OPTIMIZACIÓN DE PESO PARA ARMADURAS 3-D EMPLEANDO PSO (*PARTICLE SWARM OPTIMIZATION)* Y MSPSO (*MULTI-SPECIES PARTICLE SWARM OPTIMIZATION)*

RAFAEL ANDRÉS RUEDA MONSALVE

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍA CIVIL
BUCARAMANGA
2017

OPTIMIZACIÓN DE PESO PARA ARMADURAS 3-D EMPLEANDO PSO (*PARTICLE SWARM OPTIMIZATION)* Y MSPSO (*MULTI-SPECIES PARTICLE SWARM OPTIMIZATION)*

RAFAEL ANDRÉS RUEDA MONSALVE

Trabajo de grado para optar al título de Ingeniero Civil

DIRECTOR
OSCAR JAVIER BEGAMBRE CARRILLO
PhD. En Ingeniería Civil

CODIRECTOR
JESUS DANIEL VILLALBA MORALES
PhD. En Ingeniería Civil

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍA CIVIL
BUCARAMANGA
2017

AGRADECIMIENTOS

A mi Familia, en especial a mi Madre.

“¡Triunfar, no es lo más importante… es lo único importante!”.

Jorge Duque Linares.

CONTENIDO

Pág.

LISTA DE ILUSTRACIONES

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: OPTIMIZACIÓN DE PESO PARA ARMADURAS 3-D EMPLEANDO PSO (*PARTICLE SWARM OPTIMIZATION)* Y MSPSO (*MULTI-SPECIES PARTICLE SWARM OPTIMIZATION).*[*]

AUTOR: RAFAEL ANDRÉS RUEDA MONSALVE[**]

PALABRAS CLAVE: Optimización, PSO, MSPSO, Armaduras 3-D

DESCRIPCION:

Las estructuras metálicas son ampliamente empleadas en la infraestructura de comunicaciones y de energía en el país, estas van extendiendo su cobertura a lo largo del territorio nacional, debido a ventajas como facilidad en el transporte al sitio de construcción y rapidez de ensamble, por diversos métodos, ya sea soldadura, pernos o remaches. Además de explorar nuevas zonas o mejorar su capacidad, el diseño puede ser una tarea compleja y meticulosa, debido a su costo elevado y la gran cantidad de tiempo que se invierte, y finalmente, éste aumenta el uso de los recursos computacionales que se llegasen a utilizar.

En esta investigación se emplean los algoritmos PSO (particle swarm optimization) [1], [6] y el multi-especie PSO - MSPSO [10] con el fin de optimizar el peso de armaduras 3-D, mediante un programa en el software Matlab®[1] [12] se realizó el proceso de optimización, el cual incluye el análisis matricial de las estructuras cumpliendo con el método Allowable Stress Design. Se analizan dos armaduras de 25 y 72 elementos respectivamente. La validación del proceso de optimización se realiza con base en lo reportado en la literatura para establecer sus semejanzas o diferencias. El peso mínimo como el número de evaluaciones de la función objetivo obtenidos, para las dos estructuras estudiadas, disminuyeron con respecto a lo conseguido en otras investigaciones.

[*] Proyecto de grado

[**] Facultad de Ingenierias Fisicomecanicas. Escuela de Ingeniería Civil. Director: Oscar Javier Begambre. Co-director: Jesus Daniel Villalba

ABSTRACT

TITLE: weigh Optimization for 3-D armors by using PSO (particle swarm optimization) and MSPSO (multi-species particle swarm optimization).*

AUTHOR: RAFAEL ANDRÉS RUEDA MONSALVE**

KEYWORDS: Optimization, PSO, MSPSO, Armaduras 3-D

DESCRIPTION:

The metallic structures are widely employed in the communicative infrastructure and energy in this country. These are spreading their coverture throughout national territory due to some advantages like transportation to the placement in construction and the velocity to build it up. By using different methods such as welding, bolts or rivets: Moreover, exploring new placements or improving its capacity, the design may be a complex work and carefully, due to its expensive cost and the quantity of time that would be invested, to finally, it can extend the use of the computational resources that can be spent.

In this research, it was employed the particle swarm optimization (PSO) [1], [6] and the multi-species PSO - MSPSO algorithms [10] in order to optimize the weight of 3-D trusses. Throughout a program performed on the software Matlab® [12]. It was used the the optimization process, which includes the matrix analysis of structures by following the methodology of Allowable Stress Design. It was analyzed in Two Steel trusses of 25 and 72 elements respectively. The validation of the optimization process was made based on what has been reported in the literature. To assimilate their similarities and differences. The minimum weight and the number of evaluations of the objective function obtained, for the two structures studied, decreased with respect to what has been achieved in other investigations.

* Undergraduate thesis

** Facultad de Ingenierias Fisicomecanicas. Escuela de Ingenieria Civil. Advisor: Oscar Javier Begambre. Co-advisor: Jesus Daniel Villalba..

1. INTRODUCCIÓN

Las estructuras metálicas son ampliamente empleadas en la infraestructura de comunicaciones y de energía en el país, debido a beneficios como: una instalación más rápida, facilidad de transporte al sitio de construcción y rapidez de ensamble (sin requerir ninguna herramienta sofisticada para su montaje). Adicionalmente, ofrecen una larga vida útil y funcionamiento seguro.

Estas estructuras son variables (en geometría y dimensiones) debido a su configuración y propósito. En este sentido, el diseño óptimo se hace necesario teniendo en cuenta su nivel de importancia y su costo de fabricación elevado (cuando se va a producir un número importante de unidades), y además los diseños toman una gran cantidad de tiempo aumentando la cantidad de recursos computacionales a utilizar.

Es por esto que resulta beneficiosa la creación de un algoritmo que permita solucionar esta problemática con el objetivo de simplificar y optimizar los diseños, la implementación de los algoritmos meta heurísticos en las estructuras metálicas ha atraído la atención de un gran número de investigadores en el área de Optimización estructural. Una razón de esta inclinación, es la motivación de reducir el uso de recursos humanos y computacionales, que tradicionalmente dependen de la experiencia del ingeniero y, por lo tanto, una gran reducción en los tiempos y costos de diseño.

Entre los métodos meta heurísticos, más empleados se destacan, el *Algoritmo de Optimización por Enjambre de Partículas (PSO), los Algoritmos Genéticos (AG), Estrategias Evolutivas (EE), Algoritmo Evolución Diferencial (DE)*, entre otras.

En este trabajo, se emplea el algoritmo *de Optimización por Enjambre de Partículas (PSO)* [1], [6], añadiendo una modificación a su método de convergencia, clasificando las partículas en conjuntos, de acuerdo a su proximidad relativa, centrándose en los óptimos globales de una función multimodal. Esta versión es llamada por sus autores *Multi-especie PSO (MSPSO) [10]*.

Posteriormente, se determinan las variables, restricciones y la función objetivo del problema. Finalmente se obtiene el mínimo global que representa el menor peso para dos armaduras de 25 y 72 elementos, seleccionadas, como casos de estudio, de las literaturas [8], [9], [11], [15]. La efectividad del proceso de optimización se establece mediante la comparación de los resultados reportados en los artículos internacionales, en donde se presentan diferentes soluciones al problema planteado. Los esfuerzos y deformaciones son verificados por el método ASD, método usados por los autores en la solución de las armaduras propuestas.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Optimización de peso para estructuras tipo cercha 3-D empleando el algoritmo *Particle Swarm Optimization* y *Multi-Species Particle Swarm Optimizer.*

2.2. OBJETIVOS ESPECIFICOS

- Programar el algoritmo de optimización *Particle Swarm Optimization* o *Multi-Specie Particle Swarm Optimizer.*

- minimizar el peso de una estructura tipo cercha (diseñada por método *Allowable Stress Design).*

- Comparar los resultados de optimización obtenidos con los reportados en la literatura.

3. MARCO TEÓRICO

3.1. PARTICLE SWARM OPTIMIZACION – PSO

La técnica de *Optimización por Enjambre de Partículas (PSO),* presentado por Eberhart y Kennedy en 1995, está inspirada en el comportamiento de los bancos de aves y peces, por ende en general, para este algoritmo se inicia ubicando aleatoriamente una población de partículas $(i = 1,2, \dots, N)$ en un espacio n-dimensional limitado, en donde las partículas evalúan la calidad de su posición temporal mediante una función objetivo, condicionada de acuerdo al problema, que puede buscar maximización o minimización. En cada iteración, las partículas comparten información entre ellas, permitiendo llegar con mayor rapidez al óptimo global.

En este caso la función objetivo determinará el mejor valor por iteración, a este valor se le llamará la mejor posición actual (Pbest). A medida que aumentan las iteraciones el programa escogerá el mejor valor denominándolo la mejor posición global (Gbest), con los valores de posición temprana, mejor posición actual y mejor posición global determinamos la velocidad a la cual la programación desplaza la partícula cambiando su ubicación en cada iteración, y posteriormente su nueva posición.

Las posiciones temporales se determinan de acuerdo con las siguientes expresiones:

$$X_i = \left(x_{i1,} x_{i2,} \dots, x_{in}\right) \qquad (1)$$

Mejor posición actual:

$P_i = (p_{i1,} p_{i2,} \dots, p_{in})$ (2)

Mejor posición global:

$G_i = (g_{i1,} g_{i2,} \dots, g_{in})$ (3)

Velocidad:

$V_i = (v_{i1,} v_{i2,} \dots, v_{in})$ (4)

Modificación en velocidad es:

$$V_{i(t+1)} = w * V_{it} + \emptyset_1 * \rho_1 * (Pbest_{it} - X_{it}) + \emptyset_2 * \rho_2 * (Gbest_{it} - X_{it}) \quad (5)$$

En donde, w es un factor de inercia entre [0, 1], V_{it} es la velocidad de la iteración anterior, $\emptyset_1$ y $\emptyset_2$ son valores aleatorias en rangos de [0, 2], ρ_1 y $\rho_2$son coeficientes de aprendizaje personal y social respectivamente, [1].

Modificación en posición viene dada por:

$$P_{i(t+1)} = X_{it} + V_{i(t+1)} \quad (6)$$

Donde, $P_{i(t+1)}$ es el vector donde se almacena las nuevas posiciones en la iteración $(t + 1)$, X_{it} es la posición temporal de la partícula, y $V_{i(t+1)}$ es la velocidad obtenida con la ecuación (5).

3.2. MULTI-SPECIE PSO – MSPSO

El *MSPSO*, desarrollado por *Masao Iwamatsu* [10], Secciona la población en conjuntos, los cuales ocupan un área determinada convirtiendo cada conjunto en

una posible solución, esto se logra, determinando la distancia euclidiana $d(i,j)$, entre su mejor posición actual $(P_i,\ P_j)$ y su posición temporal $(x_i,\ x_j)$ [10].

$$d(i,j) = \sqrt{\sum_{k=1}^{n}(P_{ik} - P_{jk})^2} \qquad (7)$$

Luego, se establece un radio (σ), donde se compara la magnitud de la distancia con el radio establecido, las partículas cuya distancia sea menor al radio, se extraerán formando el primer conjunto, llamado por el autor *especie* (P_s), el programa realizará este proceso hasta que todas las partículas $(i = 1,2,\dots,M)$ pertenezcan a una especie, idealizado por *Iwamatsu* como:

$$d(i,j) < \sigma \ \rightarrow i \in \ P_s \qquad (8)$$

Una vez definidas las especies, se empieza con el proceso iterativo, que es similar al PSO tradicional, pero en este proceso, el PSO tradicional utilizaba la mejor posición actual por iteración y la mejor posición global de todas las iteraciones, el MSPSO realizará esta acción por cada especie, es decir, cambiará su velocidad empleando la mejor posición personal y su mejor posición global por cada especie, este paso altera la ecuación (5) quedando:

$$V_{i(t+1)} = w * V_{it} + \emptyset_1 * \rho_1 * (P_{it} - X_{it}) + \emptyset_2 * \rho_2 * (P_{st} - X_{it}) \qquad (9)$$

Donde, P_s es la mejor posición de la especie a la que pertenece la partícula i [10]. Finalmente, se actualiza la posición de la partícula, usando la ecuación (6).

4. METODOLOGÍA DE DISEÑO

4.1. FUNCIÓN OBJETIVO

Un problema de optimización puede ser formulado como un proceso donde se desea encontrar el valor óptimo que minimiza o maximiza la función objetivo $f(x)$ [6]. En esta investigación, se plantea la minimización del peso de armaduras 3D optimizando el área de cada elemento satisfaciendo las restricciones y determinando las variables presentes en las armaduras, estas ecuaciones son propuestas por Sebastián Echeverri [15] así:

Función a minimizar:

$$W = \rho \sum_{i=1}^{N} l_i A_i \qquad (i = 1,2,\dots,N) \qquad (10)$$

Con las restricciones,

$$\delta_j(A_i) \leq \delta_{máx} \qquad (j = 1,2,\dots,NN) \qquad (11)$$

$$\sigma_k(A_i) \leq \sigma_{adm} \qquad (k = 1,2,\dots,N) \qquad (12)$$

Y las variables,

$$A_{min} \leq A_i \leq A_{máx} \qquad (13)$$

Donde, W es el peso de la armadura, ρ es el peso específico del acero, l_i y A_i, longitud del elemento y área de la sección transversal del elemento i, respectivamente, N es el número de elementos en la estructura, $\delta_j(A_i)$ representa el desplazamiento en el nudo j, $\delta_{máx}$ desplazamiento máximo admisible, NN numero

de nodos, $\sigma_k(A_i)$ esfuerzo máximo en el elemento k, σ_{adm} esfuerzo máximo admisible en el elemento.

4.2. ALGORITMO DE OPTIMIZACIÓN

El proceso de optimización, seguido en este trabajo, se representa en la ilustración 1. La programación comienza ingresando las propiedades geométricas de la armadura, como lo son, sus coordenadas nodales, conexiones entre elementos, condiciones de carga y sus respetivas áreas, A continuación, el programa iniciará con el PSO o el MSPSO, asignando una población de partículas, evaluando mediante un análisis matricial sus deformaciones y esfuerzos, luego se verifica que estén dentro de los rangos establecidos por el método ASD y, finalmente, se analiza la función objetivo.

El programa modifica su velocidad y posición y realizara el análisis matricial por cada iteración nueva hasta cumplir con el número de iteraciones registradas por la literatura [15], además ajusta su velocidad y posición de acuerdo con las ecuaciones (9) y (6). En el anexo B se presenta el código del PSO programado por el autor en Matlab®[1] y empleado en este estudio.

4.3. CONFIGURACIÓN DEL MSPSO

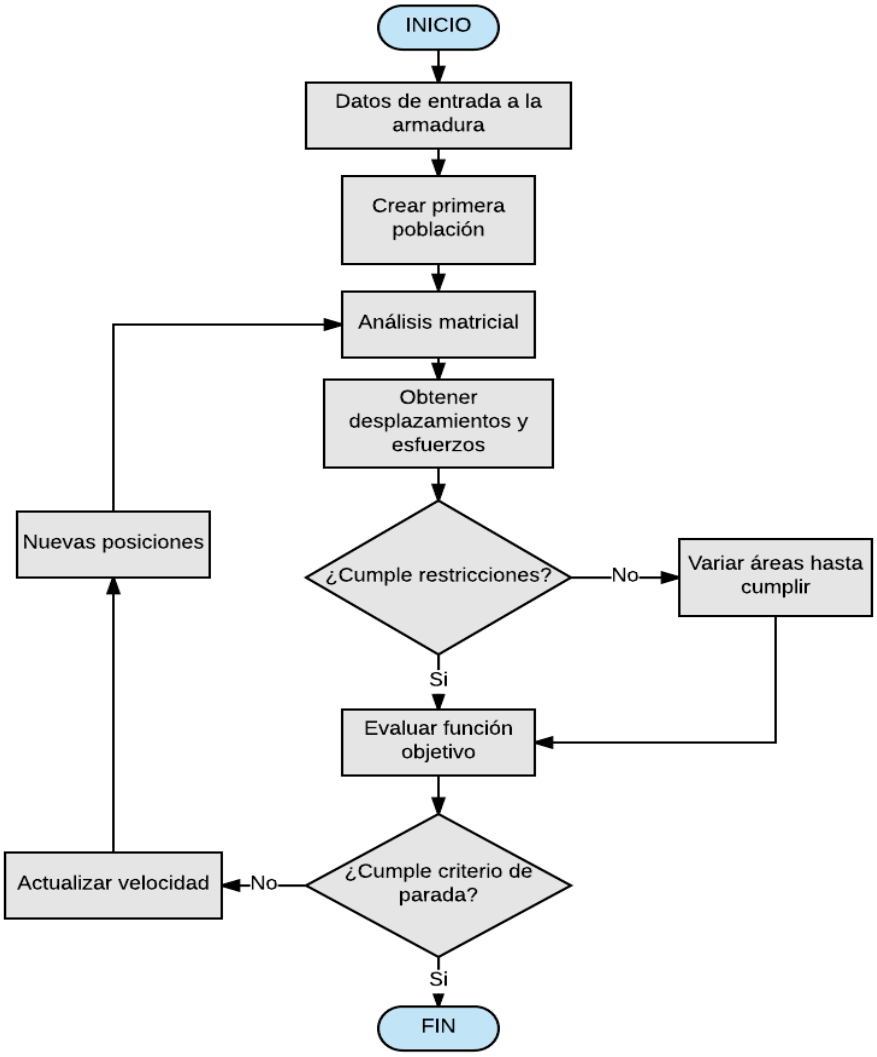
La interpretación del algoritmo se representa en la matriz de posición *(M.P.)*, en que sus columnas son el número de áreas a determinar en la estructura y las filas el número de partículas en el enjambre.

$$M.P. = \begin{bmatrix} 0.5 & 0.7 & 1 \\ 0.1576 & 0.245 & 0.453 \\ 0.356 & 0.654 & 0.324 \end{bmatrix}$$

En este ejemplo, se ubican 3 áreas y 3 partículas, en la matriz *M.P.* de tamaño 3x3, los valores 0.5, 0.7, 1 en la primera fila corresponden a diferentes áreas (A1, A2, A3) aumentando de izquierda derecha y de arriba abajo dependiendo del número de partículas, obteniendo las mejores soluciones.

Los valores asignados para coeficiente personal ρ_1y social ρ_2 son iguales a 2 para cada uno. El factor de inercia W usado es 0,95. (Ver anexo C).

Ilustración 1. Diagrama de flujo de algoritmo de optimización.

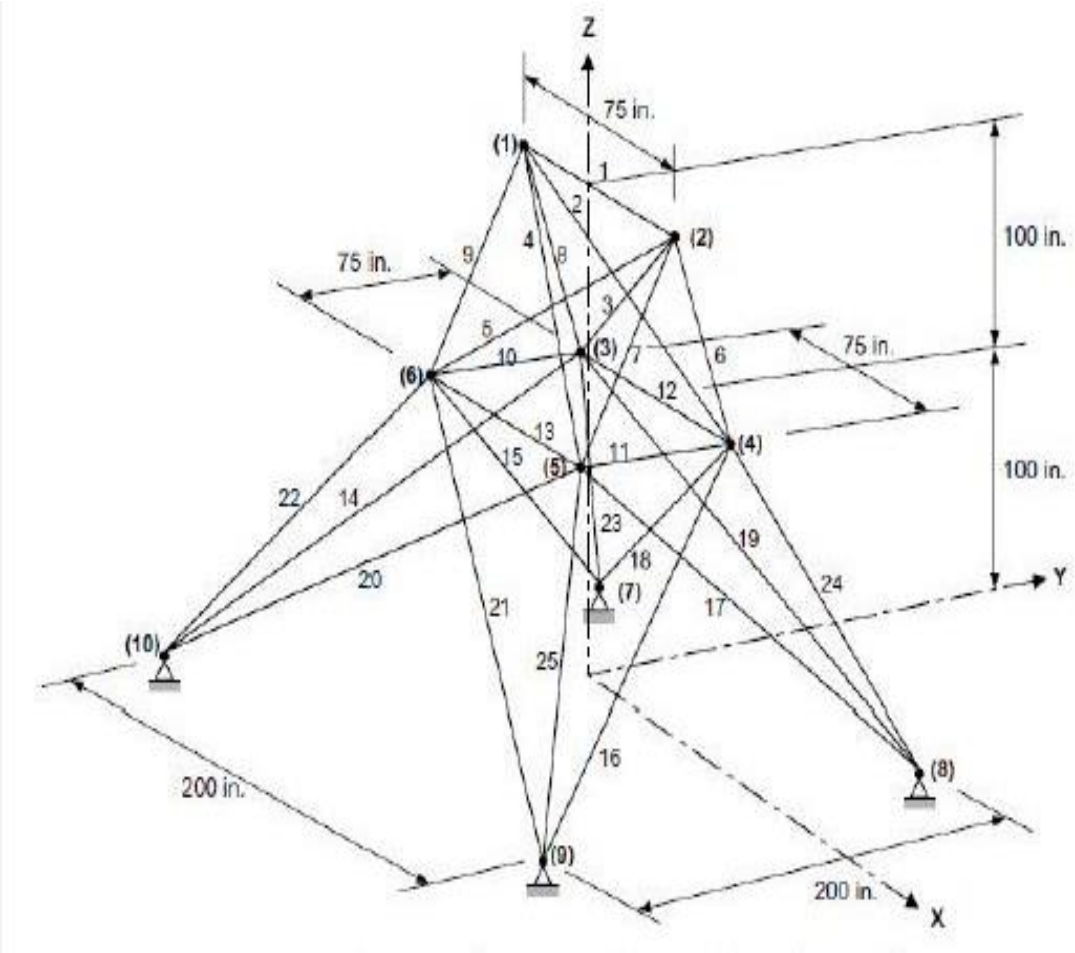


4.4. ANÁLISIS MATRICIAL

Para verificar que la programación del análisis matricial realizada en este trabajo (realizada en *MATLAB® [12]*[1]) es correcta, se determinaron los desplazamientos y esfuerzos presentes en la armadura de 25 elementos (ver ilustración 2) y se compararon estos resultados con los obtenidos en el programa *SAP2000* [3][1] como se observa en las tablas 1 y 2. En la sección 4.4.1, se presenta la topología de la armadura en el software SAP2000, sus cargas y resultados de deformación, en la sección 4.4.2 la programación en el software MATLAB®, sus matrices locales y los resultados en la Tablas 1 y 2, se consideran que todos los elementos manejan el sistema ingles de unidades y tienen las siguientes características:

- -Barras circulares de acero (Φ=0,7979 in)
- -Área (A=0,5 in^2)
- -Módulo de Elasticidad (E=29000 Ksi)

Ilustración 2. Armadura 3-D 25 elementos.



Fuente: Sebastián Echeverri [15]

4.4.1. Programación software SAP2000. En esta sección se realiza la estructura en el software SAP2000, reportando sus resultados de desplazamientos y esfuerzos en las tablas 1 y 2, respectivamente, en la ilustración 3, se observa la estructura ingresada en SAP2000, sus cargas en la ilustración 4 y los resultados de la deformación en los nodos en la ilustración 5.

Ilustración 3. Topología en el software SAP2000.

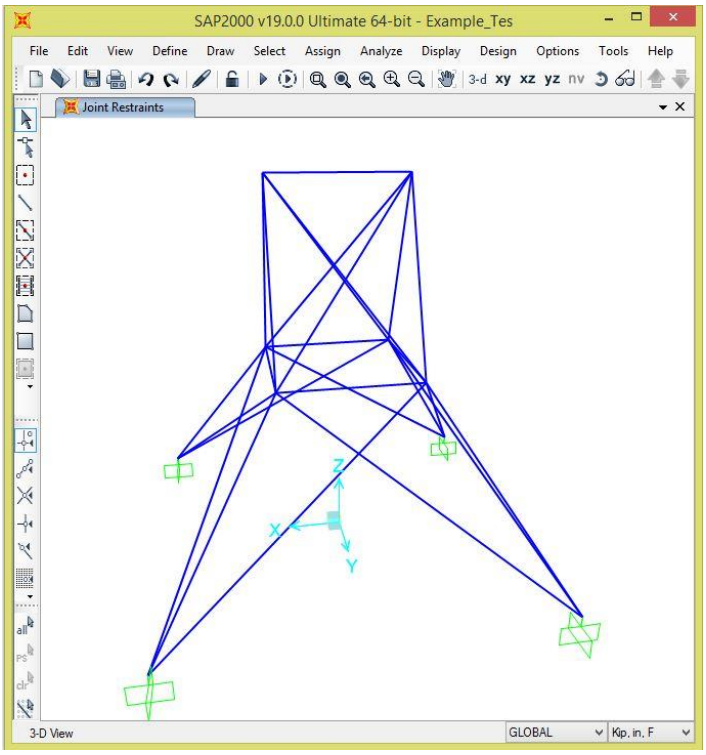


Ilustración 4. Condiciones de carga en SAP2000.

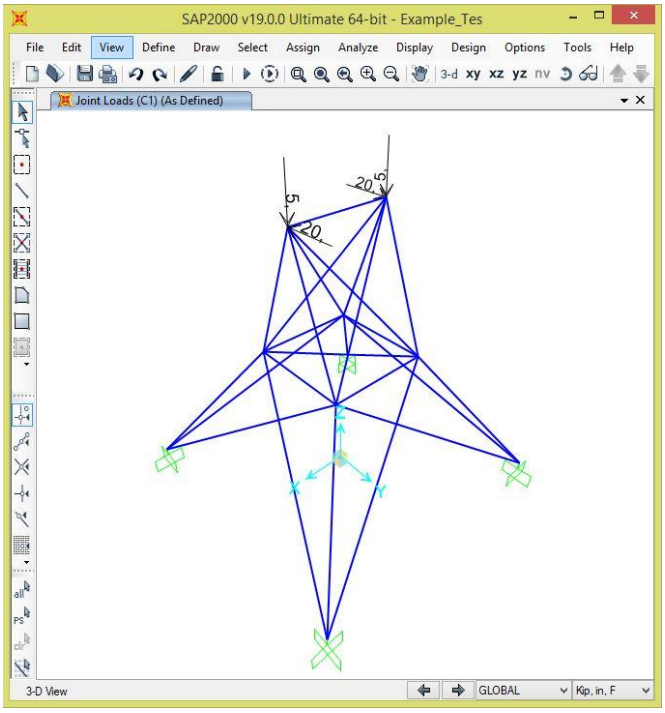
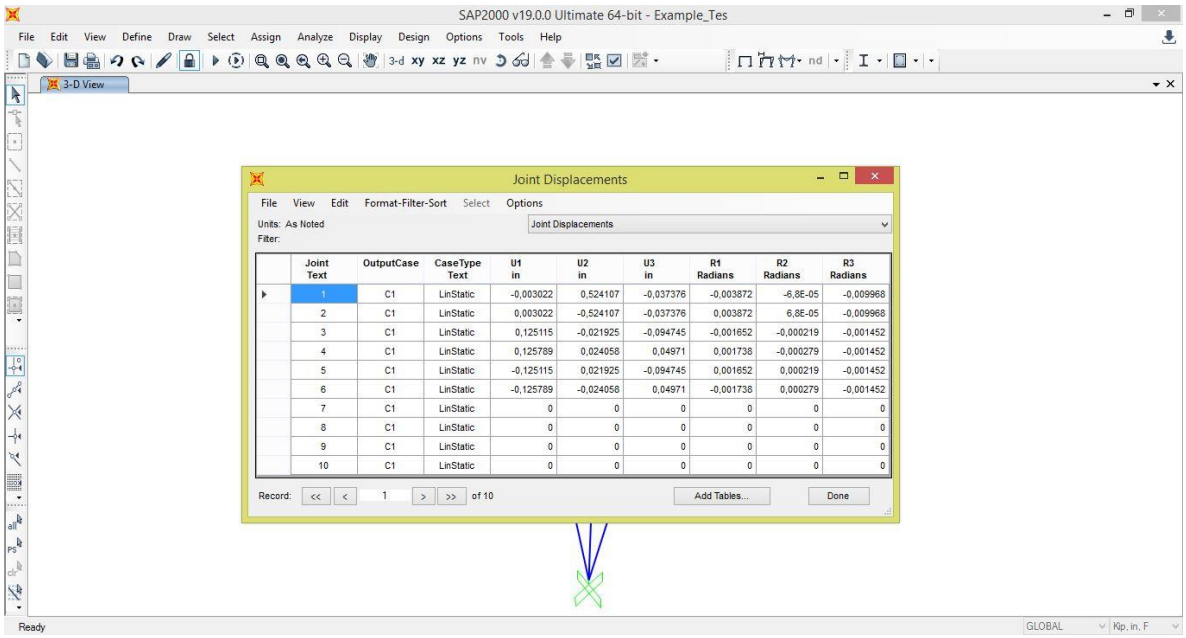


Ilustración 5. Resultados desplazamientos SAP2000.



4.4.2. Programación software MATLAB®. Se ingresan los datos de la armadura como su geometría, conexión, restricciones, cargas, para iniciar con el proceso del análisis matricial, en la ilustración 6 se muestran los datos de entrada de la armadura y en las ilustraciones 7 a 10, se muestran las matrices locales de rigidez, seguidos de sus resultados en el software Matlab® (Ver anexo A).

Ilustración 6. Propiedades de la armadura en Matlab.

```
%=========================================================================
%PARAMETROS DEL PROBLEMA

%Número de barras
nnod=10;
%Número de nodos
nbar=25;
%Densidad en lb/in3
dens=0.1;
%Modulo de young en ksi
elas=29000;
%Coordenadas en in
coor=[-37.5,0,200;  37.5,0,200;  -37.5,37.5,100;  37.5,37.5,100; 37.5,-37.5,100;
         -37.5,-37.5,100;   -100,100,0;   100,100,0;   100,-100,0;     -100,-100,0];
%Conexiones nodo inicial y final
conx=[1,2;1,4;2,3;1,5;2,6;2,4;2,5;1,3;1,6;
      6,3;4,5;3,4;6,5;3,10;6,7;4,9;5,8;
      4,7;3,8;5,10;6,9;6,10;3,7;4,8;5,9];
%Condicion de carga 1 en kips
carg=[0,20,-5;  0,-20,-5;  0,0,0;  0,0,0;
    0,0,0;    0,0,0;     0,0,0;  0,0,0;
    0,0,0;    0,0,0];
%Definir nodos con apoyos ("1" si lo hay)
apoy=[0,0,0;  0,0,0;  0,0,0;  0,0,0;
      0,0,0;  0,0,0;  1,1,1;  1,1,1;
     1,1,1;  1,1,1];
%Áreas en in2
area=[0.5;0.5;0.5;0.5;0.5;0.5;0.5;0.5];
%variacion de areas
vare=[1;4;4;2;2;4;4;4];
%=========================================================================
```

Ilustración 7. Matriz de rigidez local elemento 1.

```
rig5 =

  193.3333         0         0 -193.3333         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
 -193.3333         0         0  193.3333         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

Ilustración 8. Matriz de rigidez local elementos 2 al 5.

```
rig5 =

  111.1078         0         0 -111.1078         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
 -111.1078         0         0  111.1078         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

Ilustración 9. Matriz de rigidez local elementos 6 al 9.

```
rig5 =

  135.7677         0         0 -135.7677         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
 -135.7677         0         0  135.7677         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

Ilustración 10. Matriz de rigidez local elementos 10 al 13.

```
rig5 =

  193.3333         0         0 -193.3333         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
 -193.3333         0         0  193.3333         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

Ilustración 11. Matriz de rigidez local elementos 14 al 21.

```
rig5 =

   80.0476         0         0  -80.0476         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
  -80.0476         0         0   80.0476         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

Ilustración 12. Matriz de rigidez local elementos 21 al 25.

```
rig5 =

  108.6440         0         0 -108.6440         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
 -108.6440         0         0  108.6440         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

4.4.3. Resultados SAP2000 y MATLAB®. Según se desprende las tablas 1 y 2, el análisis matricial programado en *MATLAB*® proporciona resultados correctos.

Tabla 1. Comparación de los desplazamientos en los nodos.

Desplazamientos			
Nodo	SAP2000®		
	U1	**U2**	**U3**
1	-0,003022	0,524107	-0,037376
2	0,003022	-0,524107	-0,037376
3	0,125115	-0,021925	-0,094745
4	0,125789	0,024058	0,04971
5	-0,125115	0,021925	-0,094745
6	-0,125789	-0,024058	0,04971
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
Nodo	MATLAB®		
	U1	**U2**	**U3**
1	-0,003	0,5244	-0,0374
2	0,003	-0,5244	-0,0374
3	0,1252	-0,022	-0,0948
4	0,1259	0,0242	0,0498
5	-0,1252	0,022	-0,0948
6	-0,1259	0,0242	0,0498
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

Tabla 2. Comparación de los esfuerzos en los elementos.

Elemento	Fuerzas Internas (KN)	
	SAP2000®	MATLAB®
1	1,168	1,1684
2	-15,155	-15,1598
3	13,122	13,1267
4	13,122	13,1267
5	-15,155	-15,1598
6	15,062	15,0676
7	-18,738	-18,7437
8	-18,738	-18,7437
9	15,062	15,0676
10	0,412	0,4124
11	0,412	0,4124
12	0,13	0,1303
13	0,13	0,1303
14	-2,064	-2,0699
15	0,184	0,1907
16	0,184	0,1907
17	-2,064	-2,0699
18	9,176	9,1833
19	-11,184	-11,1915
20	-11,184	-11,1915
21	9,176	9,1833
22	-3,577	-3,581
23	-0,232	-0,228
24	-3,577	-3,581
25	-0,232	-0,228

Nota: las fuerzas se indician si son a tensión o compresión por el signo.

5. EJEMPLOS ANALÍTICOS PSO

En esta sección se definen en detalle dos ejemplos clásicos reportados en la literatura. Estos ejemplos han sido analizados en diversas investigaciones y con variadas metodologías de optimización. Se empezó demostrando la validez del algoritmo con 3 funciones extraídas de la literatura [10] y con el proceso de análisis de las dos armaduras, finalizando con la comparación de los resultados que se obtienen mediante el uso del algoritmo PSO (ver tabla 7).

El algoritmo PSO (ver anexo B) se validó por medio de las funciones de la literatura [10], para demostrarlo se tomaron tres funciones de la literatura (ver ecuaciones 14, 15 y 16).

$$f(x) = \begin{cases} \frac{-160}{15}(15 - x), & 0 \leq x \leq 15 \\ \frac{-200}{5}(x - 15), & 15 \leq x \leq 20 \end{cases} \quad (14)$$

$$f(x) = \begin{cases} \frac{-160}{10}X, & 0 \leq X < 10 \\ \frac{-160}{5}(15 - X), & 10 \leq X < 15 \\ \frac{-200}{5}(X - 15), & 15 \leq X \leq 20 \end{cases} \quad (15)$$

Los autores proponen usar una población de 50 particulas, con una velocidad máxima de $v_{máx} = 0.5$, y un máximo de iteraciones $MAX = 40$ para la ecuaciones 14 y 15, los parámetros se muestran en la ilustración 13 y las funciones programadas en MATLAB® en la ilustración 14 y 15, respectivamente.

Ilustración 13. Parámetros algoritmo PSO (ecuación 14 y 15).

```
%=================================================================
%PARAMETROS DEL PROBLEMA
ejec=1; %ejecuciones pso
iter=40; %iteraciones pso
part=50; %particulas pso
wpso=0.95; %w pso [valor entre 0 y 1]
c1pso=2; %c1 pso [valor entre 0 y 2]
c2pso=2; %c2 pso [valor entre 0 y 2]
%=================================================================
```

Ilustración 14. Función de calidad (ver ecuación [14]).

```
%CALIDAD
for ccc1=1:part
    if posi(ccc1,1)<15
        cali(ccc1,1)=(-160/15)*(15-posi(ccc1,1));
    end
    if posi(ccc1,1)>15
        cali(ccc1,1)=(-200/5)*(posi(ccc1,1)-15);
    end
end
```

Ilustración 15. Función de calidad (ver ecuación [15]).

```
%CALIDAD
for ccc1=1:part
    if posi(ccc1,1)<10
        cali(ccc1,1)=(-160/10)*(posi(ccc1,1));
    end
    if posi(ccc1,1)>10 && posi(ccc1,1)<15
        cali(ccc1,1)=(-160/5)*(15-posi(ccc1,1));
    end
    if posi(ccc1,1)>15
        cali(ccc1,1)=(-200/5)*(posi(ccc1,1)-15);
    end
end
```

Para la ecuación 16, disponen una $v_{máx} = 0.1$ y $MAX = 10$ mantiene su población de 50 partículas, como se muestra a continuación sus parámetros (ver ilustración 16) y función (ver ilustración) programada en MATLAB.

$$f(x) = -sin^6(5\pi x) \quad 0 \leq x \leq 1 \qquad (16)$$

Ilustración 16. Parámetros algoritmo PSO (ecuación 16).

```
%=================================================================
%PARAMETROS DEL PROBLEMA
ejec=1; %ejecuciones pso
iter=10; %iteraciones pso
part=50; %particulas pso
wpso=0.95; %w pso [valor entre 0 y 1]
c1pso=2; %c1 pso [valor entre 0 y 2]
c2pso=2; %c2 pso [valor entre 0 y 2]
%=================================================================
```

Ilustración 17. Función de calidad (ver ecuación [16]).

```
%CALIDAD
for ccc1=1:part
    cali(ccc1,1)=-(sin(5*pi()*posi(ccc1,1)))^6;
end
```

Encontrando que el algoritmo converge al óptimo global un 80 porciento de 10 ejecuciones independiente de la función programada, validando el algoritmo realizado en el MATLAB®.

5.1. PRIMER EJEMPLO

El primer ejemplo consiste en una armadura de 25 elementos, 4 apoyos de tercer grado y 18 grados de libertad. La configuración se muestra en la ilustración 2, el problema de optimización está condicionado por áreas discretas de 0.1, 0.2, 0.3,

0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 3.0, 3.2 y 3.4 in^2. Las variables de diseño, condiciones de carga y restricciones de la estructura se muestran en las tablas 3 a 6.

Tabla 3. Propiedades de los materiales para armadura de 25 elementos.

Propiedades de los materiales para armadura de 25 elementos	
Módulo de Elasticidad (ksi)	10000
Densidad del Material (lb/in^3)	0.10

Tabla 4. Restricciones para armadura de 25 elementos.

Restricciones para armadura de 25 elementos		
Esfuerzos Limite		
Variables de Diseño (in^2)	Esfuerzos Limite a Compresión (ksi)	Esfuerzo Limite a Tensión (ksi)
A_1	35.09	40.00
A_2	11.59	40.00
A_3	17.31	40.00
A_4	35.09	40.00
A_5	35.09	40.00
A_6	6.96	40.00
A_7	6.96	40.00
A_8	11.08	40.00

Desplazamientos admisibles
$\Delta_j \leq 0.35$ in en las direcciones *x* y *y*, j=1,2,…25.

Tabla 5. Condiciones de carga para la armadura de 25 elementos.

Condiciones de carga para la armadura de 25 elementos						
Nodo	Condición de carga (kips)			Condición 2 (kips)		
	F_x	F_y	F_z	F_x	F_y	F_z
1	0.0	20.0	-5.0	1.0	10.0	-5.0
2	0.0	-20.0	-5.0	0.0	10.0	-5.0
3	0.0	0.0	0.0	0.5	0.0	0.0
6	0.0	0.0	0.0	0.5	0.0	0.0

Tabla 6. Variables de diseño para la armadura de 25 elementos.

Variables de diseño para la armadura de 25 elementos	
Variables de Diseño (in^2)	Elementos
A_1	1
A_2	2,3,4,5
A_3	6,7,8,9
A_4	10,11
A_5	12,13
A_6	14,15,16,17
A_7	18,19,20,21
A_8	22,23,24,25

5.1.1. Resultados y discusión. En la ilustración 18, se muestra la mejor solución encontrada por el algoritmo con poblaciones de 100 y 200 partículas. El algoritmo tuvo un parámetro de parada de 50 corridas. La ilustración 18, mantiene una línea recta debido a sus restricciones (ver tabla 5).

El peso final fue el mínimo valor entre todas las corridas, como se muestra en la ilustración 19, seguido, se extrae el cuadro comparativo de la literatura [15] (ver tabla 7), el cual se registra el autor, el algoritmo utilizado, la fecha de publicación del artículo, el número de análisis de la función objetivo, peso mínimo, si lo registra, y el valor por área.

Ilustración 18. . PSO, peso total vs Número de iteraciones, Armadura de 25 Elementos.

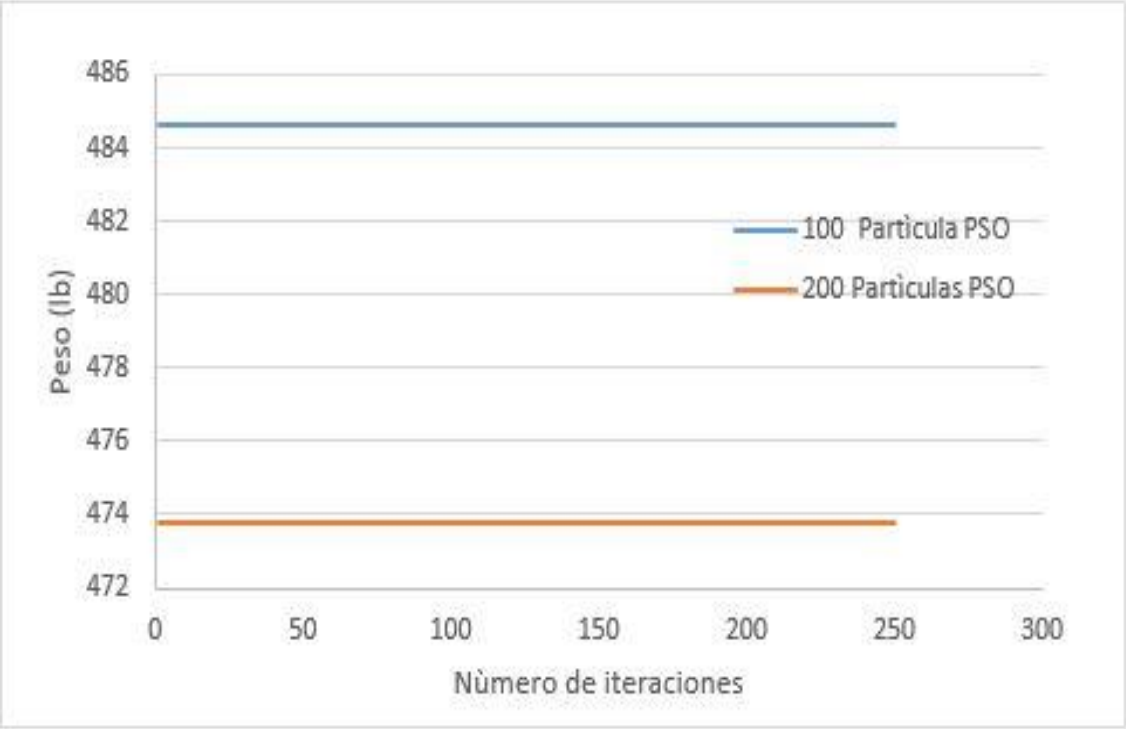


Ilustración 19. PSO, peso por corrida armadura de 25 elementos.

La mejor solución obtiene un peso mínimo en 50 corridas de $473{,}76\ lb$, con una desviación estándar de $26{,}008\ lb$, el algoritmo efectúa 250 iteraciones por corrida, sea de 100 o 200 partículas la población, por consiguiente, la función objetivo realizará 25.000 evaluaciones de la función objetivo para una población de 100 partículas y 50.000 evoluciones para una población de 200 partículas. En esta investigación se observa que el mejor resultado tiene una diferencia de aproximadamente $4{,}3\%$, con respecto al mínimo valor reportado por la literatura (ver tabla 7) [15].

Tabla 7. Tabla de comparación armadura 25 Elementos.

			Variables de Diseño (in^2)							
Método	Peso (lb)	Análisis *	A1	A2	A3	A4	A5	A6	A7	A8
Talaslioglu (2009) MPGA ps=300	486.29	12000	0.10	0.50	3.40	0.10	1.50	0.90	0.60	3.40
Talaslioglu (2009) BGAwEIS ps=300	485.90	60000	0.10	0.10	3.40	0.10	1.90	1.00	0.70	3.40
Li et al. (2009) HPSO	484.85	-	0.10	0.30	3.40	0.10	2.10	1.00	0.50	3.40
Hasancebi et al. (2009) PSO	484.85	1600	0.10	0.30	3.40	0.10	2.10	1.00	0.50	3.40
Hasancebi et al. (2009) HSH	484.85	2100	0.10	0.30	3.40	0.10	2.10	1.00	0.50	3.40
Hasancebi et al. (2009) SA	484.85	6624	0.10	0.30	3.40	0.10	2.10	1.00	0.50	3.40
Hasancebi et al. (2009) ES	485.05	4350	0.10	0.50	3.40	0.10	1.90	0.90	0.50	3.40
Hasancebi et al. (2009) AC	485.05	10050	0.10	0.50	3.40	0.10	1.90	1.00	0.40	3.40
Hasancebi et al. (2009) SGA	485.38	9050	0.10	0.20	3.40	0.10	2.00	1.00	0.60	3.40
Hasancebi et al. (2009) TS	485.57	1626	0.10	0.40	3.40	0.10	1.80	0.90	0.60	3.40
Dede et al. (2011) ps=20	535.57	2200	0.60	0.40	3.00	0.20	0.10	1.10	1.90	3.00
Dede et al. (2011) ps=40	506.58	4400	0.90	1.10	2.80	0.10	1.50	1.00	0.30	3.40
Dede et al. (2011) ps=200	484.85	29200	0.10	0.30	3.40	0.10	2.10	1.00	0.50	3.40
Echeverrí (2014) AGM ps=100	495.33	14800	0.10	1.10	3.20	0.10	0.10	1.10	0.50	3.40
PSO (2017) ps=100	484.61	25000	0.90	2.30	2.50	0.60	0.60	0.80	1.90	0.70
PSO (2017) ps=200	473.76	50000	0.60	2.20	2.30	0.80	1.10	0.60	2.10	0.60

ps: tamaño de la población, GA: Genetic Algorithm, GAOS: Genetic Algorithm Based Optimum Structural Design, HSH: Harmony Search Heuristic, BB-BC: Big-Bang Crunch Algorithm, PSO: Particle Swarm Optimization, HPSO: Heuristic Particle Swarm Optimization, AC: Ant Colony Optimization, MPGA: Multipopulation Based Genetic Algorithm, BGAwEIS: Bipopulation Based Genetic Algorithm with Enhanced Interval Search, SA: simulated Annealing; ES: Evolution Strategies, TS: Tabu Search; SGA: Simple Genetic Algorithm, AGM: Algoritmo Genético Multicromosoma.

*El número de análisis corresponde a la cantidad de evaluaciones de la función objetivo, es decir, al número de iteraciones requeridas multiplicado por el tamaño de la población.

Fuente: Sebastián Echeverrí [15]

Para validar que la gráfica 18, la cual mantiene la línea recta por sus fuertes restricciones se opta por modificar la tabla 4, unificando el valor del esfuerzo a compresión como se puede observar en la tabla 8. A continuación se evaluó el algoritmo con poblaciones de 100 partículas, 250 iteraciones y un criterio de parada de 50 corridas, los resultados se observan en las ilustraciones 20 y 21, respectivamente.

Tabla 8. Restricciones modificadas para armadura de 25 elementos.

Restricciones modificadas para armadura de 25 elementos		
Esfuerzos Limite		
Variables de Diseño (in^2)	Esfuerzos Limite a Compresión (ksi)	Esfuerzo Limite a Tensión (ksi)
A_1	20.00	40.00
A_2	20.00	40.00
A_3	20.00	40.00
A_4	20.00	40.00
A_5	20.00	40.00
A_6	20.00	40.00
A_7	20.00	40.00
A_8	20.00	40.00

Desplazamientos admisibles
$\Delta_j \leq 0.35$ in en las direcciones *x* y *y*, j=1,2,…25.

Ilustración 20. PSO, peso total vs Número de iteraciones, Armadura de 25 Elementos (restricciones Tabla 8).

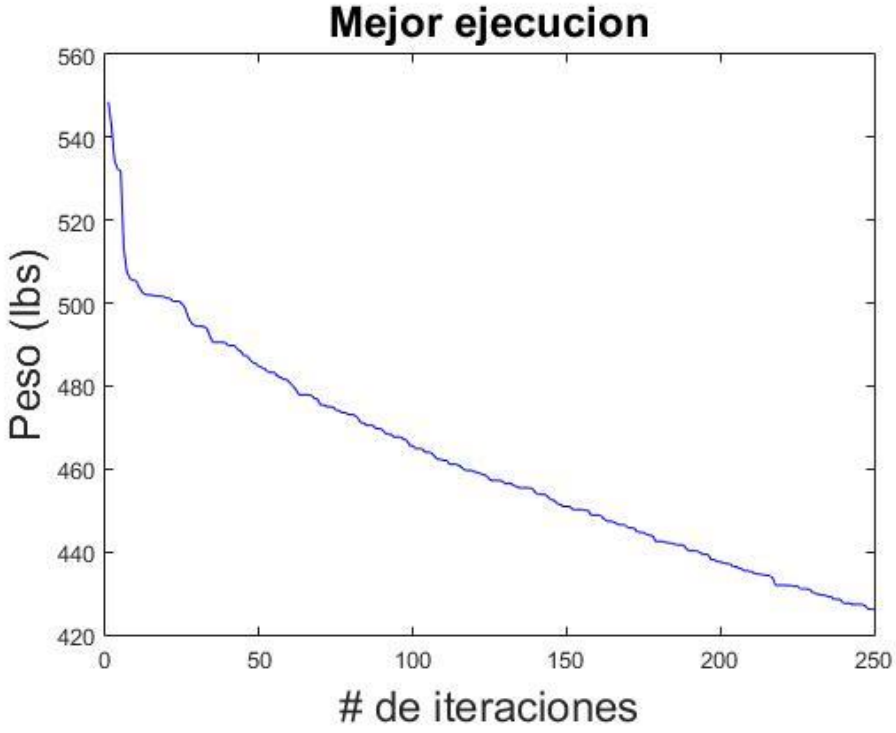
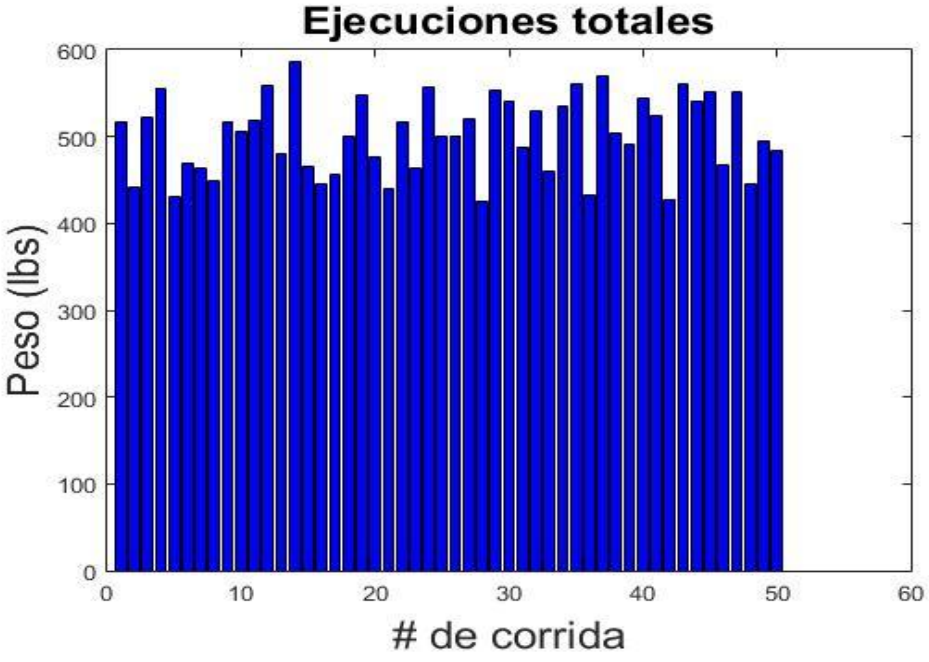


Ilustración 21. PSO, peso por corrida armadura de 25 elementos (restricciones tabla 8).

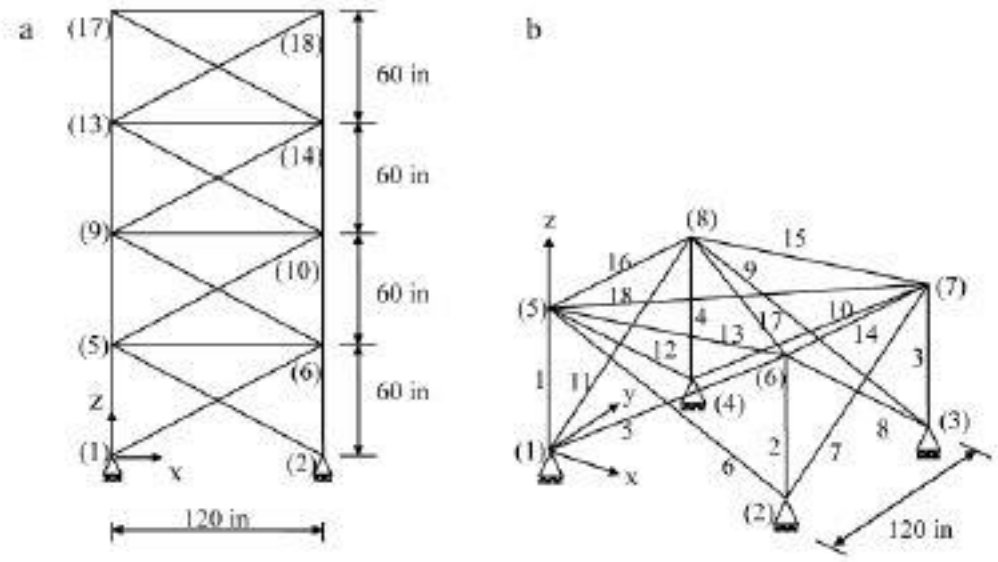


Al alterar sus restricciones tenemos en la ilustración 20 la convergencia del algoritmo y en la ilustración 21 los valores de las 50 corridas ejecutadas, validando la programación realizada en *MATLAB®.* (El valor obtenido de $426\,lb$ por esta ejecución no se tendrá en cuenta en la investigación).

5.2. SEGUNDO EJEMPLO

El segundo ejemplo consiste en una armadura de 72 elementos, 4 apoyos y 48 grados de libertad, la configuración geométrica se presenta en la ilustración 22. Las áreas discretas se modificaran empezando desde la menor área transversal 0.1 in^2 que tendrá un incremento de área de 0.001 in^2. Esta modificación es empleada por Sebastián Echeverri [15]. Para el análisis solo se trabajó la condición de carga número 2, basado en la literatura [15]. Las variables de diseño, condiciones de carga y restricciones de la estructura se muestran en las tablas 8 a 11.

Ilustración 22. Armadura 3-D 72 elementos.



Fuente: Sebastián Echeverri [15].

Tabla 9. Propiedades de los materiales para armadura de 72 elementos.

Propiedades de los materiales para armadura de 72 elementos	
Módulo de Elasticidad (ksi)	10000
Densidad del Material (lb/in^3)	0.10

Tabla 10. Restricciones para armadura de 72 elementos.

Restricciones para armadura de 72 elementos
Esfuerzos Limite
$-25\ ksi \leq (\sigma_a)_j \leq 25\ ksi, j = 1,2,\dots,72.$
Desplazamientos admisibles
$\Delta_j \leq 0.35$ in en las direcciones *x* y *y*, j=1,2,…20.
Secciones de Elementos
$0.10\ in^2 \leq A_j, j = 1,2,\dots,72.$

Tabla 11. Condiciones de carga para la armadura de 72 elementos.

Condiciones de carga para la armadura de 72 elementos						
Nodo	Condición 1 (kips)			Condición 2 (kips)		
	F_x	F_y	F_z	F_x	F_y	F_z
1	5.0	5.0	-5.0	0.0	0.0	-5.0
2	0.0	0.0	0.0	0.0	0.0	-5.0
3	0.0	0.0	0.0	0.0	0.0	-5.0
6	0.0	0.0	0.0	0.0	0.0	-5.0

Tabla 12. Condiciones de carga para la armadura de 72 elementos.

Condiciones de carga para la armadura de 72 elementos	
Variables de Diseño (in^2)	Elementos
A_1	1,2,3,4
A_2	5,6,7,8,9,10,11,12
A_3	13,14,15,16
A_4	17,18
A_5	19,20,21,22
A_6	23,24,25,26,27,28,29,30
A_7	31,32,33,34
A_8	35,36
A_9	37,38,39,40
A_{10}	41,42,43,44,45,46,47,48
A_{11}	449,50,51,52
A_{12}	53,54
A_{13}	55,56,57,58
A_{14}	59,60,61,62,63,64,65,66
A_{15}	67,68,69,70
A_{16}	71,72

5.2.1. Resultados armadura 72 elementos. Los resultados de la armadura de 72 elementos, al igual que la de 25 elementos, mostrarán su convergencia por medio de una gráfica para poblaciones de 100 partículas y 200 partículas, empleando el PSO, al igual, que las evaluaciones de la función objetivo y el valor de las 50 corridas por ejecución (ver ilustración 24), finalmente se llegara a una tabla comparativa con las investigaciones reportadas (ver tabla 13) extraída de la literatura [15].

Ilustración 23. PSO, peso total vs Número de iteraciones, Armadura de 72 Elementos.

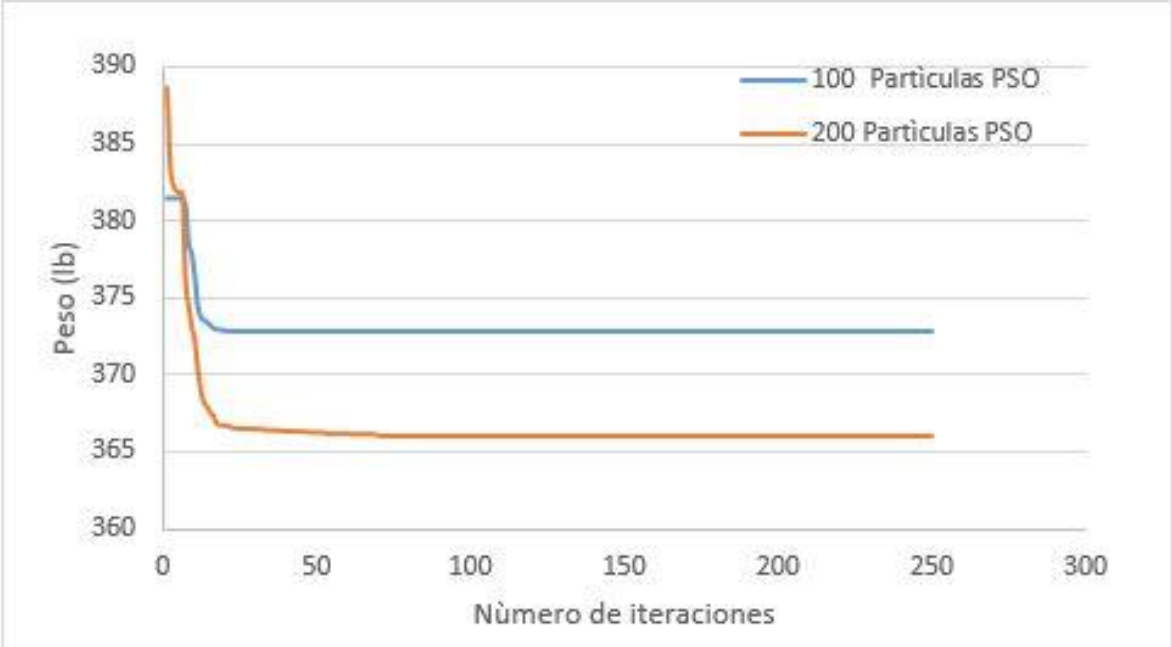


Ilustración 24. PSO, peso por corrida armadura de 72 elementos.

Para la armadura de 72 elementos, la mejor solución con una desviación estándar de $36{,}5\ lb$, obtiene el peso mínimo de $366{,}04\ lb$ en 50 corridas, validando este resultado con el peso mínimo reportado en la tabla 13. El algoritmo efectúa 25.000 evaluaciones de la función objetivo para 100 partículas y 50.000 para 200 partículas. Para este caso, se observa que la mejor respuesta tiene una diferencia de aproximadamente 4,6%, con respecto al mínimo valor reportado por la literatura [15] de $383.87\ lb$, (ver tabla 13).

Tabla 13. Tabla de comparación armadura 72 Elementos.

			Variables de Diseño (in^2)															
Método	Peso (lb)	Análisis *	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
Barbaso an Lemonge (2003) APM	385.46	-	0.151	0.556	0.501	0.681	0.576	0.423	0.100	0.160	1.286	0.439	0.100	0.114	2.125	0.522	0.100	0.100
Barbaso an Lemonge (2003) CPM	384.13	-	0.153	0.577	0.294	0.726	0.655	0.573	0.100	0.103	1.316	0.483	0.102	0.100	1.862	0.462	0.100	0.100
Sedaghati (2005) FM	379.62	-	0.157	0.546	0.410	0.570	0.524	0.517	0.100	0.100	1.268	0.512	0.100	0.100	1.886	0.512	0.100	0.100
Camp (2007) BB-BC	379.85	19621	1.858	0.506	0.100	0.100	1.248	0.527	0.100	0.101	0.521	0.517	0.100	0.100	0.157	0.551	0.382	0.592
Pérez y Behdinan (2007) PSO	381.91	-	0.162	0.509	0.497	0.561	0.514	0.546	0.100	0.110	1.308	0.519	0.100	0.100	1.743	0.519	0.100	0.100
Li et. al (2007) HPSO	369.65	125000	1.857	0.505	0.100	0.100	1.255	0.503	0.100	0.100	0.496	0.506	0.100	0.100	0.100	0.524	0.400	0.534
Talaslioglu (2009) MPGA ps = 500	594.81	25000	0.675	0.253	0.601	0.437	0.841	0.861	0.460	1.513	1.910	0.789	0.132	0.936	1.840	0.899	0.244	0.183
Talaslioglu (2009) BGAwEIS ps = 500	380.78	200000	0.156	0.555	0.370	0.510	0.620	0.530	0.100	0.100	1.250	0.523	0.101	0.105	1.860	0.513	0.100	0.100
Farshi y Ziazi (2010) MC	379.65	-	0.1565	0.546	0.411	0.570	0.524	0.517	0.100	0.100	1.269	0.512	0.100	0.100	1.886	0.512	0.100	0.100
Dede et al. (2011) ps = 100	407.37	19100	2.046	0.477	0.174	0.174	0.146	0.508	0.174	0.287	0.431	0.508	0.220	0.220	0.174	0.587	0.431	0.431
Dede et al. (2011) ps = 100	382.35	19100	1.702	0.496	0.100	0.100	1.288	0.469	0.100	0.100	0.505	0.550	0.109	0.118	0.154	0.604	0.442	0.604
Degertekin y Hayalioglu (2013) TLBO	379.63	19709	1.906	0.506	0.100	0.100	1.262	0.511	0.100	0.100	0.532	0.516	0.100	0.100	01.56	0.549	0.410	0.570
Kaveh y Khayatazad (2013) RO	380.45	19084	1.836	0.502	0.100	0.100	0.125	0.503	0.100	0.100	0.573	0.550	0.100	0.100	0.158	0.522	0.436	0.597
Echeverrí (2014) AGM ps = 200	383.87	25200	0.158	0.546	0.410	0.511	0.610	0.553	0.100	0.100	1.116	0.553	0.100	0.100	1.808	0.525	0.100	0.100
PSO (2017) ps=100	372.85	25000	1.547	0.333	0.439	0.183	0.343	0.632	0.223	0.129	0.472	0.157	0.519	0.503	1.058	0.315	1.005	0.173
PSO (2017) ps=200	366.04	50000	0.907	0.268	1.055	0.630	0.361	0.382	0.751	0.896	0.627	0.206	0.255	0.312	0.637	0.346	0.275	0.127

ps: tamaño de la población, GA: Genetic Algorithm, SOA: Second-Order Approximation, APM: Adaptive Penalty Method, CPM: Constant Penalty Method, FM: Force Method, GAOS: Genetic Algorithm Based Optimum Structural Design, HSH: Harmony Search Heuristic, BB-BC: Big-Bang Crunch Algorithm, PSO: Particle Swarm Optimization, HPSO: Heuristic Particle Swarm Optimization, AC: Ant Colony Optimization, MPGA: Multipopulation Based Genetic Algorithm, BGAwEIS: Bipopulation Based Genetic Algorithm with Enhanced Interval Search, SA: Simulated Annealing; ES: Evolution Strategies, TS: Tabu Search; SGA: Simple Genetic Algorithm, TLBO: Teaching Learning Based Optimization, RO: Ray Optimization, AGM: Algoritmo Genético Multicromosoma.
*El número de análisis corresponde a la cantidad de evaluaciones de la función objetivo, es decir, al número de iteraciones requeridas multiplicado por el tamaño de la población.

Fuente: Sebastián Echeverrí [15].

6. EJEMPLOS ANALÍTICOS MSPSO

El algoritmo MSPSO se validó con las mismas funciones que el algoritmo PSO (ver ecuaciones 14, 15 y 16). Los parámetros del algoritmo MSPSO para los ejemplos se muestran en las ilustraciones 25 y 26, sus funciones mantiene la misma estructura vista en las ilustraciones 14, 15 y 17 según el caso.

Ilustración 25. MSPSO, Parámetros algoritmo (ecuación 14 y 15), software MATLAB®.

```
%=========================================================================
%PARAMETROS DEL PROBLEMA
ejec=1; %ejecuciones pso
iter=40; %iteraciones pso
part=50; %particulas pso
wpso=0.95; %w pso [valor entre 0 y 1]
c1pso=2;%c1 pso [valor entre 0 y 2]
c2pso=2;%c2 pso [valor entre 0 y 2]
radio=2; %radio mpso
%=========================================================================
```

Ilustración 26. MSPSO, Parámetros algoritmo (ecuación 17), software MATLAB®.

```
%=====================================================
ejec=1; %ejecuciones pso
iter=10; %iteraciones pso
part=50; %particulas pso
wpso=0.95; %w pso [valor entre 0 y 1]
c1pso=2;%c1 pso [valor entre 0 y 2]
c2pso=2;%c2 pso [valor entre 0 y 2]
radio=2; %radio mpso
%=====================================================
```

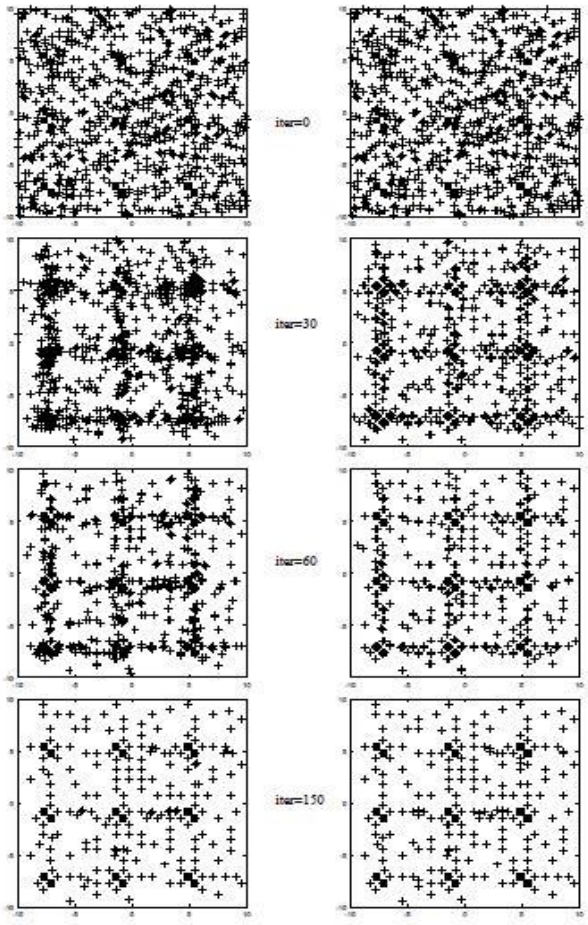
Al analizar las estructuras con el MSPSO, se pudo observar resultados desfavorables, (ver tabla 14),

El algoritmo MSPSO definido en la sección 2.1 presento estancamiento en la caracterización del mínimo global, ya que al seleccionar las especie y determinar la mejor, el algoritmo atraerá las demás partículas a que pertenezcan a la mejor especie, esto con lleva a que si la mejor especie, es un mínimo local, el algoritmo se estanque juntando las partículas en esta especie, esto se puede evidenciar en la figura 11 y 12, extraídas de la literatura [10].

Tabla 14. Comparación de resultados PSO vs MSPSO

Armadura	Peso optimo (lb)	
	PSO	MSPSO
25 elementos	473.76	546,81
72 Elementos	366.04	447,14

Ilustración 27. MSPSO, comportamiento de las partículas..

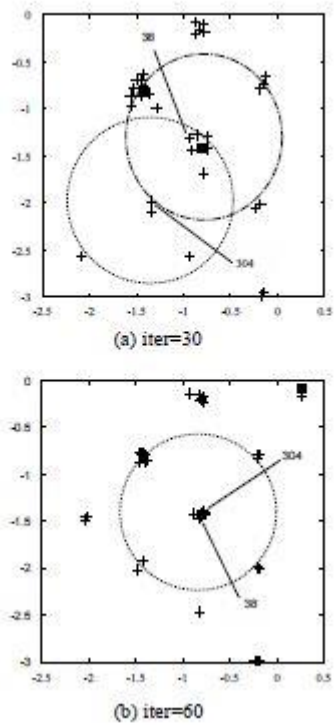


Fuente: Iwamatsu [10]

El autor *Iwamatsu [10],* realiza un ejemplo con número de población de 800 partículas, velocidad máxima de 1.0, radio de 0.8 y un máximo 150 iteraciones por ejecución, en la ilustración 27, se observa como las partículas son atraídas al centro de la especie más cercana a medida que aumenta el número de iteraciones. Un acercamiento se puede ver en la ilustración 28.a. La cual en su iteración 30 tendrá dos mínimos (partícula N° 38 y N° 304) cada una perteneciendo a una especie diferente, y en su iteración 60 (ver ilustración 28.b), las dos partículas (N°38 y N°304) pertenecerán a una sola especie, en ese momento se encontrara el mínimo global, del problema.

Sin embargo, para el análisis de las armaduras de 25 y 72 elementos, el algoritmo no logra determinar el óptimo global por la pobre exploración de las partículas, que limita el movimiento en el campo de búsqueda, dando como solución una especie con mínimo local, este comportamiento se presentó durante la evaluación de las estructuras a optimizar (armaduras de 25 y 72 elementos), convergiendo en mínimos locales en comparación con los resultados del PSO.

Ilustración 28. MSPSO, comportamiento de las partículas (a) iteración 30 y (b) iteración 60.



Fuente: Iwamatsu [10].

Finalmente, se determina que entre menos dimensiones más efectivo el algoritmo por la poca exploración que necesitan estos problemas.

El valor mínimo para las armaduras de 25 y 72 elementos hallados por el MSPSO con una población de 100 partículas, son $546{,}81\ lb$ y $447{,}14\ lb$, respectivamente. (Ver ilustración 29 y 30).

Ilustración 29. MSPSO, Peso por corrida armadura de 25 elementos.

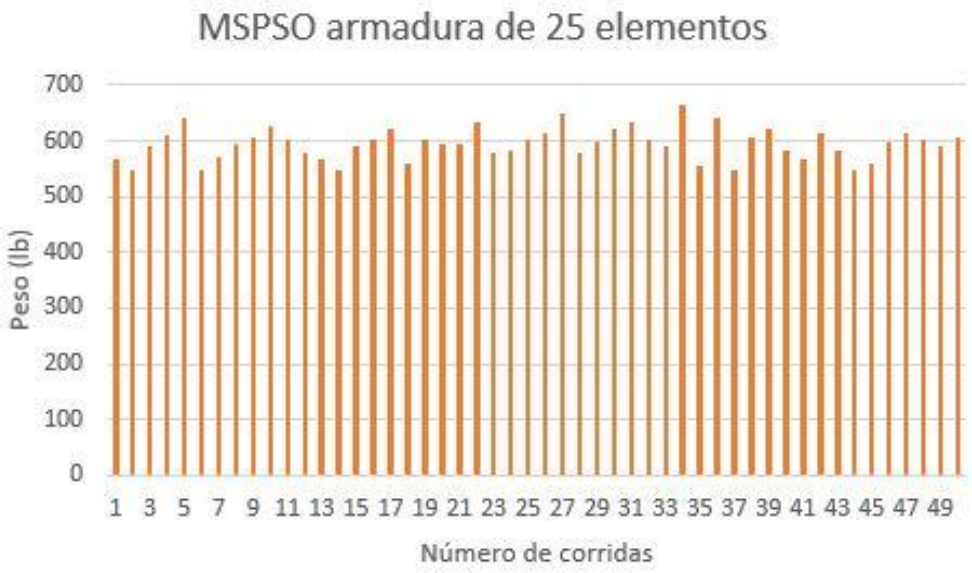
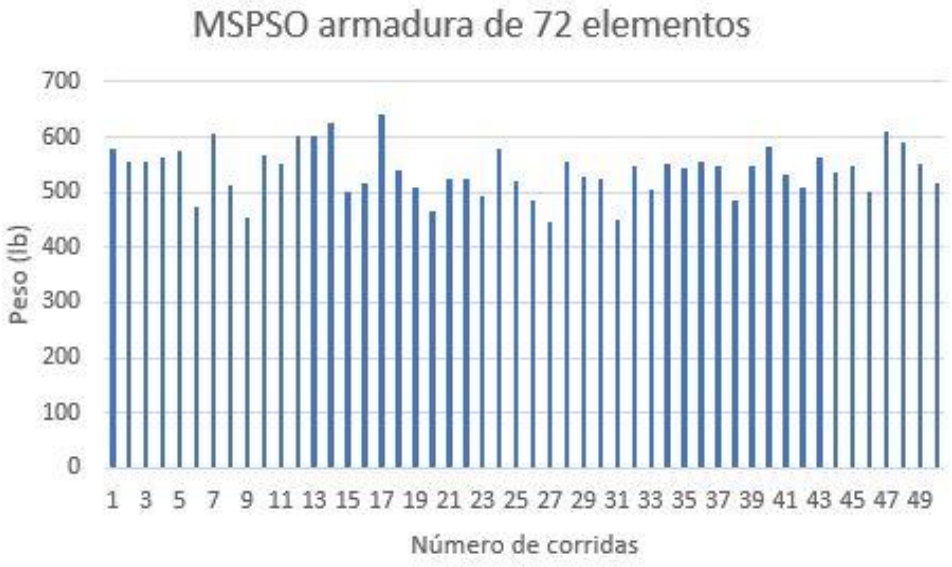


Ilustración 30. MSPSO, Peso por corrida armadura de 72 elementos.



7. CONCLUSIONES

De los resultados obtenidos y comparaciones realizadas con lo reportado en la literatura [15], se concluye:

- Mediante el procedimiento y los algoritmos usados en este estudio, se obtiene el peso óptimo de dos armaduras 3-D, cuya configuración geométrica es definida en la referencia [15], cumpliendo con las restricciones y parámetros de diseño por el método AISC, los resultados obtenidos con el PSO tradicional se exponen en la sección 4, los cuales en comparación con la referencia [15] tiene una mejora del 4,3% en el peso óptimo, y en la sección 5, se presentan los resultados con el algoritmo MSPSO, con un incremento aproximado del 15% en el peso optimo hallado por el PSO tradicional.

- El PSO convencional logra una mejor respuesta que su modificación MSPSO, a causa de que no restringe los parámetros de explotación y exploración de las partículas, permitiendo un tránsito libre en todo el espacio de búsqueda, mientras que el MSPSO está limitado al espacio de búsqueda establecido por sus especies, esto no garantiza que luego de determinar la mejor especie permita un muy movimiento dinámico amplio a las demás partículas, impidiendo encontrar un mejor solución al problema.

- El primer objetivo específico consiste en programar el algoritmo de optimización PSO, en el software MATLAB, este objetivo es logrado como se evidencia en la sección 4, el código del PSO programado por el autor se encuentra en el anexo.

RECOMENDACIONES

- Como recomendación final a una futura extensión del anterior trabajo, se propone la implementación de una variación del MSPSO que permita una mayor capacidad de exploración a las partículas, con el fin de evitar un estancamiento en un mínimo local, sin aumentar el costo computacional.

CITAS BIBLIOGRAFICAS

[1] AZAR, Miguel; PAZ, Fabiola y HERRERA, Analía. Análisis de convergencia temprana en algoritmos PSO con función objetivo lineal. Trabajo de grado en Licenciatura en Sistemas. Provincia de Jujuy: Universidad Nacional de Jujuy. Facultad de Ingeniería, 2015, 5p.

[2] BARRAGÁN, Alejandro. Análisis de una torre de transmisión eléctrica ante efectos de viento atmosférico y de tromba: relación del costo de la torre en función de la velocidad del viento. Tesis de pregrado en Ingeniería Civil. Ciudad de México: Universidad Nacional Autónoma de México. Facultad de Ingeniería, 2013, 79p.

[3] COMPUTERS & STRUCTURES INC. SAP 2000 v19. [En línea]. (Recuperado en 8 octubre 2017). Disponible en https://www.csiamerica.com/products/sap2000

[4] CASTELEIRO, M.; PARÍS, J.; MARTÍNEZ, S.; COLOMINAS, I. y NAVARRINA, F. Optimización estructural de torres de alta tensión. [Base de datos en línea]. 15-18 Noviembre 2010. Asociación Argentina de Mecánica Computacional, 29, 273-291. (Recuperado en 9 octubre 2017). Disponible en http://amcaonline.org.ar/ojs/index.php/mc/article/viewFile/3027/2958

[5] COUCERIO, I.; PARÍS, J.; MARTÍNEZ, S.; COLOMINAS, I.; NAVARRINA, F. y CASTELEIRO, M. Structural optimization of lattice steel transmission towers. [Base de datos en línea]. 15 Junio 2016. Engineering Structures, 177, 274-286. (Recuperado en 9 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0141029616300256#!

[6] CUEVAS JIMÉNEZ, Erik Valdemar; OSUNA ENCISO, José Valentín; OLIVA NAVARRO, Diego Alberto y DÍAZ CORTÉS, Margarita Arimatea. Optimización, Algoritmos programados con MATLAB. México: Alfaomega Grupo Editor, 2016. p. 208. ISBN 978-958-778-211-0

[7] DA SILVA, J.; VELLASCO, P.; DE ANDRADE, S. y DE OLIVEIRA, M. Structural assessment of current steel design models for transmission and telecommunication towers. [Base de datos en línea]. Agosto 2005. Journal of Constructional Steel Research, 61(8), 1108-1134. (Recuperado en 10 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0143974X05000416

[8] DEGERTEKIN, S. Improved harmony search algorithms for sizing optimization of truss structures. [Base de datos en línea]. Febrero 2012. Computers and Structures, 92-93, 229-241. (Recuperado en 10 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S004579491100277X

[9] FARSHI, Behrooz y ALINIA-ZIAZI, Ali. (2010). Sizing optimization of truss structures by method of centers and force formulation. [Base de datos en línea]. Septiembre 2010. International Journal Of Solids And Structures, 47(18-19), 2508-2524. (Recuperado en 10 octubre 2017). Disponible en http://www.sciencedirect.com/ science/article/pii/S0020768310001794

[10] IWAMATSU, Masao. Multi-species Particle Swarm Optimizer for Multimodal Function Optimization. [Base de datos en línea]. 1 Marzo 2006. IEICE TRANSACTIONS on Information and Systems, 89(3), 1181-1187. (Recuperado en 11 octubre 2017). Disponible en https://search.ieice.org/bin/summary.php?id=e89-d_3_1181

[11] LAMBERTI, L. An efficient simulated annealing algorithm for design optimization of truss structures. [Base de datos en línea]. Octubre 2008. Computers and

Structures, 86(19-20), 1936-1953. (Recuperado en 11 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0045794908000448

[12] MATRIX LABORATORY, (MATLAB R2015a). [software]. 2009. The MathWorks, Inc.

[13] MCCORMAC, Jack. Diseño de estructuras de metalicas: metodo ASD. México: Alfaomega, 1999. p.740. ISBN 9701502221

[14] PÉREZ LÓPEZ, Jesús Ramón. Contribución a los métodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo próximo. Tesis doctoral. Santander: Universidad de Cantabria. Departamento de Ingeniería de Comunicaciones, 2005, 80p.

[15] RAMÍREZ ECHEVERRI, Sebastian. Metodología basada en Algoritmos Genéticos y Programación en Paralelo para el Diseño Óptimo de Armaduras de Acero. Trabajo de grado de Maestría en Ingeniería Civil. Bogotá: Pontificia Universidad JAVERIANA. Facultad de Ingeniería, 2014, 158p.

[16] VILLANUEVA DOMÍNGUEZ, Eduardo. Análisis estructural de una torre de alta tensión. Tesis de pregrado en Ingeniería Técnica Industrial. Leganés: Universidad Carlos III de Madrid. Departamento de Mecánica de Medios continuos y Teoría de Esctructuras, 2014, 120p.

BIBLIOGRAFÍA

AZAR, Miguel; PAZ, Fabiola y HERRERA, Analía. Análisis de convergencia temprana en algoritmos PSO con función objetivo lineal. Trabajo de grado en Licenciatura en Sistemas. Provincia de Jujuy: Universidad Nacional de Jujuy. Facultad de Ingeniería, 2015, 5p.

BARRAGÁN, Alejandro. Análisis de una torre de transmisión eléctrica ante efectos de viento atmosférico y de tromba: relación del costo de la torre en función de la velocidad del viento. Tesis de pregrado en Ingeniería Civil. Ciudad de México: Universidad Nacional Autónoma de México. Facultad de Ingeniería, 2013, 79p.

CASTELEIRO, M.; PARÍS, J.; MARTÍNEZ, S.; COLOMINAS, I. y NAVARRINA, F. Optimización estructural de torres de alta tensión. [Base de datos en línea]. 15-18 Noviembre 2010. Asociación Argentina de Mecánica Computacional, 29, 273-291. (Recuperado en 9 octubre 2017). Disponible en http://amcaonline.org.ar/ojs/index.php/mc/article/viewFile/3027/2958

COMPUTERS & STRUCTURES INC. SAP 2000 v19. [En línea]. (Recuperado en 8 octubre 2017). Disponible en https://www.csiamerica.com/products/sap2000

COUCERIO, I.; PARÍS, J.; MARTÍNEZ, S.; COLOMINAS, I.; NAVARRINA, F. y CASTELEIRO, M. Structural optimization of lattice steel transmission towers. [Base de datos en línea]. 15 Junio 2016. Engineering Structures, 177, 274-286. (Recuperado en 9 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0141029616300256#!

CUEVAS JIMÉNEZ, Erik Valdemar; OSUNA ENCISO, José Valentín; OLIVA NAVARRO, Diego Alberto y DÍAZ CORTÉS, Margarita Arimatea. Optimización, Algoritmos programados con MATLAB. México: Alfaomega Grupo Editor, 2016. p. 208. ISBN 978-958-778-211-0

DA SILVA, J.; VELLASCO, P.; DE ANDRADE, S. y DE OLIVEIRA, M. Structural assessment of current steel design models for transmission and telecommunication towers. [Base de datos en línea]. Agosto 2005. Journal of Constructional Steel Research, 61(8), 1108-1134. (Recuperado en 10 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0143974X05000416

DEGERTEKIN, S. Improved harmony search algorithms for sizing optimization of truss structures. [Base de datos en línea]. Febrero 2012. Computers and Structures, 92-93, 229-241. (Recuperado en 10 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S004579491100277X

FARSHI, Behrooz y ALINIA-ZIAZI, Ali. Sizing optimization of truss structures by method of centers and force formulation. [Base de datos en línea]. Septiembre 2010. International Journal Of Solids And Structures, 2010 47(18-19), 2508-2524. (Recuperado en 10 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0020768310001794

IWAMATSU, Masao. Multi-species Particle Swarm Optimizer for Multimodal Function Optimization. [Base de datos en línea]. 1 Marzo 2006. IEICE TRANSACTIONS on Information and Systems, 89(3), 1181-1187. (Recuperado en 11 octubre 2017). Disponible en https://search.ieice.org/bin/summary.php?id=e89-d_3_1181

LAMBERTI, L. An efficient simulated annealing algorithm for design optimization of truss structures. [Base de datos en línea]. Octubre 2008. Computers and Structures,

86(19-20), 1936-1953. (Recuperado en 11 octubre 2017). Disponible en http://www.sciencedirect.com/science/article/pii/S0045794908000448

MATRIX LABORATORY, (MATLAB R2015a). [software]. 2009. The MathWorks, Inc.

MCCORMAC, Jack. Diseño de estructuras de metalicas: metodo ASD. México: Alfaomega, 1999. p.740. ISBN 9701502221

PÉREZ LÓPEZ, Jesús Ramón. Contribución a los métodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo próximo. Tesis doctoral. Santander: Universidad de Cantabria. Departamento de Ingeniería de Comunicaciones, 2005, 80p.

RAMÍREZ ECHEVERRI, Sebastian. Metodología basada en Algoritmos Genéticos y Programación en Paralelo para el Diseño Óptimo de Armaduras de Acero. Trabajo de grado de Maestría en Ingeniería Civil. Bogotá: Pontificia Universidad JAVERIANA. Facultad de Ingeniería, 2014, 158p.

VILLANUEVA DOMÍNGUEZ, Eduardo. Análisis estructural de una torre de alta tensión. Tesis de pregrado en Ingeniería Técnica Industrial. Leganés: Universidad Carlos III de Madrid. Departamento de Mecánica de Medios continuos y Teoría de Esctructuras, 2014, 120p.

ANEXOS

Anexo A. Programación de matricial

```
%PROGRAMACIÓN METODO MATRICIL
%=============================================================
=============
clc
clear

%=============================================================
=============

%PARAMETROS DEL PROBLEMA

%número de barras
%número de nodos
%densidad en lb/in3
%módulo de Young en ksi
%coordenadas en in
%conexiones nodo inicial y final
%cargas en kips
%1 nodo con apoyo
%áreas en in2
%variación de áreas

%=============================================================
=============
```

```
%ANALISIS ESTRUCTURAL

%LONGITUDES EN IN

for c1=1:nbar

        long(c1,1)=sqrt(((coor(conx(c1,1),1)-coor(conx(c1,2),1))^2)+...
            ((coor(conx(c1,1),2)-coor(conx(c1,2),2))^2)+...
            ((coor(conx(c1,1),3)-coor(conx(c1,2),3))^2));

End

%AREAS EN IN2

c2=1;

for c3=1:size(vare,1)

   for c4=1:vare(c3)

      arto(c2,1)=area(c3);
      c2=c2+1;

   end

end
%APOYOS

c5=1;
```

```
for c6=1:size(apoy,1)

   for c7=1:size(apoy,2)

      dapo(c5,1)=apoy(c6,c7);
      c5=c5+1;

   end

end

res1=find(dapo==1);
res2=find(dapo==0);

%FUERZAS

c8=1;

for c9=1:size(carg,1)

   for c10=1:size(carg,2)

      dcar(c8,1)=carg(c9,c10);
      c8=c8+1;

   end

end

%MATRIZ DE RIGIDEZ
```

```
rigk=zeros(nnod*3,nnod*3);

for c11=1:nbar

          rig1=(coor(conx(c11,2),1)-coor(conx(c11,1),1))/(long(c11,1));
          rig2=(coor(conx(c11,2),2)-coor(conx(c11,1),2))/(long(c11,1));
          rig3=(coor(conx(c11,2),3)-coor(conx(c11,1),3))/(long(c11,1));
          rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
            0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
         rig5=[(arto(c11)*elas/long(c11)),0,0,-(arto(c11)*elas/long(c11)),0,0;
             0,0,0,0,0,0;0,0,0,0,0,0;
            -(arto(c11)*elas/long(c11)),0,0,(arto(c11)*elas/long(c11)),0,0;
            0,0,0,0,0,0;0,0,0,0,0,0]
        rig6=rig4'*rig5*rig4;
        rig7=[3*conx(c11,1)-2,3*conx(c11,1)-1,3*conx(c11,1),3*conx(c11,2)-
        2,3*conx(c11,2)-1,3*conx(c11,2)];
        rig8=zeros(nnod*3,nnod*3);
        rig8(rig7,rig7)=rig6;
         rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
```

```
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

        esf1=(coor(conx(c12,2),1)-coor(conx(c12,1),1))/(long(c12,1));
         esf2=(coor(conx(c12,2),2)-coor(conx(c12,1),2))/(long(c12,1));
         esf3=(coor(conx(c12,2),3)-coor(conx(c12,1),3))/(long(c12,1));
        esf4=[3*conx(c12,1)-2,3*conx(c12,1)-1,3*conx(c12,1),3*conx(c12,2)-
        2,3*conx(c12,2)-1,3*conx(c12,2)];
          esf5=dtot(esf4);
         etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,-
                esf3,esf1,esf2,esf3]*esf5/long(c12);

end

%RESULTADOS

for c13=1:nnod

   despl(c13,1)=dtot((3*c13)-2,1);
   despl(c13,2)=dtot((3*c13)-1,1);
   despl(c13,3)=dtot((3*c13),1);
```

```
end

despl;

Dmax;

etot;
```

Anexo B. Programación algoritmo PSO en MATLAB

```
%PROGRAMACIÓN ALGORITMO PSO PARA ARMADURA 3-D
%=============================================================
=============
clc
clear
tic
%=============================================================
=============
%PARAMETROS DEL PROBLEMA

%número de barras
%número de nodos
%densidad en lb/in3
%módulo de Young en ksi
%coordenadas en in
%conexiones nodo inicial y final
%cargas en kips
%1 nodo con apoyo
%áreas en in2
%variación de áreas

%delta de área en in2 para función de restricción
%limite desplazamiento
%límite esfuerzo tensión
%límite esfuerzo compresión
%ejecuciones pso
```

```
%iteraciones pso
%partículas pso
%w pso [valor entre 0 y 1]
%c1 pso [valor entre 0 y 2]
%c2 pso [valor entre 0 y 2]

%==========================================================
=============

%LONGITUDES EN IN

for c1=1:nbar

      long(c1,1)=sqrt(((coor(conx(c1,1),1)-coor(conx(c1,2),1))^2)+...
                              ((coor(conx(c1,1),2)-coor(conx(c1,2),2))^2)+...
                              ((coor(conx(c1,1),3)-coor(conx(c1,2),3))^2));

End

%APOYOS

c5=1;

for c6=1:size(apoy,1)

   for c7=1:size(apoy,2)

      dapo(c5,1)=apoy(c6,c7);
      c5=c5+1;
```

```
    end

end

res1=find(dapo==1);
res2=find(dapo==0);

%FUERZAS

c8=1;

for c9=1:size(carg,1)

    for c10=1:size(carg,2)

        dcar(c8,1)=carg(c9,c10);
        c8=c8+1;

    end

end

%RESTRICCION DESPLAZAMIENTO

rare1=zeros(max(vare),size(vare,1));

for c15=1:vare(1)

    rare1(c15,1)=c15;
```

```
end

for c16=2:1:size(vare,1)

    for c17=1:1:vare(c16)

        rare1(c17,c16)=rare1(vare(c16-1),c16-1)+c17;

    end

end

for c18=1:1:size(rare1,2)

    rare2=rare1(:,c18);
    rare2(rare2==0)=[];

    for c19=1:1:size(rare2,1)

        rare3(c19,1,c18)=conx(rare2(c19,1),1);
        rare3(c19,2,c18)=conx(rare2(c19,1),2);

    end

end

%CONSTANTES PSO

if c1pso+c2pso<4
    fipso=4;
```

```
else

    fipso=c1pso+c2pso;
end

ccpso=2/(abs(2-fipso-sqrt((fipso^2)+(4*fipso))));

%LIMITES VECTORES

lpos=[0.1,2];
lvel=[((lpos(2)-lpos(1))/5)*(-1),((lpos(2)-lpos(1))/5)];

%============================================================
=============

%PSO

ipso=0;

for c13=1:ejec

    %VALORES INICIALES
    posi=round(random('unif',lpos(1),lpos(2),part,size(vare,1)),3);
    velo=round(random('unif',lvel(1),lvel(2),part,size(vare,1)),3);

    %CALIDAD

    for c14=1:part

        %AREAS EN IN2
```

```
c2=1;

for c3=1:size(vare,1)

   for c4=1:vare(c3)
      arto(1,c2)=posi(c14,c3);
      c2=c2+1;
   end

end
%MATRIZ DE RIGIDEZ

rigk=zeros(nnod*3,nnod*3);

for c11=1:nbar

                  rig1=(coor(conx(c11,2),1)-coor(conx(c11,1),1))/(long(c11,1));
                  rig2=(coor(conx(c11,2),2)-coor(conx(c11,1),2))/(long(c11,1));
                  rig3=(coor(conx(c11,2),3)-coor(conx(c11,1),3))/(long(c11,1));
                  rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
                        0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
                  rig5=[(arto(1,c11)*elas/long(c11,1)),0,0,-
                  (arto(1,c11)*elas/long(c11,1)),0,0;
                    0,0,0,0,0,0;0,0,0,0,0,0;
                        -(arto(1,c11)*elas/long(c11,1)),0,0,
                        (arto(1,c11)*elas/long(c11,1)),0,0;
                    0,0,0,0,0,0;0,0,0,0,0,0];
                  rig6=rig4'*rig5*rig4;
```

```
                rig7=[3*conx(c11,1)-2,3*conx(c11,1)-
                1,3*conx(c11,1),3*conx(c11,2)-2,3*conx(c11,2)-
                1,3*conx(c11,2)];
        rig8=zeros(nnod*3,nnod*3);
      rig8(rig7,rig7)=rig6;
       rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

        esf1=(coor(conx(c12,2),1)-coor(conx(c12,1),1))/(long(c12,1));
```

```
            esf2=(coor(conx(c12,2),2)-coor(conx(c12,1),2))/(long(c12,1));
            esf3=(coor(conx(c12,2),3)-coor(conx(c12,1),3))/(long(c12,1));
            esf4=[3*conx(c12,1)-2,3*conx(c12,1)-1,3*conx(c12,1),3*conx(c12,2)-
            2,3*conx(c12,2)-1,3*conx(c12,2)];
    esf5=dtot(esf4);
            etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,-
            esf3,esf1,esf2,esf3]*esf5/long(c12);

end

esfuemax=max(etot);
esfuemin=min(etot);

%PENALIZACION RESTRICCIONES

while Dmax>ldes|esfuemax>lest|esfuemin<lesc
    posi(c14,:)=posi(c14,:)+deltarea;

    %AREAS EN IN2

    c27=1;

    for c28=1:size(vare,1)

        for c29=1:vare(c28)
            arto(1,c27)=posi(c14,c28);
            c27=c27+1;
        end

    end
```

```
%MATRIZ DE RIGIDEZ

rigk=zeros(nnod*3,nnod*3);

for c30=1:nbar

                rig1=(coor(conx(c30,2),1)-coor(conx(c30,1),1))/(long(c30,1));
                rig2=(coor(conx(c30,2),2)-coor(conx(c30,1),2))/(long(c30,1));
              rig3=(coor(conx(c30,2),3)-coor(conx(c30,1),3))/(long(c30,1));
              rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
              0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
              rig5=[(arto(1,c30)*elas/long(c30,1)),0,0,-
              (arto(1,c30)*elas/long(c30,1)),0,0;
                 0,0,0,0,0,0;0,0,0,0,0,0;
                  -(arto(1,c30)*elas/long(c30,1)),0,0,
                arto(1,c30)*elas/long(c30,1)),0,0;
                 0,0,0,0,0,0;0,0,0,0,0,0];
               rig6=rig4'*rig5*rig4;
               rig7=[3*conx(c11,1)-2,3*conx(c30,1)
                -1,3*conx(c30,1),3*conx(c30,2)-2,3*conx(c30,2)-
                1,3*conx(c30,2)];
                rig8=zeros(nnod*3,nnod*3);
              rig8(rig7,rig7)=rig6;
               rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
```

```
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

                    esf1=(coor(conx(c12,2),1)-
                    coor(conx(c12,1),1))/(long(c12,1));
                    esf2=(coor(conx(c12,2),2)-
                    coor(conx(c12,1),2))/(long(c12,1));
                    esf3=(coor(conx(c12,2),3)-
                    coor(conx(c12,1),3))/(long(c12,1));
                    esf4=[3*conx(c12,1)-2,3*conx(c12,1)-
                    1,3*conx(c12,1),3*conx(c12,2)-2,3*conx(c12,2)-
                    1,3*conx(c12,2)];
                    esf5=dtot(esf4);
                    etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,
                    -esf3,esf1,esf2,esf3]*esf5/long(c12);
```

```
        end

        esfuemax=max(etot);
        esfuemin=min(etot);

    end

    %VOLUMEN

    cali(c14,1)=sum((arto'.*long)*dens);

end

%PBEST

pbest=zeros(part,size(vare,1)+1);

for c37=1:1:size(posi,2)

    for c38=1:1:size(posi,1)

        pbest(c38,c37)=posi(c38,c37);

    end

end

pbest(:,end)=cali(:,1);

%GBEST
```

```
    gbest=zeros(1,size(posi,2)+1);
    [magbest,posgbest]=min(cali);
    gbest=posi(posgbest,:);
    gbest(1,end+1)=min(cali)
;
    %CHECKEO

    ipso=ipso+1
    pbest(:,end);
    gbest(:,end)

    %GUARDA GBEST

    plotbest(ipso-(iter*(c13-1)),c13)=gbest(:,end);
    aretotal(ipso-(iter*(c13-1)),:,c13)=gbest(:,1:end-1);


%=========================================================
=========

    %ITERACIONES

    for c39=2:1:iter

        for c40=1:1:part

            for c41=1:1:size(posi,2)

                %VELOCIDAD
```

```
    velo(c40,c41)=round(ccpso*((wpso*velo(c40,c41))+(c1pso*rand*(pbe
    st(c40,c41)-posi(c40,c41)))+(c2pso*rand*(gbest(1,c41)-
    posi(c40,c41)))),3);

if velo(c40,c41)>lvel(2)

   velo(c40,c41)=lvel(2);

end

if velo(c40,c41)<lvel(1)

   velo(c40,c41)=lvel(1);

end

%POSICION
posi(c40,c41)=round(posi(c40,c41)+velo(c40,c41),3);

if posi(c40,c41)>lpos(2)

   posi(c40,c41)=lpos(2);

end

if posi(c40,c41)<lpos(1)

   posi(c40,c41)=lpos(1);
```

```
        end

    end

end

%CALIDAD

for c14=1:part

    %AREAS EN IN2

    c2=1;

    for c3=1:size(vare,1)

        for c4=1:vare(c3)

            arto(1,c2)=posi(c14,c3);
            c2=c2+1;

        end

    end

    %MATRIZ DE RIGIDEZ

    rigk=zeros(nnod*3,nnod*3);

    for c11=1:nbar
```

```
 rig1=(coor(conx(c11,2),1)-
coor(conx(c11,1),1))/(long(c11,1));
rig2=(coor(conx(c11,2),2)-
coor(conx(c11,1),2))/(long(c11,1));
rig3=(coor(conx(c11,2),3)-
coor(conx(c11,1),3))/(long(c11,1));
rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-
rig3,0,rig1,0,0,0;
  0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
rig5=[(arto(1,c11)*elas/long(c11,1)),0,0,-
(arto(1,c11)*elas/long(c11,1)),0,0;
   0,0,0,0,0,0;0,0,0,0,0,0;
   -(arto(1,c11)*elas/long(c11,1)),0,0,
   (arto(1,c11)*elas/long(c11,1)),0,0;
    0,0,0,0,0,0;0,0,0,0,0,0];
rig6=rig4'*rig5*rig4;
rig7=[3*conx(c11,1)-2,3*conx(c11,1)
   -1,3*conx(c11,1),3*conx(c11,2)-2,3*conx(c11,2)-
   1,3*conx(c11,2)];
rig8=zeros(nnod*3,nnod*3);
        rig8(rig7,rig7)=rig6;
rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);
```

```
%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

                    esf1=(coor(conx(c12,2),1)-
                    coor(conx(c12,1),1))/(long(c12,1));
                    esf2=(coor(conx(c12,2),2)-
                    coor(conx(c12,1),2))/(long(c12,1));
                    esf3=(coor(conx(c12,2),3)-
                    coor(conx(c12,1),3))/(long(c12,1));
                    esf4=[3*conx(c12,1)-2,3*conx(c12,1)-
                    1,3*conx(c12,1),3*conx(c12,2)-2,3*conx(c12,2)-
                    1,3*conx(c12,2)];
                    esf5=dtot(esf4);
                    etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,-
                    esf3,esf1,esf2,esf3]*esf5/long(c12);

end
```

```
esfuemax=max(etot);
esfuemin=min(etot);

%PENALIZACION RESTRICCIONES

while Dmax>ldes|esfuemax>lest|esfuemin<lesc

        posi(c14,:)=posi(c14,:)+deltarea;

   %AREAS EN IN2

   c27=1;

   for c28=1:size(vare,1)

      for c29=1:vare(c28)

         arto(1,c27)=posi(c14,c28);
         c27=c27+1;

      end

   end

   %MATRIZ DE RIGIDEZ

   rigk=zeros(nnod*3,nnod*3);

   for c30=1:nbar
```

```
                    rig1=(coor(conx(c30,2),1)-
                    coor(conx(c30,1),1))/(long(c30,1));
                    rig2=(coor(conx(c30,2),2)-
                    coor(conx(c30,1),2))/(long(c30,1));
                    rig3=(coor(conx(c30,2),3)-
                    coor(conx(c30,1),3))/(long(c30,1));
                    rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-
                    rig3,0,rig1,0,0,0;
                    0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
                    rig5=[(arto(1,c30)*elas/long(c30,1)),0,0,-
                    (arto(1,c30)*elas/long(c30,1)),0,0;
                      0,0,0,0,0,0;0,0,0,0,0,0;
                            -(arto(1,c30)*elas/long(c30,1)),0,0,
                    (arto(1,c30)*elas/long(c30,1)),0,0;
                      0,0,0,0,0,0;0,0,0,0,0,0];
                    rig6=rig4'*rig5*rig4;
                    rig7=[3*conx(c11,1)-2,3*conx(c30,1)-
                    1,3*conx(c30,1),3*conx(c30,2)-2,3*conx(c30,2)-
                    1,3*conx(c30,2)];
                    rig8=zeros(nnod*3,nnod*3);
                    rig8(rig7,rig7)=rig6;
                    rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);
```

```
%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

                esf1=(coor(conx(c12,2),1)-
                coor(conx(c12,1),1))/(long(c12,1));
                esf2=(coor(conx(c12,2),2)-
                coor(conx(c12,1),2))/(long(c12,1));
                esf3=(coor(conx(c12,2),3)-
                coor(conx(c12,1),3))/(long(c12,1));
                esf4=[3*conx(c12,1)-2,3*conx(c12,1)-
                1,3*conx(c12,1),3*conx(c12,2)-2,3*conx(c12,2)-
                1,3*conx(c12,2)];
                 esf5=dtot(esf4);
                etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,-
                esf3,esf1,esf2,esf3]*esf5/long(c12);

end
```

```
        esfuemax=max(etot);
        esfuemin=min(etot);

    end

    %VOLUMEN

    cali(c14,1)=sum((arto'.*long)*dens);

end

%PBEST

for c42=1:1:size(cali,1)

    if cali(c42,1)<pbest(c42,end)

        pbest(c42,1:end-1)=posi(c42,:);
        pbest(c42,end)=cali(c42,1);

    end

end

%GBEST

[magbest,posgbest]=min(cali(:,1));

if magbest<gbest(1,end)
```

```
            gbest(1,1:end-1)=posi(posgbest,:);
            gbest(1,end)=magbest;

        end

        %CHECKEO

        ipso=ipso+1
        pbest(:,end);
        gbest(:,end)

        %GUARDA GBEST

        plotbest(ipso-(iter*(c13-1)),c13)=gbest(:,end);
        aretotal(ipso-(iter*(c13-1)),:,c13)=gbest(:,1:end-1);

    end

end

%=============================================================
=============

for i=1:1:ejec

    areafinal(i,:)=aretotal(end,:,i);
end

for i=1:1:size(plotbest,2)
```

```
    total(i)=plotbest(end,i);

end

[magtotal,postotal]=min(total);
equisite=1:1:iter;
equiseje=1:1:ejec;

for i=1:1:size(plotbest,1)

    yeite(i)=plotbest(i,postotal);

end

yeeje=total;
areafinala=zeros(1,size(areafinal,2)+1);
areafinala(1,1:end-1)=areafinal(postotal,:);
areafinala(1,end)=magtotal;
areafinala
%=========================================================
=============

%GRAFICAS

%EJECUCIONES DE LA MEJOR

figure
plot(equisite,yeite,'b')
title('Mejor ejecucion','fontsize',18);
xlabel('# de iteraciones','fontsize',18);
```

```
ylabel('Peso (lbs)','fontsize',18);

%EJECUCIONES DE LA MEJOR

figure
bar(equiseje,yeeje,'b')
title('Ejecuciones totales','fontsize',18);
xlabel('# de corrida','fontsize',18);
ylabel('Peso (lbs)','fontsize',18);

%============================================================
=============

%ARCHIVOS EXPORTABLES

%peso en iteraciones y ejecuciones

fidoo = fopen('peso.txt','wt');
[fk,ck]=size(plotbest);

for i=1:fk

   for j=1:ck

      fprintf(fidoo,'%f\t',plotbest(i,j));

   end

   fprintf(fidoo,'\n');
```

```
end

fclose(fidoo);

%mejor area en ejecuciones

fidooo = fopen('areas.txt','wt');
[fk,ck]=size(areafinal);

for i=1:fk

    for j=1:ck

        fprintf(fidooo,'%f\t',areafinal(i,j));

    end

    fprintf(fidooo,'\n');

end
fclose(fidooo);

%mejor total

fidoooo = fopen('total.txt','wt');
[fk,ck]=size(areafinala);

for i=1:fk

    for j=1:ck
```

```
        fprintf(fidoooo,'%f\t',areafinala(i,j));

    end

    fprintf(fidoooo,'\n');

end

fclose(fidoooo);
toc
```

Anexo C: Programación algoritmo MSPSO en MATLAB

```
% PROGRAMACIÓN ALGORITMO MSPSO PARA ARMADURA 3-D
%=============================================================
=============
clc
clear
tic
%=============================================================
=============
%PARAMETROS DEL PROBLEMA

%número de barras
%número de nodos
%densidad en lb/in3
%módulo de Young en ksi
%coordenadas en in
%conexiones nodo inicial y final
%cargas en kips
%1 nodo con apoyo
%áreas en in2
%variación de áreas

%delta de área en in2 para función de restricción
%limite desplazamiento
%límite esfuerzo tensión
%límite esfuerzo compresión
%ejecuciones mspso
```

```
%iteraciones mspso
%partículas mspso
%w mspso [valor entre 0 y 1]
%c1 mspso [valor entre 0 y 2]
%c2 mspso [valor entre 0 y 2]
%radio MSPSO

%============================================================
=============

%LONGITUDES EN IN

for c1=1:nbar

        long(c1,1)=sqrt(((coor(conx(c1,1),1)-coor(conx(c1,2),1))^2)+...
            ((coor(conx(c1,1),2)-coor(conx(c1,2),2))^2)+...
            ((coor(conx(c1,1),3)-coor(conx(c1,2),3))^2));

End

%APOYOS

c5=1;

for c6=1:size(apoy,1)

   for c7=1:size(apoy,2)

      dapo(c5,1)=apoy(c6,c7);
      c5=c5+1;
```

```
    end

end

res1=find(dapo==1);
res2=find(dapo==0);

%FUERZAS

c8=1;

for c9=1:size(carg,1)

    for c10=1:size(carg,2)

        dcar(c8,1)=carg(c9,c10);
        c8=c8+1;

    end

end

%RESTRICCION DESPLAZAMIENTO

rare1=zeros(max(vare),size(vare,1));

for c15=1:vare(1)

    rare1(c15,1)=c15;
```

```
end

for c16=2:1:size(vare,1)

    for c17=1:1:vare(c16)

        rare1(c17,c16)=rare1(vare(c16-1),c16-1)+c17;

    end

end

for c18=1:1:size(rare1,2)

    rare2=rare1(:,c18);
    rare2(rare2==0)=[];

    for c19=1:1:size(rare2,1)

        rare3(c19,1,c18)=conx(rare2(c19,1),1);
        rare3(c19,2,c18)=conx(rare2(c19,1),2);

    end

end

%CONSTANTES PSO

if c1pso+c2pso<4
```

```
    fipso=4;

else

    fipso=c1pso+c2pso;

end

ccpso=2/(abs(2-fipso-sqrt((fipso^2)+(4*fipso))));

%LIMITES VECTORES

lpos=[0.1,2];
lvel=[((lpos(2)-lpos(1))/5)*(-1),((lpos(2)-lpos(1))/5)];

%============================================================
=============

%PSO TRADICIONAL

ipso=0;

for c13=1:ejec

    %VALORES INICIALES

    posi=round(random('unif',lpos(1),lpos(2),part,size(vare,1)),3);
    velo=round(random('unif',lvel(1),lvel(2),part,size(vare,1)),3);

    %CALIDAD
```

```
for c14=1:part

   %AREAS EN IN2

   c2=1;

   for c3=1:size(vare,1)

      for c4=1:vare(c3)

         arto(1,c2)=posi(c14,c3);
         c2=c2+1;

      end

   end

   %MATRIZ DE RIGIDEZ

   rigk=zeros(nnod*3,nnod*3);

   for c11=1:nbar

      rig1=(coor(conx(c11,2),1)-coor(conx(c11,1),1))/(long(c11,1));
      rig2=(coor(conx(c11,2),2)-coor(conx(c11,1),2))/(long(c11,1));
      rig3=(coor(conx(c11,2),3)-coor(conx(c11,1),3))/(long(c11,1));
      rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
         0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
     rig5=[(arto(1,c11)*elas/long(c11,1)),0,0,-(arto(1,c11)*elas/long(c11,1)),0,0;
         0,0,0,0,0,0;0,0,0,0,0,0;
```

```
        -(arto(1,c11)*elas/long(c11,1)),0,0,(arto(1,c11)*elas/long(c11,1)),0,0;
        0,0,0,0,0,0;0,0,0,0,0,0];
    rig6=rig4'*rig5*rig4;
   rig7=[3*conx(c11,1)-2,3*conx(c11,1)-1,3*conx(c11,1),3*conx(c11,2)-
  ,3*conx(c11,2)-1,3*conx(c11,2)];
    rig8=zeros(nnod*3,nnod*3);
    rig8(rig7,rig7)=rig6;
    rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar
```

```
   esf1=(coor(conx(c12,2),1)-coor(conx(c12,1),1))/(long(c12,1));
   esf2=(coor(conx(c12,2),2)-coor(conx(c12,1),2))/(long(c12,1));
   esf3=(coor(conx(c12,2),3)-coor(conx(c12,1),3))/(long(c12,1));
  esf4=[3*conx(c12,1)-2,3*conx(c12,1)-1,3*conx(c12,1),3*conx(c12,2)-
  2,3*conx(c12,2)-1,3*conx(c12,2)];
   esf5=dtot(esf4);
  etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,-esf3,esf1,esf2,esf3]*esf5/long(c12);

end

esfuemax=max(etot);
esfuemin=min(etot);

%PENALIZACION RESTRICCIONES

while Dmax>ldes|esfuemax>lest|esfuemin<lesc
   posi(c14,:)=posi(c14,:)+deltarea;

   %AREAS EN IN2

   c27=1;

   for c28=1:size(vare,1)

      for c29=1:vare(c28)

         arto(1,c27)=posi(c14,c28);
         c27=c27+1;

      end
```

```
end

%MATRIZ DE RIGIDEZ

rigk=zeros(nnod*3,nnod*3);

for c30=1:nbar

    rig1=(coor(conx(c30,2),1)-coor(conx(c30,1),1))/(long(c30,1));
    rig2=(coor(conx(c30,2),2)-coor(conx(c30,1),2))/(long(c30,1));
    rig3=(coor(conx(c30,2),3)-coor(conx(c30,1),3))/(long(c30,1));
    rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
          0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
   rig5=[(arto(1,c30)*elas/long(c30,1)),0,0,-
      (arto(1,c30)*elas/long(c30,1)),0,0;
          0,0,0,0,0,0;0,0,0,0,0,0;
          -(arto(1,c30)*elas/long(c30,1)),0,0,
         (arto(1,c30)*elas/long(c30,1)),0,0;
          0,0,0,0,0,0;0,0,0,0,0,0];
    rig6=rig4'*rig5*rig4;
    rig7=[3*conx(c11,1)-2,3*conx(c30,1)-1,3*conx(c30,1),3*conx(c30,2)
        -2,3*conx(c30,2)-1,3*conx(c30,2)];
    rig8=zeros(nnod*3,nnod*3);
    rig8(rig7,rig7)=rig6;
    rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
```

```
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

   esf1=(coor(conx(c12,2),1)-coor(conx(c12,1),1))/(long(c12,1));
   esf2=(coor(conx(c12,2),2)-coor(conx(c12,1),2))/(long(c12,1));
   esf3=(coor(conx(c12,2),3)-coor(conx(c12,1),3))/(long(c12,1));
  esf4=[3*conx(c12,1)-2,3*conx(c12,1)-1,3*conx(c12,1),3*conx(c12,2)
       -2,3*conx(c12,2)-1,3*conx(c12,2)];
   esf5=dtot(esf4);
 etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,
       -esf3,esf1,esf2,esf3]*esf5/long(c12);

end

esfuemax=max(etot);
```

```
        esfuemin=min(etot);
    end
    %VOLUMEN
    cali(c14,1)=sum((arto'.*long)*dens);
end
%PBEST
pbest=zeros(part,size(vare,1)+1);
for c37=1:1:size(posi,2)
    for c38=1:1:size(posi,1)
        pbest(c38,c37)=posi(c38,c37);
    end
end
pbest(:,end)=cali(:,1);
%MPSO
contmpso=1;
fampso=zeros(size(cali,1),size(cali,1));
```

```
cmpso=cali;
[magmpso,posmpso]=min(cmpso);

for mc1=1:size(posi,1)

   for mc2=1:size(posi,2)

      pmpso(mc1,mc2)=((posi(mc1,mc2)-posi(posmpso,mc2))^2);

   end

end

pmpso=sum(pmpso,2);
famipso=find(pmpso<radio);

for mc3=1:size(famipso,1)

   fampso(mc3,contmpso)=famipso(mc3);

end

for mc4=1:size(famipso,1)

   cmpso(famipso(mc4))=9999;

end

while min(cmpso)<9999
```

```
for mc4=1:size(famipso,1)

    cmpso(famipso(mc4))=9999;

end

contmpso=contmpso+1;
[magmpso,posmpso]=min(cmpso);

for mc1=1:size(posi,1)

    for mc2=1:size(posi,2)

        pmpso(mc1,mc2)=((posi(mc1,mc2)-posi(posmpso,mc2))^2);

    end

end

pmpso=sum(pmpso,2);
rcmpso=find(cmpso==9999);

for mc5=1:size(rcmpso,1)

    pmpso(rcmpso(mc5),1)=99;

end

famipso=find(pmpso<radio);
```

```
    for mc3=1:size(famipso,1)

        fampso(mc3,contmpso)=famipso(mc3);

    end

end

delfampso=find(fampso(1,:)==0);
fampso(:,delfampso)=[];

%GBEST

gbest=zeros(size(fampso,2),size(vare,1)+1);

for cc39=1:1:size(fampso,2)

    famili=fampso(:,cc39);
    zzz1=find(famili==0);
    famili(zzz1)=[];
    famil=pbest(famili,end);
    gbest(cc39,end)=min(famil);
    zz2=find(gbest(cc39,end)==pbest(:,end));

    for cc40=1:size(posi,2)

        gbest(cc39,cc40)=posi(zz2,cc40);

    end
```

```
    end

    %CHECKEO

    ipso=ipso+1
    gbest(:,end);
    min(pbest(:,end))

%=========================================================
=========

    %ITERACIONES

    for c39=2:1:iter

        for c40=1:1:part

            %IDENTIFICADOR

            zxc1=c40;
            zxc2=find(fampso==zxc1);
            zxc3=ceil(zxc2/part);

            for c41=1:1:size(posi,2)

                %VELOCIDAD

                    velo(c40,c41)=round(ccpso*((wpso*velo(c40,c41))+(c1pso*rand*(pbe
                    st(c40,c41)-posi(c40,c41)))+(c2pso*rand*(pbest(zxc3,c41)-
                    posi(c40,c41)))),3);
```

```
            if velo(c40,c41)>lvel(2)
              velo(c40,c41)=lvel(2);
            end
           if velo(c40,c41)<lvel(1)
              velo(c40,c41)=lvel(1);
           end
        %POSICION
          posi(c40,c41)=round(posi(c40,c41)+velo(c40,c41),3);
            if posi(c40,c41)>lpos(2)
              posi(c40,c41)=lpos(2);
           end
            if posi(c40,c41)<lpos(1)
              posi(c40,c41)=lpos(1);
      end
end
```

```
end

%CALIDAD

for c14=1:part

   %AREAS EN IN2

   c2=1;

   for c3=1:size(vare,1)

      for c4=1:vare(c3)

         arto(1,c2)=posi(c14,c3);
         c2=c2+1;

      end

   end

   %MATRIZ DE RIGIDEZ

   rigk=zeros(nnod*3,nnod*3);

   for c11=1:nbar

      rig1=(coor(conx(c11,2),1)-coor(conx(c11,1),1))/(long(c11,1));
      rig2=(coor(conx(c11,2),2)-coor(conx(c11,1),2))/(long(c11,1));
      rig3=(coor(conx(c11,2),3)-coor(conx(c11,1),3))/(long(c11,1));
```

```
    rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
        0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
    rig5=[(arto(1,c11)*elas/long(c11,1)),0,0,
         -(arto(1,c11)*elas/long(c11,1)),0,0;
        0,0,0,0,0,0;0,0,0,0,0,0;
        -(arto(1,c11)*elas/long(c11,1)),0,0,
         (arto(1,c11)*elas/long(c11,1)),0,0;
        0,0,0,0,0,0;0,0,0,0,0,0];
    rig6=rig4'*rig5*rig4;
    rig7=[3*conx(c11,1)-2,3*conx(c11,1)-1,3*conx(c11,1),3*conx(c11,2)
         -2,3*conx(c11,2)-1,3*conx(c11,2)];
    rig8=zeros(nnod*3,nnod*3);
    rig8(rig7,rig7)=rig6;
    rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));
```

```
%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

   esf1=(coor(conx(c12,2),1)-coor(conx(c12,1),1))/(long(c12,1));
   esf2=(coor(conx(c12,2),2)-coor(conx(c12,1),2))/(long(c12,1));
   esf3=(coor(conx(c12,2),3)-coor(conx(c12,1),3))/(long(c12,1));
   esf4=[3*conx(c12,1)-2,3*conx(c12,1)-1,3*conx(c12,1),3*conx(c12,2)
       -2,3*conx(c12,2)-1,3*conx(c12,2)];
   esf5=dtot(esf4);
   etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,
       -esf3,esf1,esf2,esf3]*esf5/long(c12);

End

esfuemax=max(etot);
esfuemin=min(etot);

%PENALIZACION RESTRICCIONES

while Dmax>ldes|esfuemax>lest|esfuemin<lesc

   posi(c14,:)=posi(c14,:)+deltarea;

   %AREAS EN IN2

   c27=1;
```

```
for c28=1:size(vare,1)

   for c29=1:vare(c28)

      arto(1,c27)=posi(c14,c28);
      c27=c27+1;

   end

end

%MATRIZ DE RIGIDEZ

rigk=zeros(nnod*3,nnod*3);

for c30=1:nbar

   rig1=(coor(conx(c30,2),1)-coor(conx(c30,1),1))/(long(c30,1));
   rig2=(coor(conx(c30,2),2)-coor(conx(c30,1),2))/(long(c30,1));
   rig3=(coor(conx(c30,2),3)-coor(conx(c30,1),3))/(long(c30,1));
   rig4=[rig1,rig2,rig3,0,0,0;-rig2,rig1,0,0,0,0;-rig3,0,rig1,0,0,0;
      0,0,0,rig1,rig2,rig3;0,0,0,-rig2,rig1,0;0,0,0,-rig3,0,rig1];
   rig5=[(arto(1,c30)*elas/long(c30,1)),0,0,
    -(arto(1,c30)*elas/long(c30,1)),0,0;
      0,0,0,0,0,0;0,0,0,0,0,0;
      -(arto(1,c30)*elas/long(c30,1)),0,0,
    (arto(1,c30)*elas/long(c30,1)),0,0;
      0,0,0,0,0,0;0,0,0,0,0,0];
   rig6=rig4'*rig5*rig4;
```

```
    rig7=[3*conx(c11,1)-2,3*conx(c30,1)
     -1,3*conx(c30,1),3*conx(c30,2)-2,3*conx(c30,2)-1,3*conx(c30,2)];
    rig8=zeros(nnod*3,nnod*3);
    rig8(rig7,rig7)=rig6;
    rigk=rigk+rig8;

end

rig9=rigk(res1,res1);
rig10=rigk(res1,res2);
rig11=rigk(res2,res1);
rig12=rigk(res2,res2);

%DESPLAZAMIENTOS

fue1=dcar(res2);
desp=inv(rig12)*fue1;
fue2=rig10*desp;
dtot=zeros(nnod*3,1);
dtot(res2)=desp;
Dmax=max(abs(dtot));

%ESFUERZOS

esfu=zeros(nbar,1);

for c12=1:nbar

    esf1=(coor(conx(c12,2),1)-coor(conx(c12,1),1))/(long(c12,1));
    esf2=(coor(conx(c12,2),2)-coor(conx(c12,1),2))/(long(c12,1));
```

```
            esf3=(coor(conx(c12,2),3)-coor(conx(c12,1),3))/(long(c12,1));
            esf4=[3*conx(c12,1)-2,3*conx(c12,1)
             -1,3*conx(c12,1),3*conx(c12,2)-2,3*conx(c12,2)-1,3*conx(c12,2)];
            esf5=dtot(esf4);
            etot(c12,1)=elas*arto(c12)*[-esf1,-esf2,
             -esf3,esf1,esf2,esf3]*esf5/long(c12);

        end

        esfuemax=max(etot);
        esfuemin=min(etot);

    end

    %VOLUMEN

    cali(c14,1)=sum((arto'.*long)*dens);

end

%PBEST

for c42=1:1:size(cali,1)

    if cali(c42,1)<pbest(c42,end)

        pbest(c42,1:end-1)=posi(c42,:);
        pbest(c42,end)=cali(c42,1);

    end
```

```
end

%MPSO

contmpso=1;
fampso=zeros(size(cali,1),size(cali,1));
cmpso=cali;
[magmpso,posmpso]=min(cmpso);

for mc1=1:size(posi,1)

   for mc2=1:size(posi,2)

      pmpso(mc1,mc2)=((posi(mc1,mc2)-posi(posmpso,mc2))^2);

   end

end

pmpso=sum(pmpso,2);
famipso=find(pmpso<radio);

for mc3=1:size(famipso,1)

   fampso(mc3,contmpso)=famipso(mc3);

end

for mc4=1:size(famipso,1)
```

```
    cmpso(famipso(mc4))=9999;
end
while min(cmpso)<9999
    for mc4=1:size(famipso,1)
        cmpso(famipso(mc4))=9999;
    end
    contmpso=contmpso+1;
    [magmpso,posmpso]=min(cmpso);
    for mc1=1:size(posi,1)
        for mc2=1:size(posi,2)
            pmpso(mc1,mc2)=((posi(mc1,mc2)-posi(posmpso,mc2))^2);
        end
    end
    pmpso=sum(pmpso,2);
    rcmpso=find(cmpso==9999);
    for mc5=1:size(rcmpso,1)
```

```
        pmpso(rcmpso(mc5),1)=99;

    end

    famipso=find(pmpso<radio);

    for mc3=1:size(famipso,1)

        fampso(mc3,contmpso)=famipso(mc3);

    end

end

delfampso=find(fampso(1,:)==0);
fampso(:,delfampso)=[];

%GBEST

gbest=zeros(size(fampso,2),size(vare,1)+1);

for cc39=1:1:size(fampso,2)

    famili=fampso(:,cc39);
    zzz1=find(famili==0);
    famili(zzz1)=[];
    famil=pbest(famili,end);
    gbest(cc39,end)=min(famil);
    zz2=find(gbest(cc39,end)==pbest(:,end));
```

```
            for cc40=1:size(posi,2)

                gbest(cc39,cc40)=posi(zz2,cc40);

            end

        end

        %CHECKEO

        ipso=ipso+1
        gbest(:,end);
        min(pbest(:,end))

    end

end

gbest
```