IMPLEMENTACION Y EVALUACIÓN DE ALGORITMOS DE ENJAMBRES DE PARTÍCULAS

Jonathan Durand,Gerar Quispe
Maestría en ciencias de la computación, Universidad Católica San Pablo
Arequipa, Perú
E-mail: jonathan.drnd@gmail.com, gerar.quispe@ucsp.edu.pe

El presente trabajo consiste en emprender el estudio, la implementación y evaluar la calidad del modelo de Particles Swarm Gravitational Search Algorithm (GSA) [Rashedi et al., 2009], probamos su funcionamiento en tres funciones benchmark multidimensionales, la función de Ackley, la función de Schwefel y una variación de la función de Schaffer.

Se presenta detalles de la implementación y algunas pruebas de desenpeño, las funciónes benchmark son probadas con dimensión d=2 y d=10, con 100 iteraciones, también mostramos los resultados en BoxPlots de los 100 experimentos para todas las iteraciones.

Introducción

El campo de la inteligencia artificial está basada en actividades de busqueda que buscan simular el pensamiento y el comportamiento humano, gracias a los avances de los enfoques matematicos. En este campo existen diferentes áreas de interés donde los investigadores desarrollan diferentes estudios, para obtener soluciones efectivas para poblemas del mundo real. Es en este punto, donde los algoritmos de enjambres de partículas (Swarm Intelligence Algorithms) es una de las más sobresalientes áreas de estudio. La cual atrae el interés de los investigadores. El alcance de la inteligencia de enjambres de partículas esta asociada con los enfoques para la solución de problemas, las cuales utilizan una colección de comportamientos naturales o artificiales de sistemas autorganizados, en este contexto diferentes tipos de algoritmos han estado siendo diseñados y desarrollados por los científicos, para ofrecer diferentes alternativas de solución para la solución de problemas.

Con respecto a estos algoritmos de investigación de enjambres de partículas, estos han sido una tendencia para diseñar y desarrollar nuevos algoritmos enfocándose en asegurar la mejor solución para cierto problema, de modo que para alcanzar estos resultados, investigadores gastan su tiempo en examinar dinámicas naturales de diferentes enjambre de partículas y tratar de diseñar algunos algoritmos matemáticos para proveer modelos similares de dinámicas naturales mientras obtenemos diferentes marcos de soluciones. Es importante que la naturalesa tenga un importante role inspirador, en la creación de nuevas estructuras de algoritmos para conbinar ambos eventos matematicos y físicos, los cuales nosotros presenciamos en la vida real.

Cuando examinamos la presente literatura, podemos observar que existe un amplio uso de algoritmos de enjambre de partículas como el algoritmo de búsqueda gravitacional, que sera desarrollado en este informe, este algoritmo entre otros tienen importante popularidad para la resolución de problemas.

I. Marco teórico

Inteligencia de enjambres o SwarmParticles Intelligence

La inteligencia de enjambres es una area de la inteligencia artificial que estudia el comportamiento colectivo de los sistemas descentralizados, autoorganizados, naturales o artificiales. El concepto se utiliza en los trabajos de inteligencia artificial. Estos algoritmos estan Inspirados por la naturaleza, especialmente por ciertos sistemas biológicos, los sistemas de inteligencia de enjambre están típicamente formados por una población de agentes simples que interactúan localmente entre ellos y con su medio ambiente. Los agentes siguen reglas simples y, aunque no existe una estructura de control centralizado que dictamine el comportamiento de cada uno de ellos, las interacciones locales entre los agentes conduce a la emergencia de un comportamiento global complejo.

Gravitational Search Algorithm El algoritmo de búsqueda gravitacional se basa en la ley de la gravedad y la noción de las interacciones de comunicación. El algoritmo GSA utiliza la teoría de la física newtoniana y sus agentes buscadores son la colección de masas. En el GSA existe un sistema aislado de masas. Usando la fuerza gravitatoria, cada masa en el sistema puede ver la situación de las otras masas; por lo tanto, la fuerza gravitatoria es una manera de transferir información entre diferentes masas. En GSA, los agentes se consideran como objetos y su desempeño se mide por sus masas. Todos estos objetos se atraen entre sí por una fuerza gravitatoria, y esta fuerza causa un movimiento de todos los objetos globalmente hacia los objetos con masas más pesadas correspondiendo estas últimas con las buenas soluciones del problema. La posición del agente corresponde a una solución del problema, y su masa se determina utilizando una función de aptitud. En un lapso de tiempo, las masas se sienten atraídas por la masa más pesada lo que presentaría idealmente una solución óptima en el espacio de búsqueda. El GSA podría ser considerado como un sistema aislado de masas ya que se trataría de un

pequeño mundo artificial de masas que obedecen las leyes de Newton de gravitación y movimiento.

Funciones Benchmark Son funciones que pueden ser usadas para probar el performance de cualquier enfoque de optimización y problemas relacionados, como tasa de convergencia , presición, robustes y performance en general, para nuestro caso de estudio utilizaremos 3 de estas funciones que son: la Función Ackley, la función Schwefel y la función de Schaffer modificado.

Función Ackley

La función Ackley se utiliza ampliamente para probar algoritmos de optimización. En su forma bidimensional, como se muestra en la gráfica 1, se caracteriza por una región externa casi plana y un gran agujero en el centro. La función plantea un riesgo para los algoritmos de optimización, en particular los algoritmos de subida, que puede que se queden atrapados en uno de sus muchos mínimos locales.

Función matemática:

$$f(x_0 \cdots x_n) = -20exp(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e) \quad (1)$$

$-32{,}768 \leq x_i \leq 32{,}768$

Mínimo Global $\text{Min} f(0, \cdots, 0) = 0$

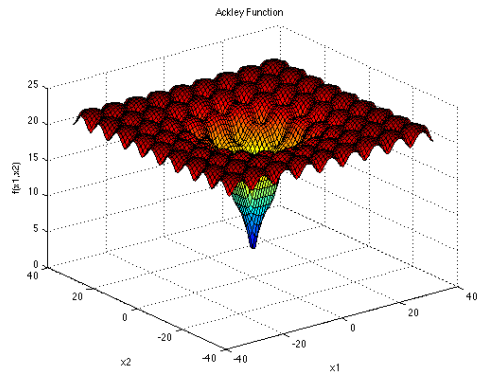

Figura 1– funcion de Ackley para una dimension igual a dos.

Función Schwefel

La función de Schwefel es compleja, con muchos mínimos locales. El gráfico 2 muestra la forma bidimensional de la función.

Función matemática:

$$f(x_1 \cdots x_n) = \alpha \cdot n - \sum_{i=1}^{n}(x_i sin(\sqrt{|x_i|})) \quad (2)$$

$\alpha = 418{,}982887$; $-500 \leq x_i \leq 500$

Mínimo Global

$\text{Min} f(420{,}968746, 420{,}968746, \cdots, 420{,}968746) = 0$

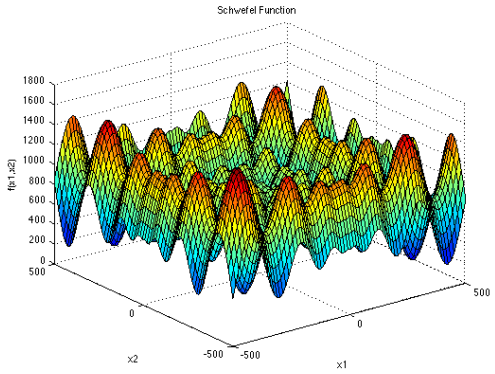

Figura 2– función de schwef para una dimension igual a dos.

Función 3 (Schaffer modificado)

La segunda función de Schaffer. Para este caso se muestra en un dominio de entrada más pequeño en el gráfico 3.

Función matemática:

$$f(x_1 \cdots x_n) = 0{,}5 - \frac{sin^2(\sqrt{\sum_{i=1}^{n} x_i^2}) - 0{,}5}{[1 + 0{,}001 \cdot (\sum_{i=1}^{n} x_i^2)]^2} \quad (3)$$

$-100 \leq x_i \leq 100$

Máximo Global $\text{Max} f(0, \cdots, 0) = 1$

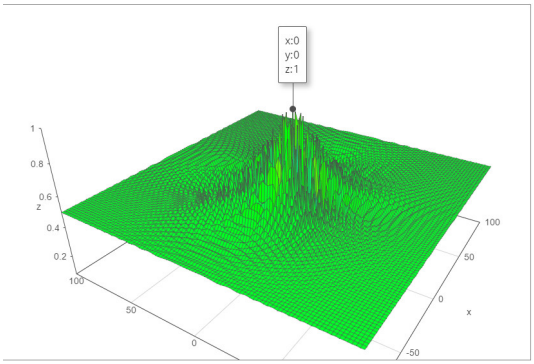

Figura 3– función de Schaffer f(x,y) = f(0,0) = 1. (Sólucion a buscar en d=2).

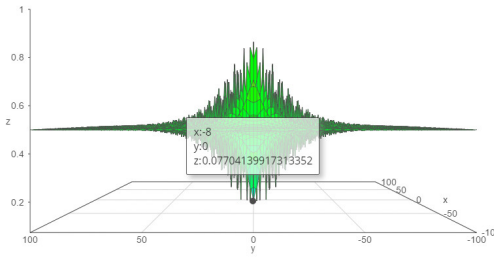

Figura 4– función de Schaffer con modificaciones accorde a la asignación.

II. Modelamiento y descripción del Algoritmo

El algoritmo de busqueda gravitacional fue propuesta por Rashedi, como una simulación del

comportamiento de la fuerza gravitacional de Newton. El GSA inicia con un conjunto de agentes, seleccionados random o en base a algun criterio ubicados en cierta posicion, lo cual representa la solucion al problema. En un sistema de N agentes, la posicion del agente i^{th} es definido como.

$$X_i = (x_i^1, \cdots, x_i^d, \cdots, x_i^n) \;\; for\; i = 1, 2, \cdots, N \tag{4}$$

Donde X_i^d presenta la posicion del agente i^{th} en la dimension d y n es la dimnension del espacio de busqueda.

En el tiempo t una fuerza actua hacia la masa i desde la masa j. Esta fuerza es definida como:

$$F_{ij}^d = G(t)\frac{M_{pi}(t) x M_{aj}(t)}{R_{ij} + \epsilon}(x_j^d(t) - x_i^d(t)) \tag{5}$$

Donde M_{aj} es la masa activa del agente j, M_{pi} es la masa gravitacional pasiva del agente i, G(t) es la contante gravitacional en el tiempo t, $\epsilon esunapequeaconstantey$ $R_{ij}(t)$ es la distancia euclideana entre los agentes i y j.

$$R_{ij}(t) = ||X_i(t).X_j(t)|| \tag{6}$$

La fuerza total que actua en la masa i, en la dimension d en el tiempo t esta dado:

$$F_i^d(t) = \sum_{j \epsilon K best, j!=i}^{M} rand_i F_{ij}^d(t) \tag{7}$$

Donde $rand_j$ es un numero random en el intervalo [0,1], K es el mejor conjunto de los primeros K agentes que esten mas cerca a la solucion. La aceleración relacionada con la masa i en el tiempo t en la dimensión d_th se da como sigue

$$a_i^d = \frac{F_i^d(t)}{M_{ii}(t)} \tag{8}$$

Donde M_{ii} es la masa inercial del agente i_{th} La siguiente velocidad del agente puede ser calculada como una fraccion de la actual velocidad mas su aceleracion. Tanto la posicion como la velocidad puede ser calculados como:

$$v_i^d(t+1) = rand_i.v_i^d(t) + a_i^d(t) \tag{9}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{10}$$

Donde $rand_i$ es un numero random en el intervalo [0,1]. La constante gravitacional G es inicializado en la busqueda y sera reducido y controlado respecto a cada iteracion como se muestra en la siguiente formula.

$$G(t) = G_0.e^{-\alpha t/T} \tag{11}$$

Donde T es el numero de iteraciones, G0 y α son constantes. La masa gravitacional y la masa inercial son actualizados por las siguientes ecuaciones:

$$\mathrm{M}_{ai} = M_{pi} = M_{ii} = M_i, \quad i = 1, 2, \cdots, N$$

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{12}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{13}$$

Donde $fit_i(t)$ representa la evaluación a la funcion a optimizar del agente i en el tiempo t y worst(t) y best(t) estan dado para un problema de minimizacion.

$$best(t) = min fit_j(t), j\epsilon[1 \ldots N] \tag{14}$$

$$worst(t) = max fit_j(t), j\epsilon[1 \ldots N] \tag{15}$$

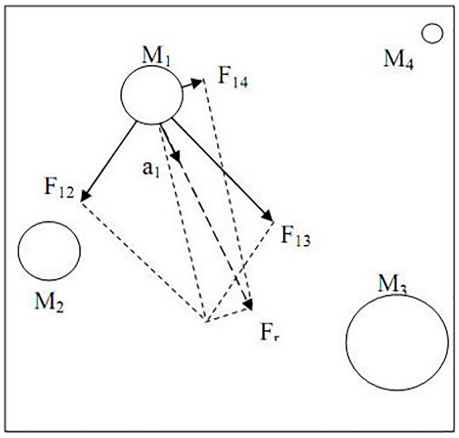

Figura 5– Cada masa acelera hacia la fuerza resultante que actúa de las otras masas.

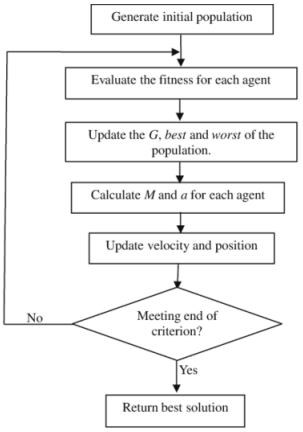

Figura 6– Visión general de GSA.

1.- Generar una población de N agentes de manera aleatoria.

2.- Actualizar G(t), evaluar el mejor y el peor agente de la población.

3.- Para cada agente i hacer:

3.1.- evaluar la cualidad de la i-ésima partícula

3.2.- evaluar $Masa_i$
3.3.- evaluar la fuerza de $Masa_i$
3.4.- evaluar la aceleración de $Masa_i$
3.5.- actualizar la velocidad de $Masa_i$
3.6.- buscar la nueva posición del $Agente_i$
3.6.1.- evaluar solución, si es mejor que la ultima actualizar.
4.- Si no se encuentra el criterio buscado, ir al punto 2, caso contrario terminar.

III. Ajuste de parámetros y parametrización

Los parametros utilizados varian de acuerdo a las funciones, los valores utilizados para dimensión igual a 2 se pueden observar en la figura 7, mientras que para dimensión igual a 10 en la figura 8.

FUNCION 1			
Nro.particulas	**Dimensión**	**Dominio**	**Constante gravitatoria**
N = 1000	d=2	Lower[i]=-32.768; Upper[i]=32.768;	double Gconst(int iterr) { return 800000000.0 * exp(-1.5 * iterr /maxiter); }
FUNCION 2			
Nro.particulas	**Dimensión**	**Dominio**	**Constante gravitatoria**
N = 10000	d=2	Lower[i]=-500; Upper[i]=500;	double Gconst(int iterr) { return 800000000.0 * exp(-1.5 * iterr /maxiter); }
FUNCION 3			
Nro.particulas	**Dimensión**	**Dominio**	**Constante gravitatoria**
N = 10000	d=2	Lower[i]=-100; Upper[i]=100;	double Gconst(int iterr) { return 800000000.0 * exp(-1.5 * iterr /maxiter); }

Figura 7– Parametros utilizados para d igual a 2.

FUNCION 1			
Nro.particulas	**Dimensión**	**Dominio**	**Constante gravitatoria**
N = 30000	d=10	Lower[i]=-32.768; Upper[i]=32.768;	double Gconst(int iterr) { return 90000.0 * Math.exp(-3* iterr /maxiter); }
FUNCION 2			
Nro.particulas	**Dimensión**	**Dominio**	**Constante gravitatoria**
N = 30000	d=10	Lower[i]=-500; Upper[i]=500;	double Gconst(int iterr) { return 7000000.0 * Math.exp(-4* iterr /maxiter); }
FUNCION 3			
Nro.particulas	**Dimensión**	**Dominio**	**Constante gravitatoria**
N = 3000	d=10	Lower[i]=-100; Upper[i]=100;	double Gconst(int iterr) { return 60000.0 * Math.exp(-4* iterr /maxiter); }

Figura 8– Parámetros utilizados para d igual a 10.

N corresponde a la cantidad de agentes en el sistema Lower y Upper corresponde al dominio donde se encuentra la solucion. la funcion Gconst corresponde a la constante gravitatoria que conecta cada una de las partículas.

IV. Resultados obtenidos

Primera función (Ackley) para dimensión igual a 2

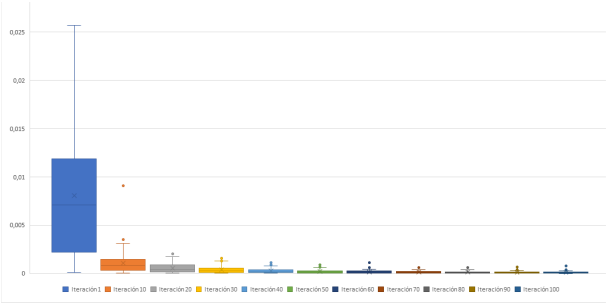

Figura 9– 10 boxplots cada 10 iteraciones para la función de Ackley.

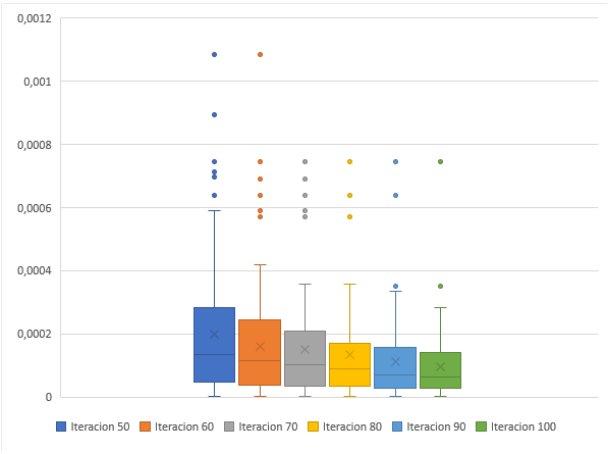

Figura 10– Desde la iteración 50 tomados de diez en diez para Ackley.

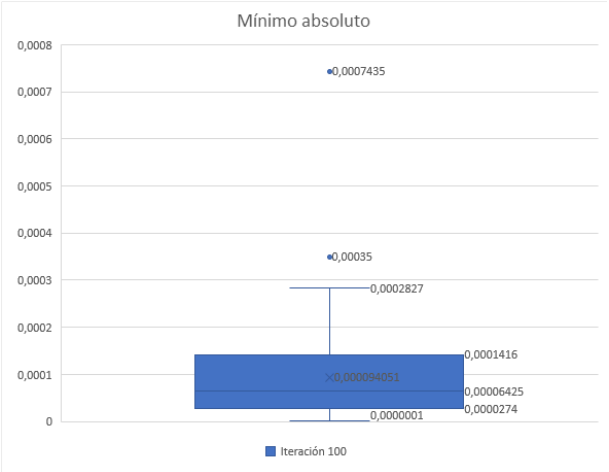

Figura 11– Resultados depués de 100 iteraciones función Arckley.

Segunda función para dimensión igual a 2

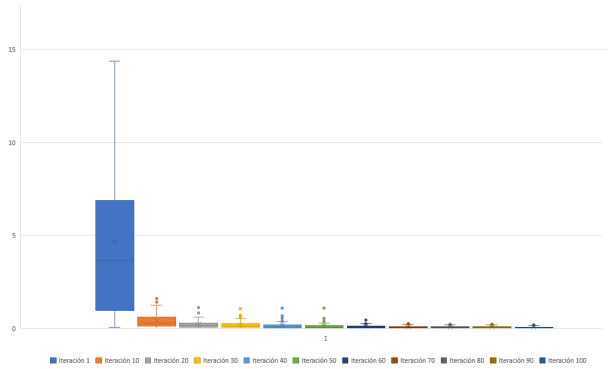

Figura 12– 10 boxplots cada 10 iteraciones para la funcion de Schwefel

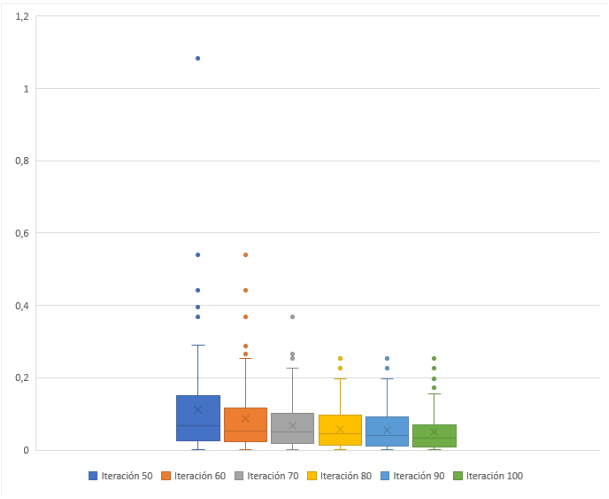

Figura 13– Desde la iteración 50 tomades de diez en diez en Schwefel

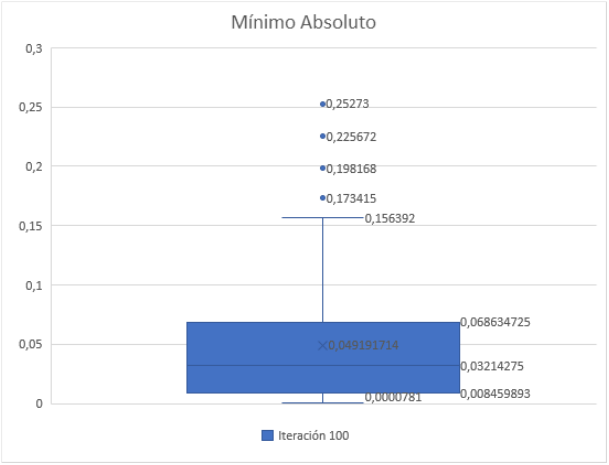

Figura 14– Resultados depués de 100 iteraciones función Schwefel

Tercera función para dimensión igual a 2

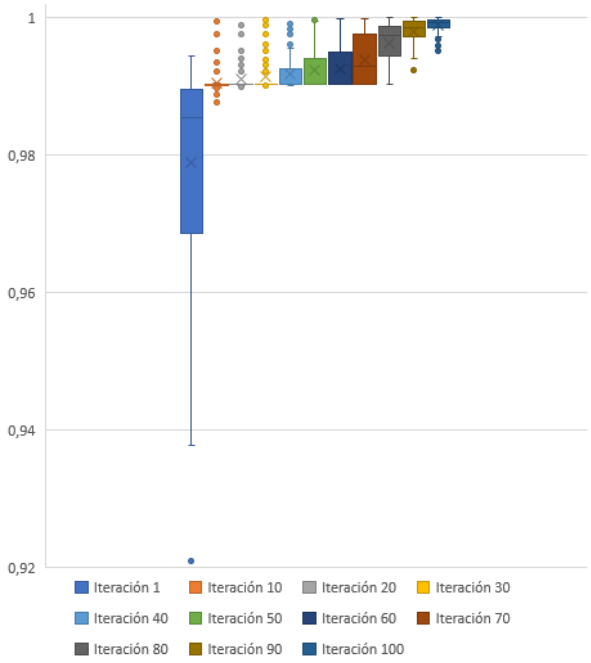

Figura 15– 10 boxplots cada 10 iteraciones para la función de Schaffer modificado.

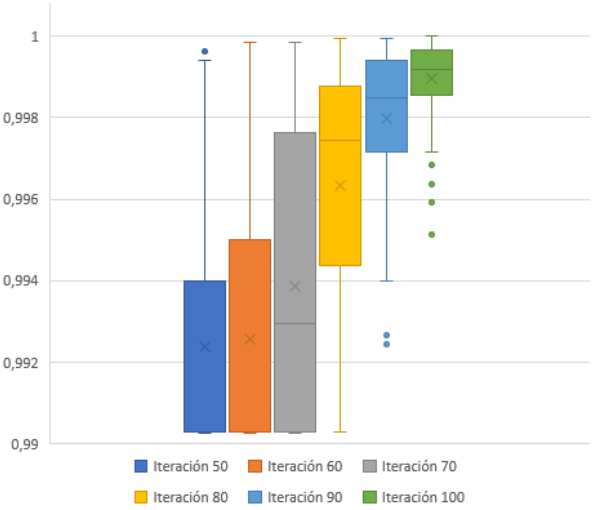

Figura 16– Desde la iteración 50 tomados de diez en diez para Schaffer.

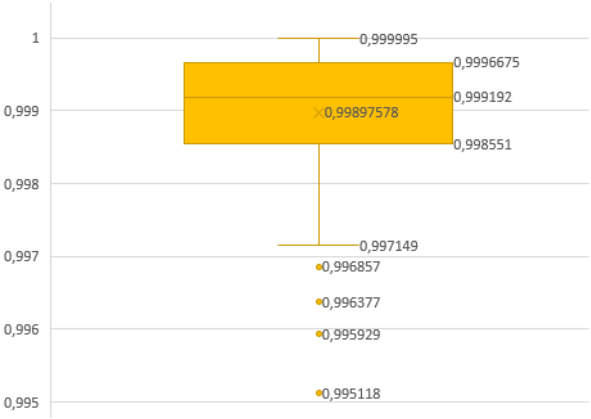

Figura 17– Resultados depués de 100 iteraciones función Schaffer.

Primera función para dimensión igual a 10

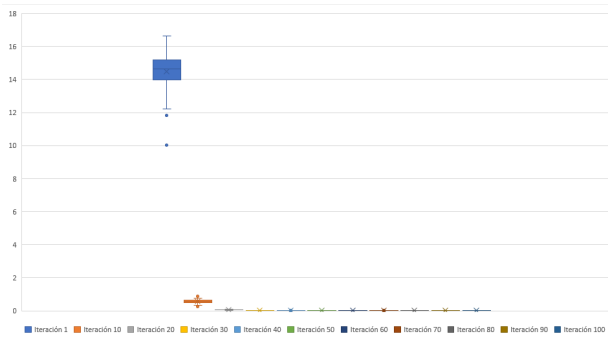

Figura 18– 10 boxplots cada 10 iteraciones para la función de Ackley.

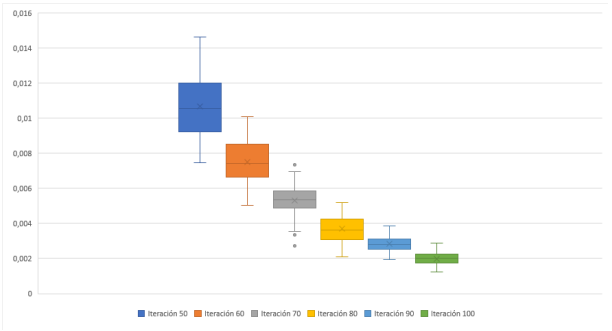

Figura 19– Desde la iteración 50 tomados de diez en diez para Ackley.

Figura 20– Resultados depués de 100 iteraciones función Arckley.

Segunda función para dimensión igual a 10

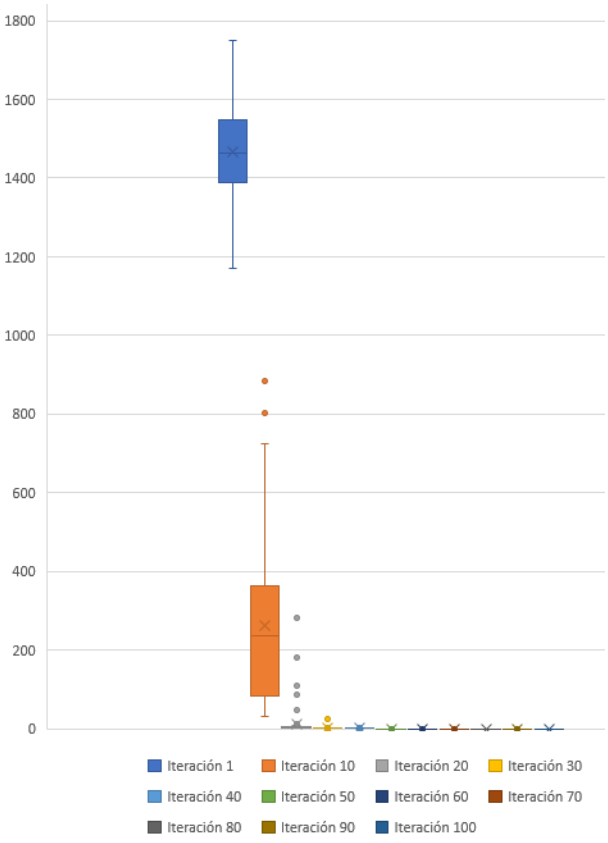

Figura 21– 10 boxplots cada 10 iteraciones para la funcion de Schwefel

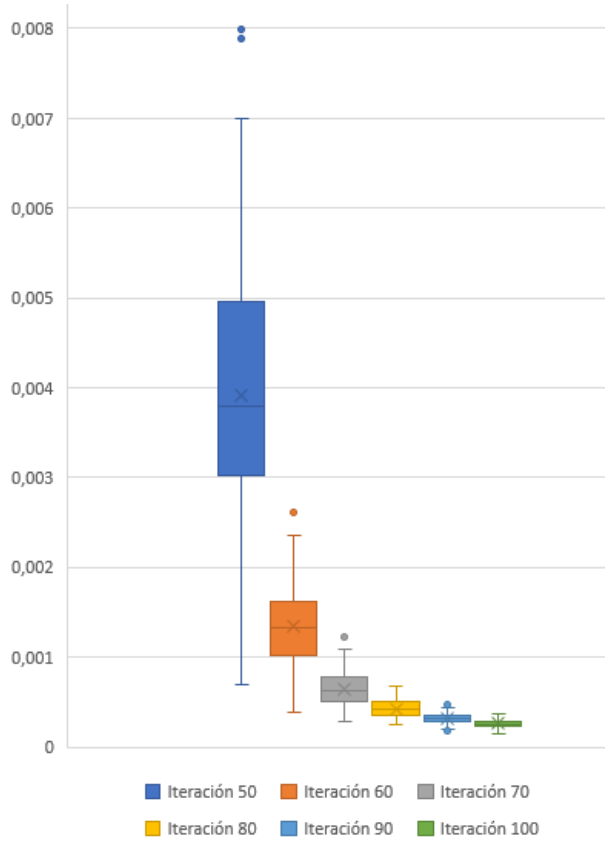

Figura 22– Desde la iteración 50 tomades de diez en diez en Schwefel

Figura 23– Resultados depués de 100 iteraciones función Schwefel

Tercera función para dimensión igual a 10

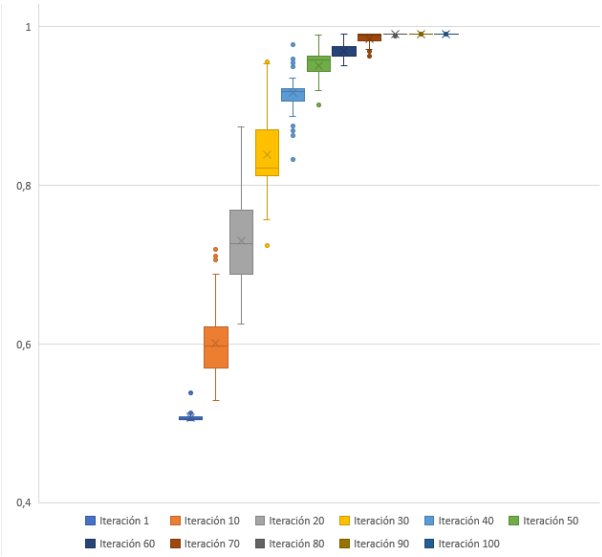

Figura 24– 10 boxplots cada 10 iteraciones para la función de Schaffer modificado.

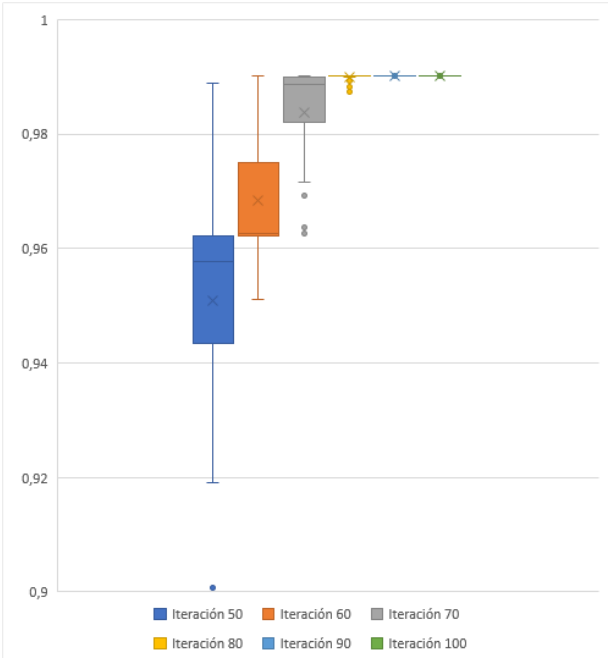

Figura 25– Desde la iteración 50 tomados de diez en diez para Schaffer Schaffer.

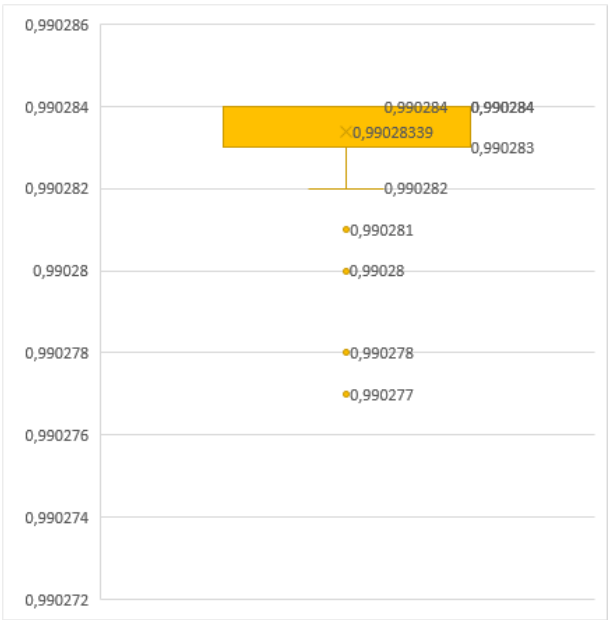

Figura 26– Resultados depués de 100 iteraciones función Schaffer Schaffer.

V. Tiempos de ejecución y complejidad algorítmica

Para la toma de tiempos de ejecución, cabe recalcar que los parametros utilizados son los mismo que mostramos en la figura 7 y figura 8, tanto para dimension 2 como para dimensión 10 respectivamente.

Mejores Aproximaciones obtenidas con los algoritmos implementados, como podemos observar en la figura 27 tenemos buena aproximación para encontrar el mínimo y el máximo absoluto de la función estudiadas.

Mejores aproximaciones		
FUNCION	Dimensión	Extremo Absoluto
f1-Ackley	2	0,000000100
f2-Schwefel	2	0,000078100
f3-Schaffer-modificado	2	0,995118000
f1-Ackley	10	0,001217340
f2-Schwefel	10	0,000150922
f3-Schaffer-modificado	10	0,990284000

Figura 27– Mejores aproximaciones obtenidas

Tiempos de ejecución				
Función	Dimensión	Nro.Particulas	Tiempo (Seg)	Tiempo en 100 iteraciones(min)
Ackley	2	1000	0,1300	0,2167
Schwefel	2	10000	1,2615	2,1025
Schaffer-Modificado	2	10000	1,2185	2,0308
Ackley	10	30000	14,5265	24,2108
Schwefel	10	30000	14,4550	24,0917
Schaffer-Modificado	10	3000	1,4325	2,3875

Figura 28– Tiempos de ejecución obtenidos, para cada función benchmark

Variables	f1 d=10	f2 d=10	f3 d=10	f1 d=2	f2 d=2	f3 d=2
x1	-0.00000613	420,9721088	0,40041222	0,004724	420,935500	-0.00188411
x2	0.00007920	420,9659990	-0,84640391	-0,015193	420,934530	0.00107847
x3	-0.00011409	420,9718832	-2,16494194			
x4	0.00065213	420,9651939	0,68076353			
x5	-0.00030516	420,9614327	0,15106904			
x6	-0.00033283	420,9745292	-0,84977559			
x7	-0.00051506	420,9760651	0,61898399			
x8	-0.00006525	420,9709560	0,39763245			
x9	-0.00001921	420,9686129	-0,36580305			
x10	0.00001895	420,9695757	-1,55013782			

Figura 29– Valores de x para las funciones analizadas

VI. Conclusiones

- El Particles Swarm Gravitational Search Algorithm a comparación de otros algoritmos que utilizan inteligencia de partículas resulta ser lento, pero a cambio de ello el algoritmo llega a un buen proceso de búsqueda de la solución, esto puede ver en los resultados mostrados en este informe.
- El algoritmo GSA, provee buenos resultados incluso en funciones benchmark complejas que poseen multiples minimos locales.

VII. Código fuente

El código fuente se encuentra en el siguiente link:

https://github.com/jonathandrnd /SistemasInteligentes/tree/master/GSA

El código se encuentra programado en C++, utilizando OPENMP para paralelizar y optimizar el tiempo de ejecucion. Tambien se ha realizado una version del código en Java.

VIII. Referencias

1 Rashedi, E., Nezamabadi-pour, H., and Saryazdi, S. (2009). "Gsa: A gravitational search algorithm". Information Sciences, 179(13):2232–2248. cited By 1261

2 Taisir Eldos, Rose Al Qasim, .On The Performance of the Gravitational Search Algorithm". (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No. 8, 2013

3 Present State of Swarm Intelligence and Future Directions, Encyclopedia of Information Science and Technology, Third Edition by Mehdi Khosrow-Pour at Usak University, Turkey.