Scientia Et Technica
ISSN: 0122-1701
scientia@utp.edu.co
Universidad Tecnológica de Pereira
Colombia

Obando Paredes, Edgar Dario
Algoritmos genéticos y PSO aplicados a un problema de generación distribuida.
Scientia Et Technica, vol. 22, núm. 1, marzo, 2017, pp. 15-23
Universidad Tecnológica de Pereira
Pereira, Colombia

Algoritmos genéticos y PSO aplicados a un problema de generación distribuida.

PSO and genetic algorithms applied to a distributed generation problem

Obando Paredes Edgar Dario[1].
[1]*edgar.obandop@campusucc.edu.co.*
[1]*Grupo de Investigación ESLINGA Universidad Cooperativa de Colombia-Sede Pasto*

***Resumen*—En este informe se presentan los resultados de la aplicación de algoritmos genéticos y PSO (*Particle Swarm Optimization*), para optimizar un problema de generación distribuida (GD) de potencia que debe cumplir ciertas restricciones. Para la implementación del algoritmo genético se utiliza el *toolbox* de Matlab ya implementado variando algunos parámetros como fracción de mutación, población etc. Lo anterior para compararlo con la función *fmincon* ya implementada dentro del ambiente Matlab y sacar conclusiones en cuanto a tasa de convergencia y error entre los datos. El Algoritmo PSO fue implementado teniendo en cuenta procesos estocásticos basados en suerte, definiendo propiedades intrínsecas a él, tal como tamaño de población, factor de inercia etc.**

***Palabras Claves*—Optimización con restricciones, Algoritmos Genéticos, PSO, Matlab, Generación.**

***Abstract*-In this report the results of the application of genetic algorithms and PSO (Particle Swarm Optimization) are presented to optimize a problem of distributed generation (DG) of power that must meet certain restrictions. For the implementation of the genetic algorithm toolbox of Matlab it is used and implemented fraction varying parameters such as mutation, population etc. This for comparison with the function fmincon already implemented within the Matlab environment and draw conclusions regarding convergence and error rate between data. The PSO algorithm was implemented taking into account stochastic processes based on luck, defining him intrinsic properties, such as population size, inertia factor etc.**

Keywords-Constrained Optimization, Genetic Algorithm, PSO, Matlab, Generation.

I. INTRODUCCION

La optimización con restricciones plantea el problema de encontrar un punto o puntos mínimos de una función restringida a ciertas restricciones, a unos valores de banda, (*Up-Down*), y a una función de restricción. En el problema a resolver se presenta una función de generación dependiente de 6 generadores distribuidos, donde cada uno de ellos tiene unos valores de banda de potencia y la restricción general es una demanda fija de potencia que se debe cumplir. Para solucionar este problema se usan algoritmos genéticos y PSO (*particle Swarm Optimization*), algoritmos específicamente elaborados para solucionar problemas de este tipo, los dos son algoritmos Heurísticos, que dependen mucho de los parámetros de entrada. En el caso de algoritmos genéticos se usa el *toolbox* de Matlab ya implementado variando diferentes propiedades. El algoritmo PSO, al tratarse de un "enjambre", una agrupación de partículas se implementa teniendo en cuenta la velocidad y posición iníciales para que converja al punto minino, dependiendo fuertemente de las propiedades de este, factor de inercia, tamaño de la población. Los dos algoritmos se comparan con los valores dados por la función *fmincon* que usa matlab para encontrar el mínimo de funciones con restricciones.

II. MARCO TEÓRICO

A. *Problema a analizar.*

En un sistema de distribución con 6 generadores distribuidos (GD) se desea realizar un despacho económico de las unidades para minimizar el costo de generación, teniendo en cuenta que se debe satisfacer una demanda y las restricciones técnicas de los generadores.

De acuerdo con esto, el problema consiste en encontrar la potencia que debe producir cada generador para la siguiente función y su restricción:

$$\min f_{tot}(p) = \sum_{i=1}^{6} f_i(p_i)$$

$$s.t \sum_{i=1}^{6} (p_i) = p_d$$

Donde cada generador produce una potencia de la forma:

$$f_i(p_i) = a_i + b_i p_i + c_i p_i^2$$

y a_i, b_i, c_i son coeficientes dados para cada generador. Para analizar el despacho a una hora determinada del día, se asume

Fecha de Recepción: 23 de noviembre de 2016
Fecha de Aceptación: 16 de marzo de 2017

que la potencia demandada es fija en el valor de $P_d = 1150$.

B. *Algoritmos Genéticos*

Loa algoritmos genéticos son un tipo de algoritmo de optimización que se utilizan para encontrar una solución óptima, a un problema computacional, en el cual se imitan procesos biológicos de reproducción y selección natural. Al igual que en la evolución, muchos de esto procesos de algoritmos genéticos son la alzar, aunque mediante el *toolbox* de Matlab y en general se puede establecer el nivel de aleatoriedad y control.

Debido a que los algoritmos genéticos simulan procesos biológicos mucha terminología relevante de biología debe ser usada, Los componentes básicos de un algoritmo genético son:

- Una función de optimización, denominada función *fitness.*
- Población de cromosomas.
- Selección de reproducción de cromosomas
- Mezcla de los productos de la siguiente generación de cromosomas y un factor aleatorio de mutación en la nueva generación.

En el *toolbox* de Matlab, ya vienen integrados todos estos parámetros, además de diversas funciones de selección de individuos, tales como selección estocástica, ruleta y elitista.

C. *PSO*

El algoritmo PSO, siglas de *particle Swarm Optimization*, optimización por enjambre de partículas, que al igual que el algoritmo genético basado en heurísticas y siendo necesario la inicialización del sistema en una población candidata de soluciones, las cuales según el algoritmo se "moverán" a la solución más óptima, a cada potencial solución se le denomina partícula la cual tiene asociada a ella una velocidad y posición iníciales. Cada partícula realiza un seguimiento de coordenadas determinadas por el espacio del problema asociada con el mejor valor de la función objetivo. El algoritmo consiste en dar un paso a la vez cambiando la velocidad de la partícula, donde esta tendrá valores determinados por la función objetivo, el *pbest* es mejor valor local encontrado, y el *gbest* es el mejor valor global encontrado. La aceleración está determinada por un número aleatorio que hace que el enjambre de partículas busque en el espacio determinado la solución óptima.

El algoritmo completo debe seguir los siguientes pasos:

1. Inicializar una población con posiciones y velocidades aleatorias en d dimensiones dadas por el problema.
2. Para cada partícula evaluar la optimización deseada de la función *fitness* en *d* variables.
3. Compare la evaluación de la función con el mejor punto de la misma, *pbest*. Si el nuevo valor es mejor que el *pbest*, ese *pbset* será el nuevo valor.
4. Cambiar la velocidad y la posición de la partícula de acuerdo con las ecuaciones 1 y 2.

$$v_{id} = v_{id} + c_1 * rand(\ \ ) * (p_{id} - x_{id}) + c_2 * rand(\ \ )(p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

5. Repetir desde el ítem 2) teniendo en cuenta un criterio de parada suficientemente bueno o considerando un número máximo de iteraciones.

D. *fmincon*

fmincon es la función por defecto que Matlab utiliza para resolver problemas de optimización con restricciones, para efectos de comparación lores resultados de la aplicación de esta función serán considerados los más acercados a la realidad.

La sintaxis por defecto, para este ejemplo donde se manejan vectores de banda, es:

```
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
```

Donde, se debe tener en cuenta la función a optimizar ya debidamente implementada, un punto inicial que debe estar en el rango de funcionamiento, las inecuaciones de restricción, si existen, las restricciones de igualdad y los vectores de banda.

III. Procedimiento y Resultados.

A. *Usando la función fmincon, por defecto.*

Para usar la función *fmincon*, se hace necesario implementar la función objetivo en un script de Matlab, donde estén definidos los coeficientes a_i, b_i y c_i, (*vease simple_fitness.m*). por definición se usará la misma función objetivo para el *toolbox* de algoritmos genéticos y PSO, al usar la función debe definirse los vectores de restricción, de la forma:

$$A_{eq} = [111111]$$
$$b_{eq} = [1150]$$

Para que cumpla la función restricción:

$$p_1 + p_2 + p_3 + p_4 + p_5 + p_6 = 1150$$

Donde A_{eq} representa los coeficientes de la potencias p_i producidas por cada generador y b_{eq} es la restricción de demanda anteriormente definida. Los vectores LB y UB necesarios para restringir la potencia de producción de cada generador deben ser definidos anteriormente, para usarlos en la evaluación. (véase *fmincon_pot.m*). Tomando un vector de potencias iníciales x_o, teniendo en cuenta las potencias mínimas y máximas a las que esta sujeto cada generador en cuestión.

Los resultados se muestran en la Tabla 1.

Potencia Usando *fmincon (KW)*
334.24
82
237.31
176.45
60
260

Tabla 1. Resultado de minimización usando *fmincon.*

Hay que resaltar que la sumatoria de todas las potencias entregadas cumplen con la restricción de la potencia

demandada de 1150 KW.

B. Usando el Toolbox de Optimización aplicando algoritmos geneticos.

Debido a que se debe comparar los resultados arrojados por el algoritmo y los resultados arrojados por la función *fimncon*, se hace necesario implementar una función de error relativo para determinar que tan convergente es el algoritmo en relación con los valores teóricos. (vease error.m), además se debe tener en cuenta que el cromosoma del individuo está representado por cada una de las potencias de generación.

PRUEBA ALGORITMOS GENÉTICOS, OPCIONES POR DEFECTO.

Se calibra el toolbox de Matlab incluido con las opciones de la función objetivo que se vaya a minimizar, en nuestro caso las potencias de generación, y se establecen los vectores de umbral de cada una de los generadores. Tal como se muestra en la Figura 1. Las demás opciones como Población (*Population*), Escala Fitness, etc, se dejan con los valores predeterminados. La población inicial se toma como 20 potencias aleatorias que posiblemente minimizaran el problema, el parámetro de selección por defecto es Estocástico Uniforme, que, al ser de carácter no lineal, tiene en cuenta los padres, es decir, las potencias iníciales y escoge la siguiente generación en base a la expectativa de vida o en este caso a la potencia producida, por medio de un parámetro aleatorio.

El parámetro de reproducción está determinado por defecto, en 2, es decir el número de individuos que sobrevivirán a la siguiente generación, en todos los casos se debe tener en cuenta que este parámetro determinado por elitismo, es decir donde los mejores resultados pasen a la siguiente generación, debe ser menor o igual al tamaño de la población. El factor de mezcla debe ser un numero contendido entre 0 y 1, en este caso 0.8; representando una tasa de reproducción al azar del 80% es decir pasan a la siguiente generación con mayores características de las potencias "padres".

Usando esta configuración, las potencias optimas de cada generador se muestran en la Figura 1, debido al parámetro de restricción, las sumatoria de todas estas potencias debe cubrir la demanda a esa hora 1150 KW.

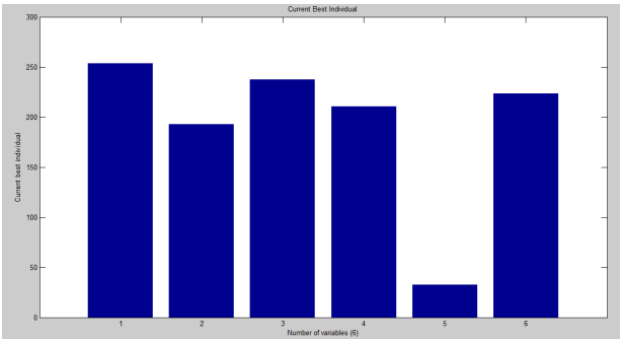


Figura 1. Potencias mínimas entregadas por generador usando opciones por defecto.

En la Figura 2 se muestra los valores de la función objetivo vs generación de los individuos; se encuentra que existe un individuo que pasa a las siguientes generaciones, aproximadamente hasta la generación N° 19, lo que implica que la generación de potencia de un generador, cumplía a cabalidad con las restricciones y funcionaba a su potencia nominal máxima, en la generación 50 el algoritmo encuentra el mínimo debido a que no existen cambios en la población

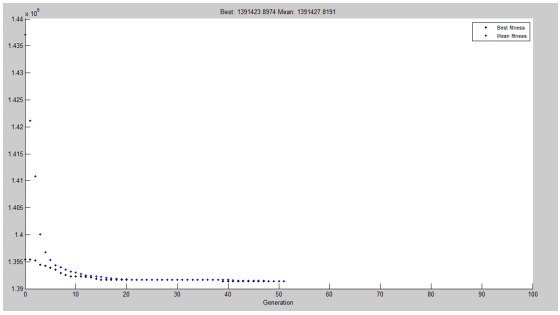


Figura 2. Mejor valor de la función objetivo vs Generación.

En la Tabla 2 se muestran los resultados de la comparación de los resultados de potencia generada usando Algoritmos Genéticos y la función predefinida *fmincon*, usada por Matlab para determinar mínimos con restricciones, cabe resaltar que la suma de las potencias mínimas generadas usando *fmincon*, debe respetar la restricción de igualdad definida anteriormente.

Potencia Usando GA Toolbox	Potencia Usando *fmincon*	Error %
261	334.24	21.911
150.91	82	84.032
282.61	237.31	19.091
194.5	176.45	10.229
31.135	60	48.10
229.84	260	11.6

Tabla 3. Comparación de resultados potencia generada.

PRUEBA ALGORITMOS GENÉTICOS CAMBIANDO EL VALOR DE FRACCIÓN DE CRUCE Y EL TAMAÑO DE LA POBLACIÓN.

Para la siguiente prueba, se cambia el tamaño de la población, al doble, 40, teniendo 40 individuos, que serán evaluados en la función objetivo como espacio de trabajo, además se cambia el factor de cruce, a 1, es decir el 100% de la población se cruza y esos cruces pasarán a la siguiente generación. En la Figura 3 se muestra el mejor valor de la función vs la generación. El mejor valor de un individuo cualquiera, pasa solamente hasta aproximadamente la generación 10, debido a que al cruzar toda la población el mejor valor (individuo) se pierde antes de pasar a la siguiente generación debido a la recombinación de todos los padres nuevamente.

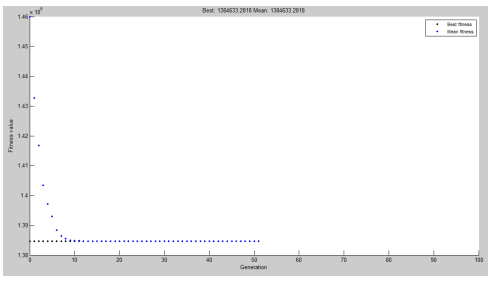


Figura 3. Mejor valor de la función objetivo vs Generación, usando una población mayor.

En la Figura 4, se muestra las potencias mínimas a las que opera cada generador, y que cumplen con los parámetros de restricción.

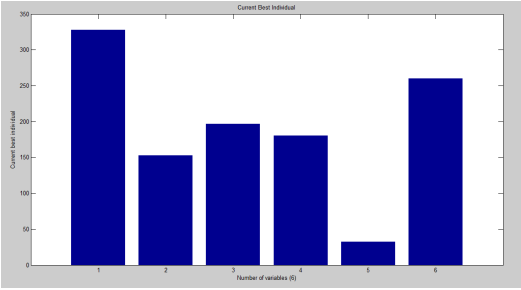


Figura 4. Potencias generadas con una población de 40 individuos.

Existen errores representados en la Tabla 4, que demuestran cambios drásticos en los resultados de potencia del algoritmo genético, se varían parámetros de población lo que implica que habrá mayor espacio de trabajo y su respectiva implicación en el tratamiento del error, además de considerar el cruce de toda la población, originando estos posibles degeneraciones y haciendo que el mejor valor se pierda.

Potencia Usando GA Toolbox	Potencia Usando *fmincon*	Error %
268.17	334.24	19.767
183.27	82	123.5
231.36	237.31	2.5044
191.48	176.45	8.5163
29.795	60	50.342
245.92	260	5.4171

Tabla 4. Comparación de valores de potencia entregados, por dos diferentes técnicas de optimización.

PRUEBA ALGORITMOS GENÉTICOS CAMBIANDO EL PARÁMETRO DE SELECCIÓN DE INDIVIDUOS.

Para esta prueba, se cambia el parámetro de selección optando por un tipo de selección ruleta, donde cada individuo tiene una probabilidad igual de pasar a la siguiente generación, el algoritmo usa un numero aleatorio para definir el área de probabilidad de la ruleta. En la Figura 5. se muestra el mejor valor de la función vs la generación, se ve una tendencia constante de mejor valor a lo largo de la generación, esto en parte a un factor de selección de 1 y al tipo de escogencia de ruleta, mientras que el valor medio de la función decrece hasta aproximadamente la generación 50, esto debido a que todos los individuos pasaran a la siguiente generación, dado que es un promedio de los valores de la función.

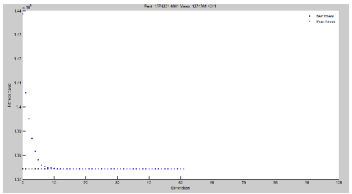


Figura 5. Mejor valor de la función vs generación.

En la Figura 6, se muestran los mejores valores de potencias individuales generadas para esta prueba, y en la Tabla 5, la comparación con los valores entregados por *fmincon*, en todas las pruebas se verifica que se cumple la restricción.

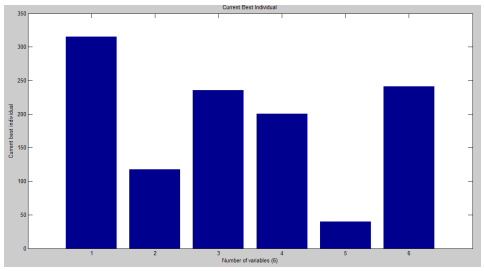


Figura 6. Potencias individuales de cada generador.

Potencia Usando GA Toolbox	Potencia Usando *fmincon*	Error %
315.25	334.24	5.6803
117.3	82	43.053
235.55	237.31	0.73797
200.59	176.45	13.68
40.079	60	33.202
241.22	260	7.2249

Tabla 5. Comparación de valores de potencia entregados, por dos diferentes técnicas de optimización.

C. Usando algoritmo PSO.

El algoritmo PSO implementado y utilizado en el presente informe basa su estructura general en la información ofrecida en el artículo "*Particle Swarm Optimization: Algoritm and it's codes in MATLAB*" publicado en marzo del presente año por *Mahamad Nabab Alama* investigador del departamento de Ingeniería Eléctrica del Instituto de Tecnología Roorke en India. La información disponible en la red frente a la forma de implementar este tipo de algoritmos, limitan su estructura al cumplimiento de restricciones de desigualdad que para el caso de estudio son las potencias máximas y mínimas de trabajo de cada uno de los generadores; sin embargo, la forma en la cual se deben tratar las restricciones de igualdad solo fue posible de solucionar gracias al artículo anteriormente citado, definiendo una función de restricción de igualdad basada en penalización.

La estrategia se utiliza de tal manera que la búsqueda del enjambre de partículas cumpla tanto con las restricciones de desigualdad como la de igualdad fue el siguiente:

- Restricciones de desigualdad:

El algoritmo, luego de generar la población inicial aleatoria, en base a los valores de pbest (mejor partícula individual), gbest (mejor partícula global) y velocidad inicial, calcula la nueva ubicación del enjambre, en dicho momento se compara la posición de cada partícula, que para este caso representa la potencia generada por cada equipo, si esta está fuera de la

banda se reemplaza la potencia de dicho equipo a la máxima o mínima de la misma según el caso, de esta forma se asegura que se cumpla con la restricción capacidad de generación de cada equipo.

- Restricción de igualdad.

La forma en que este problema se mitiga resulto ser bastante curiosa, se basa principalmente en un ejercicio de premio y castigo reflejado en una condición de penalización sobre la fitness evaluada en posición de cada partícula. En el presente caso de estudio la restricción de igual es:

$$\sum_{k=1}^{6} x_k = 1150\ kW$$

En donde x_k representa la potencia aportada por cada generador tal que solvente en todo instante una demanda fija de 1150 kW. Frente a este requerimiento, partículas que cumplan estrictamente con la restricción son premiadas mediante la no modificación del valor de su respectiva fitness, es decir el costo de producción de 1150kW por parte de los generadores no se modifica. Ahora si la partícula no cumple con la restricción y por el contrario la potencia total generada del conjunto de equipos supera la demanda fija, este comportamiento se penaliza fuertemente por lo que se multiplica la fitness con una constante que hace que su costo sea elevado, esto implicaría que en el proceso de selección de la mejor fitness subsecuente, esta partícula no sería tomada en cuenta como una posible pbest. Sin embargo puede darse el caso en que la partícula no cumpla con la restricción, pero que la potencia generada sea menor, en tal caso la penalización se da pero no se hace tan drástica, se penaliza medianamente con el fin de que la población trate de encontrar un mínimo más adecuado. Para efectos de esta implementación esto se hace de la siguiente forma:

```
c=x(1)+x(2)+x(3)+x(4)+x(5)+x(6)-1150;    %restriccion donde es <=0
if c>0
     c=1;
elseif c<0
     c=0.5;
else
     c=0;
end
penalty=1000000000000000000;  % penalidad por incumplimiento de
f=y+penalty*c;                % restriccion
```

En donde *f* es la función de costo evaluada en la posición de la partícula y c el porcentaje de penalización. Posteriormente el algoritmo escoge la partícula con menor fitness y se reasigna los nuevos pbest y gbesta para la siguiente iteración. El algoritmo completo desarrollado en MATLAB se adjunta como anexo a este informe.

Vale la pena aclarar que el programa desarrollado tiene dos parámetros uno que controla las iteraciones de reposicionamiento de partículas (ite), y un segundo que controla la cantidad de veces que se prueba una nueva población aleatoria usando los mejores resultados de la prueba anterior, a este parámetro se lo denomina (run).

Con el fin de comparar los resultados obtenidos con el algoritmo frente a su comportamiento en la búsqueda del mínimo global, se plantean las siguientes pruebas:

PRUEBA ALGORITMO PSO CAMBIANDO LOS FACTORES DE INERCIA.

Los factores de inercia en el algoritmo PSO se definen como coeficientes numéricos que hacen parte activa del cálculo de la velocidad de desplazamiento de cada partícula iteración a iteración. Estos coeficiente junto a los valores de pbes y gbest determinan la dirección y magnitud de movimiento de cada partícula dentro del espacio de búsqueda, estos coeficientes permiten que el enjambre de partículas iteración a iteración se muevan hacía un posible óptimo de forma más rápida o lenta según se escojan. Tras realizar diferentes pruebas teniendo en cuenta que el coeficiente de inercia está siendo calculado de la siguiente forma:

```
w=wmax-(wmax-wmin)*ite/maxite;
```

Donde ite es la iteración actual, maxite es el número máximo de eteraciones, wmax y wmin el rango de velocidades que definen el factor de inercia, se encuentra que dicho factor afecta directamente la velocidad a la cual la partícula se mueve hasta acercarse a un mínimo global. Sin embargo su efecto global no es significativo para este caso de estudio ya que en cada búsqueda, la mejor partícula se encuentra muy rápidamente debido al fuerte efecto de la restricción de igual. Pruebas realizadas antes de incluir la restricción de igualdad indicaban de mejor manera la velocidad a la cual se aproximaba al mínimo.
El objetivo del factor de inercia es permitir que los puntos se muevan en pasos más grande con el fin de no quedarse en mínimos locales si no buscar un mínimo global, de esta manera para las siguientes pruebas, aunque el efecto no es grande, se tomara wmax y wmin bastante separados con el fin de que la inercia en la iteración 1 sea la máxima y que vaya decreciendo a medida que aumentan las iteraciones.

PRUEBA ALGORITMO PSO CAMBIANDO LOS COEFICIENTES DE APRENDIZAJE

Para esta prueba se considera inicialmente coeficientes de aprendizaje iguales, estableciéndolos en $c_1 = c_2 = 0.1$, en la Figura 6, se muestran el valor que toma la función fitness vs iteración de producción de individuos (run).

Condiciones de prueba:

Población	n =100
Máximo de iteraciones	ite =100

Máximo de iteraciones de población	maxrun =600
Coeficiente de inercia máximo	wmax = 10
Coeficiente de inercia mínimo	wmin = 0.01
Coeficiente de aprendizaje individual	c1= 0.1
Coeficiente de aprendizaje global	c2= 0.1

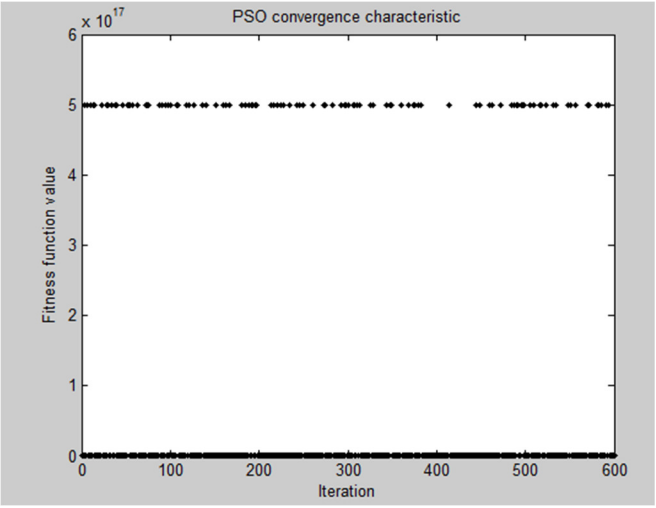


Figura 6. Valor función fitness vs iteración de producción de individuos.

Se puede observar que en un buen porcentaje de las pruebas con diferentes poblaciones aleatorias, se obtiene una mayor concentración de fintes bajas, lo cual indica que hay una tendencia por encontrar un mínimo global.

En la Tabla 6. se muestran los valores mínimos de potencia generados, y su correspondiente error, comparado con los resultados obtenidos con la función *fmincon* de matlab.

Potencia Usando PSO	Potencia Usando *fmincon*	Error %
297	334.24	10.61
117	82	42.68
246	237.31	3.66
194	176.45	9.94
54	60	10
242	260	6.92

Tabla 6. Comparación PSO vs *fmincon*

Para esta prueba, el mejor valor de potencia estuvo en la población generada número 324; cabe resaltar que entre más grande sea la población mejores resultados se darán en términos de convergencia; pero esto tiene repercusiones computacionales de forma demandante.

Paso seguido se realiza una segunda prueba, se toma como coeficientes de aprendizaje $c_1 = 100$ y $c_2 = 10$, definiendo con esto un factor de aprendizaje individual mucho mayor que el aprendizaje global. En la Figura 2 se muestra los resultados obtenidos.

Condiciones de prueba:

Población	n =100
Máximo de iteraciones	ite =100
Máximo de iteraciones de población	maxrun =500
Coeficiente de inercia máximo	wmax = 10
Coeficiente de inercia mínimo	wmin = 0.01
Coeficiente de aprendizaje individual	c1= 100
Coeficiente de aprendizaje global	c2= 10

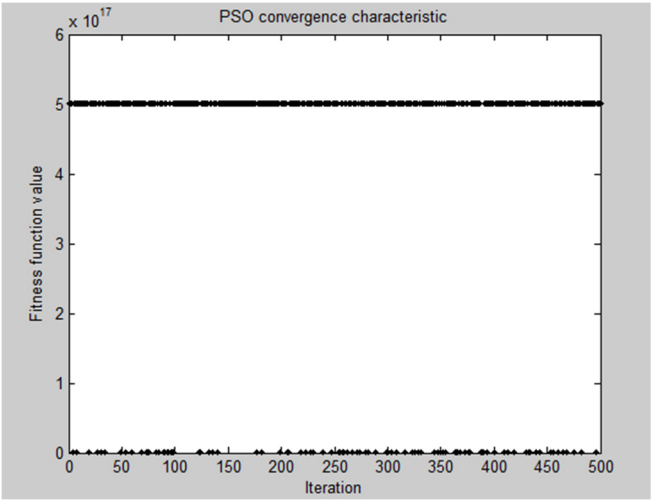


Figura 7. Valor de la función fitness vs iteración de producción de individuos.

Como se puede observar a diferencia de los resultados obtenidos anteriormente, las partículas prueba a prueba tienden a localizarse en un mínimo local, lo cual ratifica la importancia que se le da al aprendizaje individual sobre el grupal.

En la Tabla 7, se muestran los valores dados en la prueba y su comparación con la función *fmincon.*

Potencia Usando PSO	Potencia Usando *fmincon*	Error %
354	334.24	5.911
135	82	0.64
183	237.31	0.29
203	176.45	15.05
27	60	55
248	260	4.62

Tabla 7. Comparación PSO vs *fmincon*

Frente al error de aproximación al mínimo global, es difícil asegurar una mejora, sin embargo se puede observar que en aquellos generadores donde disminuye el error, lo hace en una proporción aceptable.

Finalmente se realiza una última prueba, se toma como coeficientes de aprendizaje $c_1 = 10$ y $c_2 = 100$, definiendo con esto un factor de aprendizaje global mucho mayor que el aprendizaje individual. En la Figura 8 se muestra los resultados obtenidos.

Condiciones de prueba:

Población	n =100
Máximo de iteraciones	ite =100
Máximo de iteraciones de poblacion	maxrun =500
Coeficiente de inercia máximo	wmax = 10
Coeficiente de inercia mínimo	wmin = 0.01
Coeficiente de aprendizaje individual	c1= 100
Coeficiente de aprendizaje individual	c1= 10
Coeficiente de aprendizaje global	c2= 100

Los resultados obtenidos son los siguientes

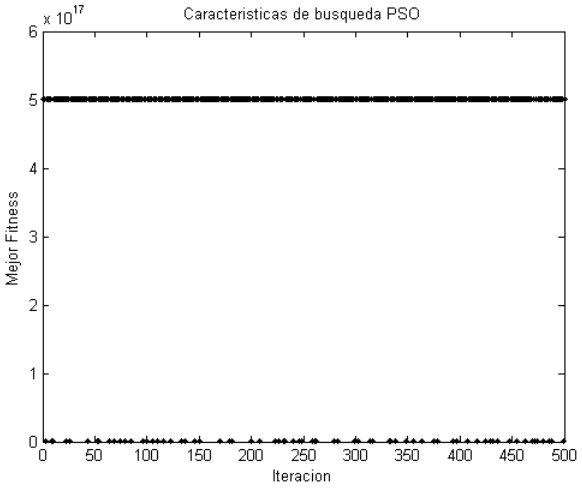


Figura 8.Valor de función fitness vs Iteración.

Es evidente nuevamente que hay una tendencia por ubicar la mejor partícula en un mínimo local, aun así la forma en que está diseñado el presente algoritmo permite identificar en las 500 poblaciones cual fue la mejor partícula. En la Tabla 7, se muestran los valores dados en la prueba y su comparación con la función *fmincon.*

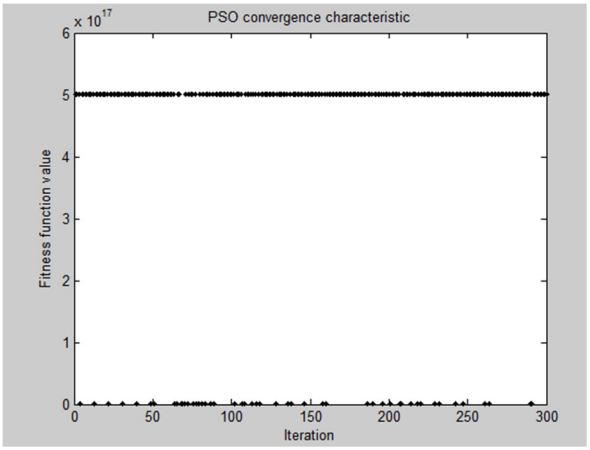
Potencia Usando PSO	Potencia Usando *fmincon*	Error %
311	334.24	8.75
180	82	19.512
245	237.31	13.775
152	176.45	9.94
24	60	36.67
238	260	5.78

Tabla 7. Comparación PSO vs *fmincon*

Como se puede ver los resultados presentan un incremento del error con respecto a la prueba anterior y esto se repite en proporciones similares en varias pruebas realizadas en las mismas condiciones de trabajo; en base a estos resultados se puede concluir que el mejor desempeño se da cuando los coeficientes de aprendizaje son iguales.

PRUEBA CAMBIANDO LOS VALORES DE ITERACIÓN DE POBLACION DE INDIVIDUOS ALETAORIOS.

Para estas pruebas se cambia la cantidad de poblaciones de individuos aleatorios a generar, comenzando en 300 iteraciones de población cada una de ellas de 100 individuos aleatorios que trataran de llegar a la solución óptima. En la Figura 4, se muestra el mejor valor de la función vs iteración de población, en este caso se muestra que los individuos



buscan en una región de mínimo local, antes de converger a un mínimo global.

Figura 9. Función fitness vs iteración de poblaciones

En la Tabla 8, se muestra la generación de potencia con PSO, comparada con fmincon, de matlab.

Potencia Usando PSO	Potencia Usando *fmincon*	Error %
226	334.24	32.39
116	82	41.46
242	237.31	1.98
248	176.45	40.55
59	60	1.67
259	260	1.67

Tabla 8. Comparación PSO vs *fmincon*

Para esta prueba el algoritmo converge en un tiempo de un minuto.

Paso seguido se prueba aumentando el número de iteraciones de población a 900, manteniendo los demás parámetros de búsqueda. En la Figura 10 se pueden observar los resultados obtenidos.

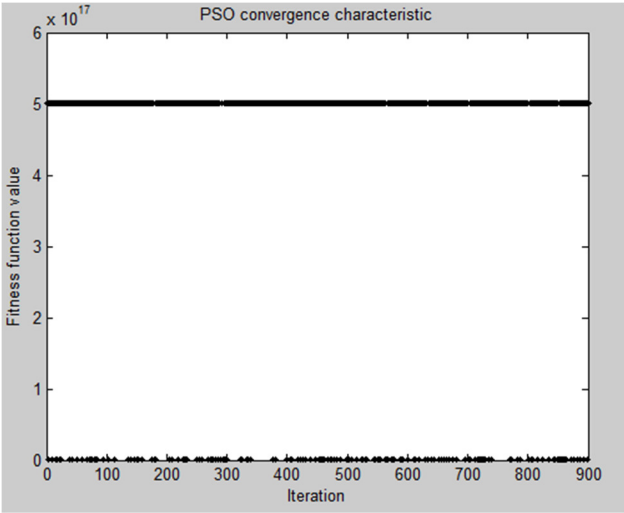


Figura 10. Función fitness vs iteración de población

En la Tabla 9, se muestra la generación de potencia con PSO, comparada con fmincon, de matlab.

Potencia Usando PSO	Potencia Usando *fmincon*	Error %
391	334.24	16.98
83	82	1.22
242	237.31	1.98
157	176.45	11.02
27	60	55
250	260	3.85

Tabla 9. Comparación PSO vs *fmincon*

Este algoritmo converge para encontrar el mínimo en un tiempo de 4 minutos, sin embargo se puede evidenciar que no hay un cambio concluyente frente al error de aproximación, se supondría que cada población futura por el hecho de contar con un gbest más cercano al global debería mejorar su comportamiento, pero el componente aleatorio sobre un mismo tamaño de población hace que esta mejora no sea perceptible. Por otra parte se puede observar que la tendencia por ubicarse en un mínimo local persiste sin importar el número de poblaciones en prueba.

Es así como finalmente resta hacer una última prueba en la cual se incremente el número de individuos por población de 100 a 1000, los resultados obtenidos se muestran en la Figura 11.

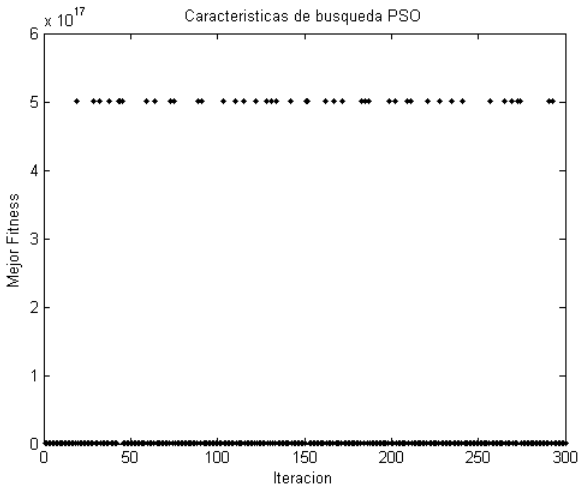


Figura 11. Función fitness vs iteración de población

Como se puede observar la tendencia de los resultados es mucho mejor frente a la búsqueda población a población de un mínimo global, esto se debe a que hay una mayor probabilidad de que en una población mayor exista una partícula que esté más cerca del mínimo.

En la Tabla 10, se muestra la generación de potencia con PSO, comparada con fmincon, de matlab.

Potencia Usando PSO	Potencia Usando *fmincon*	Error %
343	334.24	2.62
136	82	65.85
193	237.31	18.67
164	176.45	7.05
54	60	10
260	260	0

Tabla 10. Comparación PSO vs *fmincon*

El error de aproximación al mínimo global, se prueba a prueba aún sigue siendo no concluyente, sin embargo con pruebas donde la población es muy grande en general la frecuencia de consecución en algunos generadores de errores nulos es mayor.

IV. CONCLUSIONES

- Los errores de valor alto aproximadamente mayores a 50% se deben en parte a la degeneración presentada en operaciones de cruce y mutación entre los individuos de la población sujetos a la restricción de banda y general.
- Los parámetros de calibración son de vital importancia para determinar las características de la población y su reproducción, como se puede determinar por las pruebas realizadas cambios en la forma de la mutación y selección de los individuos hacen que la función converja pero ceñida a restricciones del algoritmo en utilización.
- PSO conclusiones

- El factor de inercia para este caso no afecta significativamente la búsqueda del mínimo debido a la fuerza que tiene la restricción de igualdad.
- El desempeño que tiene la búsqueda cuando se utiliza la técnica de pruebas sucesivas que mantienen información de la mejor partícula encontrada en una prueba inmediatamente anterior, genera mínimos globales más adecuados que si solo se corriera el algoritmo con una sola población.
- El algoritmo trabaja mejor cuando los coeficientes de aprendizaje tanto global como individual son iguales.
- El logro de mejores resultados de búsqueda está asociado directamente con el tamaño de la población y la cantidad de veces que se generen nuevas poblaciones con memoria de la mejor partícula global, esto implica que el costo computacional es mayor debido al número de iteraciones finales, lo cual se ve reflejado en tiempo de búsqueda.
- Un incremento en la población inicial, incrementa la posibilidad de encontrar un mínimo global.
- El hecho de que en cada prueba la generación de la población inicial sea totalmente aleatoria, hace que dos pruebas consecutivas nunca sean iguales, aunque los parámetros de búsqueda sean idénticos.

REFERENCES

[1] Mahamad Nabab Alama, “Particle Swarm Optimization: Algoritm and it’s codes in MATLAB”, en revisión, Marzo 8 de 2016. Departamento de Ingeniería Eléctrica del Instituto de Tecnología Roorke, India.

[2] Carr J, An An Introduction to Genetic, Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[3] Pohlheim H. Tutorial: Genetic and Evolutionary Algorithm Toolbox for use with Matlab Documentation for: Genetic and Evolutionary Algorithm Toolbox for use with Matlab version: toolbox 1.95 documentation 1.95 (August 1999)

[4] Help with Matlab Script. Available http://www.mathworks.com/help/optim/ug/fmincon.html [01/05/2016]

[5] Yuhui Shi , Particle swarm optimization: Development, applications and resources, Conference Paper · February 2001 DOI: 10.1109/CEC.2001.934374 · Source: IEEE Xplore

[6] Gon¸calo Pereira , Particle Swarm Optimization INESC-ID and Instituto Superior T´ecnico Porto Salvo, Portugal