Análisis de las condiciones iniciales para el algoritmo de optimización basado en enjambres partículas con comportamiento de vorticidad*

Analysis of initial conditions for optimization algorithm based on particle swarm with vorticity behavior

Helbert Eduardo Espitia Cuchango**
Jorge Iván Sofrony Esmeral***

Fecha de recepción: enero de 2014
Fecha de aceptación: marzo de 2014

Resumen

Este artículo analiza el efecto que tienen diferentes configuraciones de condiciones iniciales para el algoritmo de optimización basado en enjambres de partículas con comportamiento de vorticidad. El algoritmo propuesto combina la búsqueda basada en gradiente y un comportamiento de enjambre de partículas, por lo cual, este algoritmo puede ser afectado por las condiciones iniciales dadas para las partículas. Para observar las características del algoritmo se emplea una función de prueba 2D.

Palabras clave: Condiciones iniciales, enjambre de partículas, optimización.

* Artículo de investigación.

** Ingeniero Electrónico y Mecatrónico, Magister en Ingeniería Industrial y Mecánica, docente de planta Universidad Distrital Francisco José de Caldas, Bogotá-Colombia. Contacto: heespitiac@udistrital.edu.co

*** Ingeniero Eléctrico, Doctor en Sistemas de Control, docente de planta Universidad Nacional de Colombia, Bogotá-Colombia. Contacto: jsofronye@bt.unal.edu.co

Abstract

This paper analyzes the effect of different configurations of initial conditions for the optimization algorithm based on particle swarms with vorticity behavior. The proposed algorithm combines the gradient-based search and particle swarm behavior, thus, this algorithm can be affected by the initial conditions for the particles. Finally, a 2D test function is used to observe the characteristics of the algorithm.

Keywords: Initial conditions, optimization, particle swarm.

1. Introducción

Una de las ramas interesantes de la computación flexible consiste en el desarrollo de algoritmos de optimización basados en el comportamiento de seres vivos que, por lo general, está dado por la interacción que se presenta entre un individuo y sus vecinos, siendo esto útil para búsqueda de alimento como también para evadir depredadores.

La descripción del comportamiento de muchos seres vivos se caracteriza por exhibir movimiento cooperativo coordinado, tal como poblaciones de bacterias las cuales presentan movimientos basados en quimiotaxis, también se observa este tipo de comportamiento en bandadas de aves, cardúmenes de peces e incluso en microorganismos como el zooplancton [1]. De los comportamientos de interés se registran el movimiento circular de partículas alrededor de un punto denominado vórtice [2], [3].

2. Optimización bioinspirada

Sobre técnicas de optimización que emulan el comportamiento de seres vivos, se tienen los algoritmos basados en colonias de hormigas, tal como se puede apreciar en [4] y [5]. Bajo esta misma orientación, el comportamiento de abejas ha permitido el desarrollo de algoritmos tanto de optimización como de búsqueda y exploración [6], [7]. Otra técnica de búsqueda inspirada en la biología consiste en el comportamiento que presentan seres unicelulares como las bacterias cuando buscan alimento, dando lugar a la técnica denominada optimización basada en quimiotaxis bacteriana [8]. En relación con el desarrollo de aplicaciones que involucran enjambres de individuos, originalmente su implementación en procesos de optimización fue propuesta por James Kennedy y Russell Eberhart [9]. De igual manera, existen otros trabajos a considerar, los cuales consisten en el desarrollo de algoritmos de optimización basados en enjambres de partículas que emplean elementos de mecánica cuántica [10], como también la aplicación el oscilador armónico [11].

Acerca de otros algoritmos de computación flexible bio inspirados se pueden apreciar propuestas basadas en el comportamiento de murciélagos (*Bat Algorithm* BA) [12], [13], luciérnagas (*Firefly Algorithm* FA) [14] y primates [15]. También se tienen enfoques bajo principios y leyes físicas como la ley de Coulomb para partículas cargadas (*Charged System Search* CSS) [16], la sintonía en frecuencia de una nota musical (*Harmony Search* HS) [17] y la caída de agua en una cascada (*Intelligent Water Drops* IWD) [18].

2.1 Algoritmos de enjambres de partículas con estrategias para evasión de mínimos locales

De los diferentes algoritmos de optimización bio-inspirados, según Bratton y Kennedy [19] los algoritmos de enjambres de partículas son una buena alternativa, sin embargo, estos tienden a presentar convergencia temprana en mínimos locales [20], [21]. Adicionalmente, son susceptibles a una mala selección de sus parámetros, tal como se muestra en [22] y [23]. Por lo anterior, se han desarrollado modificaciones y propuestas con las cuales se busca evadir mínimos locales teniendo una mejor exploración del espacio de soluciones.

Una estrategia general para el escape de mínimos locales consiste en realizar un proceso de dispersión —explosión— luego de tener una convergencia a un mínimo local. Un ejemplo de este concepto se pueden apreciar en [24] con el algoritmo de optimización Supernova, en [25] para el algoritmo de optimización *Glowworm* y en [26] con el algoritmo de optimización basado en forrajeo de bacterias.

En particular, para el algoritmo PSO una primera modificación consiste en adicionar un factor de inercia modulada tal como se propone en [27] y [28]. Con este enfoque se busca controlar la exploración del algoritmo sobre el espacio de búsqueda. En [27] y [28] se expone que un factor grande de inercia acelera la convergencia, mientras que un valor pequeño mejora la capacidad de búsqueda. Una modificación adicional del algoritmo PSO consiste en reiniciarlo cuando se considera que hay un estancamiento de este [29].

Aparte del enfoque de inercia modulada en [30], se propone un método denominado olas de enjambres de partículas (*Waves of Swarm Particles* WoSP) con el cual se busca impulsar el enjambre para que pueda escapar de un mínimo local y así continuar con el proceso de exploración. Por otro lado, en [31] se propone emplear técnicas de repulsión para cada mínimo local encontrado, esperando así evadir soluciones encontradas previamente. Adicionalmente, en [32] se presenta una variante del algoritmo PSO denominada aprendizaje integral, la cual consiste en emplear la información histórica de las partículas para actualizar su velocidad. Este enfoque busca conservar la diversidad del enjambre evitando la convergencia prematura.

Otra variación del algoritmo PSO consiste en incorporar un término de turbulencia, dando lugar de esta forma a varias propuestas. En [33] se incluye este operador en el algoritmo PSO tradicional como una variable estocástica. Según [34], con este operador se busca mantener la diversidad de la población. Desde un punto de vista de algoritmos evolutivos, el término de turbulencia tiene la misma función de un operador de mutación [35]. Por otro lado, en [36] se propone el algoritmo TPSO (*Turbulence in the Particle Swarm Optimization*) donde se incluye la turbulencia para resolver el problema de la convergencia prematura, con esta propuesta se busca impulsar las partículas perezosas para llevarlas a explorar nuevos espacios de búsqueda. TPSO utiliza un umbral de velocidad mínima para controlar la velocidad de las partículas evitando su aglomeración y manteniendo la diversidad de la población en el espacio de búsqueda. Un enfoque adicional al factor de turbulencia consiste en realizar una perturbación al enjambre proporcional a la distancia entre una determinada partícula y otra, la cual se toma de forma aleatoria, en esta propuesta el factor de turbulencia ayuda a escapar al enjambre de mínimos locales [37], [38].

Un acercamiento desde un enfoque analítico se presenta en [23], donde se efectúa un estudio de la convergencia al algoritmo PSO.

Con este análisis se busca mejorar la selección de parámetros y de esta forma también su convergencia.

3. Estrategia de búsqueda basada en dispersión

La propuesta de búsqueda emplea el comportamiento de vorticidad como un mecanismo de dispersión para lograr una búsqueda global. En la figura 1 se puede apreciar el diagrama de flujo para la estrategia de búsqueda propuesta. En un primer lugar, se inicializa el enjambre y se procede a encontrar el mínimo local más cercano, almacenando el valor del mínimo encontrado; luego, para lograr que el enjambre escape del mínimo encontrado se realiza el proceso de dispersión, empleando para esto el comportamiento de vorticidad. Con el anterior proceso se espera encontrar un valor mínimo menor al encontrado previamente, en el caso que no se encuentre un valor menor se detiene el algoritmo considerando para esto una dispersión máxima de las partículas sobre el espacio de búsqueda.

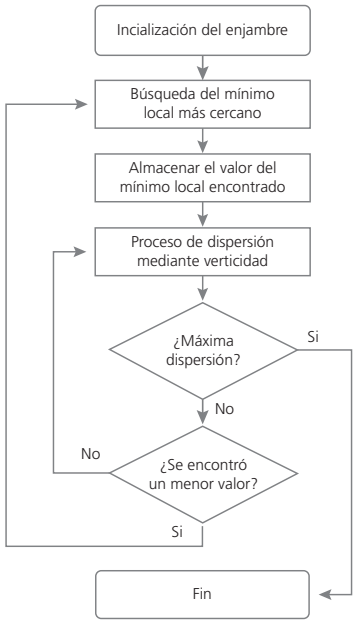

Figura 1. Esquema propuesto para el proceso de búsqueda

Fuente: elaboración propia

4. Algoritmo de optimización basado en partículas con comportamiento de vorticidad

El *Algoritmo de optimización basado en enjambres partículas con comportamiento de vorticidad* (Vortex Particle Swarm Optimization, VPSO) se fundamenta en realizar un proceso de dispersión de las partículas para lograr que estas escapen de un mínimo local.

La dispersión se logra mediante el comportamiento de vorticidad que presentan las partículas, para esto se tomó como referencia un modelo dinámico que describe la forma de locomoción del zooplancton Daphnia, el cual presenta movimientos circulares y lineales que le permiten evadir depredadores y buscar alimento [39].

Las ecuaciones para calcular la posición y la velocidad son:

$$\vec{r}_i(n) = \vec{r}_i(n-1) + \vec{v}_i \Delta t \quad (1)$$

$$\vec{v}_i(n) = \vec{v}_i(n-1) + (\vec{F}_{pro,i} + \vec{F}_{int,i} + \vec{F}_{obj,i})\Delta t \quad (2)$$

Donde, se tiene la fuerza de auto-propulsión la cual se considera como:

$$\vec{F}_{pro,i} = (-\alpha + \beta \mid v_i \mid^2)\vec{v}_i \quad (3)$$

La fuerza de interacción de las partículas está dada por:

$$\vec{F}_{int,i} = -a(\vec{r}_i - \vec{R})$$

$$\vec{F}_{int,i} = -\frac{a}{N}\sum_{j=1}^{N}(\vec{r}_i - \vec{r}_j)$$

Donde $\vec{R}$ corresponde al centro de masa del enjambre:

$$\vec{R} = \frac{1}{N}\bullet_{j=1}^{N}\vec{r}_j \quad (4)$$

La información de la función objetivo esta dada por U_{obj}. La fuerza sobre cada partícula que se produce por el potencial U_{obj} es:

$$\vec{F}_{obj,i} = -k_f \vec{\nabla}_i U_{obj}(\vec{r}_i) \quad (5)$$

Donde k_f es una constante que pondera la influencia de la función objetivo.

Para lograr que el enjambre escape de mínimos locales y así cubrir de forma adecuada el espacio de búsqueda, se propone aumentar la energía de propulsión del enjambre α cuando éste alcanza un mínimo local y disminuirla cuando el enjambre escapa. Para lo anterior se considera una variable U_{min} que se actualiza con el valor mínimo del campo potencial de $U_{obj}(\vec{R})$, esto puede ser expresado como:

$$U_{min} = \begin{cases} U_{obj}(\vec{R}), & \text{si} U_{min} \geq U_{obj}(\vec{R}); \\ U_{min}, & \text{si} U_{min} < U_{obj}(\vec{R}). \end{cases} \quad (6)$$

La adición de la energía se logra con el factor de propulsión α. En esta propuesta α se calcula como:

$$\alpha(n) = \begin{cases} \alpha(n-1) + \tau_c \Delta t, & \text{si} U_{min} \leq U_{obj}(\vec{R}); \\ 0, & \text{si} U_{min} > U_{obj}(\vec{R}). \end{cases} \quad (7)$$

Es de apreciar que al llegar el enjambre a un mínimo local U_{min} " $U_{obj}(\vec{R})$ el término α se incrementa, aumentando de esta forma la energía de propulsión. Esta acción continuará sucediendo hasta que las partículas escapen del mínimo local. El aumento de la energía viene dado por el parámetro τ_c. El algoritmo de optimización propuesto se puede apreciar en la figura 2.

Algoritmo 1: Algoritmo de optimización propuesto

```
1  Inicializar el enjambre en el espacio solución:
2  begin
3      while Bajo algún criterio de finalización (Dispersión o Iteraciones)
       do
4          Calcular U_act y U_min;
5          Calcular α;
6          for i=1 hasta N do
7              Calcular la nueva velocidad de las partículas, empleando la ecuación 1;
8              Calcular la nueva posición de las partículas, empleando la ecuación 2.
9          end
10     end
11     Establecer el mínimo encontrado.
12 end
```

Figura 2. Algoritmo de optimización basado en enjambres de partículas con comportamiento de vorticidad

Fuente: elaboración propia.

4.1 Criterio de parada

El criterio de parada propuesto establece que, si la dispersión del enjambre σ_G es mayor que la dispersión calculada para el espacio de búsqueda dado σ_g, entonces el algoritmo se finaliza. En esta propuesta, la medida de la dispersión se considera como la desviación típica generalizada.

Para determinar σ_g como primera medida se debe considerar el teorema de Chebyshev donde se establece que para un conjunto de

datos en una dimensión $(D=1)$, por lo menos $(1-1/k^2)100\%$ de los datos están contenidos dentro de $(\bar{x}-k\sigma, \bar{x}+k\sigma)$ donde $\bar{x}$ es el valor medio de los datos y σ es la desviación típica.

Adicionalmente observando que el $(1-1/k^2)100\%$ de datos tiene rango de $L=2k\sigma$, es posible calcular la desviación de los datos como $\sigma = L/2k$ donde L puede ser considerado como el rango de la función objetivo para una dimensión.

Para varias dimensiones con rangos $L_1, L_2, ..., L_D$ y considerando que las variables no están correlacionadas, la desviación estándar generalizada puede ser estimada como $\sigma_g = \sigma_1\sigma_2\cdots\sigma_D$ por lo tanto:

$$\sigma_g = (L_1/2k)(L_2/2k)\cdots(L_D/2k) \quad (8)$$

Por otro lado, para determinar el valor de dispersión del enjambre de partículas, estas se pueden representar como un conjunto de datos de la forma:

$$X = x_{ij} = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}$$

Donde N es el total de partículas, D es el número de dimensiones, i es el índice de cada partícula y j el índice para cada dimensión.

El valor medio de la j-ésima variable de la matriz de datos X es:

$$\bar{x}_j = \frac{1}{N}\bullet_{i=1}^{N} x_{ij}$$

Donde $\bar{x}_j$ corresponde al vector promedio $\bar{X}$.

$$\bar{X} = \begin{bmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_D \end{bmatrix}$$

La matriz de varianza y covarianza muestral S corresponde a:

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1D} \\ \vdots & \ddots & \vdots \\ s_{D1} & \cdots & s_{DD} \end{bmatrix}$$

Si $i \uparrow j$ se tiene:

$$s_{ij} = \frac{1}{N}\sum_{k=1}^{N}(x_{ki}-\bar{x}_i)(x_{kj}-\bar{x}_j)$$

por otro lado si $i=j$ entonces:

Finalmente, la varianza generalizada del conjunto de datos X se puede calcular como el determinante de S y la desviación generalizada como:

$$\sigma_G = \sqrt{det(S)} \quad (9)$$

4.2 Análisis cualitativo del algoritmo

Para mostrar el funcionamiento del algoritmo, se realiza una simulación considerando un potencial cuadrático de la forma $U_{obj} = 2(x^2+y^2)$. En la figura 3 se puede apreciar el potencial y la evolución que tiene la posición de las partículas en la medida que pasan las iteraciones. Es de apreciar que, luego de encontrar el mínimo local, las partículas inician el proceso de dispersión realizando un movimiento circular.

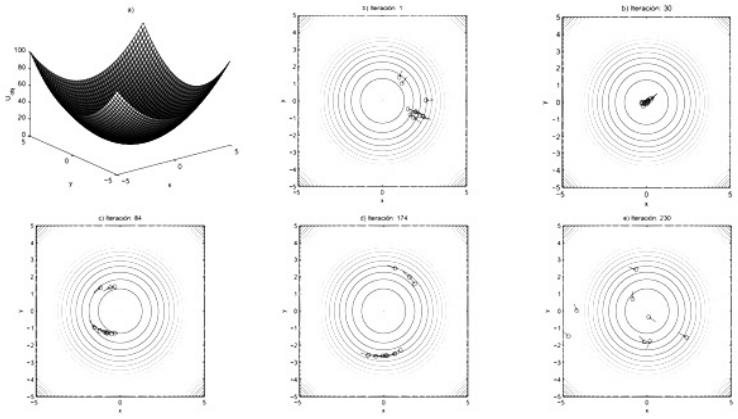

Figura 3. Representación gráfica del funcionamiento del algoritmo en un potencial parabólico

Fuente: elaboración propia.

5. Resultados experimentales

Los experimentos se realizan buscando establecer si las condiciones iniciales de las partículas afectan el desempeño del algoritmo. Las condiciones iniciales de las partículas se generan tomando un punto aleatorio y una dispersión de las partículas alrededor de este punto aleatorio, para esto se emplea la siguiente expresión:

$$\vec{r}_i(0) = \vec{G}L_G(1-\gamma) + \vec{g}_i L_g \gamma \qquad (10)$$

Donde $\vec{G}$ es un vector de números aleatorios en el intervalo $[-1,1]$ el cual es el mismo para todos las partículas, $\vec{g}_i$ también es un vector de números aleatorios en el intervalo $[-1,1]$ pero diferente para cada partícula, L_G y L_g son escalares que corresponden al valor máximo sobre el cual se quiere realizar la dispersión y γ es un número en el intervalo $[0,1]$ el cual permite determinar la configuración de condiciones iniciales a estudiar.

5.1 Función de prueba

Para observar el comportamiento que presenta el algoritmo, se emplea la función objetivo propuesta en [8]. La expresión para la función de prueba es:

$$\begin{aligned} U_{obj} &= +0.01(x^2+y^2) \\ &+5e^{-0.8((x)^2+(y-1.7)^2)} - 2e^{-0.64((x-1.7)^2+(y-0)^2)} \\ &+3e^{-0.64((x-3.3)^2+(y+1.7)^2)} + 2e^{-0.8((x+1.7)^2+(y+1.7)^2)} \\ &-2e^{-4((x+3.3)^2+(y+1.7)^2)} - 4e^{-0.8((x-0)^2+(y+3.3)^2)} \\ &-2e^{-4((x+2.3)^2+(y-3.3)^2)} - 2e^{-4((x-2.0)^2+(y-3.3)^2)} \\ &+2e^{-4((x-3.3)^2+(y-0.3)^2)} + 2e^{-4((x+3.3)^2+(y+0.3)^2)} \end{aligned}$$

Esta función presenta varios mínimos y máximos los cuales se pueden apreciar en la figura 4.

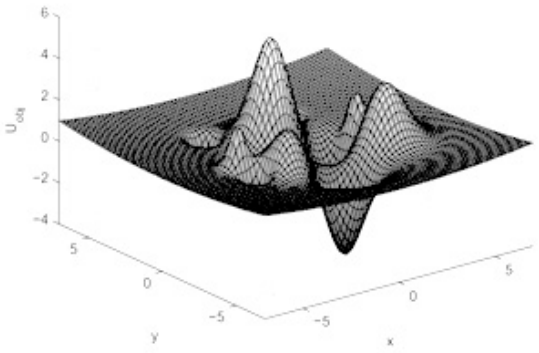

Figura 4. Función de prueba

Fuente: elaboración propia.

5.2 Cálculo del criterio de parada

Considerando un espacio de búsqueda en dos dimensiones donde se tiene que $-7 \le x \le 7$ ($L_x = 14$) y $-7 \le y \le 7$ ($L_y = 14$), entonces, la desviación típica generalizada para diferentes valores de k es:

- Para: $k = 2$ $(75,0\%), \sigma_g = 12,25$.
- Para: $k = 3 (88,8\%), \sigma_g = 5,44$.
- Para: $k = 4$ $(93,7\%)$, $\sigma_g = 3,06$.

Es de apreciar que con $k = 2$ se obtiene una mayor dispersión por lo cual se toma $\sigma_g = 12,25$.

5.3 Configuración de parámetros y condiciones iniciales

Para la implementación del algoritmo se tomó $\tau_c = 1$, $\beta = 1$, $\varnothing t = 0,1$, por otro lado, para observar el comportamiento del algoritmo con diferentes condiciones iniciales se consideró la siguiente configuración de parámetros:

- Configuración 1: $k_f = 1$ y $a = 1$.
- Configuración 2: $k_f = 0,5$ y $a = 0,5$.

Los valores de γ se toman como: 0,1, 0,3, 0,7 y 1. Se realizan 50 ejecuciones del algoritmo para cada caso de condiciones iniciales obteniendo los resultados de la tabla 1.

Tabla 1. Resultados para 50 ejecuciones

Casos	**Configuración 1**		**Configuración 2**	
Caso 1	Desempeño	Iteraciones	Desempeño	Iteraciones
Max	0,3208	522	0,2633	341
Min	-3,8656	114	-3,8656	78
Promedio	-2,0435	232,28	-2,1238	218,7
STD	1,3375	63,82	1,4817	63,19
Caso 2	Desempeño	Iteraciones	Desempeño	Iteraciones
Max	0,3621	552	0,3251	324
Min	-3,8655	145	-3,8655	57
Promedio	-2,2409	243,1	-2,3042	182,62
STD	1,2644	75,01	1,2727	68,98
Caso 3	Desempeño	Iteraciones	Desempeño	Iteraciones
Max	0,1659	378	0,0963	454
Min	-3,8656	121	-3,8656	105
Promedio	-2,5416	260,64	-2,6203	209,58
STD	1,3237	55,93	1,1816	79,14
Caso 4	Desempeño	Iteraciones	Desempeño	Iteraciones
Max	0,1434	448	2,1854	362
Min	-3,8656	104	-3,8656	59
Promedio	-3,0057	265,28	-2,5152	203,04
STD	1,2148	80,89	1,4597	73,27

Fuente: elaboración propia.

5.4 Análisis estadístico de resultados

Con el fin de seleccionar entre pruebas paramétricas y no paramétricas para realizar la comparación entre grupos, primero se implementa la prueba de normalidad de Kolmogorov-Smirnov (K-S) y la prueba de homocedasticidad de Levene [40]. Si se cumplen los supuestos de normalidad y homocedasticidad se implementa la prueba para comparación entre grupos de ANOVA y, en caso contrario, se emplea la prueba de Kruskal-Wallis (K-W). Para cada prueba se calcula el respectivo p -value de tal forma que si éste es mayor que el nivel de significancia de 0,05 se acepta la hipótesis nula de normalidad, homocedasticidad o igualdad entre grupos según sea el caso que se este considerando. Los resultados de estas pruebas se pueden apreciar en la tabla 2.

Tabla 2. Análisis estadístico

	Configuración 1		**Configuración 2**	
Prueba	Desempeño	Iteraciones	Desempeño	Iteraciones
K-S	$1,5298\times10^{-13}$	0,3189	$1,57\times10^{-10}$	0,5388
Levene	0,4274	0,0567	0,5494	0,1513
ANOVA	-	0,066	-	0,0783
K-W	$5,017\times10^{-4}$	-	0,1740	-

Fuente: elaboración propia.

Para la configuración 1 donde se observa el resultado obtenido de la función objetivo, las comparaciones multiples se realizaron mediante la prueba de Bonferroni —para mayor detalle sobre la prueba ver [41] y [42]—

obteniendo la figura 5 donde se aprecia que existe diferencia entre el caso 4 con el 1 y 2.

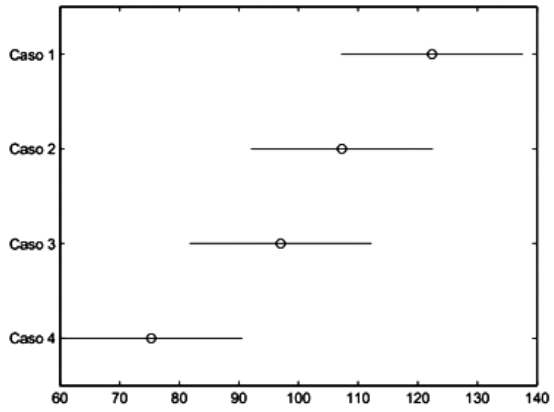

Figura 5. Comparación mediante el método de Bonferroni

Fuente: elaboración propia.

6. Conclusiones

Se observó que el algoritmo logra encontrar el mínimo global de la función propuesta para los diferentes parámetros seleccionados como también para las condiciones iniciales seleccionadas.

De los resultados experimentales se observó estadísticamente que las condiciones iniciales no son un factor dominante en el comportamiento del algoritmo, sin embargo se observa un caso donde es favorable tener una buena dispersión del enjambre sobre todo el espacio de búsqueda.

A pesar que el algoritmo emplea el cálculo de gradientes para converger a un mínimo local, para los casos estudiados el algoritmo no se ve fuertemente afectado por las condiciones iniciales.

Referencias

[1] U. Erdmann, W. Ebeling, L. Schimansky, A. Ordemann, F. Moss, "Active brownian particle and random walk theories of the motions of zooplankton: application to experiments with swarms of daphnia", *Journal of Theoretical Biology 9,* February. 2008.

[2] Y. Hsin, "Emergence of vortex swarming in daphnia", *Term Paper for Emergent State of Matter*, Spring 2006.

[3] E. Werner, "Nonequilibrium statistical mechanics of swarms of driven particles", *Elsevier Physica*. 2002.

[4] M. Dorigo, M. Birattari, T. Stützle, "Ant Colony Optimization. Artificial Ants as a Computational Intelligence Technique", *IEEE Computational Intelligence Magazine*, November. 2006.

[5] A. Zecchin, A. Simpson, H. Maier, J. Nixon, "Parametric Study for an Ant Algorithm Applied Water Distribution System Optimization", *IEEE Transactions On Evolutionary Computation,* Vol. 9, No. 2, April. 2005.

[6] D. Karaboga, "An Idea Based On Honey Bee Swarm For Numerical Optimization", *Technical Report-TR06*, October. 2005.

[7] S. Thakoor, J. Morookian, C. Javaan, B. Hine, S. Zornetzer, "BEES: Exploring Mars with Bioinspired Technologies", *IEEE Computer Society*. 2004.

[8] K. Passino, "Biomimicry of bacterian foragin for distributed optimization and control", *IEEE Control Systems Magazine*, June. 2002.

[9] E. Russell, K. James, "Particle Swarm Optimization", *IEEE Proceedings Neural Networks*. 1995.

[10] L. Coelho, "A quantum particle swarm optimizer with chaotic mutation operator", *Elsevier Chaos Solitons and Fractals*. 2006.

[11] L. Coelho, V. Cocco, "Particle swarm approach based on quantum mechanics and harmonic oscillator potential well for economic load dispatch with valve-point effects", *Elsevier Energy Conversion and Management*. 2008.

[12] D. Sedighizadeh, E. Masehian, "Particle swarm optimization methods, taxonomy and applications", *International Journal of Computer Theory and Engineering*, Vol. 1, No. 5, December. 2009.

[13] X. Yang, "A new metaheuristic bat-inspired algorithm", *Nature Inspired Cooperative Strategies for Optimization (NICSO)*. 2010.

[14] X. Yang, *Firefly algorithm, lévy flights and global optimization*, Springer London: Research and Development in Intelligent Systems XXVI, 2009.

[15] A. Mucherino, O. Seref, *Monkey search: a novel metaheuristic search for global optimization*, in Aip Conference Proceedings, Data mining systems analysis and optimization in biomedicine. 2007.

[16] A. Kaveh, S. Talatahari, "A novel heuristic optimization method: charged system search", *Springer-Verlag, Acta Mechanica*. 2010.

[17] X. Yang, *Harmony search as a metaheuristic algorithm*, Springer Berlin: Music-Inspired Harmony Search Algorithm, 2009.

[18] H. Shah, "Problem solving by intelligent water drops", *IEEE Congress on Evolutionary Computation*. 2007.

[19] D. Bratton, J. Kennedy, "Defining a Standard for Particle Swarm Optimization", *IEEE Swarm Intelligence Symposium SIS*. 2007.

[20] G. Evers, *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization*, Master Thesis: University of Texas-Pan American, 2009.

[21] J. Schutte, *Particle swarms in sizing and global optimization*, Master's Dissertation: University of Pretoria, 2002.

[22] M. Hvass, *Tuning&Simplifying Heuristical Optimization*, Ph.D. Thesis: University of Southampton, UK, 2010.

[23] F. Van den Bergh, *An Analysis of Particle Swarm Optimizers*, PhD. Thesis: University of Pretoria, Pretoria, 2001.

[24] E. Mesa, *Supernova: un algoritmo novedoso de optimización global*, Tesis de Maestría, Universidad Nacional de Colombia: Sede Medellín, 2010.

[25] K. Krishnanand, D. Ghose, "Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions", *Springer Science, Swarm Intell*. 2009.

[26] K. Passino, *Biomimicry for optimization, control, and automation*, Springer-Verlag: London, UK, 2005.

[27] C. Feng, S. Cong, X. Feng, "A new adaptive inertia weight strategy in particle swarm optimization", *IEEE Congress on Evolutionary Computation (CEC),* 2007.

[28] L. Yin, X. Liu, *A PSO Algorithm Based on Biologe Population Multiplication (PMPSO)*, in Second Symposium International Computer Science and Computational Technology (ISCSCT '09), 2009.

[29] J. García, E. Alba, *Restart Particle Swarm Optimization with Velocity Modulation: A Scalability Test*, in Springer, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 1. 1997.

[30] T. Hendtlass, "A particle swarm algorithm for high dimensional, multi-optima problem spaces", *IEEE Swarm Intelligence Symposium*, 2005.

[31] K. Parsopoulos, M. Vrahatis, "On the computation of all global minimizers through particle swarm optimization", *IEEE Transactions on Evolutionary Computation,* Vol. 8. 2004.

[32] J. Liang, A. Qin, P. Suganthan, S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", *IEEE Transactions on Evolutionary Computation*, Vol. 10. 2006.

[33] J. Fieldsend, S. Singh, "A multiobjective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence", *Workshop on Computational Intelligence*, Vol. 2723/2003, pp. 34 - 44. 2002.

[34] A. Cervantes, *Clasificación mediante enjambre de prototipos*, Tesis Doctoral: Universidad Carlos III de Madrid, Departamento de Informática, Leganés, 2009.

[35] K. Deb, N. Padhye, *Development of efficient Particle Swarm Optimizers by using concepts from evolutionary algorithms*, in 12th annual conference on Genetic and evolutionary computation, GECCO '10, pp. 55-62, 2010.

[36] H. Liu, A. Abraham, *Fuzzy adaptive turbulent particle swarm optimization*, in V international conference on hybrid intelligent systems (HIS'05), Rio de Janeiro, Brazil, November, 2005.

[37] S. He, Q. Wu, J. Wen, J. Saunders, P. Patton, "A particle swarm optimizer with passive congregation", *Biosystems,* Vol. 78, pp. 135 - 147. 2004.

[38] J. Vlachogiannis, K. Lee, "A comparative study on particle swarm optimization for optimal steady-state performance of power systems", *IEEE Transactions on Power Systems,* Vol. 21, No. 4, November. 2006.

[39] A. Ordemanna, G. Balazsi, F. Moss, "Pattern formation and stochastic motion of the zooplankton Daphnia in a light field", *Elsevier Science B.V., Physica A* 325. 2003.

[40] M. Gómez, C. Danglot, L. Vega, "Sinopsis de pruebas estadísticas no paramétricas. Cuándo usarlas", *Revista Mexicana de Pediatría,* Vol. 70 No. 2 Marzo-Abril. 2003.

[41] J. Derrac, S. García, D. Molina, H. Francisco, *Un tutorial sobre el uso de test estadísticos no paramétricos en comparaciones múltiples de metaheurísticas y algoritmos evolutivos*, en VIII Congreso Español Sobre Metauristicas, Algoritmos Evolutivos y Bioinspirados, 2012.

[42] S. García, A. Fernández, J. Luengo, F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power", *Information Sciences*, Vol. 180, No. 20, pp. 2044–2064. 2010.