Una Comparativa del Algoritmo Particle Swarm Optimization con Restricciones entre los Métodos de Factor de Constricción y Neighborhood con Coeficiente de Inercia Dinámico

Miguel Augusto Azar, Fabiola Patricia Paz, Analía Herrera Cognetta

Facultad de Ingeniería - Universidad Nacional de Jujuy
auazar@live.com, fabyppaz@gmail.com, anihco@yahoo.com.ar

Resumen. Los diferentes enfoques de optimización por enjambre de partículas ofrecen variantes que ayudan a mejorar la performance del algoritmo en una serie de aspectos tales como robustez, calidad, confiabilidad de resultados en óptimos globales, entre otros. La hibridación entre algunas de estas variantes no necesariamente garantiza un resultado exitoso en cuanto a los aspectos mencionados. El presente trabajo se concentra en estudiar los problemas de optimización con restricciones mediante la comparación entre el clásico algoritmo PSO con factor de constricción y un híbrido con topología de vecindad y factor de inercia dinámico.

Palabras Clave. Optimización, metaheurísticas, constriction, inteligencia de enjambre, particle swarm optimization, PSO, neighborhood, factor de inercia dinámico, constraints.

1 Introducción

La resolución de problemas de optimización en los cuales se involucra una función objetivo dentro de un espacio multidimensional y sujeto a una serie de restricciones puede tener diferentes soluciones dentro del campo de la investigación operativa tradicional. Tales problemas también pueden resolverse mediante heurísticas en el dominio de la inteligencia artificial y más precisamente en el ámbito de los algoritmos evolutivos bioinspirados. Estos algoritmos, a diferencia de la programación lineal, encuentran resultados que son en todos los casos aproximados dado que controlan un número finito de partículas intervinientes y son sus movimientos aleatorios los que evalúan una posible solución en el espacio de búsqueda n-dimensional.

El algoritmo de optimización por enjambre de partículas (PSO) provee un conjunto de variantes y enfoques que permiten obtener una solución óptima confiable. Estos enfoques posibilitan entre otros objetivos, evitar que las partículas se concentren en óptimos locales, asegurar una amplia exploración del espacio de búsqueda, evitar las búsquedas fuera de la dimensión máxima de la función objetivo y mejorar la convergencia.

El presente artículo estudia una de esas variantes, PSO con factor de constricción versus una hibridación del PSO con vecindad y factor de inercia dinámico. En ambos

casos se emplea una técnica de manejo de restricciones basado en las violaciones de los recursos. El objetivo buscado es analizar si la hibridación de técnicas que mejoran ciertas características del algoritmo logra una mejora global del mismo o se produce una baja de su performance general.

2 El algoritmo PSO con restricciones y sus variantes

El algoritmo Particle Swarm Optimization (PSO) fue propuesto por Kennedy y Eberhart [1] inspirado en la simulación del comportamiento social de bandada de aves. Es un enfoque de búsqueda basado en una población de la cual cada individuo vuela en el espacio de búsqueda con una velocidad que es ajustada de acuerdo a su propia experiencia de vuelo y la experiencia del vuelo de sus compañeros. A diferencia de otros algoritmos evolutivos, en PSO todas las partículas que forman parte de la población se mantienen durante todo el proceso de búsqueda; siendo la velocidad y la posición de las partículas las que se actualizan de acuerdo a la mejor posición encontrada y a las de la otras partículas. Además este es el único algoritmo evolutivo que no implementa la supervivencia del más apto [2]. Para este algoritmo constantemente surgen nuevas versiones respecto del original con el fin de mejorar el rendimiento del proceso de optimización evitando las convergencias prematuras. Entre las variantes más conocidas se pueden mencionar: factor de constricción, vecindad, factor de inercia dinámico, entre otros.

Los problemas de optimización restringida se encuentran en numerosas y diversas aplicaciones: problemas de optimización estructural, diseño de ingeniería, diseño VLSI, de economía, de asignación y ubicación son sólo algunos de los campos científicos en el que se hallan con frecuencia optimización con restricciones [3].

Originalmente, esta heurística no contempla un mecanismo para resolver problemas de optimización restringida, pero existen diversos mecanismos que han sido implementados en algoritmos evolutivos, así como en esta técnica [4]. Uno de los mecanismos más utilizado debido a su sencillez de implementación, es la técnica propuesta por Deb [5] que consiste en reglas basadas en la factibilidad de las partículas durante el ciclo de búsqueda; esto es:

1. Entre 2 soluciones factibles, la solución con un mejor valor de la función objetivo gana.
2. Si una solución es factible y la otra es no factible, la solución factible gana.
3. Si ambas soluciones son no factibles, aquella con el valor más bajo en la suma de la violación de restricción gana.

En este trabajo, se adapta dicho mecanismo para los problemas de producción, en donde las empresas disponen de recursos limitados para su producción. Se modifica el mecanismo de Deb, reemplazando la tercera regla por [6]:

3. Si ambas soluciones son no factibles, se considera aquella con el menor número de restricciones violadas. En el caso de que ambas partículas violen la misma cantidad de restricciones se elige aquella con menor valor en la suma de los porcentajes de violaciones con respecto a los recursos disponibles.

2.1 PSO con factor de inercia dinámico

En el algoritmo clásico de optimización por enjambres se incorpora un parámetro denominado factor de inercia *w*. El objetivo de *w* es equilibrar la búsqueda global y la búsqueda local, encontrando áreas más prometedoras. Las evaluaciones que han sido realizadas con este parámetro, han demostrado un impacto efectivo y muy significativo en los resultados de búsqueda [7]. Esta variable modifica la fórmula de la actualización de la velocidad quedando de la siguiente manera:

$$\overrightarrow{vel_{(i,d)}} = \boldsymbol{w} * \overrightarrow{vel_{(i,d)}} + r_1 * p_1 * \left(\overrightarrow{\boldsymbol{pbest}_{(i,d)}} - \overrightarrow{pos_{(i,d)}} \right) + r_2 * p_2 * \left(\overrightarrow{\boldsymbol{gbest}_{d}} - \overrightarrow{pos_{(i,d)}} \right) \quad (1)$$

Donde $\overrightarrow{vel_{(i,d)}}$ es el valor de la velocidad de la partícula i en la dimensión d, p_1 es el factor de aprendizaje cognitivo, p_2 es el factor de aprendizaje social poblacional, r_1 *y* r_2 son valores aleatorios uniformemente distribuidos en el rango [0, 1], $\overrightarrow{pos_{(i,d)}}$ es la posición actual de la partícula i en la dimensión d, $\boldsymbol{pbest_{(i,d)}}$ es el valor en la dimensión d de la partícula con el mejor valor objetivo encontrado por la partícula i, $\overrightarrow{\boldsymbol{gbest}_{d}}$ es el valor en la dimensión d del individuo del cúmulo (swarm) que encontró el mejor valor de la función objetivo.

El parámetro *w* puede ser constante o linealmente decreciente en el tiempo. En este último caso, se trata de un factor de inercia dinámico y es reducido linealmente utilizando 2 valores límites: *w_inicial y w_final*. Luego, mediante la ecuación (2), el factor de inercia es actualizado en cada ciclo del proceso, hasta el final de las iteraciones (*ciclo_max*).La ecuación de actualización que se utiliza es:

$$\boldsymbol{w} = \frac{(ciclo\_max - ciclo) * (w_{inicial} - w_{final})}{ciclo\_max} + w_{final} \quad (2)$$

2.2 PSO con factor de constricción

Este parámetro propuesto por Clerc y Kennedy [8], introduce una variable X estática a la ecuación de la velocidad. Este factor se usa para controlar y constreñir las velocidades de las partículas, durante el proceso de búsqueda. La ecuación propuesta es:

$$\overrightarrow{vel_{(i,d)}} = \boldsymbol{X} * \left[\overrightarrow{vel_{(i,d)}} + r_1 * p_1 * \left(\overrightarrow{\boldsymbol{pbest}_{(i,d)}} - \overrightarrow{pos_{(i,d)}} \right) + r_2 * p_2 * \left(\overrightarrow{\boldsymbol{gbest}_{d}} - \overrightarrow{pos_{(i,d)}} \right) \right] \quad (3)$$

Donde al igual que el factor de inercia *w*, el papel más importante que juega este parámetro, es el de asegurar la convergencia del algoritmo y controlar el equilibrio deseado entre el comportamiento de exploración y explotación dentro del espacio de búsqueda. Para asegurar la convergencia, las variables que definen el factor de constricción deben cumplir con los siguientes principios:

$$X = \frac{2k}{\left| 2 - \emptyset - \sqrt{\emptyset(\emptyset - 4)} \right|} \quad (4)$$

$$\emptyset = \emptyset_1 + \emptyset_2$$

$$\emptyset \geq 4$$

$$\emptyset_1 = p_1 * r_1$$

$$\emptyset_2 = p_2 * r_2$$

$k \in [0,1]$

Para valores de $\phi < 4$ el enjambre se acerca lentamente en espiral a la mejor posición encontrada pero sin garantía de convergencia mientras que para valores de $\phi > 4$ la convergencia es rápida y segura [8].

2.3 PSO con neighborhood

A menudo el PSO demuestra una convergencia más rápida en los primeros ciclos, es decir, en la primera fase de la búsqueda, y luego se ralentiza o incluso se detiene cuando el número de generaciones va aumentado. Una vez que el algoritmo se ralentiza, es difícil lograr alcanzar mejores soluciones. Para evitar la terminación o la convergencia en un mínimo local, se introduce una estructura social para determinar la comunicación entre las partículas, la cual se denomina vecindad. De acuerdo a la forma de comunicación entre las partículas que se establezca, se define una topología de vecindad. La idea básica es impulsar esas partículas a una mejor estrategia de comunicación y de esta manera lograr explorar mejor el espacio [9].

En PSO, cada partícula dentro del enjambre pertenece a una vecindad específica de comunicación. Se llevaron a cabo varios estudios con el fin de determinar si la topología de vecindad podría afectar a la convergencia del algoritmo. Estos estudios se basaron en propuestas teóricas e implementaciones de topologías de vecindad comúnmente utilizados por PSO [10]. En esos estudios, algunas topologías de vecindad han obtenido mejores resultados que otros. La forma de agrupamiento de las partículas, puede mejorar el rendimiento del algoritmo. Entre las diferentes propuestas de topologías de vecindad están:

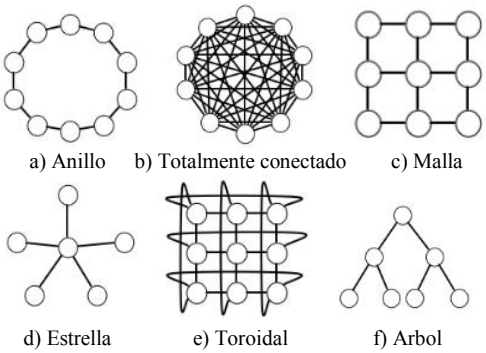

Figura 1: Topologías de vecindades utilizadas en algoritmos de enjambre.

3 Evaluación de los algoritmos y resultados

Con el propósito de lograr una comparación objetiva todas las ejecuciones se configuraron con los mismos valores de parámetros para ambas versiones del PSO, excepto el peso *w* inercia que es dinámico. Los parámetros utilizados fueron los siguientes: tamaño de población (cantidad de partículas) 20; máximo número de

iteraciones 500; factor de aprendizaje cognitivo y social 1.49445; factor de inercia inicial y final de 0.9 y 0.4 respectivamente; factor de constricción de X= 0.7298; cantidad de vecindades 3 de 10 partículas cada una; topología de vecindad estrella. En esta topología implementada todas las partículas de todo el enjambre dirige su vuelo hacia una partícula (la partícula central), y la partícula central dirige su vuelo hacia la mejor partícula de la vecindad. Esta topología debe evitar la convergencia prematura en óptimos locales.

Ambos algoritmos fueron evaluados mediante las funciones de referencia (Benchmark Functions) g04, g08, g09, g15 y g24 debido a que estas son las únicas funciones que contemplan la incorporación de restricciones con disponibilidad. Por otro lado, se evaluó la performance de todos los algoritmos para 500 ciclos dado que en todos los casos se encontró el valor óptimo dentro de ese rango.

Tabla 1: Valores obtenidos del algoritmo PSO con factor de constricción.

FUNCIONES DE PRUEBA - ALGORITMO PSO con FACTOR DE CONSTRICCION-					
G(X)	OPTIMO	MEJOR	PEOR	PROMEDIO	DESV. ESTAND
g04	-30665,539	-30665,0029	-28183,7113	-29927,895	671,561666
g06	-6961,813	**-6961,81388**	-747,977358	-6961,81388	2229,58258
g08	-0,095	**-0,09582504**	-0,02726221	-0,09582504	0,02081586
g09	680,63	680,720704	718,581609	681,336206	8,33319373
g15	961,715	**961,668777**	972,610631	967,356535	3,194949
g24	-5,50801327	**-5,50801327**	-4,03470362	-5,50801327	0,59817649

En las Tablas 1 y 2 se visualizan los resultados estadísticos obtenidos en las 20 ejecuciones para 6 funciones de benchmark con restricciones g04, g06, g08, g09, g15 y g24. Cabe destacar, que el algoritmo PSO con factor de inercia dinámico y vecindad en sus 20 ejecuciones para cada función evaluada encontró soluciones factibles; no así el algoritmo PSO con factor de constricción.

Tabla 2: Valores obtenidos del desempeño del algoritmo PSO con factor de inercia dinámico y vecindad.

FUNCIONES DE PRUEBA - ALGORITMO PSO con FACTOR INERCIA + VECINDAD -					
G(X)	OPTIMO	MEJOR	PEOR	PROMEDIO	DESV. ESTAND
g04	-30665,539	-30665,0029	-30665,00286	-30665,0029	9,24615E-06
g06	-6961,813	**-6961,81388**	-6961,813876	-6961,813876	7,56884E-10
g08	-0,095	**-0,09582504**	-0,095825041	-0,095825041	2,79E-17
g09	680,63	680,65939	685,3846961	681,1705082	1,216223845
g15	961,715	**961,5659**	972,2179663	967,0694139	3,747513459
g24	-5,50801327	**-5,50801327**	-5,508013272	-5,508013272	4,29667E-12

En la Figura 2 puede observarse la gráfica de la convergencia en función de los ciclos para una partícula evaluada en los algoritmos comparados para la función benchmark g04. Se puede observar que ambas versiones para esta función obtienen iguales resultados según Tabla 1 y Tabla 2 y en cuanto a su convergencia también son bastantes similares. En este caso se podría adoptar cualquiera de las versiones, sin embargo, el algoritmo con constricción sería el más adecuado debido a su sencillez en cuanto a programación y por su bajo costo computacional.

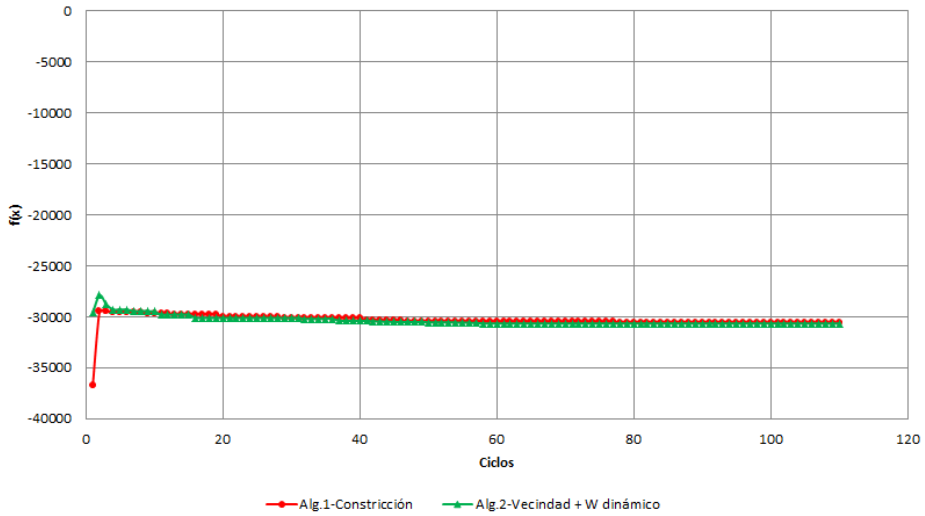

Figura 2: Comparación de algoritmos de constricción y vecindad con factor de inercia dinámico para la función g04 de benchmark.

En la Figura 3 se ilustra la misma comparativa pero evaluada mediante la función benchmark g06. El algoritmo de constricción converge al óptimo más lentamente en comparación con el algoritmo basado en vecindad y coeficiente de inercia dinámico. Es importante resaltar que para esta función, el algoritmo con vecindad se diferencia bastante del de constricción por su robustez como se visualiza en las Tabla 1 y Tabla 2, ya que éste desde el inicio de los ciclos está bastante cerca del óptimo.

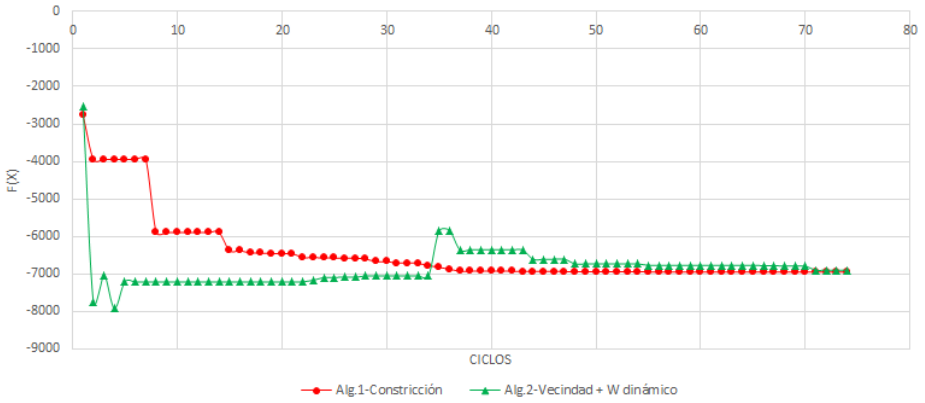

Figura 3: Comparación de algoritmos de constricción y vecindad con factor de inercia dinámico para la función g06 de benchmark.

La función de prueba evaluada, g08, se observa en la Figura 4. En este caso las partículas del algoritmo con vecindad se desplazan muy cerca del óptimo desde los primeros ciclos desde aproximadamente el ciclo 6 en adelante. Es por ello que se justifica un valor de desviación estándar de $2{,}7937*10^{-17}$ según Tabla 2, un resultado

muy superior con respecto al algoritmo de constricción. De todas las evaluaciones esta es la que ofrece la mejor robustez.

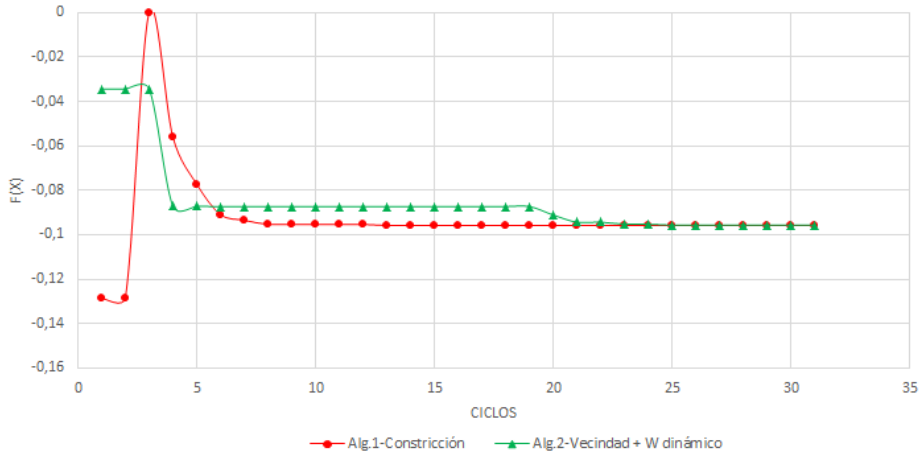

Figura 4: Comparación de algoritmos de constricción y vecindad con factor de inercia dinámico para la función g08 de benchmark.

Los algoritmos comparados mostrados la Figura 5 se evaluaron mediante la función g09 en donde ambos no alcanzan el óptimo conocido, pero se observa que el algoritmo con vecindad tiene un mejor desempeño que el de constricción, debido a que los resultados indican una calidad y robustez superior (ver Tabla 1 y Tabla 2).

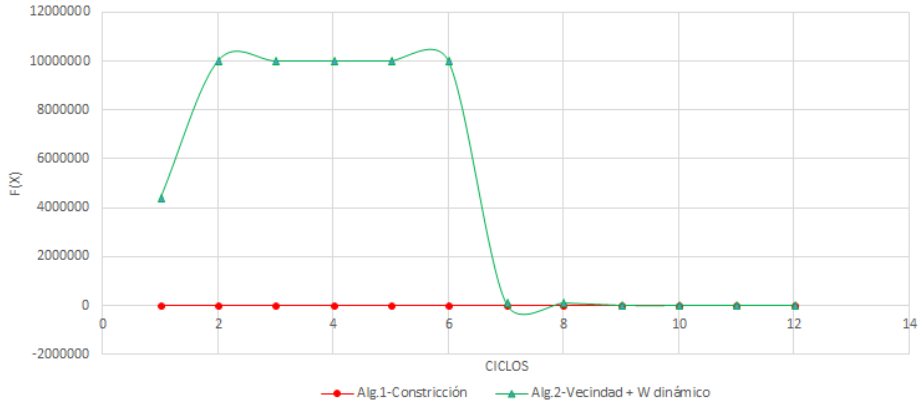

Figura 5: Comparación de algoritmos de constricción y vecindad con factor de inercia dinámico para la función g09 de benchmark.

En la Figura 6 se analizan ambos algoritmos evaluados para la función g15. Se observa encuentran una mejor solución que el óptimo conocido por otros algoritmos de la literatura especializada. En la gráfica al inicio de los ciclos de búsqueda el algoritmo de constricción se encuentra más alejado del óptimo con respecto al algoritmo con vecindad. Sin embargo obtiene una mejor robustez (Tabla 1), pero su

calidad sigue siendo inferior al algoritmo con vecindad y peso de inercia dinámico (Tabla 2). En esta función, el espacio de soluciones es bastante reducido, por lo que se hace más difícil obtener resultados factibles. No obstante, estas dos propuestas hallaron no solo soluciones factibles en las 20 corridas, sino también alcanzaron un muy buen resultado.

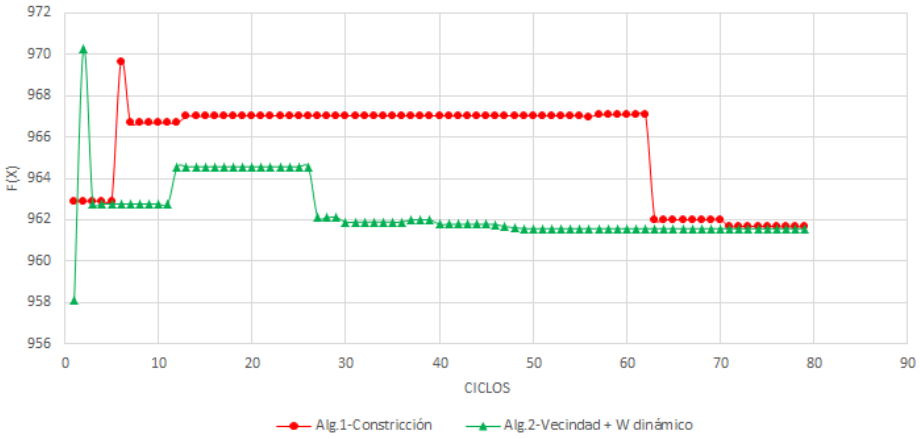

Figura 6: Comparación de algoritmos de constricción y vecindad con factor de inercia dinámico para la función g15 de benchmark.

La última función evaluada y comparada (g24) se encuentra en la Figura 7 en donde ambos algoritmos alcanzan el óptimo. Se observa en la gráfica la convergencia al óptimo y claramente justifica los resultados de la correspondiente desviación estándar observados en la Tabla 1 y Tabla 2, es decir que el algoritmo con vecindad tiene una mejor convergencia al óptimo.

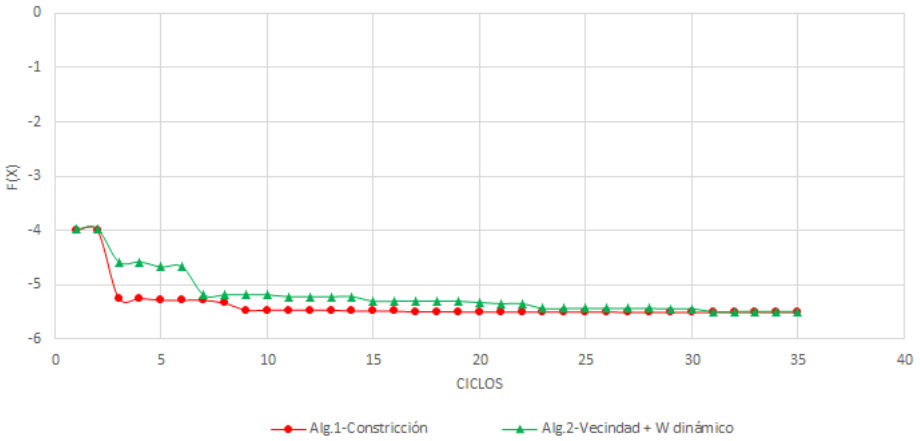

Figura 7: Comparación de algoritmos de constricción y vecindad con factor de inercia dinámico para la función g24 de benchmark.

4 Conclusiones

Los resultados de las evaluaciones presentados en las Tablas 1 y 2, fueron obtenidos con el fin de comparar y seleccionar aquel algoritmo que sea el más adecuado para determinadas funciones. El algoritmo PSO con el parámetro del factor de constricción al igual que el PSO con coeficiente de inercia dinámico y vecindad alcanza el óptimo para las funciones g06, g08, g15 y g15 y se estancan en óptimos locales muy cercanos al óptimo global para las funciones g04 y g09. La calidad de los algoritmos para estas funciones son bastantes similares, pero a su vez se diferencian bastante por su robustez ya que el algoritmo con coeficiente de inercia dinámico y vecindad es superior al de factor de constricción.

Otro aspecto importante que los diferencia en cuanto a las funciones de prueba es la factibilidad de los resultados siendo el algoritmo con factor de inercia y vecindad factible en todas sus ejecuciones.

Las observaciones y el análisis efectuado entre ambos algoritmos permiten afirmar que la hibridación de las técnicas de vecindad con factor de inercia dinámico no siempre incrementa aquellas mejoras que por separado si lo logran.

Es por ello que conviene analizar el posible diseño de hiperheurísticas que permitan la complementación de las mejores características de cada técnica.

5 Trabajos a futuro

Debido a que ambos algoritmos incorporan un mecanismo de manejo de restricciones diferente a los clásicos, es posible plantear modificaciones sobre la técnica citada y observar si se producen mejoras en cuanto a robustez y calidad en las 6 funciones analizadas.

Otro aspecto a considerar es la evaluación y comparación mediante funciones de referencia más actualizadas, esto es, el artículo presente utiliza un conjunto de pruebas de optimización (CEC 2006 Problemas Benchmark [11]) en el cual la dimensión de la función está comprendida entre 2 y 20 lo cual es considerado como un valor de dimensión bajo. Sin embargo, éstas son las funciones que se utilizan habitualmente en las investigaciones referidas al algoritmo PSO para medir su desempeño. En este contexto se pretende en futuras investigaciones estudiar y evaluar el algoritmo con las funciones de referencia restantes además de las funciones de prueba CEC 2010 y 2013[12] [13].

6 Referencias

[1] Kennedy J., Eberhart R. C., Shi Y.: Swarm Intelligence. Morgan Kaufmann, San Francisco, CA (1995)

[2] Eberhart, R. C., Shi, Y. H.: Comparison between genetic algorithms and particle swarm optimization. Annual Conference on Evolutionary Programming, San Diego (1998).

[3] Parsopoulos K.E., Vrahatis M. N.: Particle Swarm Optimization Method for Constrained Optimization Problems (2002).

[4] Coello-Coello C.A.: “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms”: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering, V.191, No. 11-12:1245–1287 (2002).
[5] Deb K.: “An efficient constraint handling method for genetic algorithms”. Computer Methods in Applied Mechanics and Engineering, v.186 (2/4):311–338 (2000).
[6] Paz F.P., Azar M.A., Herrera Cognetta A., Pérez Otero N.M., Una alternativa para el mecanismo de manejo de restricciones en algoritmos PSO, in Proceedings Second Argentine Conference on Human-Computer Interaction, Telecomunications, Informatics and Scientific Information HCITISI, Córdoba, Argentina, ISBN 978.88.96.471.25.8, Noviembre 21-22 (2013).
[7] Shi, Y. y Eberhart, R. “A modified Particle Swarm Optimizer”. En Proceeding of the 1998 IEEE World Congress on Computational Intelligence, páginas 69–73, Piscataway, NJ (1998).
[8] Clerc, M., The swarm and the queen: towards a deterministic and adaptive particle swarm optimization., Memorias del 1999 Congress on Evolutionary Computation (CEC 99) (1999).
[9] Ajith A, Hongbo Liu, Tae-Gyu Chang. Variable Neighborhood Particle Swarm Optimization Algorithm (2006).
[10] Medina A.J.R., Toscano G.P., Torres R.J.G. A Comparative Study of Neighborhood Topologies for Particle Swarm Optimizers (2009).
[11] Liang J. J., Runarsson T. P., Montes E.M, Clerc M., Suganthan P.N., Coello Coello C.A., Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization (2006).
[12] Suganthan, Lǐ, Yáng, Weise. “Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization” (2010).
[13] Li, Tang, Omidva, Yang, Qin. “Benchmark Functions for the CEC’2013 Special Session and Competition on Large-Scale Global Optimization” (2013).