UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

Una propuesta de topología dinámica para el algoritmo de enjambre de partículas (PSO)

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Licenciada en Ciencias de la Computación

PRESENTA:

Nancy Arlette Mejía Juárez

TUTOR

Dr. Arturo Hernández Aguirre

Ciudad Universitaria, CD. MX., 2016

1. Datos del alumno
Mejía
Juárez
Nancy Arlette
56 73 14 10
Universidad Nacional Autónoma de
México
Facultad de Ciencias
Ciencias de la Computación
308095469
2. Datos del tutor
Dr.
Arturo
Hernández
Aguirre
3. Datos del sinodal 1
Dra.
Katya
Rodríguez
Vázquez
4. Datos del sinodal 2
L. en C.C.
Sergio
Hernández
López
5. Datos del sinodal 3
Dr.
José de Jesús
Galaviz
Casas
6. Datos del sinodal 4
Dra.
María de Luz
Gasca
Soto
7. Datos del trabajo escrito
Una propuesta de topología dinámica para el algoritmo de enjambre de partículas (PSO)
74 p.
2016

A mis papás.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE ALGORITMOS

INTRODUCCIÓN

Día a día nos enfrentamos a problemas en los que debemos encontrar la solución óptima, aquella que nos otorgue el mejor resultado entre un conjunto de posibles soluciones, es decir, nos enfrentamos a problemas de optimización, que van desde escoger el producto más económico, el camino más corto para trasladarnos de un lugar a otro, el mejor lugar para vivir; por mencionar sólo algunos ejemplos.

Sin embargo, no sólo se encuentran problemas de optimización en nuestro día a día, también en diversas industrias se presentan problemas de optimización, por ejemplo: ¿Cómo generar la mayor producción con el menor costo? ¿Cómo fabricar herramientas con el gasto mínimo de material?, entre otros. Como se puede observar, son problemas que requieren minimizar o maximizar alguna tarea, por lo que entendemos de manera intuitiva que optimizar es el proceso de maximizar o minimizar una función de costo bajo ciertas condiciones. Esta función de costo generalmente es una expresión analítica.

Debido a que este tipo de problemas, conocidos como problemas de optimización global se presentan regularmente, se han creado diversas formas de resolverlos mediante modelos matemáticos y algoritmos. En general la solución de estos problemas depende de cada problema, por lo que no se tiene una solución única para resolverlos.

Los algoritmos de optimización global se clasifican en dos tipos: con o sin restricciones. En los algoritmos de optimización con restricciones, cada una de las variables debe de cumplir ciertas condiciones que se establecen al inicio. Por ejemplo, si se desea generar la producción máxima de un cierto producto, algunas de las restricciones serían tomar en cuenta el tiempo t que tarda la producción del producto, el número de máquinas con las que se cuentan para la producción. De acuerdo a la naturaleza del problema a optimizar se le añaden las condiciones necesarias que deben cumplir las variables.

Para diseñar una propuesta que sea capaz de resolver un problema, se deben definir las condiciones que deben cumplir cada una de nuestras variables, una función de costo a la cual nos referiremos en este trabajo como función objetivo, mediante la cual se evalúan las posibles soluciones, con el fin de ir mejorando la solución actual o saber si se ha encontrado la solución al problema; también se debe definir el espacio de búsqueda, el cual determina en donde las variables tienen valores válidos, en general cada variable X_i está acotada por dos límites, X_{min} y X_{max} de la siguiente manera: $X_{min} \leq X_i \leq X_{max}$.

Debido a que se han planteado diversas estrategias para resolver problemas de optimización global, podemos clasificarlas en dos clases: A) Los métodos basados en el cálculo del gradiente de la función objetivo; B) Los métodos basados en heurísticas.

Los métodos de la primera clase realizan el cálculo de la primera o segunda derivada de la función objetivo para encontrar el resultado. Algunos de los métodos más conocidos son: el método del descenso del gradiente, método de Newton y el método del gradiente conjugado.

Por otro lado, las heurísticas, en general son métodos estocásticos, en los cuales encontrar el valor de la solución óptima depende del valor de los parámetros introducidos y cómo vayan

evolucionando a lo largo de la ejecución. Dentro de estos métodos se encuentran los algoritmos bio-inspirados, como lo son: algoritmos genéticos, el método de búsqueda colonia de hormigas (ant-colony), el cual está inspirado en el comportamiento que presentan las colonias de hormigas para encontrar su comida; el algoritmo de enjambre de partículas (*Particle Swarm Optimization*, PSO, por sus siglas en inglés), el cual se estudia en este trabajo. Es una meta heurística de búsqueda poblacional propuesta por Kennedy y Eberhart [1], está inspirada en el comportamiento colaborativo observado en un cardumen o una parvada de diferentes especies, con diferentes fines, como lo son: obtener su alimento o huir de sus depredadores.

En el algoritmo PSO se define una población y cada uno de los individuos que la conforman es representado por una partícula. Cada partícula representa una solución potencial al problema de optimización global por resolver, mientras que la ubicación del alimento es el análogo a la solución óptima del problema.

Durante el proceso de optimización todas las partículas comparten información entre ellas y colaboran unas con otras, haciendo la analogía con el comportamiento observado en diversas especies de la naturaleza. Esta colaboración hace referencia al movimiento de las aves o peces hacia el alimento, desde diferentes direcciones, mediante la información personal y la información de la población. De la misma manera en el algoritmo PSO, es simulado mediante el movimiento de cada una de las partículas desde distintas posiciones para encontrar la solución óptima permitiendo la convergencia de la población. El algoritmo PSO es sencillo de implementar, además de que sólo es necesario ajustar un pequeño número de parámetros.

Con el fin de mejorar el rendimiento del algoritmo PSO, se emplean diferentes configuraciones llamadas topologías, que nos indican con quien o quienes compartirá información cada una de las partículas. Las dos más empleadas son las topología todos conectados y la topología de anillo, ya que podemos tratarlas como si estas fueran las cotas del número máximo y mínimo de conexiones que tendrá la topología, respectivamente. Esto debido a que en la topología de todos conectados cada una de las partículas está conectada con cada partícula del resto de la población. Por otro lado en la topología de anillo, cada partícula está solamente conectada con sus k vecinos más cercanos, en la versión más común $k = 2$.

Una medida de eficiencia de un algoritmo es mediante el número de evaluaciones de la función objetivo (NEF). Si acotamos un problema por número de evaluaciones de función que se obtienen durante la ejecución con las topologías anteriormente mencionadas, tendríamos dos cotas: NEF_{max} y NEF_{min}. Debido a que con la topología de todos conectados se favorece la etapa de explotación y por otro lado la topología de anillo favorece la etapa de exploración, las cotas NEF_{max} y NEF_{min} varían de acuerdo a la naturaleza del problema. Por ejemplo en problemas de optimización unimodales, como la función de esfera la topología todos conectados favorece la explotación, entonces permite la convergencia al óptimo en un número pequeño de evaluaciones de función, por lo que con una topología de todos conectados obtendríamos la cota NEF_{min}. Por otro lado la topología de anillo promueve la exploración, en funciones unimodales como la esfera que por su naturaleza no requieren de mucha exploración del espacio de búsqueda, la topología de anillo hará que encontrar el valor del óptimo le tome un mayor número de evaluaciones de función a comparación de la topología de todos conectados, por lo que la cota NEF_{max} del número máximo de evaluaciones de función para la función esfera estará dado por la topología de anillo.

Sin embargo, en problemas claramente multimodales, en donde es fácil converger a óptimos locales, la topología que presenta mejores resultados es la topología de anillo al promover la exploración del espacio de búsqueda.

En general se desconoce cuál es la solución al problema a optimizar, no podemos definir anticipadamente con cuál topología obtendremos mejores resultados, por lo que se ha planteado es crear una versión en donde la topología sea capaz de auto adaptarse durante la ejecución del algoritmo, tomando en cuenta el estado de la población (el mejor elemento de la población, el estancamiento de la población, etcétera.), con el fin de mejorar los resultados o eficiencia obtenidos con las topologías todos conectados y la de anillo.

Se considera que una versión del algoritmo PSO con topología dinámica tiene buen rendimiento si es capaz de hallar una solución para un conjunto de problemas en un número considerable de evaluaciones de función NEF, siendo este acotado por $NEF_{min} \leq NEF \leq NEF_{max}$, donde NEF_{max} y NEF_{min} son las cotas encontradas por las topologías todos conectados y la de anillo.

El objetivo de esta tesis es presentar una propuesta de topología auto adaptativa para el algoritmo PSO que la mayoría de las veces encuentre la solución de un problema de optimización global con un número de evaluaciones de función, NEF, que cumpla la condición: $NEF_{min} \leq NEF \leq NEF_{max}$.

Este trabajo de tesis está organizado de la siguiente manera: En el Capítulo 2, se da una introducción al algoritmo PSO, un breve análisis de los parámetros y algunos de los mejores valores propuestos para estos, en la siguiente sección del capítulo se abordan las condiciones de convergencia del algoritmo PSO, además de los criterios de paro comúnmente empleados en el algoritmo y finalmente se da una pequeña introducción a la clasificación de las topologías empleadas en el algoritmo y algunos ejemplos de estas. En el Capítulo 3, se lleva a cabo una clasificación de diversas propuestas de algunos autores con las que se desea mejorar el rendimiento y eficiencia del algoritmo PSO, se hace una clasificación de estas dependiendo del enfoque que presentan, es decir, si varían el valor de los parámetros o la topología durante la ejecución del algoritmo y a su vez, si estos cambios los hacen de manera determinista o bien de forma auto adaptativa. En el Capítulo 4 se explica con detalle nuestra propuesta con topología auto adaptativa, con la que se desea obtener mejores o resultados similares a los de otras propuestas, en un número menor de evaluaciones de función, se presentan también algunos de los comportamientos observados del incremento y decremento de las topologías en distintas funciones. En el Capítulo 5 se presentan los resultados obtenidos con nuestra propuesta y son comparados contra otras propuestas. Finalmente se presentan las conclusiones del trabajo realizado.

1

ALGORITMO PSO

En el presente capítulo se da una visión panorámica del algoritmo PSO. En la primer sección se presenta la versión canónica del algoritmo PSO. En la segunda sección se presentan y explican los parámetros que influyen en él. En la tercer sección se explican algunas condiciones de convergencia del algoritmo. En la cuarta sección se exponen los criterios de paro comúnmente empleados. Finalmente se presentan las topologías del algoritmo y la clasificación de éstas.

1.1 PSO CANÓNICO

La primer versión del algoritmo PSO fue propuesta por Kennedy y Eberhart en 1995 [1], este algoritmo propone una forma de resolver problemas de optimización por medio de la simulación del comportamiento de animales, como los observados en las parvadas y los cardúmenes. En ambos, los animales transmiten información para lograr llegar al óptimo, se entiende como óptimo en estas situaciones, al lugar en donde se encuentra la comida, al lugar en donde se encuentran a salvo del depredador, entre otros.

El algoritmo PSO es un algoritmo de tipo poblacional, que ha sido aplicado para resolver problemas de optimización con parámetros reales, y pese a que es una meta-heurística y no asegura obtener siempre el mismo resultado, se han obtenido buenos resultados en problemas de optimización global.

Se considera una heurística simple, ya que no requiere calcular el gradiente de la función a optimizar y sólo usa operaciones matemáticas simples.

En el algoritmo se define una *población o enjambre* de tamaño N, $N \in \mathbb{Z}$, y cada uno de los individuos de la población está representado por una *partícula*. Adicional a ello se define un *espacio de búsqueda* d-dimensional, $\mathbb{R}^d$, en donde habitarán las partículas.

Además se define la forma en la que las partículas comparten información por medio de una *topología*. La cual indica con quiénes comparte información cada una de las partículas.

El PSO también requiere definir la *función de costo o función objetivo*, la cual se define dentro del espacio de búsqueda de la siguiente manera: $f : \mathbb{R}^d \to \mathbb{R}$. Es una expresión analítica que nos permite evaluar las posibles soluciones. Al valor que reporta la función objetivo al ser evaluada en la posición de una partícula se le conoce como *valor de aptitud*.

El objetivo del PSO es hallar el punto $\vec{x}$ que maximiza o minimiza la función de costo o función objetivo. Se dice que un punto $\vec{x}^* \in \mathbb{R}^d$ es un *mínimo global*[17] de $f : \mathbb{R}^d \to \mathbb{R}$, sí $f(\vec{x}^*) \leq f(\vec{x})$, para todo $\vec{x} \in \mathbb{R}$. También se dice que un punto $\vec{x}^*$ es un *mínimo local*[17] de $f : \mathbb{R}^d \to \mathbb{R}$, si existe $\varepsilon > 0$ tal que $f(\vec{x}^* \leq f(\vec{x})$ para todo $\vec{x} \in \mathbb{R}$ que cumple $||\vec{x} - \vec{x}^*|| < \varepsilon$.

Una vez que se establece lo anterior, las partículas son colocadas aleatoriamente en el espacio de búsqueda y de cada una de ellas, se guarda la siguiente información:

La historia del mejor valor que ha obtenido hasta ese punto, el cual aporta el valor de la *influencia personal*. Con el mejor valor obtenido nos referimos a aquella posición en la que la evaluación de la partícula es la más cercana al valor óptimo, es decir, si hablamos de una función en donde queremos maximizar, el mejor valor obtenido de la partícula es la posición en donde la evaluación de la función nos ha dado el valor más grande hasta ese punto en la ejecución del algoritmo. Por otro lado, si el problema a optimizar requiere encontrar el mínimo, el mejor valor es aquel en el que la evaluación de la función nos da el valor más pequeño obtenido hasta ese punto.

Además se almacena el mejor valor encontrado de toda la población hasta ese punto en la ejecución del algoritmo, el cual aporta información sobre la *influencia social* que hay entre las partículas.

Finalmente cada una de las partículas guarda el valor de la velocidad, con la que se ha movido a lo largo de la ejecución del algoritmo.

Por lo anterior, cada una de las partículas está definida por tres vectores. El primero de ellos el vector de *posición* en el espacio de búsqueda d-dimensional $\vec{X} = (x_1, x_2, ..., x_d)$, donde d es el tamaño de la dimensión del espacio y cada componente de $\vec{X}$, cumple la siguiente condición: $x_{min} \leq x_i \leq x_{max}$; x_{min} y $x_{max} \in \mathbb{R}$, las cuales son cotas para los valores de la posición; esto es importante, ya que permite establecer los límites de los valores que pueden tomar las partículas y así evitar que se encuentren fuera del espacio de búsqueda.

También cuenta con un vector d-dimensional para almacenar el valor de velocidad de la partícula $\vec{V} = (v_1, v_2, ..., v_d)$, cada $v_i \in \mathbb{R}$. Y finalmente cada partícula cuenta con un vector d-dimensional que sirve como memoria, en el cual se guarda la historia de la mejor posición obtenida por la partícula, $\vec{P_{best}} = (P_{best1}, P_{best2}, ..., P_{bestd})$.

En la versión canónica del algoritmo PSO, todas las partículas se informan entre ellas, por lo que se define también el vector $\vec{G_{best}} = (G_{best1}, G_{best2}, ..., G_{bestd})$, que es el vector tal que al evaluarlo indica el mejor valor encontrado en toda la población. A este vector se le denomina el *factor social*, el cual influirá a todas las partículas de la población.

Una vez definida la estructura y los vectores de cada partícula, para dirigirlas al valor óptimo, se aplica una pequeña perturbación a la posición y velocidad actual de las mismas. Esta actualización se lleva a cabo por medio de las ecuaciones 1 y 2, respectivamente.

$$v_{ij}^{t+1} = v_{ij}^{t} + c_1 * r_{1j}^{t} * \underbrace{(P_{bestij}^{t} - x_{ij}^{t})}_{\text{Factor cognitivo}} + c_2 * r_{2j}^{t} * \underbrace{(Gbest_j^{t} - x_{ij}^{t})}_{\text{Factor social}} \tag{1}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{2}$$

Donde:

v_{ij}^{t+1}	es la velocidad que tendrá la partícula i en la iteración $t+1$, en la dimensión j
v_{ij}^{t}	es la velocidad actual de la partícula i en la iteración t, en la dimensión j
c_1 y c_2	son los coeficientes de aceleración
r_1 y r_2	son dos valores aleatorios entre [0,1]
P_{bestij}^{t}	es el mejor valor encontrado hasta la iteración t por la partícula i, en la dimensión j
G_{bestj}^{t}	es el mejor valor encontrado en la iteración t en toda la población, en la dimensión j

Con la Ecuación 2, lo único que haremos será mover la posición de la partícula al sumar ambos vectores.

De las ecuaciones anteriores hay que hacer notar dos cosas importantes, en la Ecuación 1, el término $(P^t_{best\,ij} - x^t_{ij})$ es denominado el *factor cognitivo* de la partícula y el término $(G^t_{best\,j} - x^t_{ij})$ es denominada el *factor social*.

A continuación se muestra el pseudocódigo del algoritmo PSO canónico

Algoritmo 1 Pseudocódigo Versión Canónica de PSO

Entradas: Valores para número de partículas, dimensión, parámetros c_1, c_2, función a optimizar

$T \leftarrow 0$
Inicializar la posición y velocidad aleatoriamente de cada una de las partículas de la población
Calcular el valor de aptitud de cada una de las partículas
Inicializar el valor $Pbest_i$ para cada partícula
Comparar el valor de aptitud de todas las partículas y actualizar el valor del *Gbest*
while NO se cumpla el criterio de paro **do**
    Actualizar la velocidad y posición de cada una de las partículas *i*, usando la ecuación 1 y 2
    Calcular el valor de aptitud de cada una de las partículas
    Actualizar el $P_{best\,i}$, de cada una de las partículas y el valor de G_{best}
    $T \leftarrow T + 1$
end while
return El mejor valor encontrado

Donde:

v^t_{ij}	es la velocidad de la partícula *i* en la dimensión *j* en la iteración *t*
x^t_{ij}	es la posición de la partícula *i* en la dimensión *j* en la iteración *t*
c_1 y c_2	son los coeficientes de aceleración, que determinan la influencia personal y social respectivamente
r^t_{1j} y r^t_{2j}	son dos números aleatorios uniformemente distribuidos en el intervalos [0,1], para la iteración *t*
$P^t_{best\,ij}$	es la mejor posición encontrada hasta la iteración *t* de la partícula *i* en la dimensión *j*
$G^t_{best\,j}$	es la mejor posición encontrada en toda la población hasta la iteración *t* en la dimensión *j*

En la Figura 1 se muestra geométricamente cómo se ve la influencia social y personal que tiene la partícula para llevar a cabo la actualización de la posición.

1.2 PARÁMETROS DEL PSO

El comportamiento de una meta-heurística depende de dos factores importantes: La variación de los operadores y los valores seleccionados para sus parámetros.

Como se pudo observar en la sección anterior, el algoritmo PSO emplea varios parámetros, los cuáles requieren ser establecidos, estos son:

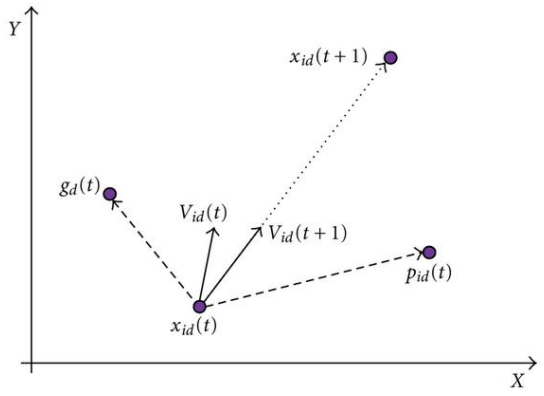


Figura. 1: Representación geométrica del PSO

- Factor de inercia
- Tamaño de la población
- Coeficientes de aceleración
- Factor de constricción

A continuación se da una breve explicación de cada uno de ellos, se explica qué son y su importancia en el algoritmo, así como algunos de los valores que generalmente son usados. La mayoría de estos valores son establecidos de forma empírica o bien por medio de otras estrategias, como lo son el uso de funciones lineales para ir variando el valor del parámetro a lo largo del algoritmo.

1.2.1 *Factor de inercia*

Este parámetro fue agregado a la versión canónica del PSO en 1998 por Shi y Eberhart [2], provee un balance entre la etapa de explotación y de exploración, algo similar a encontrar un balance entre una búsqueda local y una búsqueda global. El parámetro es generalmente denotado por ω.

La propuesta de Shi y Eberhart consiste en cambiar la ecuación de actualización de velocidad, para ahora llevarla a cabo usando la Ecuación 3:

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 r_1 (Pbest_{ij}^{t} - x_{ij}^{t}) + c_2 r_2 (G_{best\,j}^{t} - x_{ij}^{t}) \tag{3}$$

Como se puede observar, su propósito principal es controlar la influencia de la velocidad de la iteración anterior con relación a la velocidad actual.

Entre más grande sea el factor de inercia, mayor es la habilidad de búsqueda en todo el espacio, es decir, un valor más grande es mas apropiado para una búsqueda global; entre más pequeño sea el factor de inercia mayor es la habilidad de búsqueda local.

Con base en el trabajo de Shi y Eberhart, se pueden concluir algunas cosas sobre este parámetro:

Si el factor ω es mayor a 1.2 el algoritmo se comporta como una versión del PSO global y se encuentra siempre tratando de explotar nuevas áreas, lo cual provoca que diverja, ya que siempre se encuentra explorando.

Si el factor ω es un valor entre 0.8 y 1.2 el algoritmo tiene mayor probabilidad de encontrar el óptimo global en un número de iteraciones razonable.

En la versión original propuesta por Shi y Eberhart este parámetro es una constante, sin embargo, se han hecho otras propuestas en las cuáles el factor de inercia es un valor aleatorio, como en el trabajo propuesto por Shi y Eberhart [3] en el que utilizan $\omega = U(0.5,1)$. Hay otras donde emplean una función lineal decreciente para establecer el valor de ω, como la versión que se plantea en la propuesta por Xin, Xhen y Hai en [4], donde ω es establecida mediante la Ecuación 4:

$$\omega^t = \omega_2 + (\omega_1 + \omega_2)\frac{t_{max} - t}{t_{max}} \tag{4}$$

Donde:

ω_1	valor inicial de ω , $\omega_1 = 0.9$
ω_2	valor final de ω, $\omega_2 = 0.4$
t_{max}	máximo número de iteraciones
t	iteración actual

Estas son sólo algunas de las propuestas para el control del factor de inercia, lo que es un hecho es que seleccionando un valor correcto para este, el comportamiento del algoritmo PSO mejora.

Se han realizado estudios comparativos de las diferentes versiones que se han empleado para establecer un valor al factor de inercia como el presentado en [5].

Empíricamente se ha determinado que un buen valor para el factor de inercia es *0.7298* [6] cuando se tiene una topología todos conectados.

1.2.2 *Tamaño de la población*

En general el tamaño de la población dependerá del problema a optimizar, sin embargo, se ha observado que a diferencia de otros algoritmos evolutivos, como lo es el algoritmo genético, el tamaño de la población en el algoritmo PSO es más pequeño, generalmente entre un rango de 20 a 50 partículas.

El elegir una población grande, nos permite cubrir una mayor parte del espacio de búsqueda, dando paso a una búsqueda global. Sin embargo, una población grande, aumenta el número de evaluaciones de función durante la ejecución del algoritmo y por lo tanto aumenta el tiempo de ejecución, pero por otro lado se puede tener mayor probabilidad de llegar a óptimos globales.

1.2.3 *Coeficientes de aceleración*

Estos parámetros juegan un papel importante para la convergencia del algoritmo, controlan qué tanto se moverá una partícula en cada iteración. El primero de ellos usualmente denotado como c_1, es parte del factor cognitivo, que nos indica que tanto influirá la experiencia personal. Por otro lado el segundo de ellos, c_2 es parte del factor social, y nos indicará que tanta influencia tendrá el mejor vecino de la partícula.

Si se tiene un valor grande para el factor cognitivo y uno pequeño para el social, el algoritmo irá promoviendo un comportamiento de búsqueda global y una convergencia al óptimo al final de éste y por el contrario, si tenemos inicialmente un valor grande para componente social y uno pequeño para el componente cognitivo, se promueve más la búsqueda local.

Podemos analizar el comportamiento que tendría el algoritmo en los siguientes casos:

- Si $c_1 = c_2 = 0$, las partículas serán sólo influenciadas por su velocidad actual, sin ser influidas por la información cognitiva ni la social.
- Si $c_1 > 0$ y $c_2 = 0$ en este caso, las partículas serán influenciadas por la velocidad actual de la misma y el factor cognitivo, sin tomar en cuenta la parte social; como no se toma en cuanta el factor social, se puede interpretar como que todas las partículas son independientes.

  Por otro lado, si $c_2 > 0$ y $c_1 = 0$ tendremos que las partículas serán influenciadas por su velocidad actual y el factor social, sin tomar en cuenta su información cognitiva. En este caso al sólo emplear la información del mejor vecino, decimos que todas las partículas están siendo influenciadas hacia un punto.
- Si $c_1 > c_2$ Si tenemos que el factor cognitivo es mayor que el factor social,
- Si $c_1 = c_2$ igualando los valores, nos referimos a que cada una de las partículas, será influenciada por su experiencia personal y social de igual manera.

Suganthan en su trabajo sugiere establecer ambos valores a 2.0, en propuestas posteriores estos valores son controlados por medio de estrategias evolutivas, ya que se ha demostrado que tener distintos valores pueden mejorar el rendimiento del algoritmo PSO.

Alguna de las técnicas para el cambio del valor de los coeficientes de aceleración es la siguiente [15]:

$$c_1 = (c_{1f} - c_{1i})\{\frac{iter}{maxiter}\} + c_{1i} \tag{5}$$

$$c_2 = (c_{2f} - c_{2i})\{\frac{iter}{maxiter}\} + c_{2i} \tag{6}$$

Donde: c_{1i}, c_{1f}, c_{2i}, c_{2f} son constantes que indican el valor inicial y final que tendrán c_1 y c_2 respectivamente.

De igual manera Frans van den Bergh y Andries Engelbrecht, proponen en su trabajo [6], que unos buenos valores para estos coeficientes, son $c_1 = c_2 = 0.1.49618$

1.2.4 *Factor de constricción*

Este parámetro es de los más recientes que han sido agregados al algoritmo PSO, fue propuesto por Clerc y Kennedy [13] en 2000. Lo que se pretende con este parámetro es eliminar el uso del parámetro v_{max} con el que se limita la velocidad, por lo que se propuso modificar la ecuación para modificar la velocidad de las partículas de la población para ahora actualizarla con base a la Ecuación 7

$$v_{ij}^{t+1} = \chi[v_{ij}^t + \varphi_1(P_{bestij}^t - x_{ij}^t) + \varphi_2(L_{bestij}^t - x_{ij}^t)] \tag{7}$$

Donde:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \text{ , } \varphi = \varphi_1 + \varphi_2 > 4 \text{ , } \varphi_1 = c_1 r_1 \text{ , } \varphi_2 = c_2 r_2$$

Clerc y Kennedy, observaron que la convergencia del PSO depende del valor de φ y dieron las siguientes observaciones: Si $\varphi < 4$, se observó que las partículas se encontraban girando alrededor de la mejor solución, por lo que no se podía garantizar la convergencia. Si $\varphi = 4$, es muy difícil que se lleve a cabo la convergencia. Si $\varphi > 4$, se presenta una convergencia casi lineal rápidamente.

Un valor sugerido por los mismo autores, es $\chi = 0.7298$.

1.3 CONDICIONES DE CONVERGENCIA DEL PSO

Se ha demostrado [13] que el comportamiento de convergencia en el PSO, está determinado principalmente por los valores de los parámetros, como el coeficiente de inercia y los coeficientes de aceleración (el factor cognitivo y el factor social). Se conoce por algunos estudios realizados que la mala selección de estos valores puede llegar a producir trayectorias cíclicas divergentes de las partículas.

Otro de los parámetros con los que se intenta evitar la divergencia de las partículas es mediante los valores de v_{min} y v_{max}, cotas para el valor de la velocidad de las partículas. Al limitar la velocidad se evita que una partícula se salga del espacio de búsqueda.

En algunas versiones recientes, se ha eliminado el uso de un v_{min} y v_{max}, ya que se añadió el factor de constricción, y este ayuda a evitar el problema de la divergencia. Sin embargo en esta tesis empleamos la versión del PSO con velocidad acotada.

1.4 CRITERIOS DE PARO

Generalmente el valor óptimo de la función objetivo es desconocido, por ello se tienen diferentes técnicas para poder terminar el proceso de búsqueda. Algunos de los más utilizados se listan a continuación:

1. Se define un número máximo de iteraciones, tal que si se supera este número de iteraciones el algoritmo para, esto es equivalente a determinar un número máximo de evaluaciones de función y una vez que se supera este número el algoritmo para.

2. El algoritmo se detiene cuando el valor del mejor elemento es menor a un umbral de error definido previamente. Este criterio se utiliza en funciones de prueba donde se conoce el mínimo.

3. Al algoritmo se le asignará un determinado tiempo de cómputo y parará cuando este termine, y el mejor valor será el encontrado hasta ese punto.

4. El algoritmo termina la ejecución, si el mejor valor encontrado no ha cambiado a lo largo de varias iteraciones.

1.5 ETAPAS DE EXPLORACIÓN Y EXPLOTACIÓN

Podemos distinguir dos fases del algoritmo PSO a lo largo de las iteraciones: a) exploración y b) explotación. La etapa de exploración es la responsable de la detección de mejores regiones en el espacio de búsqueda [16]. Por otro la etapa de explotación promueve la convergencia de las partículas alrededor de mejores soluciones, [16].

Con base en estas definiciones es fácil darnos cuenta que basados en la varianza que presenta la población podemos determinar en que fase se encuentra el algoritmo. Aunque no hay manera de determinar completamente en qué etapa se encuentra el algoritmo, este criterio nos ofrece una buena medida de qué está ocurriendo.

1.6 TOPOLOGÍAS DEL PSO

Como se explicó anteriormente entenderemos como **topología** a la forma en la que las partículas compartirán información en el algoritmo PSO, es decir, es la forma en la que las partículas se encuentran conectadas entre sí y de acuerdo a esta configuración, cada una de las partículas determina a quien debe informar y de quién obtener información para actualizar su posición. Las topologías en el algoritmo PSO son construidas con base al índice de cada partícula, es decir, cada una de las partículas de la población tiene un identificador que es asignado al momento de la inicialización.

Las topologías juegan un papel importante en el comportamiento del algoritmo PSO, ya que de ellas depende que sea o no capaz de resolver un problema.

Estas pueden acelerar o afectar directamente la convergencia del algoritmo, que una topología funcione correctamente depende del problema que se quiere optimizar.

Podemos hacer una clasificación de las topologías en dos tipos, las topologías *estáticas* y las topologías *dinámicas*. En las estáticas la estructura permanece constante y no presenta algún cambio a lo largo de la ejecución del algoritmo; las topologías dinámicas, modifican su estructura a lo largo de la ejecución del algoritmo.

Definición La *vecindad* de la partícula i, se definirá como el conjunto de partículas con las cuales la partícula i comparte información.

A continuación se explican con mayor detalle cada una de ellas.

1.6.1 *Topologías estáticas*

Las topologías estáticas, como se mencionó, no modifican su estructura durante la ejecución del algoritmo. Sin embargo, como la topología depende del problema, el algoritmo PSO converge prematuramente o es poco eficiente en algunos problemas.

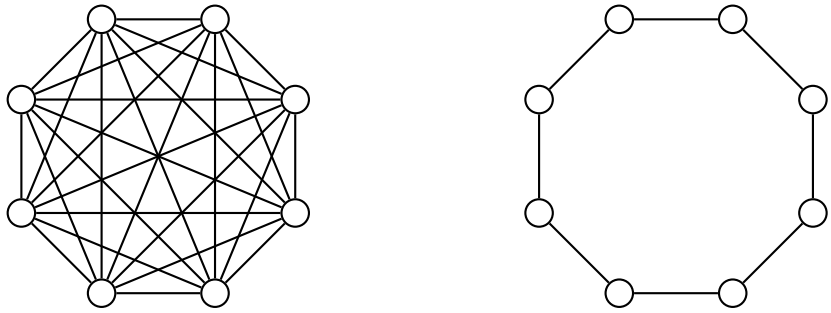
Las topologías clásicas y comúnmente usadas son la topologías *gbest* y *lbest*. En la primera de ellas, cada una de las partículas está conectada al resto de las partículas de la población; en la segunda cada una de las partículas de la población se encuentra conectada con sus k vecinos más cercanos.

A continuación, se explican con mayor detalle las topologías estáticas comúnmente utilizadas en el algoritmo PSO.

1.6.1.1 *Todos conectados*

La topología todos conectados o totalmente conectada, también conocida como *gbest*, es la más conocida y comúnmente empleada. En esta cada una de las partículas se encuentra conectada con el resto de las partículas de la población, como se puede observar en la Figura 2a. En estudios realizados, se ha analizado el comportamiento de esta topología, y se ha descubierto que converge rápidamente, sin embargo, es susceptible a converger en óptimos locales.

Esta topología es la que se emplea en la versión canónica del algoritmo PSO.



(a) Todos conectados (b) Anillo

Figura. 2: Topologías estáticas empleadas en el algoritmo PSO

1.6.1.2 *Anillo*

La siguiente topología más conocida es la topología de *anillo*, también conocida como *lbest*, en la cual cada una de las partículas sólo se comunicará con las k partículas inmediatas a ellas. Se le conoce como topología de anillo, ya que gráficamente, puede verse como un anillo.

En este tipo de topología la comunicación entre las partículas sólo se da localmente y la transmisión de información en este tipo de topología se realiza de forma más lenta entre las partículas. Sin embargo, con una topología de anillo se explorará mejor el espacio de búsqueda, a diferencia de una topología todos conectados.

En esta variante del algoritmo, se asume que las partículas sólo tienen conocimiento de los valores de las partículas que están a su alrededor, es decir, sus vecinos más próximos. En la versión más conocida, para cada una de las partículas i, se tiene que se comunica solamente con sus 2 partículas inmediatas, por lo que en un enjambre de tamaño N, tendríamos que la partícula i, estará siendo informada por la partícula $(i-1) \mod N$ y la partícula $(i+1) \mod N$, como se puede observar en la Figura 2b.

A continuación se muestra el pseudocódigo del algoritmo PSO, con una topología de anillo.

Algoritmo 2 Pseudocódigo PSO con topología de anillo

Entradas: Valores para número de partículas, dimensión, parámetros c1,c2, función a optimizar

1: $T \leftarrow 0$
2: Inicializar la posición y velocidad aleatoriamente de cada una de las partículas de la población
3: Calcular el valor de aptitud de cada una de las partículas
4: Inicializar el valor $Pbest_i$ para cada partícula
5: Comparar el valor de aptitud de todas las partículas
6: **while** NO se cumpla el criterio de paro **do**
7: Actualizar la velocidad y posición de cada una de las partículas i, usando las ecuaciones 8 y 9

$$v_{ij}^{t+1} = \omega * v_{ij}^{t} + c_1 * r_{1j}^{t} * (P_{bestij}^{t} - x_{ij}^{t}) + c_2 * r_{2j}^{t} * (L_{bestij}^{t} - x_{ij}^{t}) \tag{8}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{9}$$

8: Calcular el valor de aptitud de cada una de las partículas
9: Actualizar el P_{besti}, de cada una de las partículas
10: $T \leftarrow T + 1$
11: **end while**
12: **return** El mejor valor encontrado

Donde:

v_{ij}^{t}	es la velocidad de la partícula i en la dimensión j en la iteración t
x_{ij}^{t}	es la posición de la partícula i en la dimensión j en la iteración t
c_1 y c_2	son los coeficientes de aceleración, definen la influencia personal y social respectivamente
r_{1j}^{t} y r_{2j}^{t}	son dos números aleatorios uniformemente distribuidos en el intervalos [0,1], para la iteración t
P_{bestij}^{t}	es la mejor posición encontrada hasta la iteración t de la partícula i en la dimensión j
L_{bestij}^{t}	es la mejor posición encontrada de los vecinos de la partícula i en la dimensión j en la iteración t

1.6.1.3 *von Neumann*

La topología von Neumann, también conocinada como **toroidal**, fue propuesta por Kennedy y Mendes [7]. En esta, cada partícula está conectada a sus 4 vecinos (este, oeste, norte, sur).

La asignación de los vecinos se realiza de la siguiente manera, de modo que aún las partículas de las esquinas tengan cuatro vecinos:

1. Sea N el número de partículas en el enjambre.
2. Se ordenan las N partículas en c columnas y r renglones, tal que $N = c * r,\ c > 0\ y\ r > 0$
3. Para cada una de las partículas $i\ \varepsilon\ \{1,2,...,N\}$

a) Vecino Norte: $N_i(1) = (i-c)\ mod\ N$, si $N_i(1) = 0$, *entonces* $N_i(1) = N$

b) Vecino Oeste: $N_i(2) = i-1$, *si* $(i-1)\ mod\ c = 0$, *entonces* $N_i(2) = i-1+c$

c) Vecino Este: $N_i(3) = i+1$, *si* $i\ mod\ c = 0$, *entonces* $N_i(3) = i+1-c$

d) Vecino Sur: $N_i(4) = (i+c)\ mod\ N$, *si* $N_i(4) = 0$, *entonces* $N_i(4) = N$

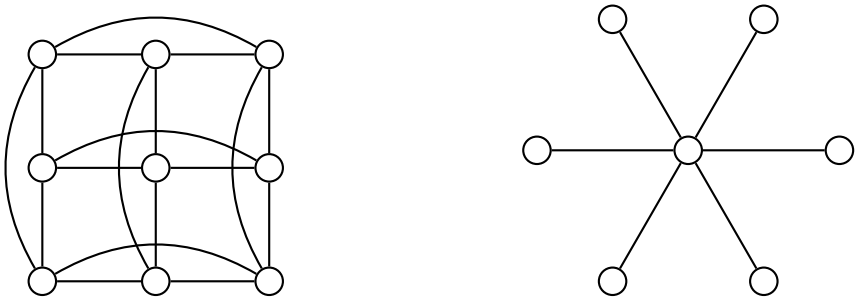
La estructura que presenta se puede observar en la Figura 3a.

1.6.1.4 *Estrella*

En esta topología, se toma de manera aleatoria a alguna de las partículas de la población de tamaño N como la **partícula central** y el resto de las partículas se encuentran conectadas a ella, es decir, cada una de las $N-1$ partículas restantes tienen sólo como vecina a la partícula central y la partícula central tiene $N-1$ vecinos.

La información de las partículas está siendo aislada del resto, mientras que la partícula central está siendo informada por el resto, dirigiéndose hacia la mejor de ellas, como sólo esta se encarga de compartir la información, se presenta una lenta transmisión de la información del mejor elemento. Esto ayuda a evitar la convergencia prematura; sin embargo, rompe un poco el enfoque de colaboración de la población.

Esta topología está ilustrada en la Figura 3b.



(a) von Neumann (b) Estrella

Figura. 3: Topologías estáticas empleadas en el algoritmo PSO

1.6.2 *Topologías dinámicas*

Existe otro tipo de topologías en las cuáles la forma en la que las partículas están conectadas, varía durante la ejecución del algoritmo, a este tipo de topologías se le conoce como *topologías dinámicas*.

Lo que se pretende con este tipo de topologías es encontrar un equilibrio entre la etapa de exploración y explotación del algoritmo PSO, ya que se desea sea capaz de adaptarse al entorno con el fin de no perder diversidad prematuramente ni converger a óptimos locales.

Se ha estudiado el comportamiento del algoritmo frente a diferentes funciones de prueba conocidas y se ha observado que de acuerdo a la clasificación de éstas, el algoritmo presenta diferentes comportamientos. Por ejemplo en funciones unimodales, una topología con un mayor número de conexiones,es decir, en donde cada partícula tiene un mayor número de vecinos, le permite llegar al óptimo en un menor número de evaluaciones de función. Por otro lado en funciones multimodales, se ha observado que al tener una topología con un número grande de conexiones, el algoritmo pierde rápidamente diversidad y converge en óptimos locales; en cambio con una topología con menor número de conexiones, no converge tan fácilmente en óptimos locales. Sin embargo, le toma un mayor número de evaluaciones de función llegar al óptimo.

A través de los años se han propuesto diferentes versiones, en donde se pretende modificar la topología de algoritmo a lo largo de las iteraciones, desde ideas básicas, como la versión del algoritmo PSO que durante el 90% de sus iteraciones genera una topología aleatoria y finalmente en el 10% de iteraciones restantes utiliza una topología completamente conectada, con el fin de realizar explotación.

Otras versiones están basadas en la teoría de los mundos pequeños (small world networks), en la que a través del algoritmo, se decide si se eliminan o bien se añaden conexiones a la topología. Tiene sentido el querer aplicar esta teoría al algoritmo, ya que podemos representar a las partículas y las conexiones entre estas como una gráfica, en la que las partículas juegan el papel de nodos y las conexiones el papel de aristas.

En el siguiente capítulo se presentan con mayor detalle algunas de las técnicas que se han empleado para realizar el cambio de topología de forma dinámica a lo largo del algoritmo.

2

TÉCNICAS DE AUTO ADAPTACIÓN DEL PSO

En el presente capítulo se presentan los dos enfoques principales que se emplean para mejorar el rendimiento y resultados obtenidos con el algoritmo PSO, estos son: mediante el ajuste de parámetros y el cambio de topología. En la primer sección se explica a detalle la forma en la que se lleva a cabo el cambio del valor de los parámetros y se presentan algunas propuestas de diversos autores donde se emplea esta técnica. En la segunda sección del capítulo, se presentan la formas en las que se realiza el cambio de topología y al igual que en la primera sección se enlistan algunas propuestas del algoritmo PSO con cambio de topología.

2.1 TÉCNICAS DINÁMICAS DE CONTROL DE PARÁMETROS

En este tipo de técnicas se desea mejorar el comportamiento del algoritmo PSO, cambiando el valor de los parámetros durante la ejecución del mismo. Para ello tendremos dos enfoques principales, a) el primero, el enfoque determinista; b) el segundo, el enfoque auto adaptativo. En el primero de ellos, el cambio de parámetros se lleva a cabo de forma determinista, es decir, se establece desde el inicio del algoritmo cómo se llevará a cabo el cambio del valor de los parámetros, mediante ecuaciones, por lo que durante la ejecución se lleva a cabo la actualización de los parámetros haciendo uso de estas ecuaciones y no se toma en cuenta el estado actual de la población en el algoritmo PSO. En el enfoque auto adaptativo se miden ciertos parámetros, como el estancamiento de la población, la distancia al mejor individuo, entre otros, y a partir de éstos se determina cómo o cuándo se realizará el cambio del valor de los parámetros.

En las siguientes secciones se presentan algunos trabajos que podemos clasificar en estos tipos de enfoques.

2.1.1 *Cambio de parámetros de forma determinista*

Fusion Global-Local-Topology Particle Swarm Optimization for global optimization problems (FGLT-PSO)[8]

El algoritmo FGLT-PSO es una versión que propone una mezcla entre una topología completamente conectada y una topología de anillo y así lograr un mejor control de las etapas de exploración y explotación, pretende evitar la convergencia prematura que presenta la versión canónica del PSO y por otro lado mejorar el porcentaje de éxito.

Podría pensarse que es una versión en donde cambia la topología, sin embargo la topología es la misma durante la ejecución del algoritmo, sólo es una combinación entre una topología de anillo y una topología de todos conectados. Por otro lado, los autores proponen que los parámetros ω y las constantes de aceleración c_1 y c_2, se modifiquen de acuerdo al número de iteraciones del algoritmo.

Sin embargo, al no tomar en cuenta el entorno y sólo modificar el valor de los parámetros con base al número de iteraciones, se puede decir que hablamos de una versión del algoritmo PSO con modificación de los parámetros de forma determinista.

La actualización de la posición y velocidad de partículas en esta versión se lleva a cabo a partir de las siguiente ecuaciones:

$$v_{ij}^{t+1} = \omega_i^t * v_{ij}^t + c_1 * r_1 * (P_{besti\,j}^t - x_{ij}^t) + c_2 * r_2 * (P_{lbest\,j}^t - x_{ij}^t) \tag{10}$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} + c_3 * r_3 * (P_{gbest\,j}^t - x_{ij}^t) \tag{11}$$

Como podemos notar en la ecuación 10, el segundo término es el factor cognitivo, por otro lado el tercer término en las ecuaciones 10 y 11 conforman el factor social de la partícula.

Los parámetros ω y los coeficientes de aceleración son calculados mediante las ecuaciones 12 y 13

$$\omega^t = \omega_{max} - \frac{t * (\omega_{max} - \omega_{min})}{T} \tag{12}$$

$$c_j^t = c_{jmin} + \frac{t * (C_{jmax} - C_{jmin})}{T}; j = 1,2,3 \tag{13}$$

Donde t es la iteración actual, T es el número máximo de iteraciones y c_{jmin} y c_{jmax} son el valor mínimo y máximo que tendrá el coeficiente de aceleración j, respectivamente. En este trabajo los autores proponen los siguientes valores para los coeficientes de aceleración: $c_{1min} = 0.5$, $c_{1max} = 2$, $c_{2min} = 1$, $c_{2max} = 2$ y $c_{3min} = 0.5$ y $c_{3max} = 1.5$

Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients [15]

En este trabajo Ratnaweera, Halgamuge y Watson proponen tres versiones del algoritmo PSO, en las cuáles se modifican el valor de los coeficientes de aceleración con el fin de controlar de forma eficiente la búsqueda local y la convergencia del algoritmo.

La primer propuesta que se presenta en este trabajo es la propuesta PSO-TVAC (Time Varying Acceleration Coefficients), en donde se varía el valor de los coeficientes de aceleración con el fin de tener un comportamiento de búsqueda global en la primer parte del algoritmo y en la parte final, guiar la convergencia de las partículas hacia el mejor. Para lograrlo, el valor del componente cognitivo es afectado mediante el decremento de c_1. Por otro lado el valor del componente social es afectado por medio del incremento del valor de c_2. Los valores de c_1 y c_2 son calculados de la siguiente manera:

$$c_1 = (c_{1f} - c_{1i}) \frac{iter}{MAXITER} + c_{1i} \tag{14}$$

$$c_2 = (c_{2f} - c_{i2}) \frac{iter}{MAXITER} + c_{2i} \tag{15}$$

Donde c_{1f}, c_{1i}, c_{2f} y c_{2i}, son constantes, $iter$ se refiere a la iteración actual y MAXITER es el número máximo de iteraciones permitidas. Los valores con los que empíricamente se obtuvieron

mejores resultados fueron variando c_1 de 2.5 a 0.5 y c_2 de 0.5 a 2.5. Además de cambiar el valor de los coeficientes de aceleración, los autores recomiendan utilizar el factor de inercia lineal [4].

La segunda propuesta que presentan es MPSO-TVAC (*Particle Swarm Optimizer with "Mutation" and Time Varying Acceleration Coefficients*). En esta propuesta agregaron un operador de mutación, cuyo propósito es aportarle a las partículas mayor diversidad. Este operador se activará cuando el mejor valor encontrado no haya mejorado a lo largo de cierto número de iteraciones. El operador de mutación funciona de la siguiente manera: una partícula de la población es seleccionada aleatoriamente y por medio de una probabilidad de mutación se selecciona aleatoriamente un componente de la velocidad de la partícula, al cual se le aplicará una pequeña perturbación, la cual es proporcional a la velocidad máxima permitida. Adicional al operador de mutación los coeficientes son modificados por medio de las Ecuaciones 14 y 15.

Algoritmo 3 Pseudocódigo MPSO-TVAC

```
1: Inicializar la posición y velocidad de las partículas aleatoriamente dentro del espacio de
   búsqueda
2: while Criterio de paro no se cumpla do
3:     for Para cada una de las partículas i do
4:         Evaluar el valor de aptitud
5:         Actualizar el valor de Pbest y P_g
6:         for Para cada una de las dimensiones d do
7:             Calcular la nueva velocidad v_id
8:             Actualizar la nueva posición de la partícula
9:         end for
10:    end for
11:    Seleccionar aleatoriamente una partícula k
12:    Seleccionar aleatoriamente una dimension l
13:    if Δglobal ≤ 0 then
14:        if rand_1 < P_m then
15:            if rand_2 < 0.5 then
16:                v_kl = v_kl + rand_3 * vmax/m
17:            else
18:                v_kl = v_kl − rand_4 * vmax/m
19:            end if
20:        end if
21:    end if
22: end while
```

Donde $rand_1$, $rand_2$, $rand_3$, $rand_4$, son números aleatorios uniformemente distribuidos en [0,1], P_m es la probabilidad de mutación y $\Delta global$ es la diferencia del mejor valor encontrado a lo largo de las generaciones.

Y finalmente, en la tercer propuesta que presentan HPSO-TVAC (Self-organizing Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients), se pretende que las partículas encuentren el óptimo global, sin utilizar el término de la velocidad en la ecuación de

actualización de velocidad. En esta versión cómo se mencionó no se toma en cuenta el término de la velocidad, sin embargo, se emplea un mecanismo de reinicio de la velocidad si esta está estancada. Y en esta propuesta también se lleva a cabo la actualización de la velocidad por medio de las ecuaciones 14 y 15.

Algoritmo 4 Pseudocódigo HPSO-TVAC

```
Inicializar la posición y velocidad de las partículas aleatoriamente dentro del espacio de búsqueda
while Criterio de paro no se cumpla do
    for Para cada una de las partículas i do
        Evaluar el valor de aptitud
        Actualizar el valor de Pbest P_id y P_g
        for Para cada una de las dimensiones d do
            Calcular la nueva velocidad v_id
            v_id = c1 * rand1 * (P_id − x_id) + c2 * rand2 * (P_gid − x_id)
            if v_id = 0 then
                if rand3 < 0.5 then
                    v_id = rand4 * v
                else
                    v_id = rand5 * v
                end if
            end if
            v_id = sign(v_id * min(abs(v_id, v_max))
            Actualizar la nueva posición de la partícula
        end for
    end for
end while
```

2.1.2 *Técnicas de auto adaptación de los parámetros*

A modified particle swarm optimizer with dynamic adaptation (DAPSO)[9]

Es una implementación del algoritmo PSO, en donde se decrementa razonablemente la aleatoriedad al actualizar la posición de la partícula y además de esta modificación, el factor de inercia es diferente para cada partícula de la población.

La velocidad y posición de las partículas será actualizada de acuerdo a las siguientes ecuaciones:

$$v_{ij}^{t+1} = \omega_i^t * v_{ij}^t + c_1 * r_1 * (P_{best\,ij}^t - x_{ij}^t) + c_2 * r_2 * (G_{best\,j}^t - x_{ij}^t) \tag{16}$$

$$x_{ij}^{t+1} = v_{ij}^{t+1} + x_{ij}^t \tag{17}$$

Donde:

ω_i^t es el factor de inercia para la partícula i en la iteración t
c_1, c_2 son los coeficientes de aceleración
r_1, r_2 son dos números aleatorios uniformemente distribuidos en el intervalo $[0,1]$
$G_{best\,j}$ es el mejor individuo de la población en la dimensión j, en la iteración t

A diferencia del PSO canónico, los valores r_1 y r_2 serán los mismos para todas las dimensiones de la partícula i y el factor de inercia es variable para cada una de las partículas, además de ser modificado de acuerdo a dos factores: el factor de velocidad y el grado de agregación, los cuales se presentan a continuación.

El factor de velocidad se calcula de la siguiente manera:

$$h_I^t = \left| \frac{\min(F(Pbest_i^{t-1}), F(Pbest_i^t))}{\max(F(Pbest_i^{t-1}), F(Pbest_i^t))} \right| \tag{18}$$

Donde, $F(Pbest_i^t)$ es el valor de aptitud de P_{besti}^t

Por otro lado el grado de agregación es calculado de la siguiente manera:

$$s = \left| \frac{\min(F_{tbest}, \overline{F_t})}{\max(F_{tbest}, F_t)} \right| \tag{19}$$

Donde, $\overline{F_t}$ es el promedio de aptitud de todas las partículas de la población, en la iteración t.

Se definirá el valor de factor de inercia para cada partícula, como una función dada por h y s

$$\omega_i^t = f(h_i^t.s) \tag{20}$$

Para el caso particular de esta variante del algoritmo PSO, el factor de inercia se calculará de la siguiente manera:

$$\omega_i^t = \omega_{ini} - \alpha(1 - h_i^t) + \beta s \tag{21}$$

Donde ω_{ini} es el valor inicial del factor de inercia, en este caso, $\omega_{ini} = 1$ y α y β son generalmente valores en el intervalo [0,1].

Como se puede notar el valor de ω estará afectado por el factor de velocidad, que toma en cuenta el valor de aptitud. Por lo tanto podemos considerar esta propuesta como un técnica de actualización de parámetros auto adaptativa, ya que toma en cuenta el estado del algoritmo para realizar la actualización de los valores de los parámetros.

2.2 TÉCNICAS DE MODIFICACIÓN DE TOPOLOGÍA

Existe otro enfoque para mejorar el rendimiento del algoritmo PSO, el cual consiste en cambiar la topología y esta modificación se puede hacer de dos formas:
a) de forma determinista; es decir, a priori disponemos de ecuaciones que nos indican cómo y cuándo se hará el cambio de topología.
b) o bien auto adaptar la topología, con base al estado que presenta el algoritmo hasta ese momento, es decir, con base al estado reconectar las partículas, añadir o eliminar conexiones, entre otros cambios.

A continuación se presentan algunas versiones del algoritmo PSO que podemos clasificar en estos dos grupos.

2.2.1 *Cambio de topologías de forma determinista*

Hierarchical Dynamic Neighborhood Based Particle Swarm Optimization for Global Optimization (Hierarchical D-LPSO)[10]

Es una variante del algoritmo PSO en la que las partículas se encuentran organizadas según una jerarquía dinámica, es decir, las partículas se encuentran organizadas mediante un árbol y durante la ejecución del algoritmo, las partículas pueden subir o bajar de nivel en la jerarquía.

Para la topología en este algoritmo la población se divide en niveles, a excepción del primer nivel que tendrá solo una partícula, los siguientes niveles tendrán n partículas, (el valor inicial de n es 2), el nivel K tendrá a lo más n^K partículas. Luego cada partícula en el nivel K, será asignado de forma aleatoria a un padre del nivel anterior, es decir, a una partícula del nivel $K-1$, excepto por la partícula del primer nivel.

Las partículas tienen la posibilidad de descender o ascender en la jerarquía, esto se hace comparando el valor de aptitud del hijo con la del padre, si el valor de aptitud del hijo es mejor que la del padre, entonces el hijo tomará el lugar del padre en la jerarquía.

Cada que se alcanza un quinto del total de las evaluaciones de función, se lleva a cabo la actualización de la topología, donde ahora $n = n+1$ y se construye nuevamente una topología, como se explicó anteriormente.

Para la actualización de la velocidad y posición de partículas, se utilizó la versión del factor de constricción.

$$v_{ij} = \chi * (\omega * v_{ij} + c_1 * random(0,1) * (P_{kj} - x_{ij}) + c_2 * random(0,1) * (Lbest_{ij} - x_{ij})) \tag{22}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{23}$$

Donde:

$$\chi = \frac{2}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \; y \; c = \sum_i c_i$$

El vecindario se forma de a lo más tres partículas seleccionadas aleatoriamente en el mismo nivel.

Este algoritmo es del tipo de topologías pre programadas, ya que el cambio de topología se define a priori, con esto nos referimos a que el cambio de topología sólo está definido al número de evaluaciones de función.

2.2.2 *Técnicas de auto adaptación de la topología*

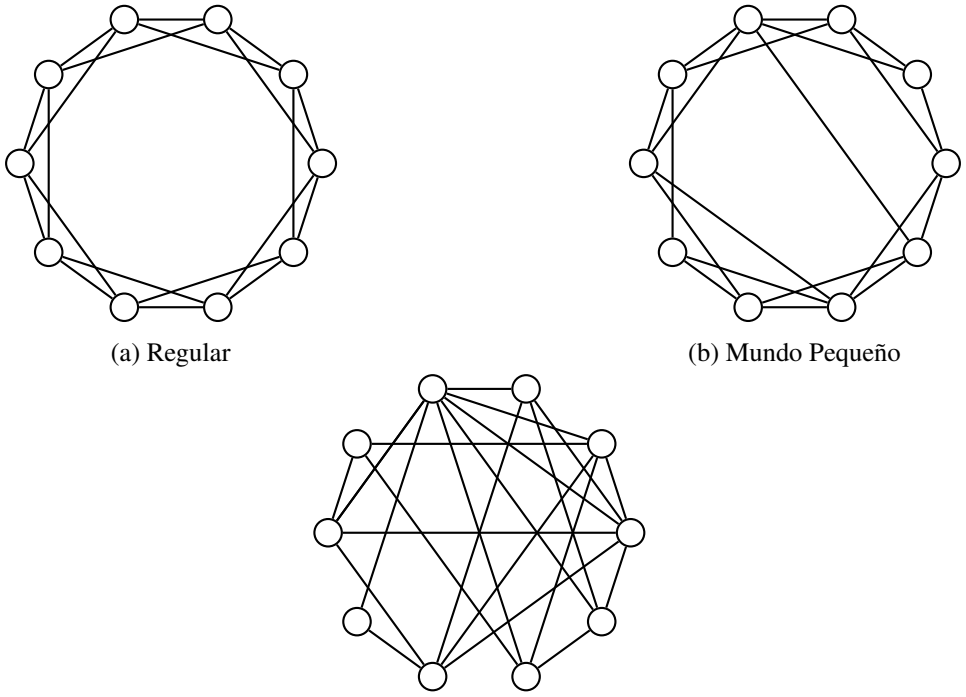
Small-World Particle Swarm Optimization with Topology Adaptation (ASWPSO)[11]

Es una versión del algoritmo del PSO, con una topología dinámica basada en la idea de los mundos pequeños. A diferencia de otros trabajos se tiene que por cada dimensión j de cada una de las partículas i, se tendrá una topología, esto con el fin de tener mayor diversidad y evitar la convergencia prematura del algoritmo.

Al estar basado en la teoría de los mundos pequeños [14], cada una de las conexiones que existen entre las partículas tienen la posibilidad de reconectarse con cualquiera de otras partículas del enjambre con cierta probabilidad.

La construcción de los mundos pequeños inicia como una gráfica regular, la cual puede reconectar las artistas con una probabilidad p, cambiando el fin de la misma hacia otro vértice en la gráfica. El valor de la probabilidad p varía entre 0 y 1.

Podemos notar que si el valor de probabilidad es 0, entonces tendremos el caso de un gráfica regular, y por otro lado si el valor de probabilidad es 1, tendremos el caso de una gráfica completamente aleatoria. Como se observa en la Figura 4



(a) Regular

(b) Mundo Pequeño

(c) Aleatoria

Figura. 4: Mundos pequeños

Otro aspecto importante del algoritmo es que durante la ejecución del mismo, el tamaño de las vecindades se modifica dinámicamente. Al igual que la probabilidad de intercambiar aristas, esta adaptación se va efectuando de acuerdo a la convergencia de la población que se presenta durante la ejecución del algoritmo, con el fin de encontrar un equilibrio entre las etapas de exploración y explotación.

La idea del algoritmo es la siguiente, al inicio cada una de las partículas se encuentra conectada con sus K vecinos sucesores, $\{i+1, i+2, ..., i+K\}$, y por cada una de las partículas se tiene la posibilidad de reconectar una arista hacia cualquier otra partícula de la población con una probabilidad P.

Como se mencionó anteriormente, cada dimensión de la partícula tiene asociada una topología, por lo que se generará un vector $Ne_i = (ne_{i1}, ne_{i2}, ..., ne_{iD})$, donde cada ne_{ij} es el índice de la partícula seleccionada en la topología de la dimensión j, para informar a la partícula i.

A diferencia del PSO canónico, en ASWPSO la actualización de la velocidad y posición de la partícula no utiliza el P_{best} y G_{best} para informarse, sino utiliza el vector generado por las topologías de cada dimensión.

Teniendo los vectores $V_{ij} = \{v_{i1}, v_{i2}, ..., v_{iD}\}$, $X_{ij} = \{x_{i1}, x_{i2}, ..., x_{iD}\}$ y $P_{besti} = \{p_{i1}, p_{i2}, ..., p_{iD}\}$ de cada partícula, la velocidad y posición se actualiza de la siguiente manera:

$$v_{ij} = \omega * v_{ij} + c * random(0,1) * (P_{kj} - x_{ij}) \tag{24}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{25}$$

Donde:

ω	es el factor de inercia
c	es el coeficiente de aceleración
$k = nei_{ij}$	índice de la partícula que informa en la dimensión j a la partícula i
P_{kj}	valor del personal best de la partícula que informará a la partícula i en la dimensión j

Como se mencionó anteriormente, la probabilidad de reconectar una partícula con otra se va adaptando al algoritmo de acuerdo a la convergencia del mismo, por ello se calcula el coeficiente S_c para medir el estancamiento de la población, de la siguiente manera:

$$S_c = \frac{\sum_{i=1}^{N} S_i / N}{S_g} \tag{26}$$

Donde N es el tamaño de la población y S_i es el número de generaciones que lleva estancada la partícula i. Se determinará que una partícula se encuentra estancada, si esta no ha mejorado con el mismo vector de vecinos que tiene actualmente. S_g es un umbral, que determina el número máximo de generaciones que tiene una partícula para estar estancada, antes del cambio de topología.

Los parámetros K, el número de vecinos de cada partícula y P, la probabilidad de reconectarse, son adaptados de acuerdo a las siguientes ecuaciones

$$K = \lfloor 2.5 + 5 * S_c \rfloor \tag{27}$$

$$P_i = 0.05 * 2^{S_c} + 0.1 * \frac{i}{N} \tag{28}$$

A continuación se muestra el pseudocódigo del algoritmo ASWPSO y el de cómo se lleva a cabo la actualización de las topologías para cada dimensión.

Algoritmo 5 Pseudocódigo ASWPSO

Entradas: N número de partículas, D dimensión, valores de coeficientes de aceleración c_1 y c_2 , valor de factor de inercia ω, f función a optimizar, S_g umbral de tolerancia

$T \leftarrow 0$
Inicializar la posición y velocidad de las partículas aleatoriamente dentro del espacio de búsqueda
Calcular el valor de aptitud de cada una de las partículas
Inicializar el valor de P_{besti} para cada una de las partículas de la población
while Criterio de paro no se cumpla **do**
    for Para cada una de las partículas i **do**
        if $s_i \geq s_g$ **then**
            Actualizar los valores de K y P_i, como se muestra en las ecuaciones 27 y 28 respectivamente
            Actualizar la topología de la partícula i con el algoritmo 6
        else
            Actualizar la velocidad y posición de cada una de las partículas con las ecuaciones 24 y 25
        end if
        Actualizar el valor de estancamiento s_i, diremos que está estancada si la partícula no mejora su mejor valor encontrado hasta esa iteración
    end for
    Actualizar el valor de S_c con la ecuación 26
    Calcular el valor de aptitud de las nuevas partículas
    Actualizar el P_{best} para cada una de las partículas
    $T \leftarrow T + 1$

end while
return El mejor valor encontrado

Algoritmo 6 Pseudocódigo de actualización de topología ASWPSO

```
for each dimensión j de la partícula i do
    r = random(0,1)
    if r < P then
        aleatoriamente elegir una partícula k de la población
        Ne_ij = k
    else
        aleatoriamente elegir una partícula k de sus K vecinos de la partícula i
        if valor de fitness de la partícula k es mejor que el de la partícula i then
            Ne_ij = j
        else
            Ne_ij = i
        end if
    end if
end for
return Nueva topología Ne
```

Particle Swarm Optimization with increasing topology connectivity (PSO-ITC)[12]

Esta versión intenta encontrar un equilibrio entre la etapa de exploración y explotación en el algoritmo, para ello se añade el módulo ITC al PSO. Este incrementa linealmente el número de conexiones en la topología, también incluye un mecanismo que ayuda a prevenir el estancamiento del enjambre; adicionalmente esta versión cambia la forma en la que la velocidad de las partículas se actualiza.

Inicialmente cada partícula estará conectada con solo un vecino del enjambre elegido aleatoriamente y a lo largo de la ejecución del algoritmo, se va incrementando la conectividad de las partículas, hasta que estas estén totalmente conectadas.

Cabe destacar que a diferencia de la versión canónica del PSO aquí las partículas no están conectadas de forma bidireccional, es decir, sí la partícula i está conectada con la partícula j, no necesariamente la partícula j estará conectada con la partícula i.

La conectividad de la partícula i en la topología cambia mediante la siguiente ecuación:

$$TC_i = \lfloor TC_{min} + TC_{max}[(k-1)/(FE_{max}-1)] \rfloor \tag{29}$$

Donde:

TC_i	es la conectividad actual de la topología
TC_{min}, TC_{max}	son el grado mínimo y máximo de conectividad, respectivamente
k	es el valor actual de evaluaciones de función hasta el momento
FE_{max}	valor máximo de evaluaciones de función

$TC_{min} = 1$ y $TC_{max} = N-1$, donde N es el número de partículas en el enjambre, y TC_{max}, es el máximo número de aristas que puede tener una partícula, es decir, si cada partícula del enjambre tiene TC_{max} conexiones, estaremos hablando de una topología de todos conectados.

En cada iteración la conectividad TC_i de cada partícula irá incrementando ΔTC_i, lo que implica seleccionar ΔTC_i nuevos vecinos del enjambre, para la partícula i.

Por otro lado para evitar el estancamiento del enjambre, si la partícula i no ayuda a mejorar el mejor valor de fitness encontrado por la vecindad durante z evaluaciones de función, se le reasignarán TC_i nuevos vecinos aleatoriamente.

Adicionalmente para evitar el estancamiento de la mejor partícula encontrada P_g, a esta se le aplicará una perturbación para ayudarla a escapar de óptimos locales, específicamente se le aplicará una pequeña perturbación en la dimensión d, de la siguiente manera:

$$P_{g,d}^{per} = r_3 P_{g,d} + (1 - r_3)(P_{x,d} - P_{y,d})$$

Donde:

$P_{x,d}$ y $P_{y,d}$	es la mejor posición encontrada de dos partículas x y y del enjambre elegidas aleatoriamente
r_3	es un aleatorio uniformemente distribuido entre [0,1]

Si al reemplazar el valor obtenido en la dimensión d por $P_{g,d}^{per}$ y si al evaluarlo el valor de aptitud es mejor al de P_g, entonces actualizamos el P_g.

En esta versión la actualización de la velocidad de la partícula se propone un entorno de aprendizaje, el cual inspirado en los factores cognitivo y social con los que tradicionalmente se modifica la velocidad de las partículas, generará dos factores $C_{exp,i}$ y $S_{exp,i}$ para guiar a la partícula i en el proceso de búsqueda. Para derivar el valor de estos factores, primero ordenaremos los vecinos de la partícula i, incluyendo a esta, de acuerdo a su mejor valor encontrado P_{besti} de forma ascendente y aquellos vecinos que estén dentro del primer cuartil, formarán el grupo $neighbor\_upper_i$ y servirán para generar $S_{exp,i}$; el resto de los vecinos formarán el grupo $neighbor\_lower_i$ y servirán para generar $C_{exp,i}$.

Para generar los ejemplares de $C_{exp,i}$ y $S_{exp,i}$ se emplea una selección ruleta de los grupos $neighbor\_lower_i$ y $neighbor\_upper_i$ respectivamente, a cada uno de los elementos k, en estos conjuntos se le asignará un peso W_k

$$W_k = \frac{f_{max} - f(P_k)}{f_{max} - f_{min}}, \forall k \in [1, K]$$

Donde:

f_{max}	es el valor del peor fitness personal de los miembros del conjunto $neighbor\_lower_i$
f_{min}	es el mejor valor de fitness personal de los miembros del conjunto $neighbor\_upper_i$
K	es la cardinalidad de los conjuntos $neigbor\_upper_i$ o $neighbor\_lower$

Una vez definido esto, se prosigue a generar los ejemplares de $C_{exp,i}$ y $S_{exp,i}$.

Para la parte de la actualización de la velocidad se considerarán el valor de $C_{exp,i}$ y el mejor elemento del enjambre P_g. Debido a que se utilizará el valor $C_{exp,i}$ y este es elegido mediante un mecanismo probabilístico. Se presentan dos escenarios, el primero de ellos es cuando $C_{exp,i}$ tiene un mejor valor de aptitud que la partícula i, y es conveniente que la partícula i sea guiada por este valor para ir dirigiendo la dirección de la partícula i hacia un mejor lugar en el espacio de búsqueda; el segundo caso es cuando el valor de aptitud de $C_{exp,i}$ es peor que la aptitud de la partícula i, en este caso no es bueno que la partícula i sea influenciada por esta.

Por lo anterior la velocidad de la partícula i, se actualizará de acuerdo a la ecuación 30

$$V_i = \begin{cases} \omega V_i + c_1 r_4 (C_{exp,i} - X_i) + c_2 r_5 (P_g - X_i), & f(C_{exp,i}) < f(P_i) \\ \omega V_i - c_1 r_6 (C_{exp,i} - X_i) + c_2 r_7 (P_g - X_i), & en\ otro\ caso \end{cases} \tag{30}$$

Donde r_4, r_5, r_6, r_7, son números aleatorios uniformemente distribuidos en el intervalo [0,1]. La actualización de la posición se realiza de la misma forma que en la versión canónica del algoritmo PSO, una vez actualizada la posición de la partícula, se compararán el valor de aptitud de la partícula i, con el valor de aptitud de P_g, si el valor de aptitud de la partícula i es mejor, reemplazaremos P_g por este valor, aplicando aquí un mecanismo elitista.

Ya que actualizar la velocidad de la partícula i no asegura que el valor de aptitud de esta sea mejor, se añadió también un operador de vecindades, para ello se crean dos arreglos $C_{candidate,i}$ y $S_{candidate,i}$, en donde se excluirán los valores $C_{exp,i}$ y $S_{exp,i}$, es decir,
$C_{candidate,i} = [C_{exp,1}, C_{exp,2}, ..., C_{exp,i-1}, C_{exp,i+1}, ... C_{exp,S}]$ y
$S_{candidate,i} = [s_{exp,1}, S_{exp,2}, ..., S_{exp,i-1}, S_{exp,i+1}, ... S_{exp,S}]$, donde S es el tamaño del enjambre.

De estos arreglos se tomarán dos elementos como guía C_{guide} y S_{guide}, mediante una selección por ruleta. Es estos dos valores se define un tercero $O_{exp,i}$, que ayudará a guiar a la partícula i en el operador de vecindades. El valor para cada dimensión d del vector $O_{exp,i}$ estará dado por valor de la dimensión d del vector C_{guide} o bien S_{guide}, de la siguiente manera:

$$O_{exp,i}(d) = \begin{cases} S_{guide,i}(d), & r < 0.5 \\ C_{guide,i}(d), & en\ otro\ caso \end{cases} \tag{31}$$

Donde r es un número aleatorio.

Y finalmente al obtener $O_{exp,i}$, se tienen nuevamente dos escenarios, el primero de ellos es cuando el valor de aptitud de este, es mejor que el valor de aptitud que la partícula i, en este caso la partícula será atraída hacia $O_{exp,i}$, en otro caso no se tomará en cuenta este valor.

Por lo que cada partícula ajustará su P_i de acuerdo a la ecuación 32

$$P_{i,temp} = \begin{cases} P_i + cr_8 (O_{exp,i} - P_i), & f(O_{exp,i}) < f(P_i) \\ P_i - cr_9 (O_{exp,i} - P_i), & en\ otro\ caso \end{cases} \tag{32}$$

El valor de aptitud de $P_{temp,i}$ es comparado contra el valor de aptitud de P_i y P_g, si su valor es mejor que este último, entonces $P_{temp,i}$, reemplazará a P_i y P_g, de lo contrario se aplicará una técnica para poder extraer información de $P_{temp,i}$ que ayude a mejorar el valor de P_g.

3

TOPOLOGÍAS DINÁMICAS DEL PSO

En la primer sección de este capítulo se presentan algunos conceptos básicos que se emplearán en la propuesta que hacemos en esta tesis, en la segunda sección se presenta la propuesta del algoritmo PSO, con una topología dinámica auto adaptable y en la última sección del capítulo se discuten algunos de los resultados observados al realizar algunas ejecuciones del algoritmo.

3.1 CRITERIOS DE LA CONECTIVIDAD DE LA TOPOLOGÍA

La conectividad de una topología se refiere a la forma y a cuantas conexiones hay dentro de la topología, que pueden variar de acuerdo al diseño de la misma.

Lo que se busca con esta nueva propuesta del PSO, es encontrar una forma de guiar al algoritmo para resolver problemas, mediante la auto adaptación del número de conexiones de la topología sin importar el problema a optimizar. Esta topología debe ser capaz de auto adaptarse de acuerdo a la fase en la que se encuentra el algoritmo, es decir, si está en una etapa de exploración, desearíamos que hubiera un menor número de conexiones presentes en la topología, en cambio en una etapa de explotación, nos gustaría tener una topología con un mayor número de conexiones.

Podemos hacer una analogía entre una topología y una gráfica, para ello debemos definir algunos conceptos antes de proseguir y dar la idea de nuestra propuesta.

Definición Una gráfica G, está definida como un conjunto V de *vértices o nodos* y un conjunto E de *aristas*, las cuales indican las conexiones que hay entre los nodos.

$$G = (E,V)$$

Haciendo la analogía entre una topología y una gráfica, cada partícula de la topología será un nodo y cada conexión que hay entre ellas, será representada por una arista.

Definición [20] Una gráfica completa G, con n vértices, es aquella en la que todos los vértices de la gráfica son adyacentes al resto de los vértices, es decir, aquella en la cual entre cualquiera dos vértices de la gráfica, existe una arista. Una gráfica completa es denotada como K_n, donde n es el número de vértices de la gráfica.

Definición El número total de aristas presente en una gráfica completa de n nodos, es:

$$|E| = \frac{n(n-1)}{2} \tag{33}$$

Definición [19] Definimos un camino en G del vértice v_0 al vértice v_n, como una secuencia no vacía, $W = v_0e_1v_1e_2v_2...e_nv_n$ en donde se alternan vértices y aristas.

Definición [19] Una trayectoria de v_0 a v_n,es un camino en G, en donde los vértices y las aristas no se repiten.

Definición [19] Una gráfica conexa es aquella que entre cualesquiera dos nodos u y v existe una trayectoria, es decir, no hay vértices inalcanzables.

3.2 PROPUESTA

Con nuestra propuesta lo que se quiere es encontrar un balance entre las etapas de exploración y explotación, sin necesidad de conocer la función a priori. La idea principal es que la topología de la población sea capaz de irse adaptando de acuerdo al estado actual de las partículas.

Nuestro criterio para incrementar o decrementar la topología, es decir, añadir o eliminar conexiones respectivamente, dependerá de cuánto se haya mejorado el mejor valor obtenido hasta ese punto en el algoritmo.

A continuación se explica con mayor detalle las etapas del algoritmo y cómo hace la actualización de velocidad y posición de las partículas, para ello nos referiremos a la topología como una gráfica.

Como conocemos una forma de acotar el número de aristas presentes en nuestra gráfica, lo que deseamos hacer es que de tener una topología con un número de aristas promedio, esta sea capaz de ir adaptándose a través de las iteraciones, añada o bien elimine aristas, según vaya mejorando el mejor valor de aptitud en la población.

Tras estudiar algunos de los últimos trabajos realizados para mejorar el rendimiento y precisión del algoritmo PSO, se observó que son pocos los trabajos que toman en cuenta el estado de las partículas a lo largo del algoritmo, la mayoría de estos hacen el cambio de parámetros y/o topología de forma determinista.

Muchos trabajos están relacionados con la teoría de los mundos pequeños o bien, con comportamientos que se han observado, desde que, una topología poco conectada ayuda a la etapa de exploración y una topología fuertemente conectada ayuda a la explotación. Sin embargo, no podemos garantizar que esta idea de los mundos pequeños funcione correctamente frente a cualquier problema.

Tratando a la topología como una gráfica, podemos deducir algunas propiedades de estas, como el número de aristas.

Por ejemplo, en una topología de anillo o Lbest, donde cada partícula tiene dos vecinos, el número de aristas que tiene la topología T con N vértices o partículas, será de:

$$|E| = N$$

La idea principal es comenzar con una gráfica que no sea totalmente conectada, ni que tenga pocas aristas, por lo que en nuestra versión la topología tendrá la libertad de ir añadiendo o bien eliminando conexiones, con la posibilidad de convertirse ya sea en una topología de todos conectados o bien una topología de anillo.

Para ello, definiremos dos cotas en el número de aristas que puede tener nuestra gráfica. El primero de ellos E_{min}, corresponde al caso del número de aristas en una topología de anillo.

$$|E_{min}| = N \tag{34}$$

El segundo de ellos E_{max}, hace referencia al total de aristas que se puede tener en una topología todos conectados.

$$|E_{max}| = \frac{N(N-1)}{2} \tag{35}$$

Para fijar el número de aristas inicial en nuestra topología, haremos un intervalo de número de posibles conexiones $I = [E_{min}, E_{max}]$, dividiremos el intervalo I en 10 partes iguales, por lo que daremos saltos, entre cada intervalo de tamaño:

$$\Delta = \lfloor \frac{E_{max} - E_{min}}{10} \rfloor$$

Ahora tomaremos el valor medio, $index = 5$, como el número de aristas con el que iniciaremos la topología de nuestro algoritmo, es decir, el número de aristas inicial que tendrá nuestra topología será:

$$|E_{ini}| = \Delta * index + E_{min}$$

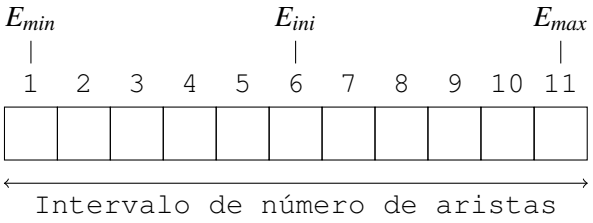


Figura. 5: Intervalos de número de aristas

Crearemos una gráfica G, con E_{ini} aristas. Esta gráfica partirá de una topología de anillo, esto sólo con el fin de tener una gráfica conectada a la que se añadirán el resto de las aristas de forma aleatoria entre los nodos de la gráfica.

Una vez que tenemos nuestra gráfica inicial, dejaremos el algoritmo iterar por 150 iteraciones, terminado esto, guardaremos el mejor valor obtenido hasta ese momento $Best_{fitness}$, el cual nos servirá posteriormente.

Adicional a ello, tomaremos dos índices del arreglo de número de aristas $index_{izq}$ e $index_{der}$, donde cada uno estará definido como $index_{izq} = index - 1$ e $index_{de} = index + 1$. Una vez definidos estos valores, crearemos dos nuevas gráficas con $|E_2|$ y $|E_3|$ aristas respectivamente

$$|E_2| = \Delta * index_{izq} + E_{min}$$

$$|E_3| = \Delta * index_{der} + E_{min}$$

Y crearemos dos nuevas gráficas G_2 y G_3 con E_2 y E_3 aristas, respectivamente. Como podemos notar la gráfica G_2 tiene Δ menos aristas que la gráfica G, estas se removerán de manera aleatoria, siempre y cuando no sean parte de las aristas del anillo; de igual manera la gráfica G_3 tiene Δ más aristas que la gráfica G, por lo cual se le añadirán estas aristas de forma aleatoria y éstas deben ser diferentes a las que actualmente tiene la gráfica G.

Copiaremos los valores del estado del algoritmo, como lo es el arreglo de partículas $P = [P_1, P_2, ..., P_N]$, los mejores valores encontrados por las partículas del enjambre: $Pbest = [Pbest_1, Pbest_2, ..., Pbest_N]$, donde N es el número de partículas en el enjambre. Definiremos las siguientes variables:

$$P_{G_2} = P,\ P_{G_3} = P,\ Pbest_{G_2} = Pbest,\ Pbest_{G_3} = Pbest$$

Y durante 50 generaciones, correremos el algoritmo con las tres gráficas, donde cada gráfica tendrá asociado su conjunto de partículas, su arreglo de las mejores posiciones encontradas y en cada una de estas las partículas tendrán asociado una vecindad diferente, es decir, la partícula i no será informada por las mismas partículas en la gráfica G, que en la gráfica G_2 o la gráfica G_3.

Debemos notar que durante las 50 iteraciones que ejecutamos el algoritmo con las 3 gráficas, el número de evaluaciones de función se incrementa en $3 * N$, donde N es el tamaño del enjambre.

Finalizadas estas iteraciones, verificaremos cuál fue el mejor valor encontrado usando las 3 gráficas diferentes, es decir, definiremos las siguientes variables:

$Best_{fitness_{G1}}$ mejor valor encontrado con el algoritmo usando la gráfica G

$Best_{fitness_{G2}}$ mejor valor encontrado con el algoritmo usando la gráfica G_2

$Best_{fitness_{G3}}$ mejor valor encontrado con el algoritmo usando la gráfica G_3

Crearemos una diferencia por cada una de estos valores, con $Best_{fitness}$ que almacenamos antes de correr el algoritmo con las tres gráficas.

Δ_G $Best_{fitness} - Best_{fitness_{G1}}$

Δ_{G_2} $Best_{fitness} - Best_{fitness_{G2}}$

Δ_{G_3} $Best_{fitness} - Best_{fitness_{G3}}$

Verificaremos con cual de estas se obtuvo un mejor valor comparado con el mejor valor obtenido antes de iniciar el algoritmo con 3 gráficas y tomaremos dicha gráfica como la nueva gráfica del algoritmo para repetir nuevamente el proceso, hasta que el criterio de paro se haya cumplido.

$$best = max[\Delta_G, \Delta_{G_2}, \Delta_{G_3}]$$

- Si $best = \Delta_G$, nos indica que el algoritmo obtuvo mejores resultados con la gráfica que estaba utilizando antes de iniciar la ejecución con las tres topologías, por lo que el número de aristas de la gráfica continuará siendo el mismo.

- Si $best = \Delta_{G_2}$, nos indica que el algoritmo obtuvo mejores resultados con una topología con menos aristas, E_2, por lo que en el siguiente paso, se ejecutará el algoritmo usando esta gráfica. Además debemos de copiar todo el estado actual, para continuar la mejora, por lo que, $Pbest = Pbest_{G_2}$, $P = P_{G_2}$, $index = index_{izq}$, $E = E_2$.

- Si $best = \Delta_{G_3}$, nos indica que el algoritmo obtuvo mejores resultados con una topología con más aristas, E_2, por lo que en el siguiente paso, correrá el algoritmo usando la gráfica

G_3 Además debemos de copiar todo el estado actual para continuar la mejora, por lo que, $Pbest = Pbest_{G_3}$, $P = P_{G_3}$, $index = index_{der}$, $E = E_3$.

Como se puede observar, no estamos forzando a cambiar el número de aristas de la gráfica con la que se ejecutará el algoritmo, damos oportunidad de continuar corriendo el algoritmo con la misma gráfica que tenía, esto porque no necesariamente incrementar o decrementar el número de aristas beneficia al algoritmo.

Para la parte de la actualización de la velocidad se utilizará la versión del factor de inercia como se muestra en la ecuación 36 y la posición de las partículas se llevará a cabo con la ecuación 37

$$\begin{aligned} v_{ij}^{t+1} = & \omega * v_{ij}^{t} + c_1 * r_{1j}^{t} * (Pbest_{ij}^{t} - x_{ij}^{t}) \\ & + c_2 * r_{2j}^{t} * (NBest_{ij}^{t} - x_{ij}^{t}) \end{aligned} \tag{36}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{37}$$

Donde:

v_{ij}^{t}	es el valor de la velocidad de la partícula i, en la dimensión j, en la iteración t
ω	es el factor de inercia
c_1 y c_2	son los coeficientes de aceleración
x_{ij}^{t}	es el valor de la posición de la partícula i, en la dimensión j, en la iteración t
$Pbest_{ij}^{t}$	es el mejor valor personal obtenido por la partícula i, en la dimensión j, hasta la iteración t
$Nbest_{ij}^{t}$	es el valor del mejor vecino de la vecindad de la partícula i, en la dimensión j, en la iteración t
r_1 y r_2	son dos aleatorios uniformemente distribuidos entre [0,1]

A continuación se muestra el pseudocódigo del algoritmo:

Algoritmo 7 Construcción de topología

Entradas: N número de partículas, E número de aristas

1: **for** i = 1, ..., N **do**
2: Generar arista (i, (i+1) mod N)
3: Añadir arista a G
4: **end for**
5: $E_{restantes} = E - N$
6: **while** $E_{restantes} > 0$ **do**
7: $nodo_1 = rand(0,N)$
8: $nodo_2 = rand(0,N)$
9: **if** $nodo_1 \; != \; nodo_2$ **then**
10: Generar arista ($nodo_1$, $nodo_2$)
11: **if** La nueva arista no está en la gráfica **then**
12: Añadir arista a G
13: $E_{restantes} \leftarrow E_{restantes} - 1$
14: **end if**
15: **end if**
16: **end while**
17: **return** Gráfica G

Algoritmo 8 Eliminar aristas de la gráfica G

Entradas: N número de partículas, E_{new} nuevo número de aristas, E número de aristas, G gráfica actual

1: **if** $E_{new} > E$ **then return** Con error: Número inválido de aristas
2: **else**
3: **while** $E_{new} > 0$ **do**
4: $edge_{random}$ = random(1, E)
5: **if** edges[$edges_{random}$] no está en el anillo **then**
6: Eliminar la arista edges[$edges_{random}$]
7: $E_{new} \leftarrow E_{new} - 1$
8: **end if**
9: **end while**
10: **end if**
11: **return** Gráfica G actualizada

Algoritmo 9 Añadir aristas a la gráfica G

Entradas: N número de partículas, E_{new} nuevo número de aristas,E número de aristas, G gráfica actual

1: **if** $E_{new} < E$ **then return** Con error: Número inválido de aristas
2: **else**
3: **while** $E_{new} > 0$ **do**
4: $node_1 = random(1,N)$
5: $node_2 = random(1,N)$
6: Crear arista ($node_1$, $node_2$)
7: **if** arista no está en G **then**
8: Agregar arista a G
9: **end if**
10: **end while**
11: **end if**
12: **return** Gráfica G actualizada

Algoritmo 10 Propuesta PSO con topología dinámica

Entradas: N número de partículas, d dimensión, valor de parámetros $c_1 = c_2 = 1.49618$, $\omega = 0.9268$. f función a optimizar, cotas de número de aristas E_{min} y E_{max}, Cotas de las variables X_{min} y X_{max}

1: $T \leftarrow 0$
2:
3: $Edges$ // arreglo de tamaño 10 con numero de aristas
4: $index = 5$ //índice para seleccionar número de aristas
5: $E_{ini} = Edges[index]$ //número inicial de aristas
6: Generar G con E_{ini} aristas con el algoritmo 7.
7: **for** i = 1,..., N **do**
8: $V_i = rand(0,1)$
9: $X_i = rand(X_{min}, X_{max})$
10: Calcular $f(X_i)$, valor de aptitud de la partícula X_i
11: **end for**
12: $contadoraux \leftarrow 0$
13: **while** NO se cumpla el criterio de paro **do**
14: **if** contadoraux >= 0 && contadoraux < 150 **then**
15: Actualizar la velocidad y posición de cada una de las partículas i, usando la ecuación 36 y 37, usando G para establecer las vecindades
16: **else if** contadoraux >= 150 && contadoraux < 200 **then**
17: **if** contadoraux == 150 **then**
18: $fitnessMin = f(x_i)$, el mejor valor encontrado con la topología actual hasta esa iteración
19: $E_2 = Edges[index - 1]$
20: Generar G_2 = eliminarAristas(E_2, E, G), con el algoritmo 8
21: $E_3 = Edges[index + 1]$
22: Generar G_3 = agregarAristas(E_3,E, G), con el algoritmo 9
23: **end if**

Actualizar la velocidad y posición de cada una de las partículas *i*, usando la ecuación 36 y 37, usando G, G_2 y G_3 para establecer las vecindades
else if contadoraux == 200 **then**
$fitnessMin_1$, mejor valor encontrado con la topología G
$fitnessMin_2$, mejor valor encontrado con la topología G_2
$fitnessMin_3$, mejor valor encontrado con la topología G_3
$diferencia_1 = fitnessMin - fitnessMin_1$
$diferencia_2 = fitnessMin - fitnessMin_2$
$diferencia_3 = fitnessMin - fitnessMin_3$
if $diferencia_2 > diferencia_1$ && $diferencia_1 > diferencia_3$ **then**
G = G_2
$index = index - 1$
$E = Edges[index]$
else if $diferencia_3 > diferencia_1$ && $diferencia_2 > diferencia_2$ **then**
G = G_3
$index = index + 1$
$E = Edges[index]$
else
Continuamos con la misma gráfica G
end if
end if
for i = 1,..., N **do**
$V_i = rand(0,1)$
$X_i = rand(X_{min}, X_{max})$
Calcular $f(X_i)$, valor de aptitud de la partícula X_i
Actualizar el $Pbest_i$ de cada partícula
end for
$T \leftarrow T + 1$
$contadoraux \leftarrow contador - aux + 1 \ mod \ 200$
end while
return Mejor valor encontrado

3.3 COMPORTAMIENTO DE LA TOPOLOGÍA PROPUESTA EN LAS FUNCIONES DE PRUEBA

En esta sección se hacen algunas observaciones importantes acerca del comportamiento de la topología al resolver cada una de las funciones de prueba de la Tabla 2.

Los experimentos se realizaron con una población de 30 partículas, en un espacio de búsqueda de dimensión 30. Bajo los siguientes parámetros $\omega = 0.7298$ y $c_1 = c_2 = 1.49618$.

Mediante la ecuación 34, se calcula el valor de E_{min}, obteniendo que $E_{min} = 30$. También se calcula el valor de E_{max} como se establece con la ecuación 35, por lo que $E_{max} = \frac{30(30-1)}{2} = 435$.

Se hace una relación entre el índice del arreglo con el número de aristas que representa en la topología para hacer el análisis de las gráficas de manera más fácil.

Índice/Etiqueta	Número de aristas
1	30
2	70
3	110
4	150
5	190
6	230
7	270
8	310
9	350
10	390
11	430

Tabla. 1: Relación de número de aristas

En las siguientes gráficas se puede observar el comportamiento de la topología propuesta frente a diferentes funciones. Las funciones de prueba están definidas en la Tabla 2.

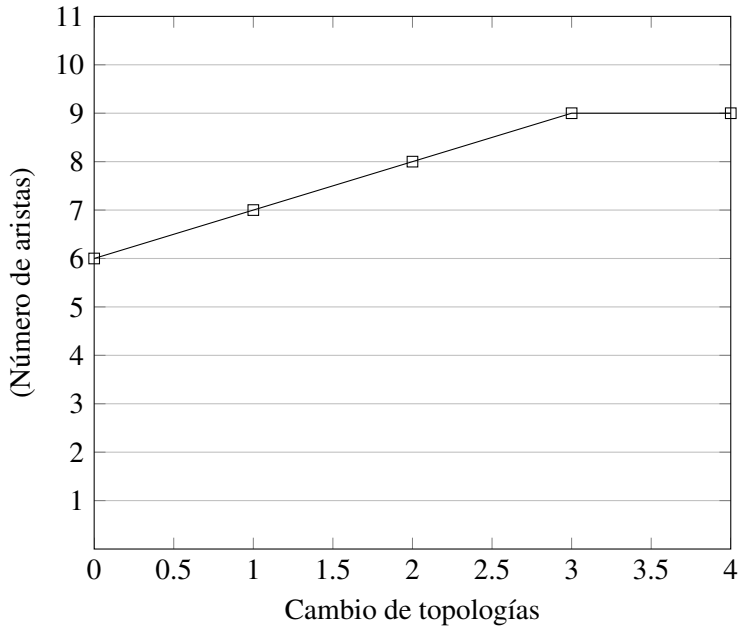


Figura. 6: Comportamiento de la topología en la función $esfera$

En la Figura 6 podemos observar que la topología tiende a incrementar su conectividad cada vez que se realiza un ajuste de topología. Lo cual nos indica que la función de esfera, obtiene mejores resultados frente a una topología completamente conectada. Este comportamiento se

puede verificar en la Tabla 3, en la que se reportan los resultados obtenidos en las pruebas, donde la topología con la que se obtienen mejores resultados es la topología de todos conectados.

En la Figura 7 podemos observar que en la primera parte de la ejecución del algoritmo la gráfica tiende a desconectarse, por lo que podemos decir, que se encuentra en una tapa de exploración, y finalmente en la última parte de la ejecución la gráfica tiende a incrementar la conectividad. Sin embargo a diferencia del comportamiento observado en la gráfica 6, la conectividad de la gráfica indica que para el PSO resuelva este problema requiere de un cierto nivel de exploración.

Se muestra en la Figura 10 el comportamiento de la topología en una ejecución de la propuesta al intentar resolver la función *Rastrigin*. Podemos observar que al inicio del algoritmo la topología intenta hacer ajustes hacia ambos lados, es decir, incrementar o decrementar el grado de conectividad, con el fin de obtener mejores resultados. Sin embargo, durante la ejecución la población se estanca en un punto, por lo tanto aún haciendo el ajuste de la topología con el fin de mejorar el valor obtenido es imposible, por lo que lo que resta de la ejecución permanece con la misma topología.

Las funciones esfera, cigar tablet y ellipsoid son convexas, por ello la conectividad de la topología se incrementa con el paso de las iteraciones del algoritmo, como se muestra en las Figuras 6, 14, 13.

En la Figura 9, se muestra gráficamente el ajuste de topología que se realizó en una ejecución no exitosa del algoritmo para resolver la función Rosenbrock. El comportamiento que se observa es que la topología oscila en el número de conexiones, con lo cual no se puede determinar si se encuentra en una etapa de explotación o exploración. Sin embargo, a diferencia de la gráfica 10, donde se nota claramente que la población se ha estancado en la ejecución del algoritmo para la función Rosenbrock esto no ocurre porque probablemente con un número mayor de evaluaciones de función se podría encontrar el valor óptimo de la función.

En la Figura 17 se muestra gráficamente el ajuste de topología de una ejecución exitosa de la función Schwefel 1.2. Se puede observar que durante la primera parte de la ejecución la topología tiende a decrementar la conectividad, lo que sugiere que se encuentra en una etapa de exploración y a lo largo de las iteraciones la conectividad aumenta. Se observa en la fase final de la ejecución que la topología termina con una conectividad alta, lo cual nos indica que se encuentra en una etapa de explotación.

Para complementar las observaciones se anexan las siguientes gráficas de ejecuciones exitosas, donde se encuentra el valor óptimo. La interpretación de las mismas donde se muestra el ajuste de las topologías se hace de forma análoga a cómo se hizo en los párrafos anteriores.

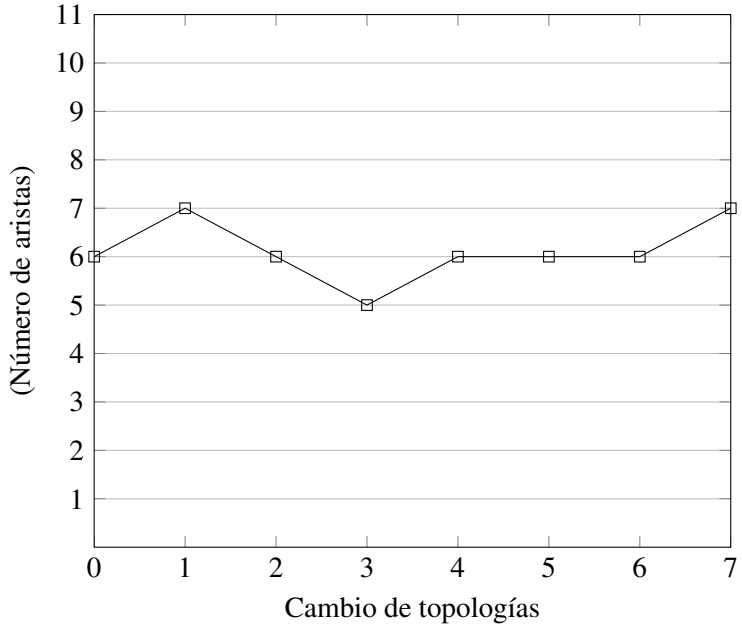


Figura. 7: Comportamiento de la topología en la función *Ackley*

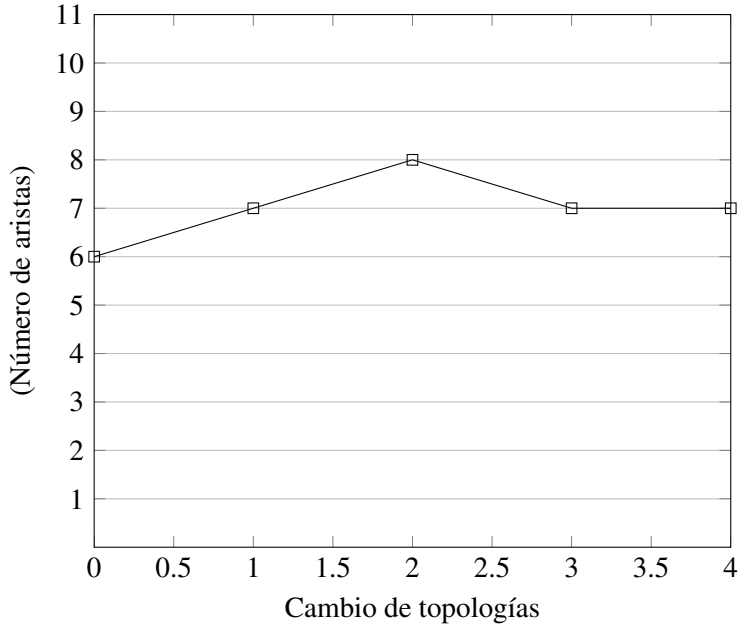


Figura. 8: Comportamiento de la topología en la función *Griewank*

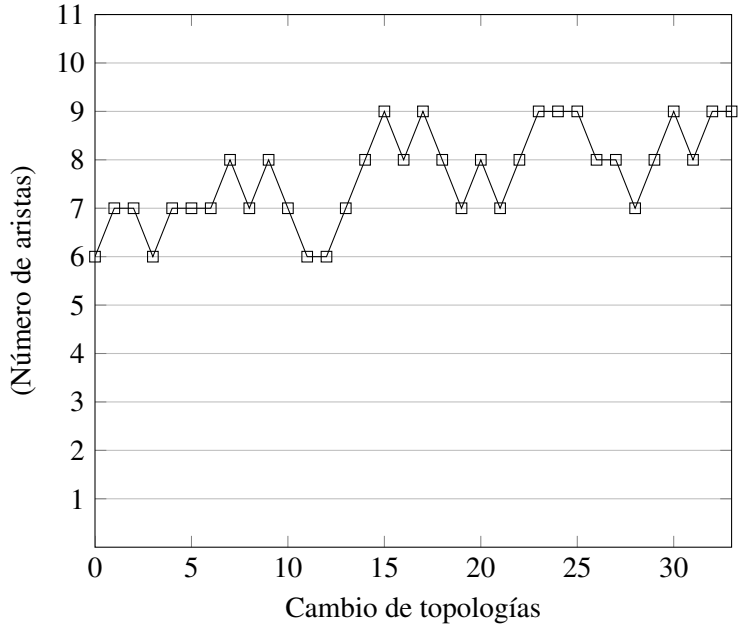


Figura. 9: Comportamiento de la topología en la función *Rosenbrock*

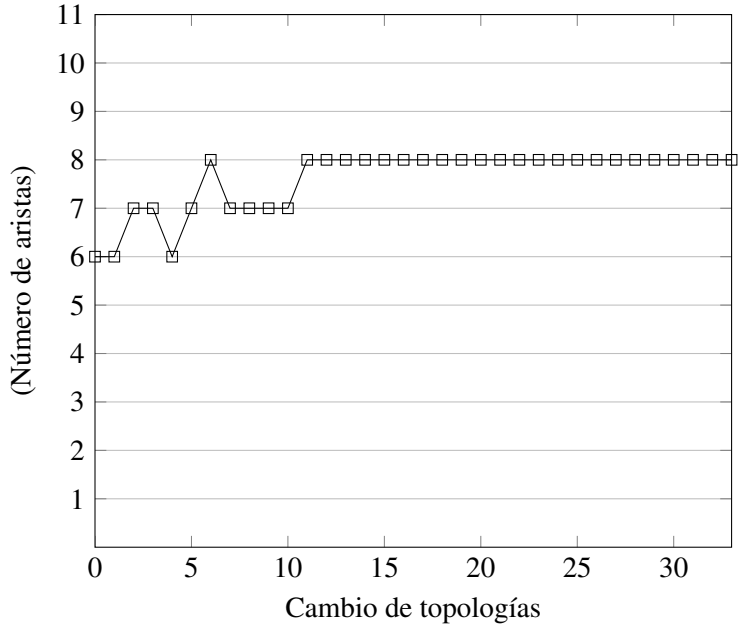


Figura. 10: Comportamiento de la topología en la función *Rastrigin*

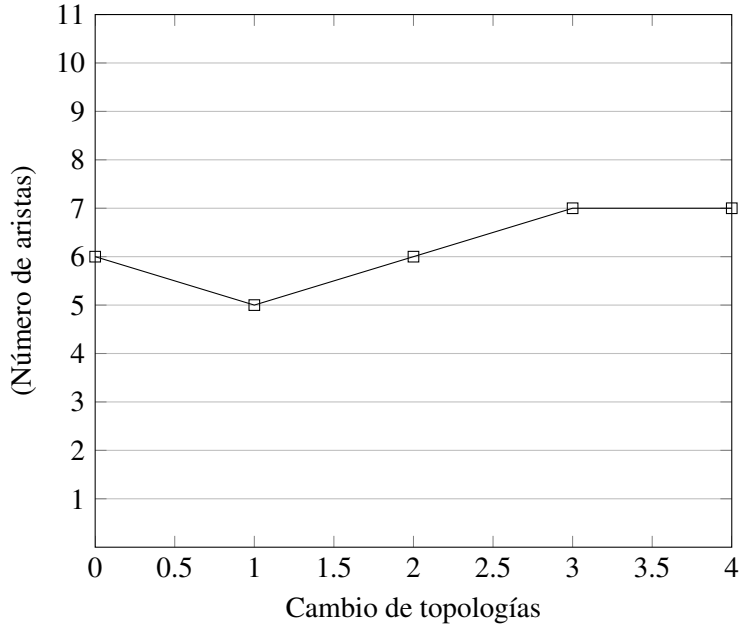


Figura. 11: Comportamiento de la topología en la función *Tablet*

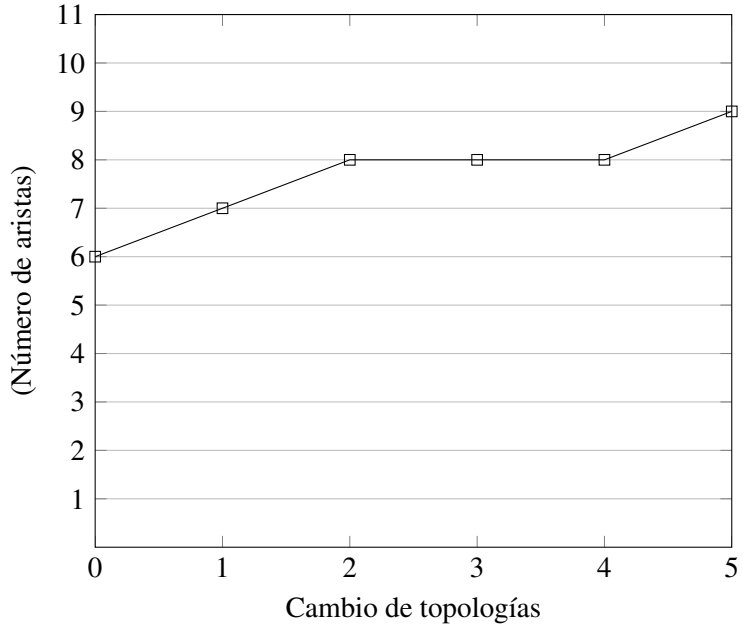


Figura. 12: Comportamiento de la topología en la función *Cigar*

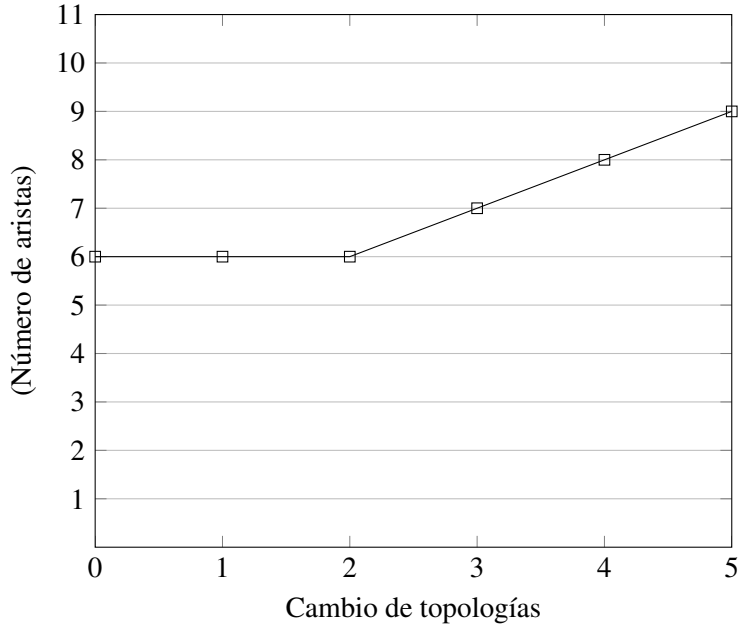


Figura. 13: Comportamiento de la topología en la función *Ellipsoid*

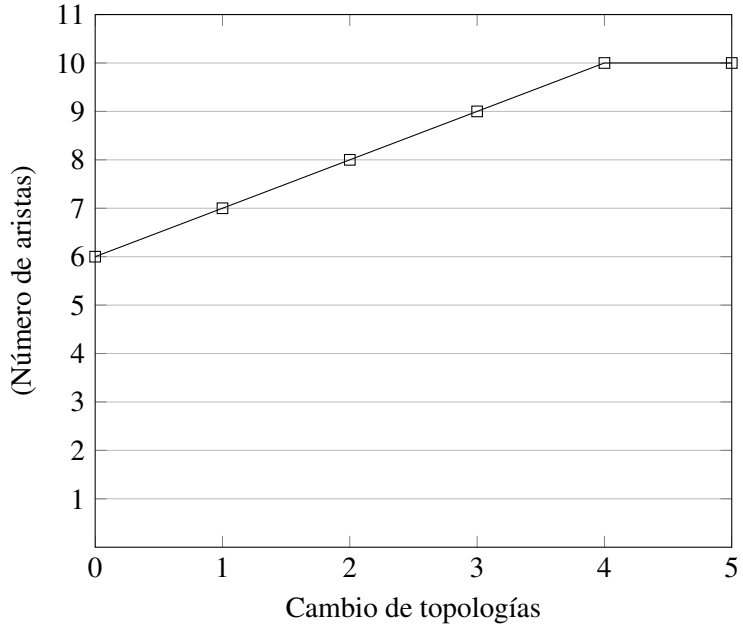


Figura. 14: Comportamiento de la topología en la función *Cigar Tablet*

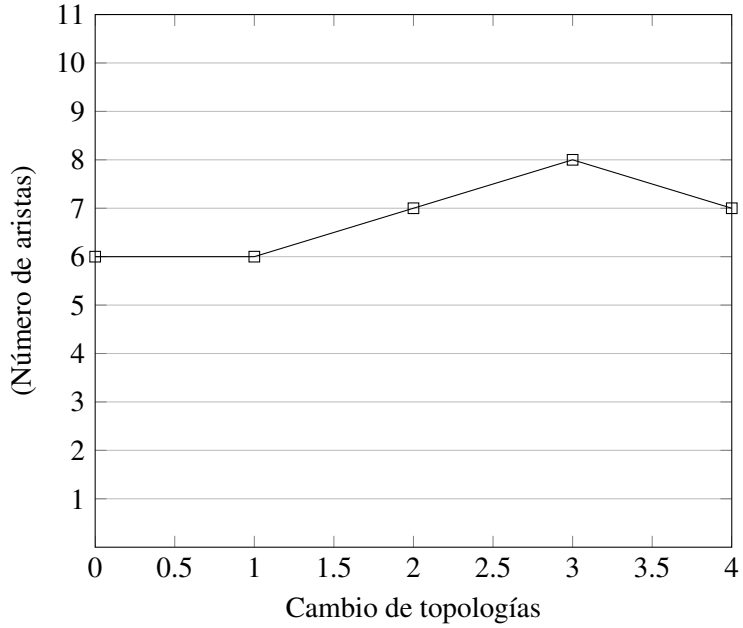


Figura. 15: Comportamiento de la topología en la función *Two Axes*

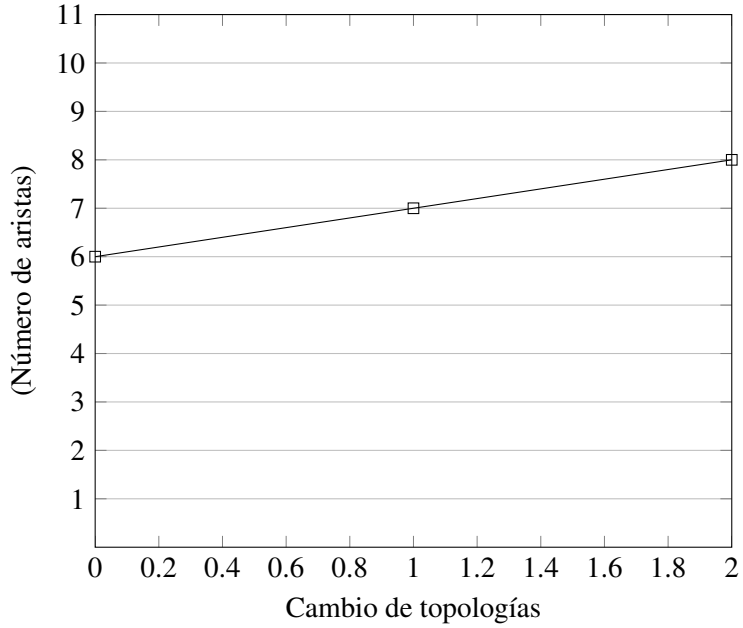


Figura. 16: Comportamiento de la topología en la función *Different Powers*

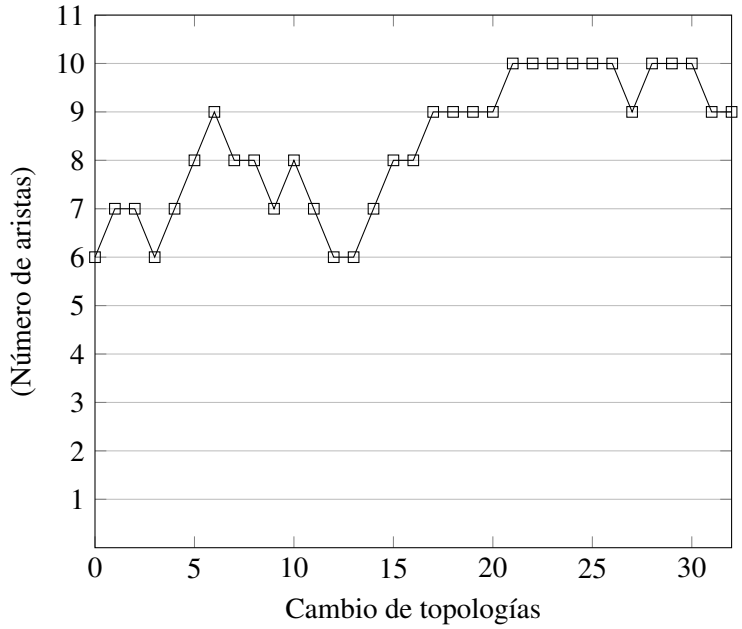


Figura. 17: Comportamiento de la topología en la función *Schwefel 1.2*

4

RESULTADOS

En este capítulo se muestran las funciones de optimización que se utilizarán para comparar diferentes versiones del PSO contra nuestra propuesta.

En la segunda parte del capítulo se muestran las tablas de resultados y finalmente se discuten éstos.

4.1 SOLUCIÓN DE UN BENCHMARK DE FUNCIONES

Sin pérdida de generalidad se resolverán un grupo de funciones comúnmente empleadas para resolver problemas de optimización, en particular problemas de minimización. Usaremos un conjunto de 12 funciones para verificar el comportamiento de nuestro algoritmo.

Función	Nombre	Fórmula	Dominio	Mínimo Global
f_1	Esfera	$\sum_{i=1}^{n} x_i^2$	$-100.0 \leq x_i \leq 100.0$	0
f_2	Ackley	$-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i^2)) + 20 + e$	$-32.0 \leq x_i \leq 32.0$	0
f_3	Griewank	$\frac{1}{4000}\sum_{i=1}^{n} x_1^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	$-600.0 \leq x_i \leq 600.0$	0
f_4	Rosenbrock	$\sum_{i=1}^{n} [100(x_{i+1} - x_i^2) + (x_i - 1)^2]$	$-30 \leq x_i \leq 30$	0
f_5	Rastrigin	$\sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	$-5.12 \leq x_i \leq 5.12$	0
f_6	Tablet	$10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$	$-10.0 \leq x_i \leq 5.0$	0
f_7	Cigar	$x_1^2 + \sum_{i=2}^{n} 10^6 x_i^2$	$-10.0 \leq x_i \leq 5.0$	0
f_8	Ellipsoid	$\sum_{i=1}^{n} 10^{6(\frac{i-1}{n-1})} x_i^2$	$-10.0 \leq x_i \leq 5.0$	0
f_9	Cigar tablet	$x_1^2 + \sum_{i=2}^{n-1} 10^4 x_i^2 + 10^8 x_n^2$	$-5.0 \leq x_i \leq 5.0$	0
f_{10}	Two Axes	$\sum_{i=1}^{\lfloor n/2 \rfloor} 10^6 x_i^2 + \sum_{\lfloor n/2 \rfloor}^{n} x_i^2$	$-5.0 \leq x_i \leq 5.0$	0
f_{11}	Different Powers	$\sum_{i=1}^{n} \mid x_i \mid^{2+10\frac{i-1}{n-i}}$	$-5.0 \leq x_i \leq 5.0$	0
f_{12}	Shwefel 1.2	$\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$-100.0 \leq x_i \leq 100.0$	0

Tabla. 2: Funciones de prueba

Definición [18] Una función es *unimodal* si tiene exactamente un sólo óptimo global y no tiene óptimos locales.

Definición [18] Una función es *multimodal*, si tiene múltiples óptimos locales y puede tener también múltiples óptimos globales.

Podemos clasificar estas funciones en 2 tipos: unimodales y multimodales, las que pertenecen al primer grupo son f_1, f_6, f_7, f_8, f_9, f_{10}, f_{11} y f_{12} ; del segundo grupo están f_2, f_3, f_4 y f_5.

La función f_4, es considera una función unimodal en dimensión 2 y 3, pero al aumentar la dimensión, es considerada una función multimodal.

Por otro lado, podemos clasificar estas funciones en dos tipos según su naturaleza: separables y no separables. Las funciones separables son la función f_1, f_{11}. Por otro lado las funciones no separables de nuestro conjunto son f_2, f_3, f_4, f_5, f_7, f_{12}.

4.2 RESULTADOS DE LAS CORRIDAS DE LOS ALGORITMOS

A continuación se presentan los resultados obtenidos con las diferentes versión del PSO, para las funciones de prueba. Los estadísticos reportados son: evaluaciones de función que le toma llegar al óptimo y el mejor valor encontrado.

Las pruebas de los algoritmos se llevaron a cabo bajo los siguientes parámetros: El tamaño de la población N, será de 30 partículas en dimensión 30.

- Gbest o PSO con topología todos conectados, $w = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{max} = 0.5(x_{max} - x_{min})$ y $v_{min} = -v_{max}$
- Lbest o PSO con topología de anillo, $\omega = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{max} = 0.5(x_{max} - x_{min})$ y $v_{min} = -v_{max}$
- von Neumann , $\omega = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{max} = 0.5(x_{max} - x_{min})$ y $v_{min} = -v_{max}$
- H. D-LPSO, $\omega = 0.578097$ y $c_1 = c_2 = 1.49445$
- PSO-ITC, $\omega_{inicial} = 0.9$, $\omega_{final} = 0.4$, $c_1 = c_2 = 2.0$, $z = 5$, $TC_{min} = 1$, $TC_{max} = N - 1$
- ASWPSO, $\omega = 0.7298$ y $c_1 = c_2 = 1.49618$
- Propuesta, $\omega = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{max} = 0.5(x_{max} - x_{min})$ y $v_{min} = -v_{max}$

El **criterio de paro**, en este caso fue alcanzar un error menor a $\varepsilon = 10^{-10}$ o bien alcanzar un número máximo de evaluaciones de función fijado en $300,000$.

4.3 COMPARACIÓN DE ALGORITMOS

A continuación se presentan las tablas de resultados donde se comparan los resultados obtenidos con los algoritmos antes mencionados.

Se reporta la media, mediana, el mejor y peor valor encontrado, tanto como para el mínimo encontrado como para el menor número de evaluaciones de función y desviación estándar, encontrada tras realizar 30 ejecuciones independientes de cada uno de los algoritmos.

En el caso de que el algoritmo haya parado tras cumplir las 300,000 evaluaciones de función, este valor es el que se reporta.

En la Tabla 3 se muestran los resultados para las funciones f_1 y f_2, en la Tabla 4 los resultados de las funciones f_3 y f_4, en la Tabla 5 los de las funciones f_5 y f_6, en la Tabla 6 de las funciones f_7 y f_8 , en la Tabla 7 de las funciones f_9 y f_{10} y en la Tabla 8, la funciones f_{11} y f_{12}.

Función	Algoritmo	Estadístico Reportado	Mejor	Peor	Media	Mediana	Desviación Estándar	Porcentaje de éxito
f_1	Gbest	Evaluaciones de función	22140	31860	25570	25290	2291.18	100%
		Mejor valor encontrado	7.52737e-11	9.9867e-11	9.4042e-11	9.7017e-11	6.3820e-12	
	Lbest	Evaluaciones de función	49800.0	56250.0	53220.0	53490.0	1461.69	100%
		Mejor valor encontrado	7.1550e-11	9.9835e-11	9.5168e-11	9.6873e-11	5.6443e-12	
	Von Neumann	Evaluaciones de función	37500.0	41670.0	39892.0	40140.0	1008.52	100%
		Mejor valor encontrado	8.4887e-11	9.9809e-11	9.4551e-11	9.5072e-11	3.7716e-12	
	H. D-LPSO	Evaluaciones de función	-	-	**31379.0**	-	1348.5	100%
		Mejor valor encontrado	9.2647e-11	6.7114e-12	9.4917e-11	7.8473e-11	9.9975e-11	
	PSO-ITC	Evaluaciones de función	-	-	77758	-	1747.3	100%
		Mejor valor encontrado	8.5021e-11	1.4196e-11	8.6391e-11	4.1146e-11	9.9893e-11	
	ASWPSO	Evaluaciones de función	72990.0	89340.0	79690.0	79020.0	4054.70	100%
		Mejor valor encontrado	7.5622e-11	9.9862e-11	9.2876e-11	9.7005e-11	7.4549e-12	
	Propuesta	Evaluaciones de función	35910.0	45870.0	40207.0	40050.0	2138.32	100%
		Mejor valor encontrado	7.5790e-11	9.94791e-11	9.1674e-11	9.5099e-11	6.7711e-12	
f_2	Gbest	Evaluaciones de función	38640.0	300000	274122.0	300000	77638.87	10%
		Mejor valor encontrado	9.6595e-11	7.9919	2.3125	2.1189	1.6767	
	Lbest	Evaluaciones de función	83880.0	93630.0	87179.0	86850.0	2177.37	100%
		Mejor valor encontrado	9.0815e-11	9.9906e-11	9.7194e-11	9.7860e-11	2.2207e-12	
	Von Neumann	Evaluaciones de función	61980.0	68220.0	64884.0	65130.0	1413.11	100%
		Mejor valor encontrado	8.9149e-11	9.9878e-11	9.7288e-11	9.8343e-11	2.7444e-12	
	H. D-LPSO	Evaluaciones de función	-	-	218020	-	111550	36.66%
		Mejor valor encontrado	9.3338e-11	2.6584	1.0847	1.2478	0.92294	
	PSO-ITC	Evaluaciones de función	-	-	114730	-	3485.6	100%
		Mejor valor encontrado	4.5934e-11	9.9907e-11	9.0042e-11	9.4801e-11	1.3013e-11	
	ASWPSO	Evaluaciones de función	127080.0	300000.0	227609.0	300000.0	82810.28	43.33%
		Mejor valor encontrado	9.1103e-11	3.4620	0.8901	1.1551	0.8963	
	Propuesta	Evaluaciones de función	58050.0	300000.0	**112297.0**	66660.0	93894.52	80%
		Mejor valor encontrado	7.7702e-11	2.1178	0.2823	9.8005e-11	0.5942	

Tabla. 3: Resultados de las funciones f_1 y f_2 en dimensión 30

Función	Algoritmo	Estadístico Reportado	Mejor	Peor	Media	Mediana	Desviación Estándar	Porcentaje de éxito
f_3	Gbest	Evaluaciones de función	23190.0	300000.0	198919.0	300000.0	132847.53	36.66%
		Mejor valor encontrado	8.4592e-11	0.2176	0.0322	0.0123	0.0467	
	Lbest	Evaluaciones de función	51990.0	300000.0	98447.0	58230.0	90268.63	83.33%
		Mejor valor encontrado	7.4535e-11	0.0148	0.0013	9.5442e-11	0.0035	
	Von Neumann	Evaluaciones de función	38310.0	300000.0	135697.0	41880.0	125019.88	63.33%
		Mejor valor encontrado	8.0075e-11	0.0516	0.0065	9.8551e-11	0.0116	
	H. D-LPSO	Evaluaciones de función	-	-	210580	-	120800	36.66%
		Mejor valor encontrado	9.1068e-11	0.061295	0.017014	0.0098573	0.019417	
	PSO-ITC	Evaluaciones de función	-	-	247530	-	96735	23.33%
		Mejor valor encontrado	6.4372e-11	0.12263	0.030324	0.023371	0.029065	
	ASWPSO	Evaluaciones de función	75300.0	300000.0	**176920.0**	91440.0	107687.66	56.66%
		Mejor valor encontrado	7.8478e-11	0.0707	0.0120	9.8878e-11	0.0175	
	Propuesta	Evaluaciones de función	38550.0	300000.0	214101.0	300000.0	121489.27	33.33%
		Mejor valor encontrado	5.8218e-11	0.0539	0.0118	0.0099	0.01254	
f_4	Gbest	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	0.0005	10.6273	2.3503	0.8828	2.9031	
	Lbest	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	1.8236e-05	73.3875	11.5304	10.0466	16.0488	
	Von Neumann	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	0.0047	17.1048	8.7449	11.7558	4.9918	
	H. D-LPSO	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	0.00261	93.119	19.248	10.203	24.969	
	PSO-ITC	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	0.041216	113.12	32.988	11.741	38.126	
	ASWPSO	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	0.0177	80.5393	31.2950	19.6086	29.4458	
	Propuesta	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	0.0056	74.2205	**10.6747**	9.7479	12.7305	

Tabla. 4: Resultados de las funciones f_3 y f_4 en dimensión 30

Función	Algoritmo	Estadístico Reportado	Mejor	Peor	Media	Mediana	Desviación Estándar	Porcentaje de éxito
f_5	Gbest	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	26.8639	82.5814	57.8733	58.7024	11.4487	
	Lbest	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	29.8487	69.6470	50.0132	49.7479	9.3650	
	Von Neumann	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	18.9042	71.6369	41.4897	41.7882	10.6050	
	H. D-LPSO	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	38.803	97.506	63.947	65.667	13.992	
	PSO-ITC	Evaluaciones de función	-	-	76835	-	1670.2	100%
		Mejor valor encontrado	3.5243e-11	9.9988e-11	**8.3899e-11**	9.0807e-11	1.8134e-11	
	ASWPSO	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	51.7378	156.2078	91.6686	89.5460	22.8720	
	Propuesta	Evaluaciones de función	300000	300000	300000	300000	0	0%
		Mejor valor encontrado	21.8891	65.6672	40.1631	39.7983	10.9067	
f_6	Gbest	Evaluaciones de función	19170.0	27060.0	22047.0	21420.0	2190.84	100%
		Mejor valor encontrado	7.4114e-11	9.9907e-11	9.4629e-11	9.7046e-11	5.8422e-12	
	Lbest	Evaluaciones de función	43140.0	47280.0	45115.0	45120.0	1255.99	100%
		Mejor valor encontrado	8.1106e-11	9.9298e-11	9.4820e-11	9.7140e-11	4.9977e-12	
	Von Neumann	Evaluaciones de función	32100.0	37080.0	34444.0	34590.0	1171.02	100%
		Mejor valor encontrado	7.0843e-11	9.9848e-11	9.3243e-11	9.4984e-11	6.8534e-12	
	H. D-LPSO	Evaluaciones de función	-	-	**28148**	-	1768.6	100%
		Mejor valor encontrado	7.1981e-11	9.9976e-11	9.3615e-11	9.5583e-11	6.4085e-12	
	PSO-ITC	Evaluaciones de función	-	-	67959	-	1752.8	100%
		Mejor valor encontrado	3.1195e-11	9.9998e-11	8.4362e-11	9.1084e-11	1.7357e-11	
	ASWPSO	Evaluaciones de función	65670.0	79530.0	71432.0	70980.0	3753.51	100%
		Mejor valor encontrado	8.2386e-11	9.9914e-11	9.3401e-11	9.4292e-11	5.3867e-12	
	Propuesta	Evaluaciones de función	32850.0	39030.0	36451.0	36960.0	1529.11	100%
		Mejor valor encontrado	7.5093e-11	9.9947e-11	9.2698e-11	9.5889e-11	6.1820e-12	

Tabla. 5: Resultados de las funciones f_5 y f_6 en dimensión 30

Función	Algoritmo	Estadístico Reportado	Mejor	Peor	Media	Mediana	Desviación Estándar	Porcentaje de éxito
f_7	Gbest	Evaluaciones de función	26940.0	36060.0	30426.0	30300.0	2253.23	100%
		Mejor valor encontrado	8.0743e-11	9.9704e-11	9.3451e-11	9.5090e-11	5.2167e-12	
	Lbest	Evaluaciones de función	61680.0	66690.0	64769.0	64680.0	1231.89	100%
		Mejor valor encontrado	7.3662e-11	9.9604e-11	9.3733e-11	9.5530e-11	6.1631e-12	
	Von Neumann	Evaluaciones de función	45630.0	51990.0	48935.0	49020.0	1338.26	100%
		Mejor valor encontrado	7.8020e-11	9.9907e-11	9.3822e-11	9.5537e-11	5.7703e-12	
	H. D-LPSO	Evaluaciones de función	-	-	**39065**	-	2657	100%
		Mejor valor encontrado	7.963e-11	9.9835e-11	9.3924e-11	9.4964e-11	5.1669e-12	
	PSO-ITC	Evaluaciones de función	-	-	97523	-	1700	100%
		Mejor valor encontrado	4.8148e-11	9.9982e-11	9.0159e-11	9.5048e-11	1.307e-11	
	ASWPSO	Evaluaciones de función	95520.0	106050.0	100568.0	101460.0	2625.741	100%
		Mejor valor encontrado	7.5498e-11	9.9673e-11	9.2925e-11	9.4516e-11	5.7741e-12	
	Propuesta	Evaluaciones de función	46950.0	56100.0	50296.0	49380.0	2464.84	100%
		Mejor valor encontrado	8.1690e-11	9.9424e-11	9.1312e-11	9.2973e-11	5.5184e-12	
f_8	Gbest	Evaluaciones de función	24270.0	34680.0	28728.0	28680.0	2977.58	100%
		Mejor valor encontrado	5.2926e-11	9.9973e-11	9.1429e-11	9.5376e-11	9.9165e-12	
	Lbest	Evaluaciones de función	51360.0	57840.0	54899.0	55140.0	1614.54	100%
		Mejor valor encontrado	8.2897e-11	9.9618e-11	9.4782e-11	9.6092e-11	4.3162e-12	
	Von Neumann	Evaluaciones de función	39960.0	43800.0	41685.0	41550.0	962.11	100%
		Mejor valor encontrado	7.5378e-11	9.9587e-11	9.2622e-11	9.5033e-11	6.0115e-12	
	H. D-LPSO	Evaluaciones de función	-	-	**33612**	-	1441.5	100%
		Mejor valor encontrado	8.1951e-11	9.9907e-11	9.5269e-11	9.5931e-11	3.9883e-12	
	PSO-ITC	Evaluaciones de función	-	-	86814	-	1689.4	100%
		Mejor valor encontrado	3.8121e-11	9.9987e-11	8.2596e-11	8.7699e-11	1.6368e-11	
	ASWPSO	Evaluaciones de función	81030.0	90510.0	86264.0	86580.0	2686.44	100%
		Mejor valor encontrado	7.1829e-11	9.9587e-11	9.1411e-11	9.2670e-11	6.6386e-12	
	Propuesta	Evaluaciones de función	36390.0	48450.0	41697.0	40500.0	3007.63	100%
		Mejor valor encontrado	6.9290e-11	9.9979e-11	9.1824e-11	9.3842e-11	7.3957e-12	

Tabla. 6: Resultados de las funciones f_7 y f_8 en dimensión 30

Función	Algoritmo	Estadístico Reportado	Mejor	Peor	Media	Mediana	Desviación Estándar	Porcentaje de éxito
f_9	Gbest	Evaluaciones de función	26100.0	36990.0	29654.0	29070.0	2443.29	100%
		Mejor valor encontrado	6.7694e-11	9.9431e-11	9.2909e-11	9.5723e-11	7.2258e-12	
	Lbest	Evaluaciones de función	54510.0	61530.0	58603.0	58830.0	1605.15	100%
		Mejor valor encontrado	7.5181e-11	9.9750e-11	9.3066e-11	9.5449e-11	6.1313e-12	
	Von Neumann	Evaluaciones de función	40980.0	46410.0	44392.0	44610.0	1406.85	100%
		Mejor valor encontrado	7.7655e-11	9.9621e-11	9.4227e-11	9.6287e-11	5.3365e-12	
	H. D-LPSO	Evaluaciones de función	-	-	**35748**	-	2016.7	100%
		Mejor valor encontrado	7.4862e-11	9.974e-11	9.2436e-11	9.5273e-11	7.4216e-12	
	PSO-ITC	Evaluaciones de función	-	-	88495	-	2112.2	100%
		Mejor valor encontrado	4.8214e-11	9.9679e-11	8.5654e-11	9.1839e-11	1.4938e-11	
	ASWPSO	Evaluaciones de función	84090.0	96930.0	90621.0	90450.0	3074.66	100%
		Mejor valor encontrado	6.7418e-11	9.9843e-11	9.4191e-11	9.6033e-11	6.2895e-12	
	Propuesta	Evaluaciones de función	40350.0	54720.0	46009.0	46050.0	2349.23	100%
		Mejor valor encontrado	5.9621e-11	9.8510e-11	9.0621e-11	9.4086e-11	9.2554e-12	
f_{10}	Gbest	Evaluaciones de función	24900.0	40230.0	29087.0	28500.0	2878.20	100%
		Mejor valor encontrado	6.9882e-11	9.9966e-11	9.5022e-11	9.7381e-11	6.0637e-12	
	Lbest	Evaluaciones de función	52260.0	56370.0	54037.0	54300.0	1003.17	100%
		Mejor valor encontrado	8.0577e-11	9.9844e-11	9.4312e-11	9.5855e-11	4.7317e-12	
	Von Neumann	Evaluaciones de función	39150.0	43710.0	41221.0	41040.0	1128.03	100%
		Mejor valor encontrado	7.8259e-11	9.9910e-11	9.4889e-11	9.6873e-11	4.9251e-12	
	H. D-LPSO	Evaluaciones de función	-	-	**34711**	-	2488.4	100%
		Mejor valor encontrado	6.8414e-11	9.989e-11	9.415e-11	9.6152e-11	6.3487e-12	
	PSO-ITC	Evaluaciones de función	-	-	90529	-	1761.7	100%
		Mejor valor encontrado	1.8166e-11	9.9819e-11	8.5876e-11	9.4814e-11	2.0341e-11	
	ASWPSO	Evaluaciones de función	78480.0	91080.0	84940.0	85320.0	2918.87	100%
		Mejor valor encontrado	6.5856e-11	9.9553e-11	9.2330e-11	9.4432e-11	7.8408e-12	
	Propuesta	Evaluaciones de función	37740.0	46710.0	42460.0	42390.0	2534.56	100%
		Mejor valor encontrado	7.2869e-11	9.9061e-11	9.0472e-11	9.2837e-11	7.4656e-12	

Tabla. 7: Resultados de las funciones f_9 y f_{10} en dimensión 30

Función	Algoritmo	Estadístico Reportado	Mejor	Peor	Media	Mediana	Desviación Estándar	Porcentaje de éxito
f_{11}	Gbest	Evaluaciones de función	9390.0	15090.0	12120.0	12270.0	1090.31	100%
		Mejor valor encontrado	6.1170e-11	9.9910e-11	8.6655e-11	8.9711e-11	1.1873e-11	
	Lbest	Evaluaciones de función	22800.0	29010.0	26264.0	26460.0	1428.32	100%
		Mejor valor encontrado	5.9790e-11	9.9963e-11	8.8210e-11	9.3340e-11	1.1093e-11	
	Von Neumann	Evaluaciones de función	16020.0	22980.0	19301.0	19470.0	1481.64	100%
		Mejor valor encontrado	4.0147e-11	9.9566e-11	8.1715e-11	8.1558e-11	1.4655e-11	
	H. D-LPSO	Evaluaciones de función	-	-	24408	-	2202.1	100%
		Mejor valor encontrado	5.3057e-11	9.9948e-11	9.0249e-11	9.3169e-11	1.1116e-11	
	PSO-ITC	Evaluaciones de función	-	-	58616	-	2131.8	100%
		Mejor valor encontrado	5.2275e-13	9.9866e-11	4.5515e-11	4.0037e-11	3.1544e-11	
	ASWPSO	Evaluaciones de función	31440.0	43950.0	37420.0	37890.0	3312.40	100%
		Mejor valor encontrado	5.7023e-11	9.8333e-11	8.7479e-11	9.1330e-11	1.0539e-11	
	Propuesta	Evaluaciones de función	17640.0	23490.0	**19387.0**	19500.0	1312.06	100%
		Mejor valor encontrado	5.8456e-11	9.9380e-11	8.5948e-11	9.1463e-11	1.1672e-11	
f_{12}	Gbest	Evaluaciones de función	149220.0	180840.0	165064.0	164880.0	7514.98	100%
		Mejor valor encontrado	8.2777e-11	9.9844e-11	9.7072e-11	9.8213e-11	3.5750e-12	
	Lbest	Evaluaciones de función	300000.0	300000.0	300000.0	300000.0	0.0	0%
		Mejor valor encontrado	5.3199e-06	0.0029	0.0004	0.0002	0.0005	
	Von Neumann	Evaluaciones de función	294840.0	300000.0	299828.0	300000.0	926.25	3.33%
		Mejor valor encontrado	9.5322e-11	1.3320e-08	3.2148e-09	2.2337e-09	3.6798e-09	
	H. D-LPSO	Evaluaciones de función	-	-	-	-	-	$-$%
		Mejor valor encontrado	-	-	-	-	-	
	PSO-ITC	Evaluaciones de función	-	-	-	-	-	$-$%
		Mejor valor encontrado	-	-	-	-	-	
	ASWPSO	Evaluaciones de función	300000.0	300000.0	300000.0	300000.0	0.0	100%
		Mejor valor encontrado	5.9489e-05	0.05799	0.0034	0.0011	0.0103	
	Propuesta	Evaluaciones de función	236370.0	300000.0	**275974.0**	276570.0	19862.27	83.33%
		Mejor valor encontrado	6.6096e-11	1.1088e-07	3.9845e-09	9.8826e-11	1.9872e-08	

Tabla. 8: Resultados de las funciones f_{11} y f_{12} en dimensión 30

4.4 DISCUSIÓN DE LOS RESULTADOS

En esta sección se compara gráficamente el número promedio de evaluaciones de función que se obtuvieron con el PSO para encontrar el valor óptimo de las funciones de prueba definidas en la Tabla 2 con las diferentes topologías: Gbest (Todos conectados), Lbest (Anillo) y la propuesta de esta tesis. Esta representación permite visualizar claramente que las ejecuciones del algoritmo PSO con las diferentes topologías, cumpla el criterio: $NEF_{min} \leq NEF \leq NEF_{max}$. Los valores fueron de la tablas de resultados 3, 4, 5, 6, 7 y 8.

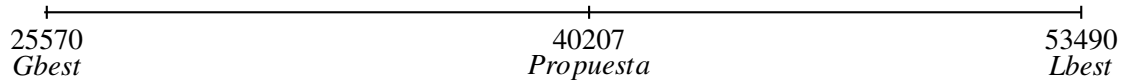


Figura. 18: Comparación del número promedio de evaluaciones de función $Esfera$

Figura. 19: Comparación del número promedio de evaluaciones de función *Ackley*

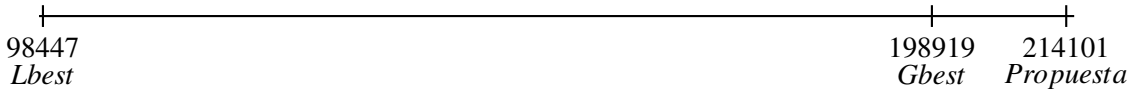


Figura. 20: Comparación del número promedio de evaluaciones de función *Griewank*

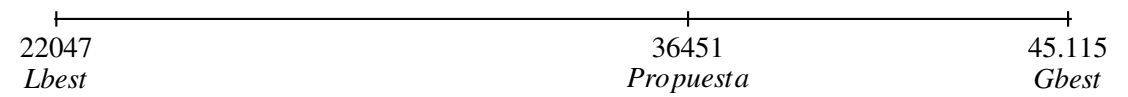


Figura. 21: Comparación del número promedio de evaluaciones de función *Tablet*

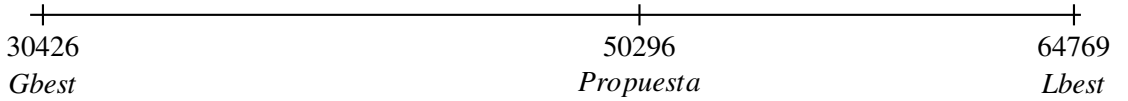


Figura. 22: Comparación del número promedio de evaluaciones de función *Cigar*

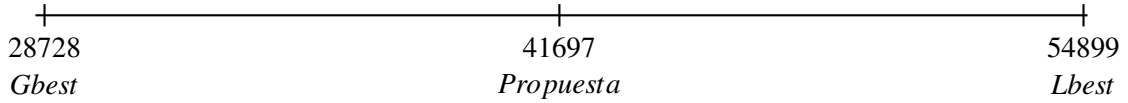


Figura. 23: Comparación del número promedio de evaluaciones de función $Ellipsoid$

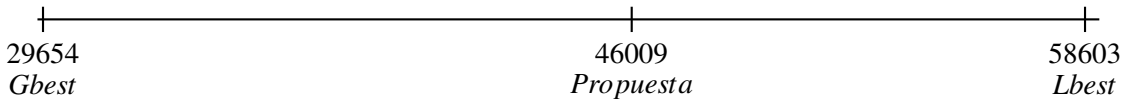


Figura. 24: Comparación del número promedio de evaluaciones de función *Cigar Tablet*

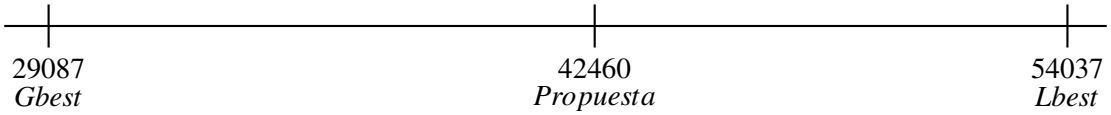


Figura. 25: Comparación del número promedio de evaluaciones de función *Two Axes*

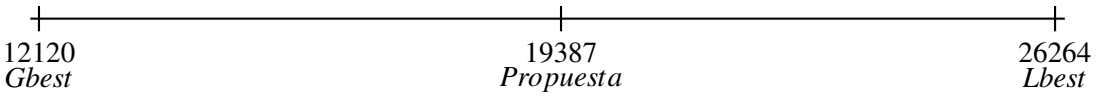


Figura. 26: Comparación del número promedio de evaluaciones de función *Different Powers*

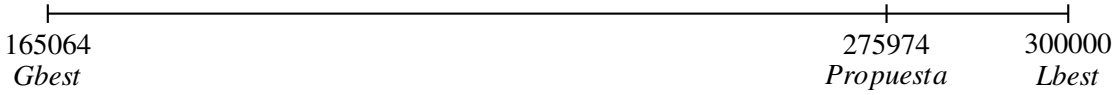


Figura. 27: Comparación del número promedio de evaluaciones de función *Schwefel* 1.2

Como se puede observar la propuesta es capaz de resolver el conjunto de problemas mostrado en la Tabla 2, salvo por la función Griewank que ejecutando el algoritmo PSO con la propuesta le toma un mayor número de evaluaciones de función, por lo que viola el criterio de $NEF_{min} \leq NEF \leq NEF_{max}$.

CONCLUSIONES

La propuesta planteada en este trabajo de tesis fue capaz de resolver un conjunto de problemas de optimización global generalmente usados para hacer pruebas de algoritmos. En la mayoría de los casos la propuesta obtuvo buenos resultados. Para ello agregamos algunos comentarios importantes de los resultados obtenidos, presentados en las Tablas 3 a la 8 del capítulo anterior. En las tablas mencionadas presentamos los resultados con las topologías Lbest (Anillo) y Gbest (Todos conectados) como referencia para tomar estos valores como cotas, como se explicó anteriormente. También se harán observaciones acerca del comportamiento de los 4 métodos con topología dinámica: H. D-LPSO, PSO-ITC, ASWPSO y la propuesta de la tesis.

Para la función de esfera se obtuvo con todos los algoritmos un porcentaje de éxito del 100% lo que nos indica que todas las versiones del algoritmo PSO fueron capaces de resolver la función. Entre las versiones Gbest (Todos conectados) y Lbest (Anillo), que hemos tomado como cotas para el número de evaluaciones de función, con la que se obtuvo el menor número de evaluaciones de función es con la topología Gbest (todos conectados) con 25570. Entre las cuatro versiones con topologías dinámicas que se presentan, con la que se obtuvo mejores resultados fue con H. D-LPSO con un valor de 31379, comparando con los resultados obtenidos con la propuesta que se presenta en este trabajo que obtuvo el óptimo en 40207 evaluaciones de función se observa que la diferencia es pequeña.

Para la función Ackley el porcentaje de éxito que se obtuvo empleando diferentes topologías en el algoritmo PSO es variado. Con la topología Gbest apenas se obtiene un porcentaje de éxito del 10%, por otro lado con la topología Lbest se obtiene un porcentaje de éxito del 100%, lo que nos indica que la función requiere de mayor exploración del espacio de búsqueda para encontrar el valor óptimo. Ya que con la topología Lbest se encuentra el valor óptimo, el número de evaluaciones de función es menor que el que le toma frente a la versión Gbest, con un total de 87179 evaluaciones de función. Comparando las versiones dinámicas, la que obtuvo mejores resultados es la propuesta presentada en este trabajo de tesis, con un total de 112297 evaluaciones de función y un porcentaje de éxito del 80%.

En el caso de la función Griewank, los resultados que se presentan en la Tabla 4 se puede observar que ningún algoritmo es decisivamente superior a otro, ya que con ninguno de ellos se obtuvo un porcentaje de éxito del 100%. Si quisiéramos hacer que las pruebas fueran más confiables, entonces se deberían llevar a cabo un mayor número de ejecuciones de los algoritmos y comparar el número de ejecuciones exitosas que se obtiene con cada algoritmo. Como se puede observar en la Gráfica 20, la propuesta presentada en esta tesis no es capaz de cumplir el criterio de: $NEF_{min} \leq NEF \leq NEF_{max}$, si se compara con la media del número de evaluaciones que les toma a los algoritmos Gbest, Lbest y a la propuesta. Sin embargo, si observamos la columna del mejor número de evaluaciones con los que se obtiene la solución, el número de evaluaciones de función con que lo logró la propuesta cumple el criterio, con un valor de 38550 evaluaciones de función, mientras que con el algoritmo Gbest le toma 23190 y el algoritmo Lbest un total de 51990 evaluaciones de función.

Analizando el comportamiento de la función Rosenbrock se obtiene que para cualquier versión del algoritmo se obtiene un porcentaje de éxito del 0%, lo que nos indica que frente a cualquiera de las versiones usando topologías estáticas o dinámicas, éstas no son capaces de encontrar un valor óptimo con un error de $\varepsilon = 10^{-10}$. Lo que se puede rescatar de los resultados obtenidos es observar con qué versión se logra disminuir más el valor de aptitud promedio obtenido con las diferentes versiones. Si comparamos entre la versión Gbest y Lbest, el valor medio obtenido se tiene un valor de 2.3503 y 11.5304, respectivamente. Por otro lado el valor medio obtenido con la propuesta de esta tesis es de: 10.6747, lo cual cumple con el criterio de obtener un valor intermedio a las versiones Gbest y Lbest y además comparando el valor obtenido con la propuesta, frente a las versiones dinámicas que se presentan, resulta ser el que obtiene un resultado menor en el valor del promedio.

Con la función Rastrigin se puede observar que sólo con el algoritmo PSO-ITC se obtiene un porcentaje de éxito del 100%, y cualquiera del resto de los algoritmo obtiene un porcentaje de éxito del 0%. Lo que se puede observar mediante la representación gráfica del ajuste de topología es que para esta función se debe tener alguna medida para la detección y control del estancamiento, de lo contrario la población se queda estancada y al no ser capaz de salir de ese estado, no se logra encontrar el valor óptimo. Si comparamos al igual que se hizo con la función Rosenbrock el mejor valor obtenido de los algoritmos que obtuvieron un porcentaje de éxito del 0%, podemos observar que el mejor valor promedio obtenido fue con la propuesta de esta tesis, obteniendo un valor de: 40.1631.

La función Tablet presenta un porcentaje de éxito del 100% con cualquiera de los algoritmos. La media de número de evaluaciones de función obtenidos con la versión Gbest y Lbest son: 22047 y 45115, comparando con la media obtenida con la propuesta de esta tesis se tiene un valor de 36451, y se cumple el criterio: $NEF_{min} \leq NEF \leq NEF_{max}$. Y la versión dinámica con la que se obtiene el menor número de evaluaciones de función es H. D-LPSO con un total de 28148.

La función Cigar obtiene con todas los algoritmos un porcentaje de éxito del 100%, lo que nos indica que todos encontraron el valor óptimo. Comparando la media del número de evaluaciones de función que se obtuvieron con las versiones Gbest y Lbest, podemos observar que se obtuvo el óptimo en un menor número de evaluaciones con la topología Gbest con respecto a la topología Lbest. Por otro lado haciendo la comparación entre las versiones con topología dinámica, la versión que obtuvo el óptimo en el menor número de evaluaciones fue: H. DL-PSO con 39065, a diferencia de la propuesta de esta tesis que lo hace en 50296 evaluaciones de función.

Los resultados para las funciones Ellipsoid y Cigar Tablet, muestran que con cualquiera de los algoritmos se obtiene un porcentaje de éxito de 100%. Al igual que con la función Cigar, la topología Gbest obtiene el óptimo en el menor número de evaluaciones de función y la función dinámica con la que se obtiene la menor media de evaluaciones de función es con la versión H. DL-PSO. Sin embargo el mejor valor encontrado por la propuesta de esta tesis es menor al encontrado por la versión H. DL-PSO.

La función Two Axes, con cualquiera de los algoritmos que se implementaron obtiene un porcentaje de éxito del 100%. Entre los algoritmos Gbest y Lbest, con el que se obtiene un menor número de evaluaciones de función es con el algoritmo Gbest, es decir, que como se vio anteriormente la función por su naturaleza requiere de explotación más que exploración para encontrar el

valor óptimo. De los algoritmos dinámicos se obtuvo un mejor número de evaluaciones de función en la media es con la versión H. D-LPSO.

Por otro lado, la función Different Powers es una función que frente a todos los algoritmos obtiene un porcentaje de éxito del 100%. Comparando las topologías Gbest y Lbest, la topología con la que se obtienen el óptimo en un menor número de evaluaciones de función es con la versión de Gbest. Si comparamos las versiones con topología dinámica podemos observar que con la que obtiene el óptimo en un número menor de evaluaciones de función es con la topología propuesta, encontrando el óptimo en una media de 19387 evaluaciones de función.

Finalmente, la función Schwefel 1.2 frente a los algoritmos Gbest y Lbest obtiene un porcentaje de éxito del 100% y 0%, respectivamente. Lo que nos da una idea de que la función requiere de una topología con un mayor número de conexiones para encontrar el valor óptimo y si la topología no tiene suficientes conexiones, encontrar la solución le tomará un mayor número de iteraciones por lo tanto un número mayor de evaluaciones de función. Los resultados obtenidos con las topología dinámicas indican que la topología propuesta obtiene mejores resultados, sin embargo, sólo lo logra con un porcentaje de éxito de 83.33%. Esto puede deberse a que inicialmente la topología no cuenta con un gran número de conexiones y al tiempo que le toma realizar el ajuste para incrementar la conectividad.

Debemos hacer también unas observaciones comparando la propuesta con el método von Neumann, que es una de las topologías estáticas con la que se han obtenido resultados en un número de evaluaciones acotado también por las topologías: Gbest y Lbest. Comparándola con la topología propuesta en esta tesis, se observa que los resultados de las Tablas antes mencionadas no difieren mucho, es decir, podemos ver que nuestra propuesta no desperdicia evaluaciones de función durante la ejecución. Debemos observar que la topología von Neumann resuelve la mayoría de los problemas, sin embargo por ser una topología estática tiene limitaciones en cuanto al conjunto de funciones que será capaz de resolver. Como se observa en la Tabla 8, con la funcion Schwefel 1.2 se obtiene solamente un porcentaje de éxito de 3.33% usando la topología von Neumann, a comparación de la propuesta de esta tesis que obtiene un porcentaje de éxito del 83.33%. Con lo que podemos observar que al tener una topología dinámica se hace más robusto el algoritmo y con ello le da oportunidad de resolver un conjunto más grande de funciones.

Algunos de los aspectos que se podrían añadir para mejorar los resultados obtenidos con la propuesta son: añadirle alguna medida para la detección y control del estancamiento de la población con el fin de intentar resolver problemas en los cuáles este problema se presenta frecuentemente, añadir algún criterio para evitar que el grado de una topología sea elegido un máximo número de veces, con el fin de evitar que el algoritmo se quede con la misma topología durante un número grande de evaluaciones de función. También se podrían añadir otras medidas además de un cierto número de evaluaciones de función para realizar el ajuste de topología, como lo es la posición de la mejor partícula, un reinicio de la posición de las partículas en caso de que se presente un estado de estancamiento en la población.

Si bien es claro, aún quedan muchos aspectos del algoritmo por probar con el fin de obtener mejores resultados, ya que dependen del problema y quizás el intentar mejorar el algoritmo nos lleve a sobre ajustar la propuesta a un solo problema.

BIBLIOGRAFÍA

[1] James Kennedy, Russell Eberhart. *Particle Swarm Optimization* Proceedings of the IEEE International Joint Conference on Neural Networks, 1995.

[2] Y.Shi,R.Eberhart. *A modified particle swarm optimizer* Evolutionary Computation Proceedings, 1998.

[3] R.C. Eberhart and Y. Shi. *Tracking and optimizing dynamic systems with particle swarms* In Evolutionary Computation, 2001.

[4] J. Xin, G. Chen, Y. Hai. *A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight* Computational Sciences and Optimization, 2009.

[5] J. C. Bansal, P. K. Singh, Mukesh Saraswat, Abhishek Verma, Shimpi Singh Jadon, Ajith Abraham. *Inertia Weight Strategies in Particle Swarm Optimization* Third World Congress on Nature and Biologically Inspired Computing, 2011.

[6] Frans Van Den Bergh, Andries Petrus Engelbrecht. *A study of particle swarm optimization trajectories* Information Sciences, 2006.

[7] James Kennedy, Rui Mendes. *Population Structure and Particle Swarm Performance* Proceedings of the IEEE Congress on Evolutionary Computation, 2002

[8] Zahra Beheshti, Siti Shamsuddin, Sarian Sulaiman. *Fusion Global-Local-Topology Particle Swarm Optimization for Global Optimization Problems* Mathematical Problems in Engineering vol. 2014, 2014

[9] Xueming Yang, Jinsha Yuan, Jiangye Yuan, Huina Mao. *A modified particle swarm optimizer with dynamic adaptation* Applied Mathematics and Computation 189, 2007.

[10] Pradipta Ghosh, Haim Zafar, Swagatam Das, Ajith Abraham. *Hierarchical Dynamic Neighborhood Based Particle Swarm Optimization for Global Optimization* Proc. IEEE Congr. Evol. Comput. 2011.

[11] Y.-j Gong y J. Zhang. *Small-World Particle Swarm Optimization with Topology Adaptation* Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, 2013.

[12] Wei Hong Lim, Nor Ashidi Mat Isa. *Particle Swarm Optimization with increasing topology connectivity* Engineering applications of artificial intelligence 27, 2014.

[13] M, Clerc, J. Kennedy. *The Particle Swarm: Explosion, Stability and Convergence in a Multidimensional Complex Space* IEEE transactions on Evolutionary Computation, 2002.

[14] Albert-László Barabási, Réka Albert. *Emergence of Scaling in Random Networks* Science 286.5439 (1999).

[15] A. Ratnaweera, K. Halgamuge, c. Watson *Self-Organizing Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients* IEEE, 2004

[16] Konstantnos E. Parsopoulos, Michael N. Vrahatis *Particle Swarm Optimization and Intelligence: Advances and Applications*

[17] Justin Solomon *Numerical Algorithms*

[18] Thomas Weise *Global Optimization Algorithms, Theory and Application* 2008

[19] J. A. Bondy, U. S. R. Murty *Graph Theory with Applications*

[20] David Avis, Alain Hertz, Odile Marcotte. *Graph Theory and Combinatorial Optimization* Springer, 2005.

[21] Ángel A. Rojas *Topologías de Particle Swarm Optimization basadas en información mutua* CIMAT, 2014.