Lámpsakos
ISSN: 2145-4086
Universidad Católica Luis Amigó

Jiménez Builes, Jovani Alberto; Arango Sanchez, Rafael Esteban; Jiménez Pinzón, Leidy Diana
Métodos de búsqueda usando los algoritmos de enjambre de partículas y genético
Lámpsakos, núm. 16, 2016, Julio-Diciembre, pp. 52-60
Universidad Católica Luis Amigó

DOI: https://doi.org/10.21501/21454086.1901

Disponible en: https://www.redalyc.org/articulo.oa?id=613964501005

Lámpsakos | N°. 16 | pp. 52-60 | julio-diciembre | 2016 | ISSN: 2145-4086 | Medellín–Colombia

Métodos de búsqueda usando los algoritmos de enjambre de partículas y genético

Search methods using the algorithms: particle swarm and genetic

Jovani Alberto Jiménez Builes, PhD.
Universidad Nacional de Colombia
Medellín, Colombia
jajimen1@unal.edu.co

Rafael Esteban Arango Sanchez, MSc.
Universidad Nacional de Colombia
Medellín, Colombia
raearangosa@unal.edu.co

Leidy Diana Jiménez Pinzón, MSc.
Universidad Nacional de Colombia
Medellín, Colombia
ldjimenezp@unal.edu.co

(Recibido el 05-04-2016. Aprobado el 16-11-2016)

Estilo de Citación de Artículo:
J, Jimenez-Builes, R. Arango-Sanchez, L. Jimenez-Pinzón, "Métodos de búsqueda usando los algoritmos de enjambre de partículas y genético", Lámpsakos, no. 16, pp 52-60, 2016
DOI: http://dx.doi.org/10.21501/21454086.1901

Resumen. La optimización de un problema para la toma de decisiones es una tarea frecuente en la vida. Existen en la literatura diferentes técnicas determinísticas y heurísticas, las cuales son utilizadas de acuerdo a las condiciones o restricciones del problema para encontrar la mejor solución. Sin embargo, para lograr una exploración en un espacio de búsqueda de posibles soluciones, se aplican los métodos metaheurísticos, los cuales se basan en el comportamiento de poblaciones y trayectorias permitiendo encontrar soluciones casi óptimas. En este artículo se presenta el estudio de dos métodos metaheurísticos basados en poblaciones, el algoritmo enjambre de partículas y el algoritmo genético, implementados para dar solución a problemas cuyo objetivo es optimizar buscando siempre el menor valor. Para llevar a cabo este estudio, se realiza una aplicación en lenguaje de programación JAVA que contiene la implementación de los dos algoritmos a ser evaluados sobre funciones no lineales. El resultado de este trabajo se muestra mediante la comparación en la precisión al obtener la solución óptima de los métodos, mostrando la evolución de los resultados de forma gráfica hasta llegar a la solución. Al finalizar se concluye que el enjambre de partículas tiene un mejor comportamiento que el algoritmo genético.

Palabras clave: Algoritmo enjambre de partículas, algoritmo genético, comparación, simulación optimización.

Abstract. This article presents the study of two metaheuristic methods based in populations, the comparison between two search algorithms, the particle swarm algorithm (PSO) and genetic algorithm (GA) for solving problems whose objective is optimize always looking for the lowest value. To carry out this study, we made an application in JAVA programming language that contains the implementation of the two algorithms to be used for evaluation of nonlinear functions. The result of this work is shown by comparing the accuracy to obtain the optimal solution of the methods listed above, showing the evolution of the results in graphical form to reach the solution. From this study it can be concluded that the particle swarm optimization has a better performance than genetic algorithm.

Keywords: Particle swarm algorithm, genetic algorithm, simulation, optimization, comparison.

1. INTRODUCCIÓN

En un problema de decisión que normalmente se presentan en cualquier ámbito de la vida cotidiana, se parte de un conjunto de recursos específicos que condicionan la elección de la mejor solución. Esto mismo, de forma matemática, se expresa como la optimización de una función objetivo, la cual está sujeta a una serie de restricciones [1].

Algunos tipos de problemas de optimización por sus características, pueden resolverse haciendo uso de técnicas determinísticas que permiten encontrar su solución óptima. Por ejemplo los problemas o funciones que son lineales, pueden ser resueltos mediante el método simplex, el cual es un procedimiento iterativo que maximiza o minimiza una función bajo unas restricciones permitiendo que se mejore la solución en cada uno de los pasos, hasta llegar a una solución que no es posible seguir optimizando [1], [2].

Sin embargo, la mayoría de los problemas reales no pueden ser resueltos con los algoritmos determinísticos, algunos porque no tienen las características para poder usar el método y otros porque el tiempo que necesitan para hallar la solución resulta ser extenso, es decir, el tiempo se hace mayor de acuerdo al tamaño del problema y por lo tanto se deben buscar otras técnicas para encontrar la solución [3]. Para ello, se hace uso de los métodos heurísticos, los cuales resuelven problemas de búsqueda y optimización encontrando una solución que aunque no necesariamente es la óptima, es una buena solución y en un tiempo razonable [4].

Existen métodos iterativos a los cuales se les aplican procesos estocásticos ya que la búsqueda debe realizarse de manera aleatoria, algunos sobre trayectorias y otros sobre poblaciones. Estos métodos son denominados metaheurísticos dado que son producto de la combinación de métodos heurísticos con el objetivo de lograr una mejor exploración en el espacio de búsqueda de forma eficiente y efectiva obteniendo soluciones casi óptimas [5], [6].

Los métodos metaheurísticos basados en trayectorias parten desde un punto y a medida que se realiza la exploración del vecindario, se actualiza la solución si es mejor o no a la anterior obteniendo al final del proceso la mejor que encontró [5], [6].

Fig. 1. Taxonomía de la computación evolutiva [8]

Los métodos metaheurísticos basados en poblaciones consisten en la iteración de una población de soluciones o individuos y el paso de los mismos en cada una de las iteraciones describiendo de esta manera una cadena clara de búsqueda, con lo cual se encuentra una solución casi óptima a partir de la evolución de un conjunto de puntos en el espacio [5], [6], [7].

De este último grupo de métodos hacen parte los algoritmos evolutivos como por ejemplo el algoritmo genético y los algoritmos de inteligencia de enjambres como el método de optimización de enjambre de partículas [5], tal como lo muestra la Fig. 1.

El método de optimización por enjambre de partículas (Particle Swarm Optimization) es un algoritmo inspirado en el comportamiento de algunos seres vivos como por ejemplo bandadas de aves, bancos de peces o manadas de mamíferos, los cuales tienen actuaciones colectivas que proporcionan comportamientos adecuados para explorar diferentes técnicas de búsqueda y optimización. Este algoritmo en lugar de utilizar operadores genéticos tradicionales, lo que hace es modificar cada una de las partículas de acuerdo a su experiencia y la experiencia de la partícula vecina [9] [10].

Por otro lado, el algoritmo genético se caracteriza por imitar los procesos naturales, procesos de búsqueda y optimización que tienen origen en la inspiración del mundo biológico. Este método permite la generación de poblaciones mediante operadores genéticos, creando poblaciones aleatorias cada vez de mejor calidad. Los operadores como la selección, la mutación y la recombinación o cruzamiento se aplican a la población encontrando el cromosoma o la solución de calidad más alta que la aplicación de los operadores haya arrojado [7].

Este artículo está distribuido de la siguiente manera: la Sección II se describe el funcionamiento del algoritmo enjambre de partículas junto con la presentación del pseudocódigo con el cual fue implementado. En la Sección III se encuentra la explicación del funcionamiento del algoritmo genético al igual que el pseudocódigo de su respectiva implementación. En la sección IV se encuentran las pruebas realizadas a los dos algoritmos estudiados sobre las funciones no lineales escogidas para la comparación entre los dos métodos y por último se encuentran la discusión y las conclusiones.

2. MÉTODOS METAHEURISTICOS

El uso de los algoritmos bioinspirados es importante en la investigación y resolución de problemas particulares en diversas áreas. Como resultado de la revisión del estado del arte, de la taxonomía de la computación evolutiva, clasificada en algoritmos evolutivos y en inteligencia de enjambres; se seleccionan dos algoritmos de optimización, los más publicados de cada área [7] para realizar su estudio y comparación.

2.1. Optimización por enjambre de partículas

La técnica de enjambre de partículas fue implementada por Kennedy y Eberhart en 1995 para simular el vuelo sincrónico de las aves. El algoritmo resultante se convirtió en un éxito para la optimización de funciones matemáticas no-lineales continuas [11], [12]. En la literatura se presentan aplicaciones en distintas áreas como la ingeniería industrial [7], la medicina [13] y la ingeniería electrónica [14], [15].

El algoritmo de optimización por enjambre de partículas empieza con un conjunto de soluciones aleatorias. A cada solución potencial se le asigna una velocidad aleatoria y de esta manera las soluciones (también denominadas partículas) “viajan” hasta encontrar la solución óptima. Este método consiste en el cambio de velocidad en cada paso del tiempo sobre todas las partículas dirigiéndolas hacia los mejores resultados, los cuales son escogidos al ser evaluadas las partículas en la función fitness, de esta manera se encuentran las mejores soluciones y la mejor solución global [16], [17], [10].

2.2. Optimización por algoritmos genéticos

Los algoritmos genéticos constituyen una técnica de búsqueda cuyo fundamento es la teoría de la evolución de las especies de Charles Darwin, donde los individuos de una población se cruzan, se reproducen y sobreviven los más aptros en cada generación [18], [19]. Esta técnica ha demostrado ser una herramienta eficiente para resolver problemas de optimización, se presentan aplicaciones en el área de la ingeniería industrial [19], [20], ingeniería electrónica [18] e ingeniería biomédica [21].

El algoritmo genético empieza con un conjunto de soluciones posibles, cada solución posible es un individuo que pertenece al conjunto denominado población. A partir de una población inicial, la cual es generada de forma aleatoria, se aplican los operadores genéticos, los cuales son: selección, mutación y cruce o recombinación.

El operador selección consiste en tomar individuos o cromosomas con el criterio de escoger siempre el mejor. La selección se realiza de manera aleatoria, esta puede presentarse de dos tipos: de manera equiprobable donde todos los individuos tienen la misma probabilidad de ser elegidos o de manera estocástica, en donde los individuos son escogidos por medio de una heurística, esta heurística tiene diferentes procedimientos estocásticos, a continuación se presentan los más usados:

Selección por ruleta: consiste en escoger un cromosoma aleatorio y a partir de esta posición se genera un nuevo aleatorio, el cual va a ser el siguiente cromosoma seleccionado.

Selección por torneo: consiste en escoger una parte de los cromosomas de la población resultante en cada iteración seleccionándose siempre los mejores, de esta manera éstos tienen mayor probabilidad de reproducirse y emigrar que el resto de los cromosomas. Esta evaluación de escoger el mejor está dada por una función denominada fitness, la cual es definida de acuerdo al objetivo que deba cumplir el algoritmo, de esta manera al evaluar los cromosomas, se quedan en la población los que tengan el menor resultado al ser evaluados.

El operador cruce o recombinación consiste en la generación de un cromosoma o individuo en función del cromosoma padre y del cromosoma madre. Este

operador es el mayor responsable de la evolución de la población y define rigurosamente las propiedades del algoritmo genético.

El operador mutación es una variación que se realiza sobre los cromosomas, es decir, es el cambio de algún gen a otro producido por algún factor externo del algoritmo genético. Por lo general se aplica a una pequeña parte de la población y se hace de manera aleatoria.

3. IMPLEMENTACIÓN DEL ALGORITMO

3.1. Optimización por enjambre de partículas

Las características de la implementación del enjambre de partículas se establecen a continuación.

Codificación de las partículas: cada partícula contiene dos variables, las cuales representan la coordenada en X y la coordenada en Y.

Criterio de inicialización: La cantidad de partículas las ingresa el usuario en la aplicación y éstas son generadas en las coordenadas positivas con una distribución de las partículas de manera uniforme. Es decir, las partículas siguen este comportamiento: (1,1), (1,2), (2,1), (2,2), (2,3), (3,2), (3,3),... (n,n); Cumpliendo con la cantidad requerida de partículas.

También se inicializa un vector con ceros, el cual corresponde a las velocidades (*v*) de cada partícula, al igual que un vector en donde están las mejores posiciones (*p*).

Criterio de parada: la optimización por enjambre de partículas termina cuando se cumple el número de iteraciones ingresadas por el usuario de la aplicación.

Función Fitness: la función fitness es la función a optimizar y en ésta se evalúa cada partícula.

Criterios de reemplazo: en cada iteración a las coordenadas de cada partícula se les suma la velocidad que le corresponde, tal como lo muestran las ecuaciones (1) y (2), es decir, la variable *X* corresponde a la coordenada en la posición X en el plano cartesiano. A ésta se le suma su velocidad *Vx.* A la variable *Y* corresponde a la coordenada en la posición Y en el plano cartesiano y se le suma *Vy*, de esta manera se actualiza la posición.

$$x_i = x_i + v_{xi} \quad (1)$$

$$y_i = y_i + v_{yi} \quad (2)$$

Luego, a partir de los mejores valores fitness se actualiza el vector de mejores posiciones (*p*) y las velocidades (*v*) se actualizan de acuerdo a las siguientes ecuaciones formuladas para cada una de las coordenadas de la partícula:

$$v_{xi} = [a_1 * w * v_{xi}] + [a_2 * c * (p_{xi} - x_i)] + [a_3 * c * (p_{gi} - x_i)] \quad (3)$$

$$v_{yi} = [a_1 * w * v_{yi}] + [a_2 * c * (p_{yi} - y_i)] + [a_3 * c * (p_{gi} - y_i)] \quad (4)$$

Donde *i* representa las iteraciones, *a1*, *a2* y *a3* son números aleatorios, *w* es el peso de la inercia, *c* es el valor cognitivo o social. *w* y *c* son constantes ingresadas por el usuario y por ultimo *pg* es la mejor posición global [1], [22], [23] .

```
Inicio;
  Generar población aleatoria de N soluciones (partículas);
  Inicializar el valor w, c;
  Para cada partícula(i);
    Calcular fitness(i);
    Asignar pi como la mejor posición de partícula i;
    Si fitness (i) es mejor que pi;
      pi= fitness(i);
    Fin;
    Asignar pg como el mejor fitness para todas las partículas;
    Para cada partícula;
      Calcular velocidad de acuerdo a las ecuaciones (3) y (4);
      Calcular posición de acuerdo a las ecuaciones (1) y (2);
    Fin;
  Fin;
Fin;
```

Fig. 2. Descripción del algoritmo de enjambre de partículas. Construcción propia

En la Fig. 2 se presenta una descripción del algoritmo de enjambre de partículas que se implementó en la aplicación descrito por Elbeltagi, Hegazy y Grierson [8].

3.2. Optimización por algoritmos genéticos

Las características de la implementación del algoritmo genético, son:

Codificación de los cromosomas o individuos: en el algoritmo genético implementado cada individuo o cromosoma almacena dos variables, las cuales indican las coordenadas X y Y.

Criterio de inicialización: la población inicial se genera de manera aleatoria y la cantidad de cromosomas o individuos es igual al tamaño de la población que ingrese el usuario en la aplicación.

Criterio de parada: el criterio de parada de un algoritmo genético es el número de generaciones. Sin embargo hay variaciones donde el algoritmo termina cuando se consigue una solución aceptable, es decir, donde se acepta cierto margen de error. En este caso el algoritmo genético termina con el número de generaciones establecidas por el usuario en la aplicación.

Función Fitness: la función fitness es la función a optimizar y en ésta se evalúa cada cromosoma.

Criterio de selección: para seleccionar qué cromosoma va a ser tratado, se aplicó la selección por torneo. Consiste en escoger de manera aleatoria dos cromosomas y el de menor fitness es el cromosoma seleccionado.

Esto con el fin de que los mejores cromosomas sean los usados para buscar una solución cada vez mejor.

Operadores genéticos: los operadores genéticos usados son Cruce y Mutación.

El operador cruce o recombinación nos permite unir una parte de un cromosoma con parte de otro. En este algoritmo genético este operador siempre está dado por la mitad de los cromosomas, como se muestra en la Fig. 3.

Fig. 3. Cruce entre cromosomas. Construcción propia

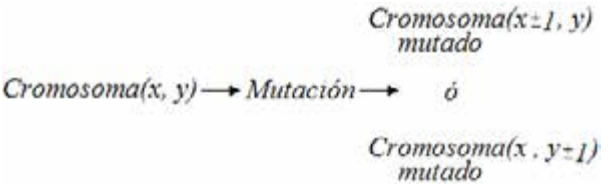

Fig. 4. Mutación entre cromosomas. Construcción propia

El operador mutación consiste en variar una parte del cromosoma, Fig. 4. Este operador por lo general se aplica a una pequeña parte de la población y de manera aleatoria. Sin embargo, es el usuario quien decide cuanto porcentaje de la población es mutada.

Criterios de reemplazo: En este algoritmo genético nuestro criterio de reemplazo de un cromosoma o individuo nuevo por uno de la población actual está dado por la calidad de la solución evaluada por la función fitness. En este caso como es minimizando el reemplazo solo se efectúa si el cromosoma nuevo tiene menor valor fitness que el cromosoma actual [1].

En la Fig. 5 se presenta una descripción del algoritmo genético que se implementó en la aplicación descrito por Elbeltagi, Hegazy y Grierson [8].

```
Inicio;
    Generar población aleatoria de P soluciones (cromosomas);
    Para i=1 a número de generaciones;
        Calcular fitness (i);
        Seleccionar dos cromosomas ia y ib;
        Generar descendencia ic= cruce(ia y ib);
        Seleccionar un cromosoma i de manera aleatoria;
        Generar descendencia ic= mutación(i);
        Calcular fitness de descendecia i;
        Si ic es mejor que el peor cromosoma entonces
            Reemplace el peor cromosoma por ic;
    Siguiente i;
Fin;
```

Fig 5. Descripción del algoritmo genético. Construcción propia

4. PRUEBAS

En este trabajo se desarrolló una aplicación en Java, Fig 6, implementando los dos algoritmos antes descritos. Se utilizó el entorno de desarrollo NetBeans, ya que permite medir el coste computacional de los algoritmos, con el fin de garantizar la comparación entre los mismos. La aplicación permite la entrada de la ecuación que se requiera resolver, no lineal, y los datos que necesita cada uno de los métodos para poder realizar su ejecución. Para el algoritmo de enjambre de partículas el usuario ingresa los valores "cantidad de partículas", "número de iteraciones", "inercia" y "corrección". En el caso del algoritmo genético el usuario ingresa los valores "cantidad de cromosomas", "número de generaciones" y "porcentaje de mutación".

Las ecuaciones escogidas para desarrollar las pruebas de los algoritmos implementados en la aplicación son las siguientes:

$z = (x+20)^2 + (y-15)^2$ (5)

$z = x^2 + y^2 + xy + x + 2y$ (6)

$z = exp(x) + exp(y) – 20x – 20y$ (7)

La finalidad del algoritmo desarrollado en java es obtener el menor valor de z.

En la parte inferior derecha de la aplicación, se encuentran tres botones "Correr PSO", "Correr GA", "Correr Ambos" y "Resultado", los cuales tal como sus nombres lo indican permiten ejecutar solamente el algoritmo de enjambre de partículas, solamente el algoritmo genético, ejecutar los dos algoritmos al tiempo y mostrar los resultados útiles para la comparación, respectivamente.

Al oprimir el botón "Resultado", la aplicación muestra en una ventana los resultados de los parámetros de comparación escogidos para hacer el análisis entre los dos algoritmos. Éstos son:

Max Fitness: indica el máximo valor que toma un cromosoma durante las iteraciones al ser evaluado en la función fitness escogida para cada algoritmo.

Min Fitness: indica el mínimo valor que toma un cromosoma durante las iteraciones al ser evaluado en la función fitness escogida para cada algoritmo.

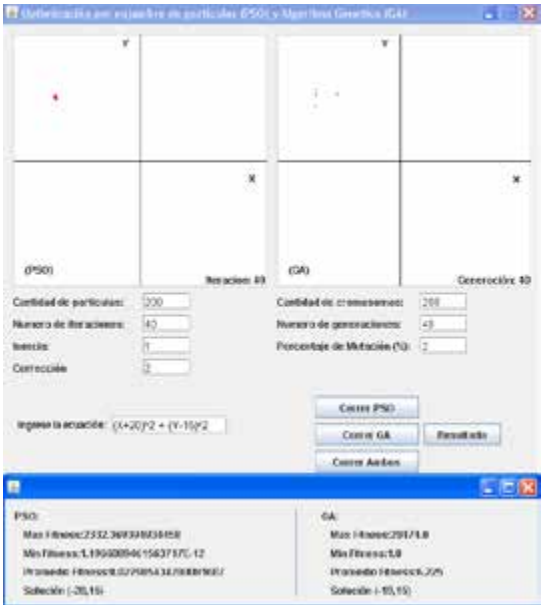

Fig 6. Interfaz gráfica de la aplicación. Construcción propia

Promedio Fitness: indica el valor promedio que toman los cromosomas durante las iteraciones al ser evaluados en la función fitness escogida para cada algoritmo.

Solución: indica la solución que arroja cada algoritmo al ser ejecutado. La solución también es visible en un plano cartesiano, cada uno de los algoritmos tiene sus propios ejes, en los cuales se ubica el punto solución que arroja la ejecución. El plano de la izquierda, Fig 6, corresponde al algoritmo enjambre de partículas y el plano de la derecha corresponde al algoritmo genético.

En la Tabla 1 se muestran los resultados de las tres ecuaciones expresadas anteriormente aplicadas al algoritmo enjambre de partículas. En la Tabla 2 se muestran los resultados de las tres ecuaciones expresadas anteriormente aplicadas al algoritmo genético

5. DISCUSIÓN

A continuación se presenta la comparación entre los dos métodos estudiados mediante las tablas que exponen los resultados de la aplicación al ejecutar los dos algoritmos. En éstas se muestra la función no lineal utilizada, las iteraciones, el valor de la función fitness (promedio, mayor y menor) y la solución de cada una de las pruebas.

Tabla 1. Resultados del algoritmo enjambre de partículas

f(x,y)	Tam	Iteraciones	Valor Fitness			Solución
			Promedio	Menor	Mayor	
1	50	10	34	0	925	(19,15)
		50	0	0	925	(-20,15)
		100	0	0	1033	(-20,15)
	100	10	37	0	1096	(-20,15)
		50	0	0	2982	(-20,15)
		100	0	0	1524	(-20,15)
	300	10	87	0	6007	(-19,15)
		50	0	0	13619	(-20,15)
		100	0	0	9131	(-20,15)
2	50	10	1	-1	168	(0,-1)
		50	-1	-1	168	(0,-1)
		100	-1	-1	168	(0,-1)
	100	10	4	-1	330	(0,-1)
		50	-1	-1	358	(0,-1)
		100	-1	-1	330	(0,-1)
	300	10	12	-1	993	(0,-1)
		50	-1	-1	3109	(0,-1)
		100	-1	-1	3377	(0,-1)
3	50	10	-72	-80	$1.76x10^9$	(3,3)
		50	-80	-80	102216	(3,3)
		100	-80	-80	3715	(3,3)
	100	10	309	-80	$-7.9x10^8$	(3,3)
		50	-80	-80	$-9.2x10^7$	(3,3)
		100	-80	-80	$-8.5x10^7$	(3,3)
	300	10	$6.03x10^7$	-80	-1	(2,3)
		50	-80	-80	-1	(3,3)
		100	-80	-80	-1	(3,3)

Las ecuaciones (5), (6) y (7) se encuentran dentro de las tablas de resultados en la columna denominada *f(x,y)* como las funciones 1, 2 y 3 respectivamente. La columna Tam hace referencia al tamaño de la población, es decir, a la cantidad de partículas o cromosomas usados para ejecutar los algoritmos respectivamente.

El algoritmo por enjambre de partículas se configuró con un peso de inercia *(w)* igual a 1 y una corrección o valor cognitivo *(c)* igual a 2 y el algoritmo genético se configuró con una mutación del 2%.

Tabla 2. Resultados del algoritmo genético

f(x,y)	Tam	Generaciones	Valor Fitness			Solución
			Promedio	Menor	Mayor	
1	50	10	58	16	20690	(-17,14)
		50	0	0	17576	(-20,15)
		100	2	2	19237	(-21,14)
	100	10	7	4	24125	(-12,9)
		50	11	10	20578	(-22,14)
		100	20	20	21610	(-18,17)
	300	10	93	1	22480	(-14,12)
		50	2	2	23330	(-18,14)
		100	0	0	23834	(-19,15)
2	50	10	128	2	27090	(6,0)
		50	56	24	24590	(4,-5)
		100	11	0	24858	(1,-2)
	100	10	8	0	25395	(-4,-3)
		50	18	2	21432	(1,1)
		100	0	0	27651	(-2,-2)
	300	10	204	-1	27936	(4,1)
		50	-1	-1	23771	(0,0)
		100	0	0	28230	(0,-1)
3	50	10	-26	-50	-1	(3,7)
		50	-73	-73	-1	(4,4)
		100	-2	-5	-1	(4,-1)
	100	10	$1.9x10^9$	-65	-1	(8,6)
		50	-80	-80	-1	(4,4)
		100	-80	-80	-1	(3,4)
	300	10	-1	-73	-1	(6,3)
		50	-79	-80	-1	(5,3)
		100	-73	-73	-1	(3,3)

Los parámetros para cada uno de los algoritmos fueron asignados de esta manera debido a que los métodos presentaban un mejor comportamiento al ser ejecutados con estos valores.

Los dos algoritmos están basados en parte por variables aleatorias y esto muestra que existe cierto criterio al azar.

Por ejemplo el algoritmo genético encontró la solución óptima con un tamaño de 50 cromosomas y una generación de 50 y en cambio con un tamaño de 300 cromosomas y una generación de 100 no la encontró.

Otro caso particular se presenta en el algoritmo enjambre de partículas donde con una cantidad de 300 partículas y 10 iteraciones no encontró la solución óptima pero con un tamaño de 100 partículas sí.

6. CONCLUSIONES

A partir de los resultados experimentales arrojados por la aplicación, se puede concluir que los dos métodos obtienen soluciones buenas en un lapso corto de tiempo. Puede que no sea la solución óptima en algunos casos, pero los dos métodos cumplen con la tarea de optimización.

Aun cuando los dos algoritmos estudiados sean métodos metaheurísticos y ambos trabajen con procesos estocásticos, se puede deducir que el algoritmo enjambre de partículas es un método más eficiente optimizando funciones no lineales que el algoritmo genético. Esto se ve reflejado en la optimización de las tres funciones utilizadas en las pruebas, donde el algoritmo de enjambre de partículas demostró ser un método certero para encontrar la solución óptima, diferente al algoritmo genético que en la mayoría de los casos obtuvo sólo soluciones cercanas a la óptima.

A partir del mayor valor de la función fitness se puede concluir que la dispersión de los cromosomas obtenidos en el algoritmo genético es mayor a la dispersión que tienen las partículas en el algoritmo de enjambre.

En cuanto al comportamiento de los métodos para la obtención de la solución, se pudo observar que el algoritmo de enjambre de partículas es más dependiente del número de iteraciones que de la cantidad de partículas con las que inicie el proceso, a diferencia del algoritmo genético el cual es más dependiente de la cantidad de cromosomas que de la cantidad de generaciones con la que es ejecutado.

Como trabajo futuro se espera implementar el algoritmo de enjambre de partículas con otra técnica heurística para la resolución de problemas en corto tiempo y con bajo costo computacional.

REFERENCIAS

[1] S. Forrest, “Genetic algorithms: Principles of natural selection applied to computation”, Science, Vol. 261, No. 5123, pp. 872-878, 1993.

[2] R. Ríos, & L. González, “Investigación de operaciones en acción: Heurísticas para la solución del TSP”, Ingenierías, Vol. 3, No. 9, pp. 15–20, 2000.

[3] J.C. Bansal, P.K. Singh, M. Saraswat, A. Verma, , S.S. Jadon, & A. Abraham, “Inertia weight strategies in particle swarm optimization”, Proceedings of the 2011 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011, pp. 633, 2011.

[4] D. Morillo, L. Moreno, & J. Díaz, “Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1”, Ingeniería y Ciencia, Vol. 10, No. 19, pp. 247–271, 2014.

[5] E. Elbeltagi, T. Hegazy, & D. Grierson, “Comparison among five evolutionary-based optimization algorithms”, Advanced Engineering Informatics, Vol. 19, No. 1, pp. 43-53, 2005.

[9] R. Eberhart, & J. Kennedy, “New optimizer using particle swarm theory”, Proceedings of the International Symposium on Micro Machine and Human Science, pp. 39, 1995.

[7] M. Márquez, “Las metaheurísticas: tendencias actuales y su aplicabilidad en la ergonomía”, Ingeniería Industrial. Actualidad y Nuevas Tendencias, Vol. 4, No. 12, pp. 108-120, 2014.

[8] M.A. Muñoz, J.A. López, & E.F. Caicedo, “Inteligencia de enjambres: sociedades para la solución de problemas”, Revista ingeniería e investigación, Vol. 28, No. 2, pp. 119-130, 2008.

[9] I.C. Trelea, “The particle swarm optimization algorithm: Convergence analysis and parameter selection”, Information Processing Letters, Vol. 85, No. 6, pp. 317-325, 2003.

[10] J. Lima, & B. Barán, "Optimización de Enjambre de Partículas aplicada al Problema del Cajero Viajante Bi-objetivo", Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, Vol. 10, No. 32, pp. 67- 76, 2006.

[11] D. Vanegas, K. Barragán & R. Correa, "Comparación de las técnicas de optimización por análisis de intervalos y la de enjambre de partículas para funciones con restricciones", Ingeniería y Universidad, Vol. 15, No. 1, pp. 47-60, 2011.

[12] V. Gonzalez, M. Villagra & B. Baran, "Optimización por Enjambre de Partículas para Satisfacción de Fórmulas Booleanas", 34th Latin-American Conference on Informatics, 2010.

[13] E. Cortes, "Aplicación del método de Optimización por Enjambre de Partículas (PSO) en el área médica de cito tecnología", Revista Iberoamericana para la Investigación y el Desarrollo Educativo, Vol. 1, No. 2, 2014.

[14] D. Vanegas, K. Barragán & R. Correa, "El método de enjambre de partículas y el criterio de mínima entropía en el diseño óptimo de un disipador de calor", Revista Ingenierías Universidad de Medellín, Vol. 11, No. 20, pp. 203-214, 2012.

[15] J. Pérez, & J. Basterrechea, "Optimización con enjambre de partículas aplicada a la reconstrucción del diagrama de radiación de antenas", XX Simposium Nacional de la Unión Científica Internacional de Radio, Gandía, 2005.

[16] R. Eberhart, & J. Kennedy, "New optimizer using particle swarm theory", Proceedings of the International Symposium on Micro Machine and Human Science, pp. 39, 1995.

[17] Z. Ma, & H. Liu, "A kind of improved uniform particle swarm optimization algorithm", Proceedings–2010 2nd WRI Global Congress on Intelligent Systems, GCIS 2010, pp. 23, 2010.

[18] I. Ruge, & M. Alvis, "Aplicación de los algoritmos genéticos para el diseño de un controlador PID adaptativo", Tecnura, Vol. 13, No. 25, pp. 81-87, 2009.

[19] Y. Solano, M. Calvo, & L. Trejos, "Implementación de un algoritmo genético para la asignación de aulas en un centro de estudio", Uniciencia, Vol. 1, No. 22, pp. 115-121, 2008.

[20] G. Mendez, "Diseño de un algoritmo genético para un sistema logístico de distribución", Ingeniría, Vol. 5, No. 1, pp. 20-27, 2000.

[21] C. Galeano, & D. Garzón, "Algoritmos Genéticos aplicados a la Ingeniería biomédica", Revista Cubana de Investigaciones Biomédicas, Vol. 20, No. 3, pp. 402-411, 2011.

[22] R.C. Eberhart, & Y. Shi, "Particle swarm optimization: Developments, applications and resources", Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, pp. 81, 2001.

[23] D. Karaboga, & B. Akay, "A comparative study of Artificial Bee Colony algorithm", Applied Mathematics and Computation, Vol. 214, No. 1, pp. 108-132, 2009.