
f_{14}	1.80e+02	5.31e+01	6.22e+01	6.05e+00	2.20e-09*	7.24e+01
	(3.3278e-04)	(1.77e+01)	(2.84e+01)	(2.24e+00)	(7.99e_09)	(5.77e+01)

*Mejor resultado

renciables.

REFERENCIAS

[1] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 01 1997.

[2] J. Holland, "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 9, pp. 297–314, 07 1962.

[3] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science (New York, N.Y.)*, vol. 220, pp. 671–80, 06 1983.

[4] B. Obaiahnahatti and J. Kennedy, "A new optimizer using particle swarm theory," 11 1995, pp. 39 – 43.

[5] S. Gholizadeh, M. Danesh, and C. Gheyratmand, "A new newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames," *Computers & Structures*, vol. 234, p. 106250, 07 2020.

[6] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 01 2016.

[7] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, 06 2018.

[8] Y. Feng, S. Deb, G.-G. Wang, and A. Alavi, "Monarch butterfly optimization: a comprehensive review," *Expert Systems with Applications*, vol. 168, 11 2020.

[9] S. Li, H. Chen, M. Wang, and A. A. Heidari, "Slime mould algorithm: A new method for stochastic optimization," *Future Generation Computer*

Tabla V: Comparación de los resultados estadísticos, D=30; N=40; T=10000.

	KGE	GE	PSO	DE	ABC	CGA
f_1	5.2255e-213*	2.56e-30	1.68e-29	6.32e-157a	4.41e-10	3.63e+00
	(0.00e+00)	(2.85e-30)	(5.19e-29)	(0.00e+00)	(5.42e-10)	(4.02e-01)
f_2	7.6327e-210*	9.07e-32	4.07e-31	1.65e-158a	1.28e-08	4.33e+01
	(0.00e+00)	(6.45e-01)	(2.30e-01)	(3.38e-02)	(3.09e-01)	(8.06e+00)
f_3	3.6569e-220*	8.66e-67	1.04e-50	0.00e+00a	1.77e-11	1.27e-01
	(0.00e+00)	(3.34e-66)	(5.52e-50)	(0.00e+00)	(2.99e-11)	(7.11e-02)
f_4	0.00e+00*	0.00e+00*	3.33e-02	0.00e+00*	0.00e+00*	0.00e+00+
	(0.00e+00)	(0.00e+00)	(1.83e-01)	(0.00e+00)	(0.00e+00)	(0.00e+00)
f_5	3.47e-211*	6.55e-16	2.87e+01	8.76e+02	2.23e-07	2.05e+04
	(0.00e+00)	(5.95e-16)	(2.97e+01)	(3.53e+02)	(3.54e-07)	(7.91e+03)
f_6	8.80e-103*	2.58e-17	6.84e-21	5.96e-92	7.69e-05	7.72e+00
	(1.1547e-102)	(1.72e-17)	(1.64e-20)	(5.70e-92)	(9.09e-05)	(7.31e-01)
f_7	2.79e+01	1.81e+01	2.44e+01	2.22e+01	9.05e-08*	1.74e+02
	(5.67e-02)	(1.66e-01)	(1.04e+01)	(1.31e+00)	(1.49e-07)	(1.08e+01)
f_8	-1.00e+00*	-1.00e+00*	-1.00e+00*	-1.00e+00*	-1.96e-76	-2.08e-26
	(2.21e-16)	(1.13e-16)	(6.10e-16)	(3.39e-16)	(4.67e-80)	(1.14e-25)
f_9	8.88e-16*	4.44e-15*	2.03e-14	7.64e-15*	1.29e-05	2.46e+00
	(0.00e+00)	(1.60e-30)	(6.71e-15)	(1.08e-15)	(1.25e-05)	(2.98e-01)
f_{10}	0.00e+00*	0.00e+00*	1.45e-02	0.00e+00a	2.45e-11	1.81e+00
	(0.00e+00)	(0.00e+00)	(1.72e-02)	(0.00e+00)	(4.74e-11)	(1.03e+00)
f_{11}	0.00e+00*	0.00e+00*	2.12e+01	1.99e-01	1.71e-07	3.17e+01
	(0.00e+00)	(0.00e+00)	(4.00e+00)	(4.05e-01)	(3.85e-07)	(1.18e+01)
f_{12}	1.7631e-71*	9.99e-02	3.77e-01	2.00e-01	5.11e-06a	3.00e-01
	(7.26e-72)	(4.81e-10)	(6.79e-02)	(1.92e-08)	(4.45e-06)	(2.80e-12)
f_{13}	4.11e+02	5.49e+02	7.14e+03	3.82e-04*	1.24e+04	6.65e+03
	(7.30e+02)	(2.03e+02)	(7.38e+02)	(2.76e-19)	(5.48e+02)	(7.22e+02)
f_{14}	4.06e+02	1.40e+02	1.82e+02	1.41e+02	2.95e-09*	2.03e+02
	(2.0576e-04)	(1.97e+01)	(8.86e+01)	(7.92e+01)	(6.39e_09)	(1.70e+02)

*Mejor resultado

Systems, pp. 300–323, 04 2020.

[10] Y. Yang, H. Chen, A. Asghar-Heidari, and A. Gandomi, "Hunger games search, visions, conception, implementation, deep analysis, perspectives and towards performance shifts (awarded a reproducible badge from code ocean platform and expert systems with applications)," *Expert Systems with Applications*, p. 114864, 03 2021.

[11] I. Ahmadianfar, A. A. Heidari, A. Gandomi, X. Chu, and H. Chen, "Run beyond the metaphor: An efficient optimization algorithm based on runge kutta method," *Expert Systems with Applications*, vol. https://aliasgharheidari.com/RUN.html, p. 115079, 04 2021.

[12] J. Tu, H. Chen, M. Wang, and A. Gandomi, "The colony predation algorithm," *Journal of Bionic Engineering*, vol. 18, pp. 674–710, 05 2021.

[13] A. Asghar-Heidari, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97 https://aliasgharheidari.com/HHO.html, pp. 849–872, 03 2019.

[14] R. Phillip, *Gradient Descent Algorithms*, 04 2018, pp. 258–373.

[15] T. Ypma, "Historical development of the newton–raphson method," *Siam Review - SIAM REV*, vol. 37, 12 1995.

[16] M. Hestenes and E. Steifel, "Method of conjugate gradients for solving linear systems," *J. Research Nat. Bur. Standards*, vol. 49, pp. 409–436, 01 1952.

[17] M. Shashi and R. Bhagwat, *Quasi-Newton Methods*, 12 2019, pp. 245–289.

[18] J. More, "The levenberg- marquardt algorithm: implementation and theory," *Numerical Analysis*, pp. 105–106, 01 1977.

[19] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 02 2012.

[20] R. Kuo and F. Zulvia, "The gradient evolution algorithm: A new metaheuristic," *Information Sciences*, vol. 316, 04 2015.

[21] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, "Gradient-based optimizer: A new metaheuristic optimization algorithm," *Information Sciences*, vol. 540, 06 2020.

[22] S. Gholizadeh, M. Danesh, and C. Gheyratmand, "A new newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames," *Computers & Structures*, vol. 234, p. 106250, 07 2020.

[23] L. Khaleel and B. Mitras, "Hybrid whale optimization algorithm with modified conjugate gradient method to solve global optimization problems," *OALib*, vol. 07, pp. 1–18, 01 2020.

[24] H. Ahmed, E. Tarik, and H. Chilmeran, "A modified bat algorithm with conjugate gradient method for global optimization," *International Journal of Mathematics and Mathematical Sciences*, vol. 2020, pp. 1–14, 06 2020.

[25] M. Probert, "Engineering optimisation: An introduction with metaheuristic applications, by xin-she yang," *Contemporary Physics - CONTEMP PHYS*, vol. 53, pp. 271–272, 05 2012.