Programación Matemática y Software (2019) 11 (3): 1-11. ISSN: 2007-3283

Ca-PSO: Coulomb atrayendo un Cúmulo de Partículas

Ca-PSO: Coulomb attracting Particle Swarms

Nayeli Joaquinita Meléndez Acosta[1], Ricardo Solano Monje[2], Cosijopii García García[3], Homero Vladimir Ríos Figueroa[4]

[1] Universidad del Istmo campus Ixtepec, Carr. Chihuitan Ixtepec S/N Ixtepec, Oax., México. C.P. 70110
[2] Instituto Tecnológico Superior de Ciudad Serdán, Avenida Instituto Tecnológico s/n, Col la Gloria, Cd Serdán, Puebla, México. C.P. 75520
[3] Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro # 1, Tonantzintla, Puebla, México. C.P. 72840
[4] Universidad Veracruzana, Sebastián Camacho 5, Zona Centro, Centro, Xalapa Enríquez, Ver. México. C.P. 91000

* Correo-e: nayelimelendez@gmail.com

PALABRAS CLAVE:

Algoritmo de Optimización, Ley de Coulomb, Funciones de Optimización, Cúmulo de Partículas.

RESUMEN

Este artículo presenta una variante del algoritmo C-PSO, al cual hemos llamado Ca-PSO, a diferencia de C-PSO que considera como cargas puntuales a $lBest_i$ y gBest, Ca-PSO considera a la partícula x_i y gBest. Al mismo tiempo se presenta una comparación de cuatro algoritmos: el algoritmo original PSO (Particle Swarm Optimization), PSO con "restricción" (Constriction PSO), C-PSO una versión que hace uso de ley de Coulomb y el algoritmo propuesto Ca-PSO. También se muestra el movimiento esquemático de una partícula en el algoritmo Ca-PSO. Los resultados que se muestran corresponden a la media de 50 corridas, cada algoritmo habiendo sido ejecutado 10000 iteraciones por función para 50 y 100 dimensiones. El algoritmo Ca-PSO mostró un rendimiento superior respecto a C-PSO en seis de las diez funciones de prueba, además se muestra que tanto C-PSO como Ca-PSO presentan un mejor rendimiento que el algoritmo original de PSO y PSO con restricción.

KEYWORDS:

Optimization Algorithm, Coulomb Law, Optimization Functions, Particle Swarm.

ABSTRACT

This article presents a variant of the C-PSO algorithm, which we have called Ca-PSO, unlike C-PSO which considers $lBest_i$ and gBest as point charges to, Ca-PSO considers the particles x_i and gBest as them. At the same time a comparison of four algorithms is presented: the original algorithm PSO (Particle Swarm Optimization), PSO with "constriction" (Constriction PSO), C-PSO a version that makes use of Coulomb's law and the proposed algorithm C-PSO. The schematic movement of a particle in the Ca-PSO algorithm is also shown. The results that are shown correspond to the mean of 50 runs, each algorithm has been executed 10000 iterations per function on 50 and 100 dimensions. The Ca-PSO algorithm showed a superior performance over the C-PSO in six of ten testing functions. Moreover, it is shown that both C-PSO and Ca-PSO present a better performance than the original algorithm of PSO and Constriction PSO.

Recibido: 28 de junio 2018• **Aceptado:** 10 de octubre de 2018 • **Publicado en línea:** 31 de octubre de 2019

1 Introducción

PSO es una técnica de búsqueda bioinspirada que simula el comportamiento social observado en grupos o cúmulos de individuos biológicos [1]. Esta técnica está basada en el principio de que la inteligencia no se encuentra en individuos, sino en un colectivo y ha sido utilizado por muchas aplicaciones de varios problemas.

El algoritmo consiste en un cúmulo de partículas que se colocan en el espacio de búsqueda de algún problema o función, donde cada partícula representa una solución potencial al problema [7]. Una partícula en sí misma no tiene casi ningún poder para resolver ningún problema; el progreso ocurre solo cuando las partículas interactúan. La resolución de problemas es un fenómeno de toda la población, que surge del comportamiento individual de las partículas a través de sus interacciones. Cada partícula ajusta su posición mediante la combinación de una atracción hacia la mejor solución que ha encontrado individualmente, y una atracción hacia la mejor solución que cualquier partícula ha encontrado [2], imitando a aquellos que tienen un mejor desempeño.

Recientemente, hay varias modificaciones de PSO original, esto para acelerar el rendimiento y mejorar las condiciones que permitan proporcionan nuevas ventajas y mayor diversidad de problemas a resolver [3].

Este artículo presenta una variación del algoritmo C-PSO, llamada Ca-PSO, además se muestra una comparación con el algoritmo PSO original, el algoritmo PSO con restricción, la versión C-PSO y una nueva variante de este último que hemos llamado Ca-PSO. El objetivo del artículo es probar el rendimiento de los algoritmos utilizando un conjunto de funciones para su optimización. El contenido del artículo se encuentra organizado de la siguiente manera, primero en la segunda sección se describe el algoritmo original PSO, luego en la tercera sección se describen dos variaciones de PSO, PSO con restricción y C-PSO, posteriormente en la cuarta sección se presenta el algoritmo propuesto en este trabajo Ca-PSO. En la quinta sección se muestran las pruebas realizadas y los resultados obtenidos. Finalmente, en la quinta sección se presentan las conclusiones.

2 Algoritmo optimización por cúmulo de partículas - PSO

El algoritmo optimización por cúmulo de partículas fue introducido por primera vez por Kennedy y Eberhart en 1995 [4]. Fue inspirado por el comportamiento social de organismos como el agrupamiento de aves. PSO es una técnica de optimización basada en la población, en donde cada solución llamada "partícula", vuela en el espacio de búsqueda de un problema buscando la posición óptima. Las partículas se colocan en el espacio de búsqueda del problema o función, y cada una evalúa la función objetivo en su ubicación actual.

Una partícula está compuesta de tres vectores: la posición actual (x_i), la velocidad (v_i) y una memoria de la mejor posición que ha obtenido ($lBest_i$– mejor local), además de tener el conocimiento de la mejor solución encontrada por el cúmulo ($gBest$-mejor global). La posición actual (x_i) se puede considerar como un conjunto de coordenadas que describen un punto en espacio. Cada partícula, a medida que el tiempo pasa a través de su búsqueda, ajusta su posición de acuerdo a tres componentes: su velocidad anterior, su propia "experiencia" (La mejor posición encontrada hasta ahora por la partícula, que se representa como $lBest_i$) y la experiencia del cúmulo de partículas (La mejor posición alcanzada por todas las partículas, que se representa como $gBest$).

La experiencia de la partícula se construye memorizando la mejor posición encontrada. Por esa razón, PSO posee una memoria, es decir, cada partícula recuerda la mejor posición que alcanzó durante el pasado, además de conservar el conocimiento de la mejor solución encontrada por todas las partículas. Las partículas en el cúmulo comparten información entre ellas. PSO combina el método de búsqueda local (a través de la experiencia propia) con métodos de búsqueda globales (a través de la experiencia del cúmulo).

Al inicio del algoritmo, los valores numéricos de la posición y la velocidad de las partículas

se asignan aleatoriamente. Las ecuaciones (1) y (2) permiten actualizar la posición y la velocidad de las partículas para iteraciones posteriores durante el proceso de búsqueda.

$$v_{i+1} = w \cdot v_i + c_l \cdot r_l \cdot (lBest_i - x_i) + c_g \cdot r_g \cdot (gBest - x_i) \quad (1)$$

$$x_i = x_{i-1} + v_i \quad (2)$$

Donde c_l y c_g son los coeficientes de decisión personal y global respectivamente, r_l y r_g son números aleatorios independientes, generados en cada iteración para cada dimensión individual, x_i es la posición de la partícula i, v_i es la velocidad de la partícula i. La siguiente iteración tiene lugar después de que todas las partículas hayan sido movidas. En cada iteración del algoritmo, la posición actual se evalúa como una solución al problema. Si esa posición es mejor que cualquiera que se haya encontrado hasta ahora, entonces las coordenadas se almacenan en el segundo vector llamado $lBest_i$. El objetivo es seguir buscando mejores posiciones x_i y actualizar $lBest_i$. Suponiendo que la solución actual es superior a la aptitud de $lBest_i$, o de $gBest$, entonces la nueva posición reemplazará a $lBest_i$o $gBest$ en consecuencia. El diagrama de flujo para el algoritmo PSO original se muestra en la Figura. 1.

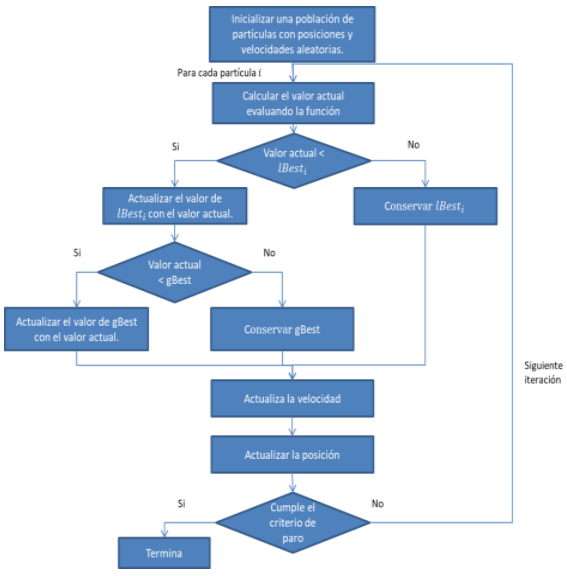


Figura 1. Diagrama general de PSO [5].

3 Variaciones de PSO

3.1 PSO con restricción (Constriction PSO)

Este es un método para equilibrar búsquedas globales y locales conocido como "*Constriction*" [3,6], es similar al método de peso por inercia, pero este método introdujo un nuevo parámetro χ, conocido como el factor de restricción, el cual se deriva de la ecuación (3):

$$\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}, \varphi = c_l + c_g \quad (3)$$

Si $\varphi > 4$, la convergencia será rápida y garantizada. Este factor de restricción se aplica a toda la ecuación de actualización de velocidad como se observa en la ecuación (4):

$$v_{i+1} = \chi \cdot \left(v_i + c_l \cdot r_l \cdot (lBest_i - x_i) + c_g \cdot r_g \cdot (gBest - x_i)\right) \quad (4)$$

3.2 Optimización por cúmulo de partículas usando la ley de coulomb (C-PSO)

C-PSO es una nueva versión de PSO que hace uso de la Ley de Coulomb [7], que considera que dos partículas son dos cargas puntuales m_1 y m_2 que están separadas a una distancia d, entonces la fuerza que experimenta una carga debido a la otra se conoce como la ley de Coulomb y está dada por la ecuación (5):

$$F = k \frac{m_1 m_2}{d^2} \quad (5)$$

Donde:
F es la fuerza magnética
m_1 y m_2 son las masas magnéticas de cada partícula, en este caso se utilizan los costos de las partículas
k es la constante de permisividad eléctrica, la cual fue remplazada por la permeabilidad magnética absoluta μ la cual se define:
μ= μ_r μ_0 Donde μ_r y μ_0 son constantes $\mu = 5000 \cdot 4\pi \times 10^{-7}$quedando la formula final como $F = \mu \frac{m_1 m_2}{d^2}$
Ahora para calcular la distancia entre las dos cargas se hace uso de la distancia entre dos puntos (ecuación (6)) para n valores:

$$d = \sqrt{\sum_{i=1}^{n} (lBestPos_i - gBestPos_i)^2} \quad (6)$$

Donde w se define de la siguiente manera ecuación (7):

$$w = \begin{cases} \mu \frac{gBestCost * lBestCost}{d^2}, & d \leq 0.5 \\ 0, & d > 0.5 \end{cases} \quad (7)$$

4 Ca-PSO

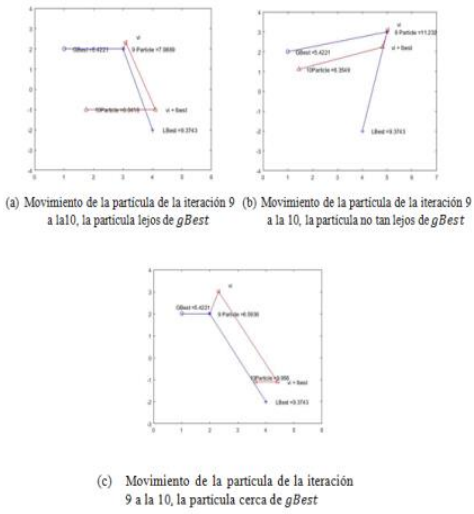
Ca-PSO es una variante de la nueva versión C-PSO, la cual también hace uso de la ley de Coulomb, pero a diferencia de la C-PSO, que considera como las dos cargas puntuales a $lBest_i$y $gBest$, Ca-PSO considera como cargas puntuales a la partícula es decir x_iy $gBest$. Haciendo esta modificación también se hace uso de la distancia entre dos puntos, pero ahora entre la partícula y $gBest$, la distancia se calcula como se muestra en la ecuación (6):

$$d = \sqrt{\sum_{i=1}^{n} (x_i - gBestPos_i)^2} \quad (8)$$

Donde w ecuación (8) se define de la siguiente manera:

$$w = \begin{cases} \mu \frac{xCost_i * lBestCost}{d^2}, & d \leq 0.5 \\ 0, & d > 0.5 \end{cases} \quad (9)$$

El movimiento esquemático de una partícula en el algoritmo Ca-PSO se ilustra en la Figura 2. La figura se muestran tres casos: primero cuando la partícula está lejos de $gBest$, en este caso el movimiento con la velocidad anterior es mínimo, segundo caso cuando la partícula no esta tan lejos de $gBest$, aquí solo hay un ligero movimiento hacia la velocidad anterior y el tercer caso cuando la partícula está cerca de $gBest$, en este último caso se corre el riesgo de alejar a la partícula de $gBest$, por lo tanto es necesario hacer un ajuste a la distancia d.



(a) Movimiento de la partícula de la iteración 9 a la10, la particula lejos de $gBest$

(b) Movimiento de la partícula de la iteración 9 a la 10, la particula no tan lejos de $gBest$

(c) Movimiento de la partícula de la iteración 9 a la 10, la particula cerca de $gBest$

Figura 2. El movimiento esquemático de una partícula

Para llevar a cabo la comparación entre los algoritmos y probar el rendimiento de la nueva versión, primero se realizó la implementación de los cuatro algoritmos: PSO original, PSO con restricciones, C-PSO y Ca-PSO, los cuales ya fueron explicados en secciones anteriores, posteriormente se han seleccionados 10 funciones de optimización (2 unimodales, 5 multimodales y otras) las cuales se muestran a detalle en la siguiente sección, luego se han ejecutado los algoritmos (10000 iteraciones para 50 y 100 dimensiones) utilizando las funciones seleccionadas y por último se han interpretado los resultados para verificar el rendimiento del nuevo algoritmo.

5 Pruebas y resultados

Para probar el rendimiento de los algoritmos explicados en secciones anteriores y la versión PSO original se seleccionaron 10 funciones de optimización las cuales se muestran en la Tabla 1, las cuales fueron seleccionadas de [8].

Tabla 1. Funciones de referencia

Ecuación	Nombre	Límites factibles
$f_1 = \sum_{i=1}^{d} x_i^2$	Sphere	$-100 \le x_i \le 100$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_2 = \sum_{i=1}^{d-1} \lvert 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \rvert$	Rosembrock	$-30 \le x_i \le 30$ $\min(f(x)) = f(1 \dots 1) = 0$
$f_3 = 20\, exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(-0.2\,\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\right) + 20 + exp(1)$	Ackley	$-32 \le x_i \le 32$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_4 = \sum_{i=1}^{d} (x_i^2 - 10\, cos(2\pi x_i) + 10)$	Rastringin	$-5.12 \le x_i \le 5.12$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_5 = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	Schwefel 2.22	$-10 \le x_i \le 10$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_6 = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Griewank	$-600 \le x_i \le 600$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_7 = 0.1 \times \{sin^2(3\pi x_1) + \sum_{i=1}^{d-1}(x_i - 1)^2\,[1 + sin^2(3\pi x_{i+1})] + (x_d - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{d} u(x_i, a, k, m)$ *Donde* $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$ $a = 5, k = 100, m = 4$	General penalizada n.° 2	$-50 \le x_i \le 50$ $\min(f(x)) = f(1 \dots 1) = 0$
$f_8 = \sum_{i=1}^{d} (\lfloor x_i + 0.5 \rfloor)^2$	Step	$-100 \le x_i \le 100$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_9 = \sum_{i=1}^{d} i x_i^4$	Quartic	$-1.28 \le x_i \le 1.28$ $\min(f(x)) = f(0 \dots 0) = 0$
$f_{10} = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{d} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{d} x_i^2}$	Salomon	$-100 \le x_i \le 100$ $\min(f(x)) = f(0 \dots 0) = 0$

Cuatro algoritmos PSO fueron comparados con el interés de demostrar el rendimiento de las técnicas: el algoritmo original PSO, PSO con restricción (PSO Constriction), C-PSO una versión que hace uso de ley de Coulomb y una nueva variante de este último que hemos llamado Ca-PSO. Las configuraciones de los parámetros para realizar las pruebas se muestran en la Tabla 2.

Todos los cúmulos fueron inicializados aleatoriamente en el espacio de búsqueda factible en cada dimensión. Cada algoritmo por cada función se ejecutó 10000 iteraciones por función para 50 y 100 dimensiones.

Estas funciones fueron elegidas por su variedad $f_1 - f_2$, son problemas unimodales simples, $f_3 - f_7$, son altamente problemas multimodales complejos con muchos mínimos locales $f_8 - f_{10}$, son otras.

Los resultados son los promedios de 50 corridas, y se muestran en la Tabla 3 y Tabla 4. Las gráficas de convergencia para funciones seleccionadas se muestran en la Figura 3.

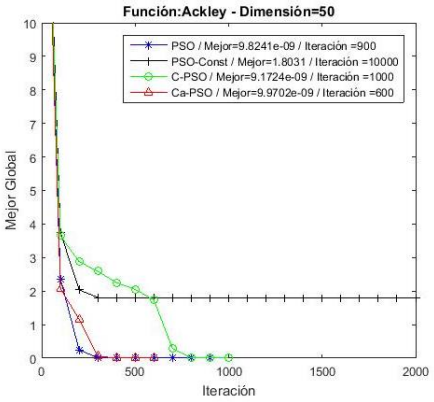
Como el rendimiento del algoritmo original PSO es más pobre que el otros todas las funciones, en las pruebas hemos colocado un valor de inercia w=0.5, de esta forma mejoramos su desempeño para poder ser comparado con el resto de algoritmos. Los resultados se muestran en la Tabla 4 para 50 dimensiones y en la Tabla 5 para 100 dimensiones.

Tabla 2. Parámetros de configuración

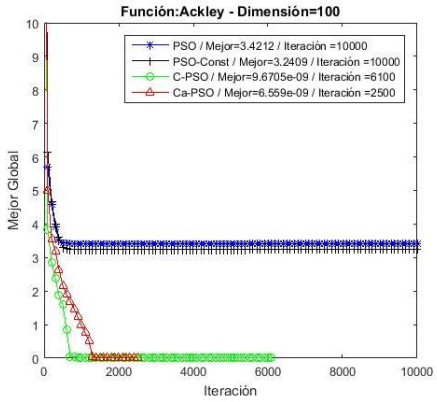
Método	Parámetro		Valor
PSO C-PSO Ca-PSO	Condición de paro	No. Máximo de Iteraciones	10000
		Error	$r < 10e-8$
	Coeficiente de decisión personal		1.8
	Coeficiente de decisión global		1.8
	Coeficiente de inercia (Solo PSO)		0.5
	Población: número de partículas		100
	Espacio de búsqueda		Indicado en la ecuación
Contraint PSO	Condición de paro	No. Máximo de Iteraciones	10000
		Error	$r < 10e-8$
	Coeficiente de decisión personal		2.05
	Coeficiente de decisión global		2.05
	Población		100
	Espacio de búsqueda		Indicado en la ecuación

El algoritmo Ca-PSO mostró un rendimiento significativamente superior sobre el C-PSO en seis de las diez funciones de prueba, mientras que el C-PSO era significativamente superior en una función unimodal y en una multimodal.

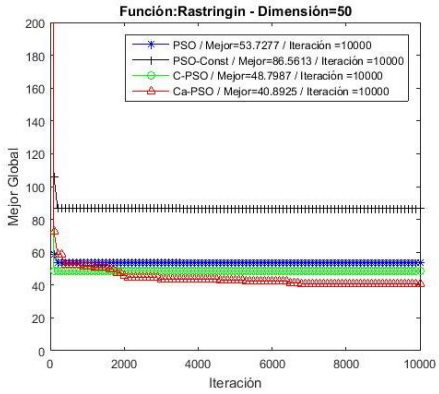
Estos resultados muestran que tanto C-PSO como Ca-PSO presentan un mejor rendimiento que el algoritmo original de PSO. Además, está claro que, en muchos de los casos de prueba, Ca-PSO devuelve mejores resultados que C-PSO. Las comparaciones de rendimiento se hicieron sobre el rendimiento promedio en un número fijo de iteraciones, pero además como se muestra en las gráficas se identifica el número de las iteraciones requeridas. Estas pruebas han sido diseñadas bajo las competencias del CEC [9].



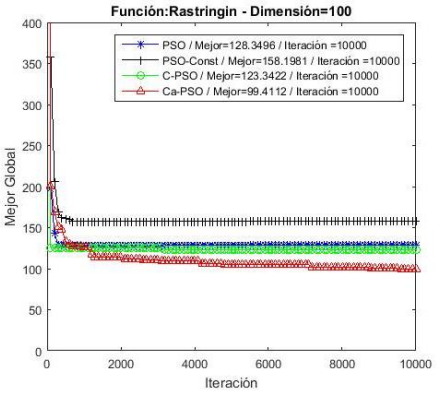
(a) f_3 Ackley para 50 dimensiones



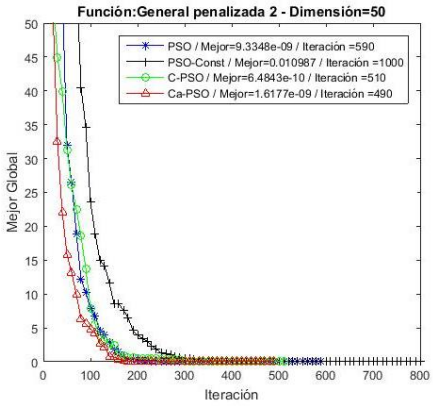
(b) f_3 Ackley para 100 dimensiones



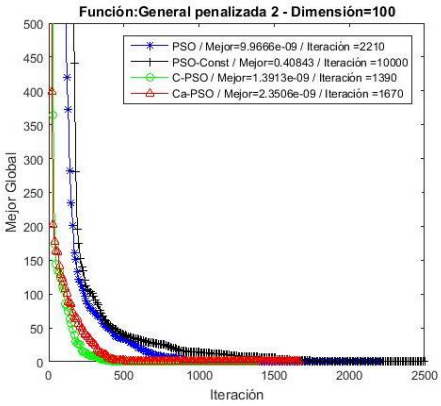
(c) f_4 Rastringin para 50 dimensiones



(d) f_4 Rastringin para 100 dimensiones



(e) f_7 General Penalizada para 50 dimensiones



(f) f_7 General Penalizada para 100 dimensiones

Figura 3. Las gráficas de convergencia para funciones seleccionadas

Tabla 3. Resultados: promedios de 50 corridas para 50 dimensiones, cada algoritmo por cada función se ejecutó 10000 iteraciones.

Función	Método	Peor	Promedio	Mediana	Mejor	Desviación Estándar
F1	PSO	9.9883e-09	9.3710e-09	9.3958e-09	8.3529e-09	4.2088e-10
	Restricción	9.9804e-09	9.3565e-09	9.4295e-09	8.1573e-09	5.1898e-10
	C-PSO	9.9426e-09	4.5991e-09	3.9750e-09	0	3.3400e-09
	Ca-PSO	9.8682e-09	4.0841e-09	2.7144e-09	6.0596e-11	3.5508e-09
F2	PSO	75.7678	17.5038	5.4858	0.0000	23.4588
	Restricción	114.8392	24.1992	15.3460	0.2003	27.5157
	C-PSO	422.1588	145.5279	154.5532	49	101.7765
	Ca-PSO	179.7920	121.2391	119.3800	36.7936	32.9866
F3	PSO	2.4784	0.5079	0.0000	0.0000	0.7589
	Restricción	2.3168	0.6971	0.0000	0.0000	0.7733
	C-PSO	3.6803	0.2988	0.0000	0.0000	0.8008
	Ca-PSO	0.9986e-08	0.9021e-08	0.9619e-08	0.4154e-08	0.1359e-08
F4	PSO	10.0000	3.2000	2.0000	0.0000	2.7030
	Restricción	39.0000	3.1200	1.0000	0.0000	6.1632
	C-PSO	2.0000	0.3000	0.0000	0.0000	0.5051
	Ca-PSO	2.0000	0.1600	0.0000	0.000	0.4219
F5	PSO	0.9999e-08	0.9318e-08	0.9700e-08	0.2174e-08	0.1448e-08
	Restricción	0.9982e-08	0.8013e-08	0.9550e-08	0.0130e-08	0.2869e-08
	C-PSO	10.9799	3.1900	2.3790	1.2664	2.0575
	Ca-PSO	0.3021e-06	0.0725e-06	0.0438e-06	0.0110e-06	0.0730e-06
F6	PSO	0.0441	0.0072	9.8768e-09	7.5187e-09	0.0104
	Restricción	0.0465	0.0052	9.8198e-09	7.3599e-09	0.0088
	C-PSO	0.1041	0.0129	9.8941e-09	0	0.0219
	Ca-PSO	0.0831	0.0168	0.0086	8.9382e-12	0.0237
F7	PSO	1.5975	0.0495	0.0000	0.0000	0.2273
	Restricción	2.5085	0.0683	0.0000	0.0000	0.3551
	C-PSO	2.0011	0.0400	0.0000	0.0000	0.2830
	Ca-PSO	0.0110	0.0002	0.0000	0.0000	0.0016
F8	PSO	23.0000	3.0000	2.0000	0	4.4309
	Restricción	5.0000	0.8000	0	0	1.2617
	C-PSO	1.0000	0.0200	0	0	0.1414
	Ca-PSO	1.0000	0.0200	0	0	0.1414
F9	PSO	0.9967e-08	0.9106e-08	0.9323e-08	0.6529e-08	0.0799e-08

	Restricción	0.9994e-08	0.9007e-08	0.9188e-08	0.6684e-08	0.0813e-08
	C-PSO	0.9981e-08	0.7797e-08	0.8600e-08	0.1022e-08	0.2160e-08
	Ca-PSO	0.9765e-08	0.7173e-08	0.8000e-08	0.0252e-08	0.2547e-08
F10	PSO	1.0999	0.7019	0.6999	0.3999	0.1491
	Restricción	0.7999	0.5999	0.5999	0.3999	0.1107
	C-PSO	1.0999	0.6182	0.5999	0.4999	0.1128
	Ca-PSO	0.8999	0.5879	0.5999	0.3999	0.1733

Tabla 3. Resultados: promedios de 50 corridas para 100 dimensiones, cada algoritmo por cada función se ejecutó 10000 iteraciones.

Función	Método	Peor	Promedio	Mediana	Mejor	Desviación Estándar
F1	PSO	0.0383	0.0023	3.8269e-04	2.5187e-05	0.0067
	Restricción	0.5047	0.0416	0.0072	7.8656e-04	0.1034
	C-PSO	6.2920	0.3818	5.6596e-08	2.3603e-09	1.3811
	Ca-PSO	0.0218	5.2152e-04	1.6490e-06	1.1326e-08	0.0031
F2	PSO	1.0500e+03	360.2895	313.6916	112.7491	161.2426
	Restricción	2.0310e+03	565.1388	457.2851	274.4563	316.7447
	C-PSO	1.1869e+03	569.1286	588.3744	99	272.3528
	Ca-PSO	4.4605e+03	678.7085	548.7721	416.7363	585.9908
F3	PSO	3.1840	2.1289	2.1709	2.2265e-09	0.5215
	Restricción	3.7027	2.3530	2.3715	1.3231	0.4505
	C-PSO	6.4525	0.7635	9.9656e-09	7.1054e-15	1.3145
	Ca-PSO	9.9993e-09	9.3205e-09	9.7390e-09	4.5473e-09	1.1348e-09
F4	PSO	256.0000	102.3000	97.0000	22.0000	53.3438
	Restricción	358.0000	75.0200	56.5000	15.0000	58.2425
	C-PSO	4.0000	1.4600	1.0000	0	1.1988
	Ca-PSO	8.0000	2.9600	2.0000	1.0000	1.7375
F5	PSO	1.0000e-08	0.6673e-08	0.8573e-08	0.0168e-08	0.3555e-08
	Restricción	0.9998e-08	0.5261e-08	0.5579e-08	0.0005e-08	0.3549e-08
	C-PSO	30.2272	6.5546	4.0520	0	6.4845
	Ca-PSO	1.9763	1.6088	1.5880	1.2863	0.1569
F6	PSO	0.0369	0.0041	9.5625e-06	8.1488e-07	0.0083
	Restricción	0.0465	0.0088	2.2371e-04	1.3492e-05	0.0127
	C-PSO	0.2369	0.0366	0.0160	9.0334e-10	0.0500
	Ca-PSO	0.1666	0.0342	0.0172	4.6039e-10	0.0408
F7	PSO	3.6414	0.3386	0.0110	0.000	0.7633
	Restricción	2.4975	0.2349	0.0110	0.0000	0.5763
	C-PSO	0.0440	0.0011	0.0000	0.0000	0.0064
	Ca-PSO	0.0110	0.0011	0.0000	0.0000	0.0033
F8	PSO	251.0000	77.7400	69.0000	27.0000	46.0438
	Restricción	285.0000	58.6000	48.0000	10.0000	49.5683
	C-PSO	6.0000	0.7600	0	0	1.1877
	Ca-PSO	1.0000	0.1600	0	0	0.3703
F9	PSO	0.9999e-08	0.9660e-08	0.9786e-08	0.7833e-08	0.0388e-08
	Restricción	0.1313e-06	0.0185e-06	0.0098e-06	0.0084e-06	0.0233e-06
	C-PSO	0.9995e-08	0.6635e-08	0.7976e-08	0.0107e-08	0.3376e-08
	Ca-PSO	0.9963e-08	0.5562e-08	0.5882e-08	0.0019e-08	0.3644e-08
F10	PSO	2.4999	1.7259	1.6999	1.2999	0.3161
	Restricción	2.3999	1.5484	1.5499	1.0999	0.2655
	C-PSO	2.0003	1.6433	1.6013	1.2999	0.1501
	Ca-PSO	2.0999	1.6183	1.5999	1.1999	0.1769

6 Conclusiones

Esta investigación de base proporcionará un medio de comparación para futuros desarrollos y mejoras que se están utilizando a la vanguardia del campo. Se reitera que esta nueva versión de PSO no debería ser considerada como la "mejor opción posible" para todos los conjuntos de problemas. Existen una gran cantidad de variaciones del algoritmo original que potencialmente ofrece un mejor rendimiento en ciertos problemas, lo que se ofrece aquí es un estándar para la comparación de algoritmos.

El algoritmo Ca-PSO mostró un rendimiento significativamente superior sobre el C-PSO en seis de las diez funciones de prueba, mientras que el C-PSO era significativamente superior en una función unimodal y en una multimodal. Además, los resultados muestran que tanto C-PSO como Ca-PSO presentan un mejor rendimiento que el algoritmo original de PSO.

Referencias

1. Xiangwei Z., Hong L. A hybrid vertical mutation and self-adaptation based MOPSO. Computers and Mathematics with Applications. 2009, 57, 2030-2038.
2. Yuxin Z., Wei Z., Haitao Z. A modified particle swarm optimization via particle visual modeling analysis. Computers and Mathematics with Applications 2009, 57, 2020-2029.
3. Rini D.P., Shamsuddin S. M., Yuhaniz S. Particle Swarm Opti-mization: Technique, System and Challenges. International Journal of Computer Applica-tions. 2011, 14(1), 19-27.
4. Bansal J. C., Singh P.K., Saraswat M., Verma A., Jadon S. S. and Abraham A. Inertia Weight Strategies Particle Swarm Optimization. Third World Congress on Nature and Biologically Inspired Computing. 2011, 633-640.
5. Dada E., Ikhwan R. E. PDPSO: The fusion of primal-dual interior point method and particle swarm optimization algorithm. Malaysian Journal of Computer Science. 2018, 31(1), 17-34.
6. Nápoles G, Grau I, and Bello R. Constricted Particle Swarm Optimization based Algorithm for Global Optimization. 2012, 46, 5-11.
7. Solano-Monje, R., Meléndez-Acosta N. J. García-García C. and Ríos-Figueroa V. C-PSO: Optimización por Cúmulo de Partículas incrustando la Ley de Coulomb. Research in Computing Science. 2018.
8. Viveros-Jiménez F., Mezura-Montes E. and Gelbukh A. Empirical analysis of a micro-evolutionary algorithm for numerical optimization. International Journal of Physical Sciences. 2012, 7(8), 1235-1258.
9. Awad N. H., Ali M. Z., Suganthan P. N., Liang J. J. and Qu B. Y. Problem Definitions and Evaluation Criterial for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical Report, 2016.

Acerca de los autores

La **M.I.A. Nayeli Joaquinita Meléndez Acosta** obtuvo el grado de Maestra en Inteligencia Artificial en 2013 por la Universidad Veracruzana y es Ingeniero en Computación egresada de la Universidad Autónoma de Tlaxcala en 2006 y además cuanta con un diplomado en Competencias Pedagógicas.
Actualmente es Profesora-Investigadora de tiempo completo en la Universidad del Istmo campus Ixtepec. Previamente fue profesora en varias universidades como en la Universidad Autónoma de Tlaxcala y la Universidad Politécnica de Tlaxcala región Poniente, entre otras.
Sus líneas de investigación son Aplicaciones Móviles, Aplicaciones Educativas, Algoritmos Bioinspirados y Procesamiento de Imágenes. En sus pasatiempos le gusta programar y leer. Es sencilla, paciente y bondadosa.

M.C. Ricardo Solano Monje es Licenciado en Ciencias de la Computación egresado de la Benemérita Universidad Autónoma de Puebla (2001). Obtuvo el grado de Maestro en Ciencias de la Computacionales en 2002 del Instituto Nacional de Astrofísica, Óptica y Electrónica.
Fue entrenador de un equipo universitario finalista en el mundial de programación de la ACM-ICPC 2006 patrocinado por IBM y ACM, representando a México y Centro América en San Antonio, Texas (2006).
Cuenta con una Maestría en Docencia por parte del Centro Veracruzano de Investigación y Posgrado (2018).
Sus líneas de investigación incluyen Machine Learning, Visión Computacional y Aplicaciones Móviles. Se ha desempeñado como académico desde 2003 en diferentes universidades y tecnológicos del país.
Actualmente es Jefe del Departamento de Ciencias Básicas del Instituto Tecnológico Superior de Ciudad Serdán (2018).

Cosijopii García García nació en Sola de Vega, Oaxaca, México en 1994. Obtuvo el título de licenciado en informática por la Universidad del Istmo, Campus Ixtepec Oaxaca. Actualmente estudia la maestría en ciencias en el área de ciencias computacionales en el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE). Sus intereses son el área de Optimización utilizando Algoritmos Bioinspirados y Algoritmos genéticos, así como el procesamiento digital de imágenes con aplicaciones en el rostro humano.

El **Dr. Homero Vladimir Ríos Figueroa** obtuvo su doctorado en Computación e Inteligencia Artificial por la Universidad de Sussex, Inglaterra en 1994. El grado de Maestro en Ciencias de la Computación por la UNAM en 1989 y la Licenciatura en Matemáticas en la Facultad de Ciencias, UNAM, 1987.
Dentro de su experiencia profesional se ha desempeñado como consultor y administrador de proyectos de TI en la iniciativa privada y en el gobierno federal y estatal por más de 24 años. Por otra parte, se ha desempeñado como académico desde 1986, en las especialidades de matemáticas, ciencias de la computación e inteligencia artificial.
Su línea de investigación es la visión artificial y el aprendizaje para el desarrollo de nuevas formas de interacción humano-computadora.
Desde el año 2000 es académico de carrera titular C en la Universidad Veracruzana y es parte del Centro de Investigación en Inteligencia Artificial.