A Gravitational Search Algorithm

Alejandra Cristina Callo Aguilar * **Raul Romaní Flores** **

* *Universidad Católica San Pablo (e-mail: alejandra.callo@ucsp.edu.pe)*
** *Universidad Católica San Pablo (e-mail: raul.romani@ucsp.edu.pe)*

Abstract: Este paper se describe el principio de funcionamiento y ecuaciones del algoritmo Gravitational Search Algorithm (GSA) descrito en el paper Rashedi et al. [2009]. Tambien se describe como se modelo para optimizar 3 funciones benchmark como se hizo para optimizar parametros para cada funcion benchmark. Tambien se hace una comparación con otras heuristicas de optimización con la prueba de suma de rangos de Wilcoxon. Ademas se hizo unas mejoras al algoritmo GSA estándar. El algoritmo GSA fue implementado en C++ 11 con Cuda y OpenMP.

Keywords: GSA, masa, fuerza, gravedad, movimiento

1. INTRODUCCIÓN

GSA es un algoritmo de optimización, inspirado en comportamientos de enjambre en la naturaleza, esta basado en la ley de gravedad de Newton y la ley del movimiento. GSA forma parte de los algoritmos basados en población. El algoritmo intenta mejorar el balance entre la exploración y explotación con leyes de gravedad y movimiento. Sin embargo, GSA ha sido criticado por no basarse genuinamente en la ley de gravedad Gauci et al. [2012]. Se informa que GSA excluye la distancia entre las masas en su fórmula, mientras que la masa y la distancia son ambas partes integrantes de la ley de gravedad. A pesar de la crítica, el algoritmo todavía está siendo explorado y aceptado por el científico comunidad.

El objetivo de este paper es implementar, optimizar y analizar el algoritmo de búsqueda gravitacional (GSA) propuesto en el paper Rashedi et al. [2009].

2. GSA

La fuerza gravitacional entre dos agentes es directamente proporcional al producto de sus masas y inversamente proporcional a el cuadrado de la distancia entre ellos. En GSA las particulas son consideradas agentes y su rendimiento es evaluado con sus masas. In GSA cada agente tiene asociado cuatro atributos: posición, masa inercial, masa gravitacional activo y masa gravitacional pasivo. La posición de los agentes provee una solución del problema, mientras la función de fitness es usado para calcular la masa inercial y gravitacional. El algoritmo al principio usa exploración para evitar caer en un mínimo local y después usa explotación. El rendimiento de GSA is mejorado haciendo un balance entre exploración y explotación.El valor kbest es usado para lograr este balance, este es decrementado linealmente con el paso del tiempo o iteraciones.

1 Identificación de espacio de búsqueda.
2 Generar población inicial.
4 Evaluar el fitness para cada agente in la población.
5 Actualizar el valor de la constante gravitacional.

$$G(t) = G(G_0, t), \quad Best(t) = \min_{i\in\{1,\dots,N\}} Fit_i(t)$$

$$Worst(t) = \max_{i\in 1\dots N} Fit_i(t)$$

6 Calcular la fuerza en diferentes direcciones(M) y aceleración (a) con las siguientes ecuaciones:

$$M_i(t) = \frac{m_i(t)}{\sum_{j-1}^{N} m_i(t)} \text{ Where } m_i(t) = \frac{Fit_i(t) - Worst(t)}{Best(t) - Worst(t)}$$

$$\text{Acceleration } a_i^d(t) = \frac{F_i^d(t)}{M_{ii(t)}}, \text{ where} F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t)$$

7 Actualizar la velocidad y posición de la partícula con las siguientes ecuaciones:

$$V_i^d(t+1) = rand_i \times V_i^d(t) \times a_i^d(t)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1)$$

8 Repetir

2.1 Del Algoritmo y la implementación

En resumen hay que resaltar que mientras se tenga mayores masas se tendrán mejores soluciones, ya que se moverán mas lento y este tiende a atraer hacia los demas puntos hacia el:

- Cada agente es inicializado aleatoriamente y a partir del cual se obtiene las posiciones con las que se calculará su primer fitness
- Una vez que tenemos el valor del fitness se calculan los valores de la masa correspondiente.
- Calculamos la fuerza con la interacción de los vecinos y sus masas respectivas hacia cada una.
- Con el valor de la Fuerza y la masa hallamos la aceleración.

- Se actualiza la velocidad y se halla la nueva posición
- Con la nueva posición se hace la siguiente iteración hasta cumplir con el número de iteraciones definido por el usuario.

Implementación

- Se crea una estructura agente que mantendrá el valor de la posición, la fuerza, velocidad, sus masas, fitnees y aceleración.
- Dentro de la clase "Gravitational" se desarrolla cada uno de los pasos del algoritmo GSA
- El archivo "Gravity" es nuestro main que recibe como parámetros la función de Benchmark a evaluar, las dimensiones, población, los valores mínimos y máximos del dominio y el número de iteraciones que por sugerencia debe ser 50.

Funciones Benchmark Con los parámetros y los dominios se implementaron las funciones Benchmark:

```
float Ackley(std::vector<float> X)
{
    float a = 20.0f, b = 0.2f, c = 2 * (float)PI,  sum = 0.0f, cosenos = 0.0f;
    unsigned int d = X.size();
    for (auto x:X)
    {
        sum += x * x;
        cosenos += cos(c*x);
    }
    return -a * exp(-b * sqrt(sum / d)) - exp(cosenos / d) + a + exp(1);
}
```

Fig. 1. Función Ackley

```
float Schwefel(std::vector<float> X)
{
    float sum = 0.0f;
    for (auto x : X)
    {
        sum += 418.9829f - x * sin(sqrt(abs(x)));
    }
    return sum;
}
```

Fig. 2. Función Schwefel

```
float Funcion_3(std::vector<float> X)
{
    float sum = 0.0f;
    for (auto x:X)
    {
        sum += x * x;
    }
    return 0.5 - (pow(sin(sqrt(sum)), 2) - 0.5) / pow(1.0 + 0.001*sum, 2);
}
```

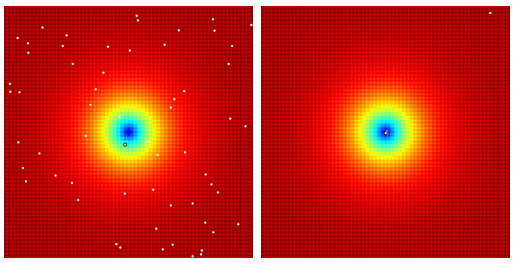
Fig. 3. Función 3

2.2 Del ajuste de parámetros

El articulo de referencia indica algunos parámetros con los que se hicieron las pruebas para la comparación de este algoritmo con otros por ejemplo:

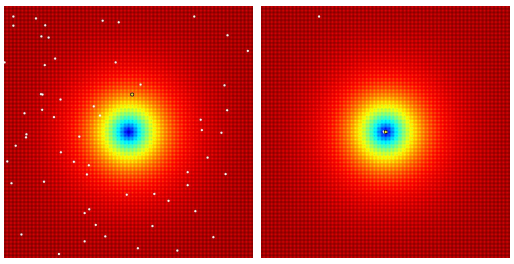
- El número de agentes es 60
- El número de iteraciones es 500
- La gravedad es 12
- El valor de Epsilon es 2.2204e-016

Al aplicar estos valores en nuestro algoritmos se tenia problemas para converger.



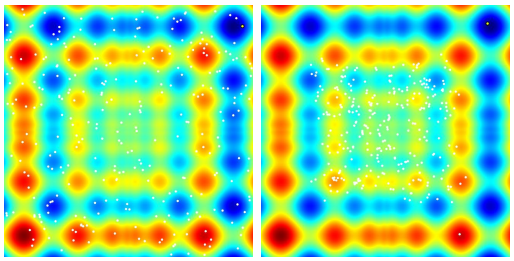
(a) Ackley inicio (b) Ackley fin

Fig. 4. Parámetros Originales
En las siguientes gráficas se vera el punto optimo de color amarillo al cual deberían acercarse los demás puntos después de ejecutarse el algoritmo.



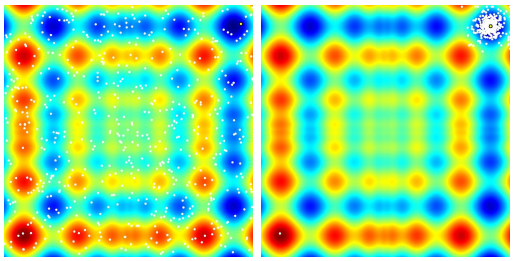
(a) Ackley inicio (b) Ackley fin

Fig. 5. Parámetros modificados
Como se pidió dentro de nuestra implementación usar como numero de iteraciones 50, era necesario modificar la población y los demás parámetros. Se ajustaron básicamente el número de la población y el valor de la gravedad.



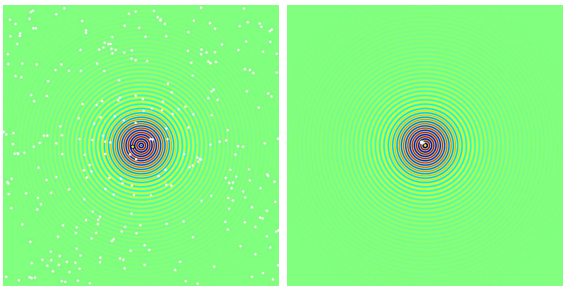
(a) Schwefel inicio (b) Schwefel fin

Fig. 6. Parámetros Originales



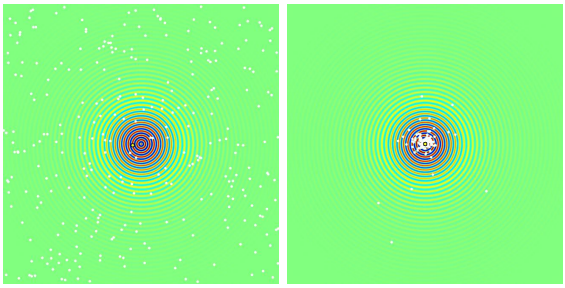
(a) Schwefel inicio (b) Schwefel fin

Fig. 7. Parámetros modificados
En esta función se nota claramente que al modificar los parámetros si hay diferencia porque los valores si se acercan al valor óptimo.



(a) Función 3 inicio (b) función 3 fin

Fig. 8. Parámetros Originales



(a) función 3 inicio (b) función 3 fin

Fig. 9. Parámetros modificados

3. PRUEBAS Y RESULTADOS

Se realizaron pruebas principalmente modificando los parámetros de la gravedad y la población, manteniendo el número de iteraciones, de lo cual se obtuvieron los siguientes resultados:

3.1 Para 2 dimensiones

Funciones Benchmark : Ackley La mejor parametrización para la función Ackley es con G=18 y Población =60

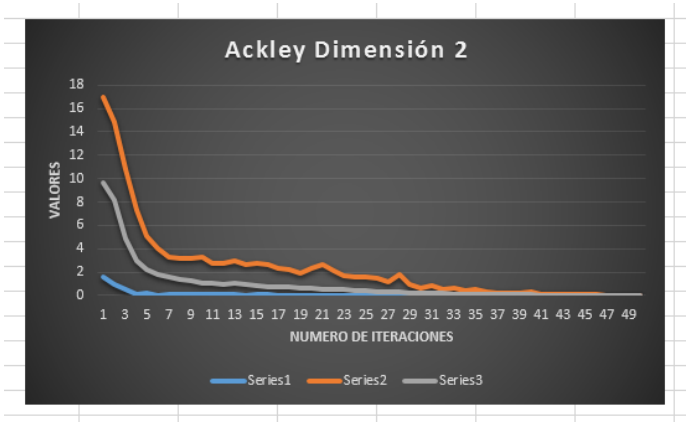


Fig. 10. Donde serie1 representa al mínimo, serie 2 al máximo y serie 3 el promedio según el número de iteraciones

Funciones Benchmark : Schwefel La mejor parametrizacion para la función Schwefel es con G=20000 y Poblacion =600

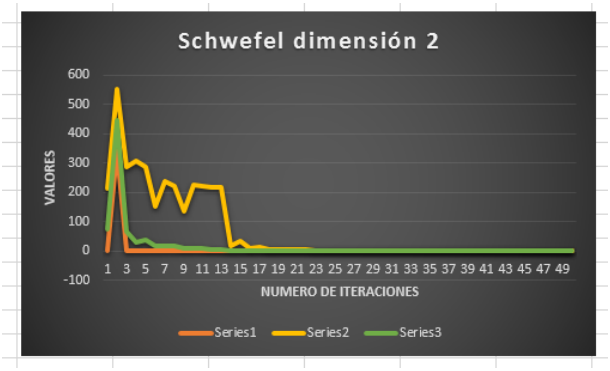


Fig. 11. Donde serie1 representa al mínimo, serie 2 al máximo y serie 3 el promedio según el número de iteraciones

Funciones Benchmark : Función 3 La mejor parametrización para la función 3 es con G=60 y Población =200

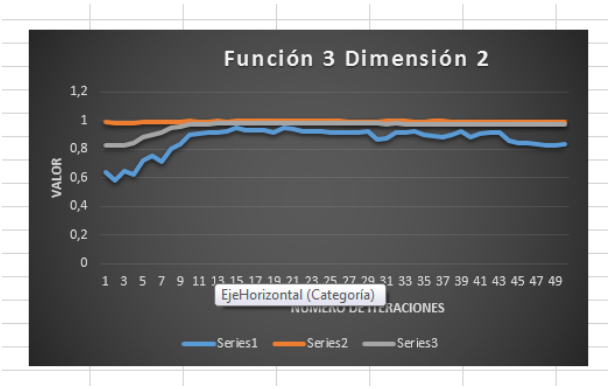


Fig. 12. Donde serie1 representa al mínimo, serie 2 al máximo y serie 3 el promedio según el número de iteraciones

3.2 Para 8 dimensiones

Funciones Benchmark : Ackley La mejor parametrizacion para la función Ackley es con G=18 y Poblacion =60

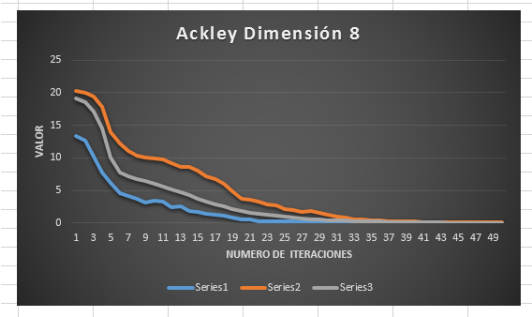


Fig. 13. Donde serie1 representa al mínimo, serie 2 al máximo y serie 3 el promedio según el número de iteraciones

Funciones Benchmark : Schwefel La mejor parametrización para la función Schwefel es con G=20000 y Poblacion =600

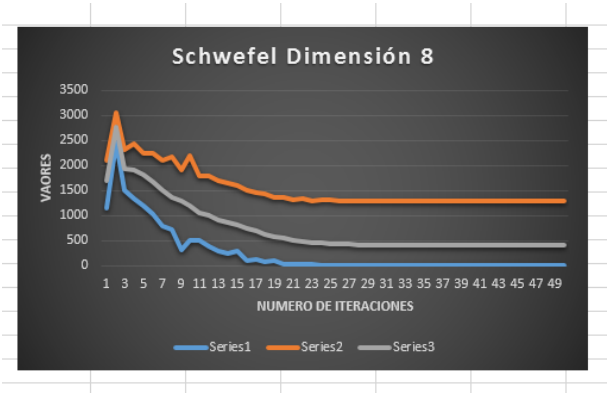


Fig. 14. Donde serie1 representa al mínimo, serie 2 al máximo y serie 3 el promedio según el número de iteraciones

Funciones Benchmark : Funcion 3 La mejor parametrizacion para la función 3 es con G=60 y Poblacion =200

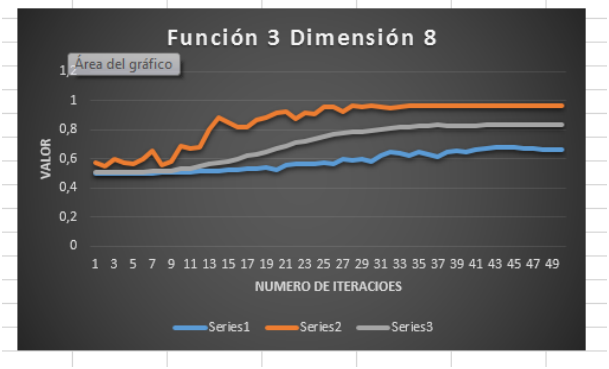


Fig. 15. Donde serie1 representa al mínimo, serie 2 al máximo y serie 3 el promedio según el número de iteraciones

4. PRUEBA DE WILCOXON

Se ejecutó el test de wilcoxon (two tailed test) con alpha igual 0.05, entre el algoritmo implementado GSA con 8 algoritmos (PSO, CUCKOO. FPA, Harmonic, BA, DE, GENOCOP y GWO), esta prueba se ejecuto con los todos los benchmark y las dos dimenensiones (2 y 8). Para ver los resultados revisar la carpeta pruebas en github. Se obtuvo como resultado que todas las comparaciones son diferentes estadísticamente (se rechazó la hipótesis nula). Debido a esto es posible hacer comparaciones con la media de los resultados, para determinar el mejor y peor algoritmo implementado.

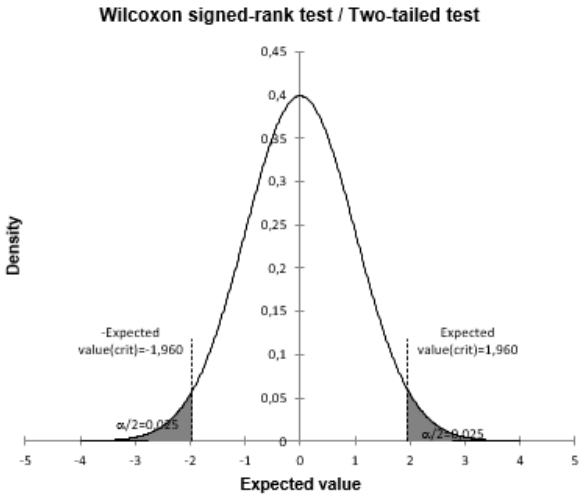


Fig. 16. Prueba de Wilcoxon two tailed

Dimension 2		Dimension 8	
Algoritmo	Ackley	Algoritmo	Ackley
PSO	0	DE	3.204E-13
DE	5.151E-16	PSO	5E-08
BA	2.082E-11	GENOCOP	7.552E-05
GWO	1.607E-08	GWO	0.0008055
CUCKOO	7.028E-06	GSA	0.0182709
GENOCOP	2.479E-05	CUCKOO	0.1999888
FPA	0.001025	FPA	0.2232025
GSA	0.0027724	BA	0.3713777
Harmonic	0.0323461	Harmonic	0.4897078

Fig. 17. Ranking de desempeño con la función Ackley para 2 y 8 dimensiones.

Dimension 2		Dimension 8	
Algoritmo	Schwefel	Algoritmo	Schwefel
GENOCOP	0	GENOCOP	0
DE	2.546E-05	CUCKOO	0.0001024
FPA	2.549E-05	DE	0.000438
CUCKOO	0.0000255	FPA	0.001869
GSA	3.651E-05	PSO	1.7968331
PSO	0.0001064	Harmonic	3.4656322
GWO	0.0001889	BA	7.903294
Harmonic	0.0008708	GSA	395.58589
BA	1.2575439	GWO	720.45968

Fig. 18. Ranking de desempeño con la función Schwefel para 2 y 8 dimensiones.

5. CONCLUSIONES

- Es importante generar gráficos para poder analizar un algoritmo. Los gráficos de las posiciones de los agentes y un respectivo heatmap de una función benchmark fue de grán ayuda para ver como convergen los agente con el paso de las iteraciones, ademas también fue

Dimension 2		Dimension 8	
Algoritmo	Funciòn 3	Algoritmo	Funciòn 3
BA	1	GENOCOP	7000000.9
PSO	0.9999853	FPA	0.9991232
FPA	0.9999687	DE	0.996873
DE	0.9998437	BA	0.992876
GSA	0.9914112	CUCKOO	0.991949
GENOCOP	0.9902841	PSO	0.9901384
CUCKOO	0.99	GSA	0.8657633
Harmonic	0.9830113	Harmonic	0.815959
GWO	-0.9948203	GWO	-0.9861578

Fig. 19. Ranking de desempeño con la función Funcion 3 para 2 y 8 dimensiones.

útil para ver el comportamiento de los agentes con diferentes parametros del algoritmo.

- El algoritmo GSA requiere una optimización de parametros de gravedad y número de población para una función benchmark.

REFERENCES

Gauci, M., Dodd, T.J., and Groß, R. (2012). Why 'gsa: a gravitational search algorithm'is not genuinely based on the law of gravity. *Natural Computing*, 11(4), 719–720.

Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences*, 179(13), 2232–2248.