Informe Final del Trabajo de Grado en la Modalidad Investigación

"COMPARACIÓN DEL DESEMPEÑO DEL ALGORITMO DE OPTIMIZACIÓN PSOSX (PE) FRENTE AL PSOSX (S)"

Presentado ante:

COMITÉ DE TRABAJOS DE GRADO E^{3}T

Por:

JAIME ANDRÉS OSMA RUIZ
Estudiante de Ingeniería Electrónica
Código UIS: 2011268

MÓNICA JULIETH VILLARREAL ARDILA
Estudiante de Ingeniería Electrónica
Código UIS: 2031759

Bucaramanga, 2009

COMPARACIÓN DEL DESEMPEÑO DEL ALGORITMO DE OPTIMIZACIÓN PSOSX (PE) FRENTE AL PSOSX (S)

JAIME ANDRÉS OSMA RUIZ

MÓNICA JULIETH VILLARREAL ARDILA

Trabajo de Grado presentado como Requisito
Parcial para optar por el Título de Ingeniero Electrónico

Director:
PH.D CARLOS RODRIGO CORREA CELY

Codirector:
PH.D OSCAR JAVIER BEGAMBRE CARRILLO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2009

DEDICATORIA

A mis padres por sus constantes y acertados consejos, su incondicional apoyo y proyección, por estar ahí cuando los necesité y por animarme cuando sentí desfallecer, por hacer de mi lo que soy ahora.

A mi novia, por apoyarme en mis decisiones, por confiar en mi juicio y estar a mi lado en los buenos y malos momentos, por darme fuerza y serenidad en aquellos momentos tormentosos.

A Dios por acompañarme en todo momento a lo largo de mi vida e iluminar mi camino siempre, por escucharme y guiarme todos los días de este largo y severo trayecto de mi vida, por darme salud y llenarme de entendimiento.

A mis amigos que siempre fueron mi soporte y apoyo, quienes me brindaron animo y concejo, los que me ofrecieron más que una amistad, a los que valoraron mi trabajo y se regocijaron junto a mí en mis triunfos y me levantaron en mis derrotas.

Le dedico este trabajo a todas las personas que hicieron posible este nuevo triunfo y me ayudaron a cumplir esta meta.

Jaime

A Dios, por su protección, compañía, por iluminarme el sendero a seguir y de esta forma permitirme cumplir tan anhelado sueño.

A mis padres, por su ayuda incondicional, sus consejos, por guiarme al transcurrir de mis años, por educarme desde niña y enseñarme lo linda que es la vida a su lado.

A mis hermanos, por su amor, comprensión, confianza y cariño.

A mis amigos, pues son quienes me brindan alegría, animo y sentimiento.

A todas las personas que de una u otra forma me dieron su apoyo el cual necesité para llegar a ser quien actualmente soy.

Mónica

AGRADECIMIENTOS

Agradecemos principalmente a Dios, por regalarnos la sabiduría y la fortaleza para continuar y obtener todas las metas propuestas.

A nuestros padres por brindarnos la posibilidad de formarnos profesionalmente.

A la Universidad Industrial de Santander, por ofrecernos los medios necesarios para poder conseguir uno de los mayores logros en nuestras vidas.

A nuestro director del trabajo de grado, el profesor Carlos Rodrigo Correa, por sus conocimientos, sus enseñanzas, su paciencia y guía tanto para la vida profesional como en la personal.

Al Codirector, el profesor Oscar Javier Begambre, por sus aportes y por su valiosa colaboración en la construcción de este trabajo.

A todos nuestros compañeros y amigos que nos han apoyado y ayudado a vencer todas las piedras encontradas en el camino.

TABLA DE CONTENIDO

LISTADO DE FIGURAS

LISTA DE TABLAS

RESUMEN EJECUTIVO D EL TRABAJO DE GRADO

Titulo:	Comparación del desempeño del algoritmo de optimización PSOSX (PE) frente al PSOSX (S).
Director:	Ph.D Carlos Rodrigo Correa Cely, crcorrea@uis.edu.co
Codirector:	Ph.D Oscar Javier Begambre Carrillo, ojbegam@uis.edu.co
Autores:	Mónica Julieth Villarreal Ardila, monike07@hotmail.com Jaime Andrés Osma Ruiz, jaime.osma@gmail.com
Modalidad:	Investigación
Costo Total:	$30.000.000 [M/C].
Plazo:	16 semanas
Posibles Entidades Interesadas en los Resultados:	UIS, Grupo de investigación en control, electrónica, modelado y simulación – CEMOS.

Breve reseña del proyecto

Particle Swarm Optimization (PSO) es un proceso de optimización que pertenece al grupo de las metaheurísticas[1]. El PSO es utilizado para encontrar la solución óptima y su objetivo es recorrer toda la zona de soluciones posibles, sin

[1] El término metaheurístico deriva de la palabra griega meta (cuyo significado es mas allá) y de la palabra griega heuriskein (que significa descubrir, encontrar). Son métodos heurísticos de más alto nivel.

quedar atrapado en ninguna de ellas.

Dos de las arquitecturas del PSO son, el PSOSX (PE) Particle *swarm optimization simplex parametric evolution* y el PSOSX (S) *Particle swarm optimization simplex sequential*, que se estudiaron en este trabajo.

Objetivo General del Proyecto:

Desarrollar una herramienta comparativa en Matlab, que muestre las ventajas y desventajas de usar el algoritmo de optimización PSOSX (PE) o el PSOSX (S).

Objetivo Específico Propuesto	Resultado Obtenido
Desarrollar los híbridos PSOSX (PE) y el PSOSX (S).	Se desarrollo el algoritmo PSO y luego se implemento el Simplex, para obtener como resultado las arquitecturas del hibrido PSOSX (PE) y del PSOSX (S) individualmente.
Realizar un programa utilizando Matlab implementando los híbridos PSOSX (PE) y el PSOSX (S).	Por medio de Matlab se ha desarrollado una herramienta útil y practica implementando los híbridos PSOSX (PE) y el PSOSX (S) (Ver Capitulo 3 del presente documento).
Validar la eficiencia de los híbridos generados para cinco funciones utilizadas en la literatura especializada en la evaluación de algoritmos de optimización.	El análisis realizado en el trabajo muestra características pertenecientes a los híbridos aplicados a las cinco funciones definidas previamente (Ver Capitulo 4 del presente documento).

Actividades de Difusión Programadas:

Futura publicación de un artículo en la Revista UIS Ingenierías donde se resalten los resultados y conclusiones obtenidas en este trabajo de grado.

Impactos del Proyecto:

La información del trabajo de grado fue analizada y presentada de una manera didáctica, esto con el objetivo de incentivar la investigación del PSO y la implementación de éste en diversas áreas.

RESUMEN

TÍTULO

COMPARACIÓN DEL DESEMPEÑO DEL ALGORITMO DE OPTIMIZACIÓN PSOSX (PE) FRENTE AL PSOSX (S)♣.

AUTORES

Jaime Andrés Osma Ruiz

Mónica Julieth Villarreal Ardila♣♣

PALABRAS CLAVES

Metaheurísticas, Particle Swarm Optimization (**PSO**), Simplex (**Nelder Mead**), Particle swarm optimization simplex parametric evolution **(PSOSX (PE))** y Particle swarm optimization simplex sequential **(PSOSX (S))**.

CONTENIDO

La gran mayoría de los problemas en el mundo real, tienen más de un objetivo a realizar, gracias a esto se ha observado la necesidad de crear algoritmos matemáticos y computacionales con el fin de resolver dichos problemas. Aunque al momento de buscar una solución a un problema con más de un objetivo a realizar, existen algunos inconvenientes, por la complejidad de los mismos. El objetivo de los algoritmos de optimización, es solucionar dichos problemas, buscando reducir el tiempo y aumentando la exactitud, en los cálculos de las respuestas de modelos matemáticos complejos.

En el desarrollo del presente trabajo de grado, se estudiaron y analizaron las arquitecturas PSOX(PE) y PSOX(S); basados en el PSO stándard y el método de Nelder Mead. El PSO y el Nelder Mead son algoritmos eficientes por separado, aunque con ciertos inconvenientes como velocidad y convergencia. En la elaboración de este trabajo de grado, se busco integrar ambos métodos por medio de 2 arquitecturas, esto con el fin de obtener los mejores resultados de cada uno y poder conseguir la solución óptima.

Para validar las arquitecturas realizadas, se verificaron cinco funciones utilizadas en la literatura especializada, en la evaluación de algoritmos de optimización; calculando sus respectivos tiempos de cómputo, exactitud, desviación estándar, media y su convergencia.

♣ Trabajo de Investigación.

♣♣ Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones. Director: Ph. D Rodrigo Correa. Codirector: Ph. D Oscar Begambre.

SUMMARY

TITLE

PERFORMANCE COMPARISON OF OPTIMIZATION ALGORITHM PSOSX (EP) FRONT PSOSX (S)[*]

AUTHORS

Mónica Julieth Villarreal Ardila[**]

Jaime Andrés Osma Ruiz[**]

KEYWORDS

Metaheuristics, Particle Swarm Optimization (**PSO**), Simplex (**Nelder Mead**), Particle swarm optimization simplex parametric evolution **(PSOSX (PE))** y Particle swarm optimization simplex sequential **(PSOSX (S))**.

CONTENT

Most of the problems in the real world has more than one objective to be perform, thanks to this the scientific community have seems the need to create mathematical and computational algorithms in order to solve those problems. Although at the time of solving a problem, there are some drawbacks, because of the complexity of their own, the objective of computational algorithms is to solve them, for this reason borns the PSO, seeking to reduce the time and increasing the accuracy in calculations on the answer of the complex mathematical models.

In the development of this grade job were studied and analyzed the architectures PSOX(PE) and PSOX(S), based on the standard PSO and the Nelder Mead method, which they are very efficient algorithms separately, although with some inconvenients like velocity finding answers and convergence in optimal points. In the developing of this grade job, seeks to integrate both methods by through two architectures, this in order to obtain the best results of each one and obtaining the optimal final solution.

Conducted to validate the architecture were checked with five functions used in the specialized literatura, in the evaluation of optimization algorithms, calculating its respective computation times, accuracy, standard deviation, half and convergence.

---

[*] Project Grade.

[**] Fisics-mecanics Engineering Faculty. School of Electrical Engineering, Electronics and Telecommunications. Director: Ph. D Rodrigo Correa. Codirector: Ph. D Oscar Begambre.

INTRODUCCIÓN

En la actualidad, el problema de buscar nuevas y mejores soluciones a sistemas complejos, es un tema presente a diario en muchas empresas. Las técnicas heurísticas, se han convertido en una herramienta imprescindible para solucionar modelos complejos. El objetivo de las heurísticas es detectar los óptimos locales, de tal manera que al momento de buscar la solución, éstas no se queden atrapadas y así poder obtener el resultado deseado (1).

Dentro de las técnicas heurísticas, existen las que se basan en la búsqueda local. Estas parten de una solución inicial y de una forma iterativa tratan de remplazarla por otra solución de su vecindario con mejor calidad, para luego continuar la búsqueda. Las heurísticas que se basan en la búsqueda local son una mejor alternativa, ya que por medio de ellas se obtienen mejores resultados, pero debido a esto requieren un mayor esfuerzo computacional (2).

En conclusión, una heurística es una estrategia usada para mejorar la eficiencia de un sistema determinado. Dentro de las técnicas heurísticas, se pueden encontrar las metaheurísticas, que son una familia de algoritmos aproximados de propósito general (3).

Las metaheurísticas son utilizadas para resolver problemas computacionales, en donde los resultados obtenidos con las técnicas heurísticas, no son satisfactorios. El objetivo de la metaheurísticas, es proporcionar una mejora local o estrategias de alto nivel. Estas técnicas crean algoritmos aproximados, para evitar los óptimos locales, analizando el espacio de búsqueda y de esta forma encontrar la solución óptima (4). Al igual que las heurísticas, existen múltiples clasificaciones de las metaheurísticas. Dentro del grupo de las metaheurísticas, se hallan las basadas en poblaciones (en las que el proceso considera múltiples puntos de búsqueda en

el espacio). Ejemplos de esta clasificación son los algoritmos Genéticos, el Scatter Search[2] y el PSO.

El PSO, es una metaheurística implementada para la resolución de problemas de optimización. Está basado, en la forma como algunos individuos intercambian información, que luego utilizan para dirigirse hacia el alimento (5). Debido a que el PSO resultó ser una respuesta a la dificultad de encontrar las soluciones a problemas complejos, este trabajo en la modalidad de investigación, describe detalladamente la comparación de dos de sus posibles arquitecturas. Antes de explicar las arquitecturas, cabe aclarar que un único algoritmo evolutivo, no funciona para todos los problemas y no existe uno de propósito general. Debido a esto, es necesario adaptar el algoritmo al problema de interés, como lo muestra el teorema del No Free Lunch (NFL) (6).

En consecuencia a la observación del NFL, en el presente trabajo se plantean dos arquitecturas. La primera es denominada PSOSX(S); ésta aplica inicialmente el método PSO basado en unos parámetros iníciales dados $(C_1,\ C_2\ y\ w)$,tiene como objetivo buscar iterativamente puntos cercanos a la respuesta óptima, para luego mediante la aplicación del método Simplex, obtener la solución. La segunda arquitectura denominada PSOSX(PE) al igual que la anterior, se lleva acabo ejecutando iterativamente como primera instancia el PSO; éste inicializa los parámetros para obtener una respuesta; en seguida el método Simplex, se encarga de optimizar los parámetros dados inicialmente al PSO, para luego ejecutar nuevamente el PSO con estos nuevos ajustes y encontrar la respuesta óptima.

El presente documento, comprende cuatro capítulos que describen los conocimientos y procedimientos necesarios, para el desarrollo del trabajo de grado. El primer capítulo, muestra los conceptos básicos del PSO, su origen y el

[2]También conocido como Búsqueda Dispersa, es un procedimiento metaheurístico basado en formulaciones y estrategias.

estado del arte, en donde se encuentran algunas aplicaciones. En el segundo, se describe el funcionamiento del PSO, Simplex, PSOCG, PSOSX(S) y PSOSX(PE). El tercero, detalla el proceso realizado para desarrollar la herramienta, su diseño e implementación. En el capítulo cuarto, con un análisis cuantitativo se muestra los resultados de las simulaciones realizadas con la herramienta, validándola con cinco de las funciones utilizadas en la literatura especializada en la evaluación de algoritmos de optimización. Finalmente, se presentan las conclusiones, recomendaciones, bibliografía y anexos.

1. ANTECEDENTES DEL PSO

Una de las grandes limitaciones en un problema de optimización, se genera cuando no se conoce ningún procedimiento "exacto" para obtener una solución; o cuando los métodos existentes son muy costosos. Debido a estas dificultades y a la escases de técnicas o algoritmos computacionales, se han venido desarrollando procedimientos de búsqueda directa o metaheurísticas, con el con el propósito de brindar una solución satisfactoria a determinados problemas.

A pesar de que las búsquedas metaheurísticas no garantizan un óptimo global, son muy eficientes para encontrar buenas respuestas, debido a que su finalidad es buscar soluciones cercanas al valor óptimo, esto dependiendo si el número de iteraciones son las necesarias, o si el punto de partida es el indicado. Las metaheurísticas son técnicas que se componen de diversas formas de solución, procedimientos de mejoras locales y estrategias de alto nivel, esto con el fin de crear un proceso capaz de emigrar de óptimos locales y realizar un análisis robusto en el espacio de búsqueda.

Dentro de las metaheurísticas se encuentra el PSO, que es una técnica aleatoria, poblacional, sin memoria, y basada en trayectorias. El PSO evita la convergencia a óptimos locales, especialmente en espacios de búsqueda complejos. En este capítulo, se describirán algunos conceptos relacionados con el PSO, tales como su origen, su definición y algunas aplicaciones, esto con el fin de familiarizar al lector con los términos utilizados en el presente trabajo de grado.

1.1 ORIGEN DEL PSO

El PSO, por sus siglas en ingles Particle Swarm Optimization, fue creado por Russell C. Eberhart [3] y James Kennedy[4], quienes lo publicaron en el "Proc. 1995 IEEE Intl. Conf. on Neural Networks, pp. 1942-1948, IEEE Press".

[3] Profesor de Ingeniería Eléctrica y Computación de Purdue University Indianapolis (IUPUI).

FIGURA 1: BANDADA DE AVES

Originalmente el PSO fue creado para la simulación del movimiento de individuos observado en bandadas de pájaros, enjambres de insectos y cardumen de peces. En este tipo de asociaciones existe un líder que condiciona el desplazamiento del grupo; además, cada individuo se guía basándose en su propia experiencia previa y la de los miembros de su grupo (7).

1.2 DEFINICIÓN DEL PSO

El PSO aplica conceptos de interacción social a la resolución de problemas de optimización; cuando el espacio de búsqueda es demasiado grande, este método puede llegar a ser una buena alternativa, en la solución óptima a un problema. El PSO es una técnica metaheurística, que se basa en cómo algunas especies, comparten información entre los miembros de un grupo y la utilizan para dirigirse a donde se encuentra el alimento buscado (8).

El algoritmo PSO, está compuesto por un grupo de individuos llamados "partículas", que se mueven por un espacio determinado, estas partículas tienen individualmente un vector posición, un vector de velocidad y un Fitness, que indica la calidad de una solución dada. Durante el proceso las partículas son inicializadas y luego se determina cuan buena es la posición de cada individuo, teniendo en cuenta su propia experiencia previa y la de sus vecinos (9).

[4] Psicólogo de investigación, en la Oficina de Estadísticas del Trabajo en Washington, DC.

A continuación, en la tabla # 1 se encuentran algunos términos, empleados en el algoritmo PSO y que se utilizaron en este trabajo.

TERMINO	SIGNIFICADO
Partícula	Individuo del enjambre.
Posición X_i	Coordenadas de una partícula en un espacio N-dimensional que representa una solución para el problema.
Enjambre	Todo el grupo de agentes, individuos o partículas.
V_i	Velocidad de la partícula.
$Fitness$	Número que muestra la calidad de una solución dada.
P_{best}	Mejor posición de la partícula, obtenida individualmente a lo largo del proceso.
G_{best}	Es la mejor localización en todo el enjambre de partículas.
V_{max}	Velocidad máxima permitida en una dirección dada.
n	Tamaño de la población.
$iter$	Contador de las iteraciones.
w	Inercia de la partícula
C_1 y C_2	Representan ponderaciones de la parte cognitiva y social, respectivamente.

TABLA 1: TÉRMINOS PSO

1.3 APLICACIONES DEL PSO

El estudio del PSO ha venido incrementando con el pasar de los años. En algunos países como China, USA, México, Argentina, Perú, entre otros; el éxito del PSO en problemas de optimización, ha demostrado el gran potencial de este algoritmo. Hoy en día existe una gran variedad de aplicaciones del PSO, en este trabajo se muestra algunas de ellas.

1.3.1 DISEÑO Y MECANIZADO DE TOLERANCIAS

Un ejemplo del uso del PSO, es el diseño y optimización de la tolerancia del mecanizado en ingeniería; ésta aplicación es muy útil debido a que afecta tanto la calidad, como el costo del ciclo general de un producto. La realización de un diseño habitualmente se divide en dos fases: la primera es el diseño del producto o tolerancias de diseño, en donde se deben distribuir sus dimensiones y la segunda es el proceso de diseño o tolerancias de fabricación, en donde se refina el producto para satisfacer la exigencia del plan de trabajo.

No obstante, en el diseño y mecanizando de tolerancias basadas en la experiencia, no se puede garantizar la tolerancia óptima, debido a que estas requieren de un costo elevado de producción. Para solucionar este inconveniente, el PSO se encarga de seleccionar las adecuadas secuencias de las tolerancias, de manera que la disminución de costos en la fabricación del producto, sería su función objetivo. Generalmente, la realización de productos mecánicos requiere de una variedad de métodos, en donde tanto el costo como el proceso son distintos para todos los casos. Por lo tanto, el precio de fabricación total del producto es la suma del valor individual de cada proceso.

El costo de fabricación del mecanizado de tolerancias y el costo total de un producto se representan en la ecuación [1] y [2] respectivamente.

$$C_{ij}(\delta_{ij}) = a_0 e^{-a_1(\delta - a_2)} + a_3 \quad i = 1, \cdots, n \qquad \textbf{[1]}$$

$$C = \sum_{i=1}^{n} \sum_{j=1}^{m_i} c_{ij} \qquad \textbf{[2]}$$

donde $C_{ij}(\delta_{ij})$ y δ_{ij} son el costo de manufactura y la tolerancia de fabricación asociado a las dimensiones ij, respectivamente. Las constantes a_0, a_1, a_2, a_3 son parámetros de control, m_i es el número de operaciones y n es el número de dimensiones.

Aprovechando las ventajas del PSO, éste se utiliza en el diseño y mecanizado de tolerancias. Primero se emplea una variable PF (Factibilidad de las partículas) para registrar si la partícula no ha cumplido con las condiciones de limitación, según la información de éste, las partículas obtienen su clasificación (si son un P_{best} o un G_{best}) limitando la situación actual la variable CF (Partícula Factible), para luego seleccionar la velocidad. Después de actualizar las mejores posiciones obtenidas, el algoritmo toma la estrategia estática con el fin de garantizar la igualdad. Como dato importante, es necesario resaltar que el algoritmo PSO no garantiza soluciones óptimas en la población inicial, sino una vez obtenidas P_{best} y G_{best}, se utilizan en la siguiente iteración (10).

1.3.2 RECONSTRUCCIÓN DE IMÁGENES DE MICROONDAS

Dentro de las muchas aplicaciones se encuentra el µPSO (microparticle swarm optimizer), este algoritmo es derivado del PSO. El µPSO al igual que el PSO, está conformado por vectores multidimensionales, que guardan sus mejores posiciones. El µPSO utiliza un pequeño grupo de partículas, las cuales son las encargadas de reiniciar el proceso cuando la población haya convergido, esto es, cuando la población obtiene una desviación estándar inferior al umbral definido previamente y a colocado la solución en la lista negra (posiciones en las cuales a las partículas se les impide volver nuevamente).

Las partículas y soluciones pertenecientes a dicha lista, son de igual polaridad y se repelen entre ellas, que según la ley de Coulomb, la fuerza de repulsión entre dos cargas puntuales es inversamente proporcional a la distancia que las separa. Para garantizar la convergencia del µPSO, se utilizan las ecuaciones [3] y [4]:

$$V_{best}(t+1) = w * V_{best}(t) + C_1 A\big(P_{best}(t) - X_{best}(t)\big) + C_2 B\big(G_{best}(t) - X_{best}(t)\big) + R_{best}(t) \quad \textbf{[3]}$$

$$V_g(t+1) = w * V_g(t) - X_g(t) + G_{best}(t) + \rho(t)(1-2D) + R_g(t) \quad \textbf{[4]}$$

donde V_{best} es la velocidad de la mejor partícula, V_g es la velocidad del resto de partículas, w es la inercia de la partícula, C_1 y C_2 son constantes, A , B y D variables aleatorias entre (0,1), $\rho(t)$ es un factor de escala y $R_g(t)$es el rechazo total de la lista negra.

El µPSO ha sido utilizado para la reconstrucción de imágenes de microondas; en especial se aplica en la reconstrucción de las propiedades dieléctricas normales y tejidos malignos de mama, en donde se utilizó MultiView[5] para reducir los costos computacionales. En esta aplicación el objetivo principal es reconstruir ε_r y σ que son los valores de los tejidos, utilizando una función de distribución gaussiana para generar 100 muestras de tejido mamario normal y tejidos malignos, en los cuales las muestras tienen una deviación y una media diferente. Después de realizar una serie de simulaciones, la ubicación de los tejidos malignos son identificados fácilmente (11).

1.3.3 OTRAS APLICACIONES

El algoritmo de optimización del diseño de máquinas de imán permanente sin escobillas, entra a ser parte de este grupo. El objetivo del algoritmo es, que la máquina multipolar de imanes permanentes (PM) maximice el rendimiento del motor, optimizando el arco y la longitud de las direcciones de magnetización de los segmentos de la máquina PM. La función objetivo del PSO es potenciar al máximo la fase del flujo de un determinado volumen de la maquina PM, para luego comparar el cálculo con los resultados de la magnetización uniforme, radial y la obtenida por el Halbach[6]. La metodología también se puede emplear en la determinación de un volumen mínimo para una carga magnética (12).

[5] Representa un control que actúa como contenedor de un grupo de controles View.
[6] Clase de magnetización discreta del rotor

Como otro ejemplo del uso del PSO, se encuentra el control de temperatura. Un inconveniente presentado en este proceso, es que cualquier cambio en la entrada del sistema, producirá una variación en la temperatura; de esta forma, la entrada se convertiría en una secuencia de variables aleatorias y éstas pueden cambiar abruptamente. Por esta razón, es que se emplea el algoritmo PSO, buscando el mejor RFC (Recurrent Fuzzy Controller) para garantizar el óptimo desempeño de la planta. Para este caso, cada RFC corresponde a una partícula del PSO (13).

Otra aplicación es la orientación general y solución altamente configurable, para simular protocolos de redes satelitales; esto con el fin de mejorar la fiabilidad y disponibilidad de la plataforma de dichas redes (14).

Entre las aplicaciones exitosas del PSO, se encuentra la segmentación de imágenes (14); la construcción de juegos (16); el diseño de un UPFC (Unified Power Flow Controller) con el fin de mejorar la estabilidad transitoria de un sistema eléctrico (16); la ingeniería geotécnica, pues el PSO se utiliza para reconocer algunos parámetros de las rocas en las construcciones de los túneles y de esta forma evitar accidentes (18); el diseño de estructuras en general (19); el diagnóstico y predicción de fallas en los motores de avión (20); la optimización del diseño de dispositivos electromagnéticos optimizando los parámetros geométricos y las densidades de corriente de las bobinas (21); el problema del viajante bi-objetivo que es una de las aplicaciones mas estudiadas (22); entre otras.

2. HIBRIDOS Y ARQUITECTURAS

2.1 PSO

El PSO parte de una población al azar para iniciar un proceso de búsqueda, donde cada individuo es una partícula, que representa un candidato a la solución del problema de optimización. La partícula se mueve a través de un espacio de búsqueda multidimensional, ajustando su posición de acuerdo a su propia experiencia y la de otras partículas en su vecindario. Es decir, una partícula hace uso de la mejor posición encontrada por si misma (P_{best}) y de la mejor posición de su vecindario (G_{best}), para dirigirse hacia la solución óptima (23). A continuación se muestra el procedimiento necesario para el buen funcionamiento del algoritmo PSO:

2.1.1 INICIALIZACIÓN DEL PSO

Se ajusta el contador de iteración $(iter = 0)$ y el tamaño de la población. Luego, se genera n partículas y una velocidad inicial, aleatoriamente. Cada partícula en la población inicial se evalúa usando la función objetivo. Para cada partícula, se establece la mejor posición individual y la mejor global.

2.1.2 ACTUALIZACIÓN DEL CONTADOR.

Se actualiza el contador de iteración.

2.1.3 ACTUALIZACIÓN DE LA VELOCIDAD.

El algoritmo mantiene una población de n partículas cuyas posiciones, $X(t)\ con\ \ t = 1,2, \dots n,$ inician con valores aleatorios (24). Las posiciones se incrementan en cada iteración del algoritmo, como se indica en la siguiente ecuación:

$$X_i(t+1) = X_i(t) + V_i(t)\ \textbf{[5]}$$

La velocidad $\overrightarrow{v_i}$ de cada partícula i, se actualiza como se observa en la ecuación [6]:

$$V_i(t+1) = w\left(V_i(t) + C_1A\left(P_{best}(t) - X_i(t)\right) + C_2B\left(G_{best}(t) - X_i(t)\right)\right) \textbf{[6]}$$

donde $\mathbf{A}$ y $\mathbf{B}$ son números aleatorios distribuidos uniformemente entre [0,1] y $\mathbf{w}$ es la inercia de la partícula. $\mathbf{C_1}$ y $\mathbf{C_2}$ son constantes positivas que representan ponderaciones de la parte cognitiva y social, respectivamente, e influencian cada partícula en dirección a las posiciones P_{best} y G_{best} . En el presente trabajo, se utilizó como parámetros iníciales $\boldsymbol{C_1 = 2}$, $\boldsymbol{C_2 = 2}$ y $\boldsymbol{w = 1}$.

2.1.4 ACTUALIZACIÓN DE LA POSICIÓN.

Según la velocidad, cada partícula cambia su posición como lo muestra la ecuación [7].

$$X_i(t+1) = X(t) + \Delta t * V_i(t+1) \textbf{[7]}$$

2.1.5 LA MEJOR POSICIÓN INDIVIDUAL

Se actualiza la mejor posición de cada partícula. Si la posición $X_i(t+1)$ es mejor que $P_{best}(t)$, entonces $P_{best}(t+1) = X_i(t+1)$, de lo contario $P_{best}(t+1) = P_{best}(t)$.

2.1.6 LA MEJOR POSICIÓN GLOBAL

Se actualiza la mejor posición grupal. Si la posición $X_i(t+1)$ es mejor que $G_{best}(t)$, entonces $G_{best}(t+1) = X_i(t+1)$, de lo contario $G_{best}(t+1) = G_{best}(t)$.

2.1.7 TERMINAR EL PROCESO

Se termina las iteraciones cuando el contador **iter** lo indique (25).

El proceso se repite cuantas veces sea necesario, reemplazando los valores anteriores por los nuevos calculados. En la figura 2, se observa el diagrama de bloques del PSO.

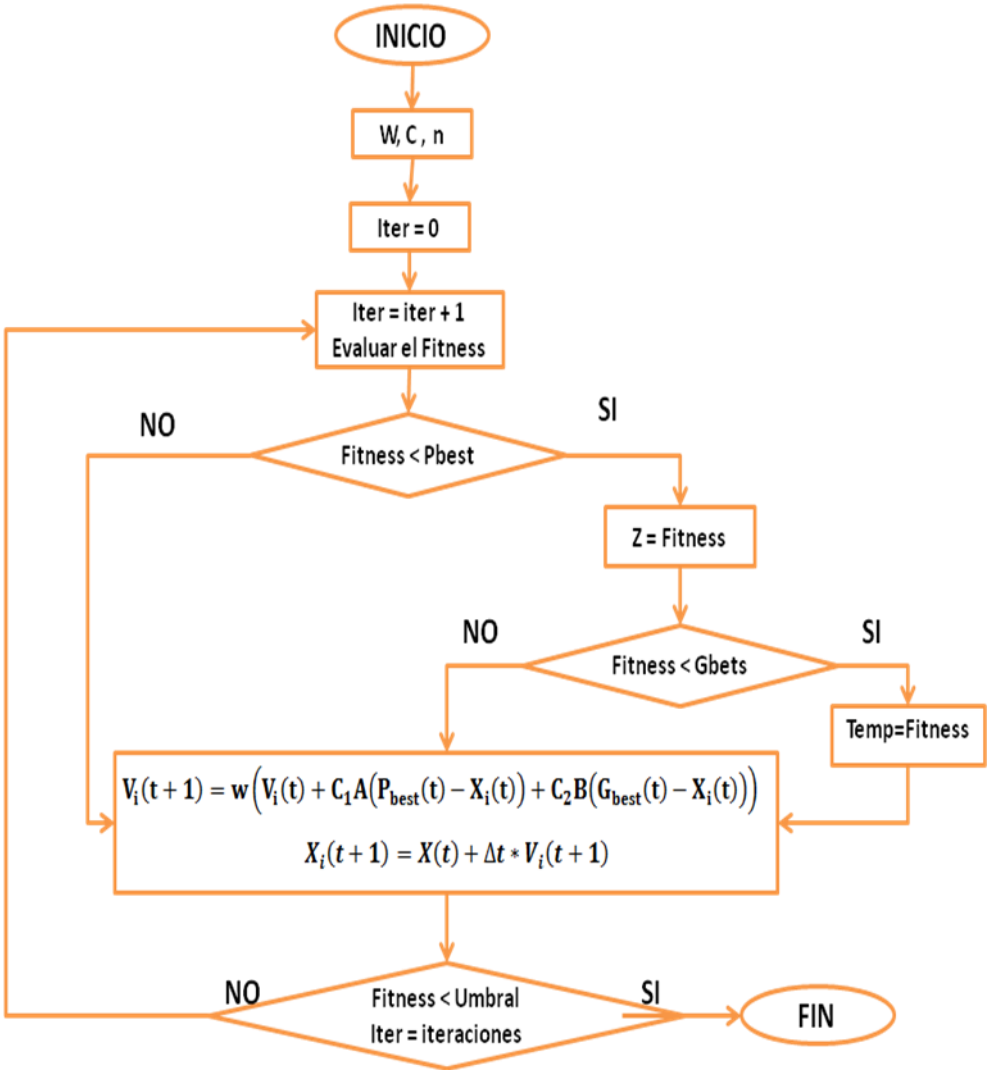

FIGURA 2: ALGORITMO PSO

2.2 SIMPLEX

El método Simplex o Nelder Mead[7], es un algoritmo iterativo, que se basa en fundamentos geométricos; éste secuencialmente se va aproximando al óptimo del problema a partir de la estimación inicial y sustituyendo los peores puntos por un nuevo punto. El algoritmo forma el mejor poliedro en el hiperespacio paramétrico. Este poliedro tiene N+1 vértices en los cuales se evalúa la función objetivo y se escoge que parámetros ajustan mejor. Para el caso de dos dimensiones, se forma un triangulo equilátero y se hace el mismo procedimiento evaluando la función objetivo en cada vértice. La labor del Simplex es mejorar el vértice anterior, generando una variedad de triángulos para encontrar las coordenadas de la solución óptima (26). El procedimiento es repetido sucesivamente, descartándose la peor respuesta. A continuación se describen los pasos a seguir:

2.2.1 Para empezar se realiza un triangulo equilátero $\overrightarrow{BGW}$mostrado en la figura 3, en los cuales el vértice $\vec{B} = (x_1, y_1)$ es el mejor, $\vec{G} = (x_2, y_2)$ es bueno y $\vec{W} = (x_3, y_3)$ es el peor.

FIGURA 3: PUNTO DE PARTIDA

2.2.2 Buscar el punto medio del segmento $\overrightarrow{BG}$, como lo indica la figura 4.

$$M = \frac{1}{2}(\vec{B} + \vec{G}) = \frac{(x_1, x_2)}{2}, \frac{(y_1, y_2)}{2}$$

[7] Es un método numérico realizado en 1965 por Nelder Mead, utilizado para minimizar una función objetiva en un espacio multidimensional.

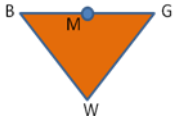

FIGURA 4: PUNTO MEDIO

2.2.3 Elegir un punto de prueba $\vec{R}$ a una distancia d para luego reflejar el triangulo por el segmento $\overrightarrow{BG}$, según se observa en la figura 5.

$$\vec{R} = \vec{M} + (\vec{M} - \vec{W}) = 2\vec{M} - \vec{W}$$

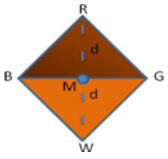

FIGURA 5: PUNTO DE PRUEBA

2.2.4 Verificar el valor de la función en $\vec{R}$, si es menor que el valor de la función en $\vec{W}$, se debe ampliar el segmento de la línea $\vec{M}$y $\vec{R}$ hasta el punto $\vec{E}$, como lo muestra la figura 6.

$$\vec{E} = \vec{R} + (\vec{R} - \vec{M}) = 2\vec{R} - \vec{M}$$

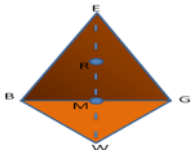

FIGURA 6: ETAPA DE EXTENSION

2.2.5 Verificar el valor de la función en $\vec{R}$, si es igual al valor de la función en $\vec{W}$, se debe probar los puntos $\overrightarrow{C_1}$ y $\overrightarrow{C_2}$ de los segmentos de las líneas

$\overrightarrow{WM}$ y $\overrightarrow{MR}$ respectivamente, para luego escoger el punto $\vec{C}$, éste es el que tienen menor valor de la función.

$$C_1 = \frac{1}{2}(\vec{M} + \vec{W})$$

$$C_2 = \frac{1}{2}(\vec{M} + \vec{R})$$

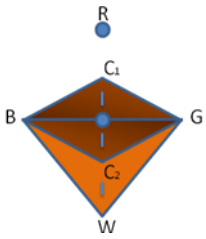

FIGURA 7: ETAPA DE CONTRACCIÓN

2.2.6 Verificar el valor de la función en $\vec{C}$, si no es menor al valor de la función en $\vec{W}$, el triangulo $\overrightarrow{BGW}$ se sustituye por $\overrightarrow{BMS}$, donde $\vec{S}$es el punto medio de la línea $\overrightarrow{BW}$. Ver figura 8.

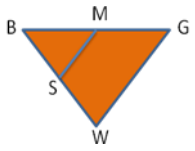

FIGURA 8: CAMBIOS DE VERTICES

En la figura 9 se encuentra el diagrama de bloques del simplex, donde $\boldsymbol{F}$ representa la función que debe ser evaluada en un punto determinado. El objetivo del método Simplex es dirigirse siempre, hacia la región de la respuesta óptima.

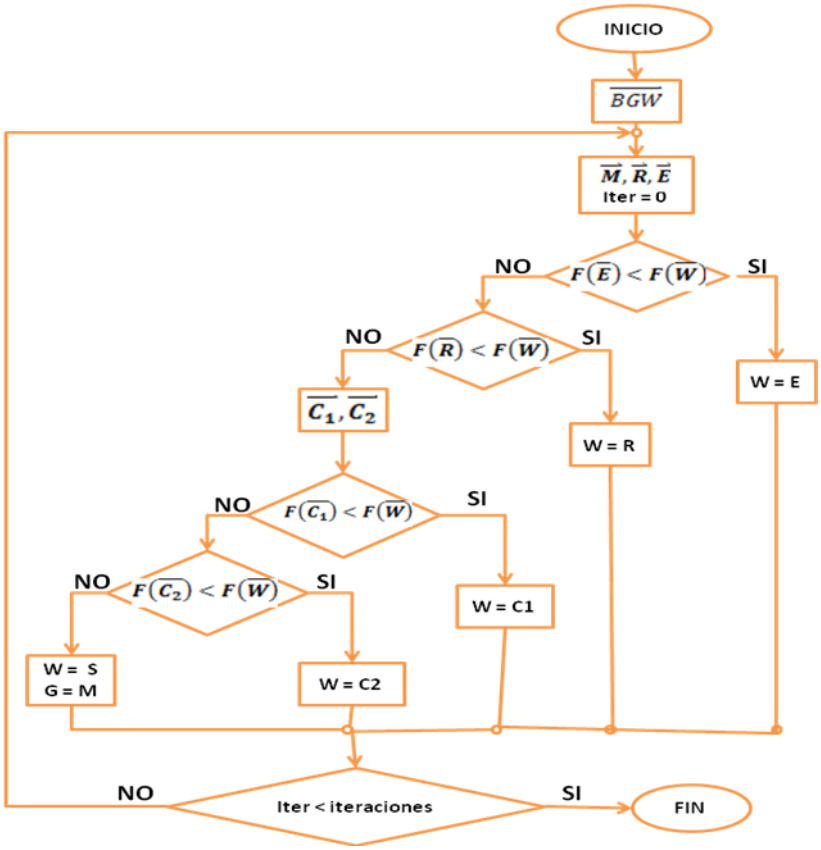

FIGURA 9: ALGORITMO SIMPLEX

2.3 PSOCG

Desde hace unos años, se ha buscado tener una fundamentación matemática del PSO tan profunda como su implementación. Basados en la experiencia y en trabajos realizados recientemente acerca del tema, han llevado a que se busque un factor de contracción en la ecuación de velocidad del PSO, con el fin de garantizar la convergencia del método. Maurice Clerc en 1999 mostró una gran propuesta para dar solución a este problema. La solución involucra algunos parámetros del mismo algoritmo. Este factor es conocido como K, el cual es función de C_1 y C_2 como se muestra en las ecuaciones [8] y [9].

$$V_{id} = K * \left[v_{id} + c_1 * rand * (p_{id} - x_{id}) + c_2 * rand * \left(p_{gd} - x_{id}\right)\right] \quad [8]$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \qquad donde\ \varphi = C_1 + C_2, \qquad \varphi > 4 \quad [9]$$

Si se evalúa el valor de $\varphi\ (al\ rededor\ de\ 4)$, se puede observar que K es 0.729. Este valor acompaña a la ecuación de velocidad en cada término y afecta toda la función. Pruebas posteriores realizadas por Maurice Clerc, mostraron que esta nueva ecuación hace que las partículas se acerquen más fácilmente a la respuesta, siempre y cuando C y W se mantengan dentro del rango apropiado. Cabe aclarar que el factor no afecta notablemente al valor numérico de la respuesta final, si no a la velocidad de convergencia del algoritmo.

2.4 PSOSX

Una novedosa estrategia para el control de parámetros del PSO, es la introducción del algoritmo Nelder–Mead (simplex) al funcionamiento del PSO. Con esto, se buscó que la convergencia del PSO, fuera independiente de las constantes heurísticas, aumentar su estabilidad y confiabilidad. Como resultado, se llegó a reducir considerablemente el número de iteraciones del PSO y a reducir el costo computacional del algoritmo. Para el presente trabajo de grado, se implementaron dos tipos de híbridos del PSOSX, el PSOSX(S) y el PSOSX(PE).

2.4.1 PSOSX(S)

La primera arquitectura denominada PSOSX(S), se muestra en la figura 10. En el PSOSX(S), el método simplex se encarga de tomar la respuesta dada por el PSO y optimizarla. Se busca que la respuesta entregada por el PSO sea la más exacta posible para que se pueda aplicar el simplex eficientemente, pues esa es la falencia que más se presenta en el Nelder–Mead; si los puntos iníciales no son apropiados, el método no converge. El PSOSX(S), se basa en la operación inicial del PSO básico.

La unión en serie de los algoritmos PSO y Simplex, reduce la necesidad de emplear un gran número de iteraciones del PSO y disminuye la posibilidad de que

el simplex diverja; conjunto que da como resultado un hibrido más confiable y rápido a la hora de aplicarlo en problemas complejos.

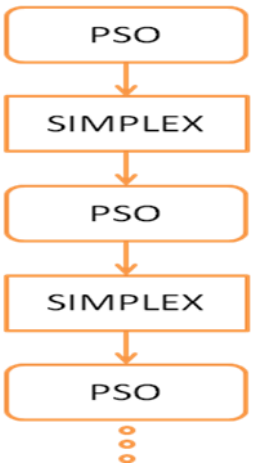

FIGURA 10: ALGORITMO PSOSX (S)

2.4.2 PSOSX(PE)

La estrategia planteada en este hibrido, conserva la estructura principal de ejecutar inicialmente el PSO, buscando ubicar una posible región donde encontrar la respuesta al problema. A continuación, se aplica el método simplex a la ecuación de velocidad del PSO, para ajustar los parámetros iníciales C y W. Esto hace que se optimice el PSO para seguir buscando la respuesta, pero con los parámetros óptimos de búsqueda.

Como consecuencia del PSOSX(PE), se reduce la posibilidad de que PSO diverja por parámetros iníciales errados o mal escogidos y aumenta la velocidad de convergencia del método PSO básico, aunque es posible que no sea tan rápido como el PSOSX(S), es más eficiente que el PSO básico. En la figura 11 se observa una breve descripción del PSOSX(PE).

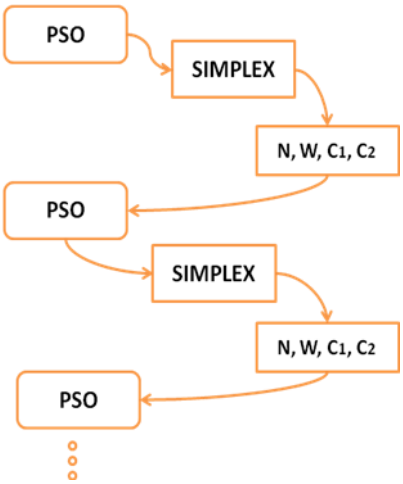

FIGURA 11: ALGORITMO PSOSX (PE)

En resumen, a diferencia del PSOSX(S), en la arquitectura PSOSX(PE) la labor del simplex es optimizar, no la respuesta, sino los parámetros del PSO $(C_1, C_2\ y\ w)$.

3. DESARROLLO DE LA HERRAMIENTA COMPUTACIONAL

En este capítulo, se presenta una descripción detallada de los pasos realizados, para la construcción de la herramienta computacional. El desarrollo se hizo, teniendo en cuenta el cumplimiento de los objetivos planteados.

3.1 DISEÑO

Todas las pruebas realizadas en el presente trabajo de grado se realizaron en un equipo en especial, para mantener las condiciones operacionales de las pruebas. Este equipo consta de un procesador AMD Turion™ X2 TL-60 de 2 GHz y 1 MB de cache, con capacidad de operación a 64 bits, pero empleado a 32, bits debido a eventuales problemas entre el sistema operativo y Matlab. El equipo consta de 3 GB de memoria DDR2 de 800MHz, tarjeta de video ATI Radeon Express 1250 de 1GB, entre otras cosas. Estas especificaciones se pueden constatar en la figura 12.

Elemento	Valor
Fabricante	Acer
Nombre del producto	TravelMate 4520
Sistema operativo	Microsoft® Windows Vista™ Ultimate
Procesador	AMD Turion(tm) 64 X2 Mobile...
Memoria del sistema	3 GB
Unidad de disco duro 1	Hitachi HTS542516K9SA00 ATA Device,...
Unidad de CD/DVD 1	TUXWPYN 9EB0DEFS SCSI CdRom...
Unidad de CD/DVD 2	HL-DT-ST DVDRAM GSA-T20N ATA...
Vídeo	ATI Radeon X1200 Series
Memoria de gráficos...	1 GB
Audio	Realtek High Definition Audio
Tarjeta LAN inalámbrica 1	Adaptador de red Broadcom 802.11g
UUID	A8499A20100011DD955AD4FB13C44EFA
Número de serie	LXTLD0Z0298160668B2000

FIGURA 12: ESPECIFICACIONES DEL EQUIPO UTILIZADO

Como primera instancia se realizo la lógica necesaria para la elaboración del código en Matlab del PSO, para esto se partió de la ubicación inicial de las partículas en el espacio de operación de la función (figura 13).

```
%% Posicion Inicial del Enjambre
incio = 1;
for i = 1 : d
    for j = 1 : d
        v(incio, 1, i) = i;
        v(incio, 1, i) = j;
        incio = incio + 1;
    end
end
v(:, 4, 1) = 1000;
v(:, 2, :) = 0;
```

FIGURA 13: POSICION INICIAL DEL ENJAMBRE

Estas posiciones fueron posibles creando n vectores que ubican “d” cantidad de partículas a lo largo de los ejes de cada dimensión, y al mismo tiempo se reescriben dos estados, con el fin de establecer un el mejor valor inicial igual a 1000 y una velocidad inicial igual a cero, para dar comienzo a los cálculos.

A continuación se define el lazo de iteraciones y número de partículas a las cuales se les actualizará su posición cada vez que se haga una iteración (figura 14).

```
for iter = 1 : iteraciones
    for i = 1 : n_particulas
        for j = 1 : d
            v(i, 1, j) = v(i, 1, j) + v(i, 2, j)/1.3;
            x(j) = v(i, 1, j);
        end
        eval = %FUNCION A UTILIZAR ( x=x(1),y=x(2)...)
            if eval < v(i, 4, 1)
                for k = 1 : d
                    v(i, 3, k) = v(i, 1, k);
                end
                v(i, 4, 1) = eval;
            end
    end
    [temp, gbest] = min(v(:, 4, 1));
```

FIGURA 14: ACTUALIZACION DE LA POSICIÓN

Se definen las variables $X(i)$ a utilizar $(X(1), X(2) \dots)$, para poder escribir la ecuación con la que se desea trabajar. Luego de introducir la función (eval), Matlab ya ha evaluado internamente las variables definidas en dicha ecuación, por consiguiente, se procede a comparar si el valor actual de eval es menor a 1000, como se había establecido inicialmente. Si eval es menor que 1000, entonces se procede a definir la variable $X_i(t+1)$, además se precisa también cual será el vector G_{best} temporal, para que luego se introduzcan todos los datos a la ecuación de velocidad.

```
%% Vector Posicion (velocidad)
    for i = 1 : n_particulas
        for j = 1 : d
            v(i, 2, j) = rand*w*v(i, 2, j) + cab*rand*(v(i, 3, j) - v(i, 1, j)) + cab*rand*(v(gbest, 3, j) - v(i, 1, j));   % x(i)
        end
    end
    for i = 1 : d
        rta_x(i)=v(gbest, 3, i);
    end

end
```

FIGURA 15: ACTUALIZACION DE LA VELOCIDAD

El lazo observado en la figura 15, muestra como se evalúa la ecuación de velocidad del PSO en cada variable $X(i)$ y se guarda el mejor global. En el proceso de reubicar y reevaluar la función velocidad se repite las veces que sean necesarias para alcanzar la respuesta deseada.

Cabe aclarar que se escogió un solo valor de C llamado $C_{ab} = C_1 = C_2$, por el simple hecho que estos dos parámetros están siendo acompañados de la función random (rand), la cual, aunque se aplique dos veces en la misma ecuación, genera 2 valores diferentes en cada termino. Por consiguiente se consideró que no era necesario incluir 2 valores de C, si igual se iban a ver afectados por el parámetro rand.

Como segunda instancia, se buscó implementar el algoritmo simplex, pero se llegó a la conclusión que la función “fminsearch” de Matlab, es la implementación del

método Nelder Mead, por consiguiente no valía la pena hacer un nuevo algoritmo, si ya existía uno implementado y aprobado por matworks.

El fminsearch tiene la ventaja de ser muy rápido, pero la gran desventaja que para una función con n dimensiones y n cantidad de puntos de partida, los puntos deben ser cercanos a la respuesta final para que Nelder Mead converja.

```
%% Aplicando Nelder-Mead (Simplex)
fun = @(x)x(1)^2-10*cos(2*pi*x(1))+10+x(2)^2-10*cos(2*pi*x(2))+10+
[x,fval] = fminsearch(fun,[rta_x]);
toc
disp('RESPUESTA X(i) RESPECTIVAMENTE (PSOSX(S))')
Respuesta = x
t = toc;
```

FIGURA 16: NEALDER – MEAD Y PSOSX(S)

Los puntos de partida del Nealder Mead, son los mejores globales entregados por el PSO, de esta forma se alimenta eficientemente el simplex para buscar mejorar la respuesta anterior. Con esto se completa el diseño básico del híbrido PSOSX(S) (Ver figura 16).

Para la implementación del PSOSX(PE), se diseñó una primera etapa, en donde el PSO se ejecuta, utilizando el mismo diseño empleado anteriormente, pero con la diferencia que ahora no se aplica fminsearch a la función a optimizar, si no a la ecuación de la velocidad del PSO básico; esto con el fin de optimizar los valores w y Cab (Ver figura 17).

```
%% Aplicando Nelder-Mead (Simplex)
fun = @(x)rand*x(1)*v(i, 2, d) + x(2)*rand*(v(i, 3, d) - v(i, 1, d)) + x(2)*rand*(v(gbest, 3, d) - v(i, 1, d));   % x(i)
[x,fval] = fminsearch(fun,[rta_x]);

disp('AJUSTANTO PARAMETROS CON SIMPLEX')

wN = x(1)                                                                    %Nuevo valor de W
cabN = x(2)                                                                  %Nuevo valor de Cab
```

FIGURA 17: NEALDER – MEAD Y PSOSX(PE)

A continuación se ejecuta nuevamente la etapa del PSO, utilizando los nuevo valores de $w\ y\ Cab$ para encontrar la respuesta final de la función. Con esto culmina el diseño del PSOSX(PE). Cabe aclarar que el diseño realizado de los híbridos es multidimensional y completamente funcional.

3.2IMPLEMENTACIÓN

La fase de implementación, tiene como objetivo producir una solución óptima. A continuación se presentan las funciones utilizadas en MATLAB para este programa:

- Guide: utilizada para la elaboración de la interfaz gráfica.
- Fminsearch: Función que cumple la labor del Simplex (Nelder Mead).
- Tic-toc: utilizada para medir el tiempo que tarda cada arquitectura en encontrar la respuesta.
- Otras: funciones utilizadas para realizar gráficas tridimensionales, tales como ezcontour, meshgrid y plot3.

El tutorial que muestra el manejo de la herramienta, se puede observar en el ANEXO I.

4. VALIDACION DE LOS MODELOS DEL PSO

En este capítulo, como primera instancia se muestran las funciones escogidas con su respectiva representación gráfica. A continuación, se observan los resultados obtenidos por las diferentes arquitecturas. Las pruebas se realizaron para verificar la calidad y desempeño de la cada hibrido.

4.1 FUNCIONES

Se eligieron seis funciones, en las que es conocido el valor de su función objetivo. Inicialmente se trabajó con dos dimensiones, debido a que sus graficas se pueden realizar. Posteriormente, se escogió una función de diez dimensiones, con el fin de verificar la veracidad de la herramienta ante funciones multidimensionales.

4.1.1 FUNCIÓN VENTER 2D

$$z = x^2 - 100\cos(x)^2 - 100\cos\left(\frac{1}{30}x^2\right) + y^2 - 100\cos(y)^2 - 100\cos\left(\frac{1}{30}y^2\right) + 1400$$

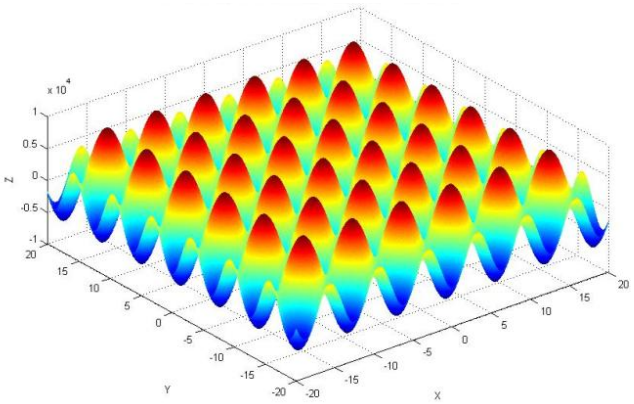

FIGURA 18: VENTER 2D

4.1.2 FUNCIÓN BOOTH

$$z = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

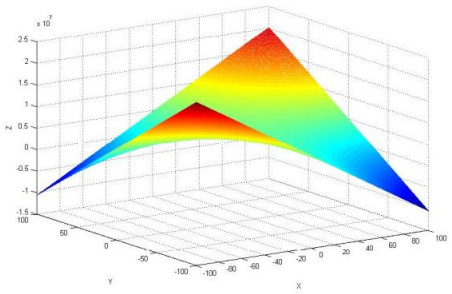

FIGURA 19: BOOTH

4.1.3 FUNCIÓN B2

$$z = {x_1}^2 + 2{x_2}^2 + 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7$$

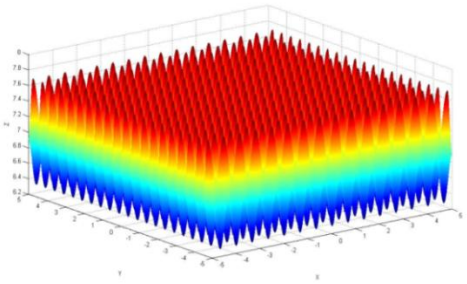

FIGURA 20: B2

4.1.4 FUNCIÓN ROSENBROCK

$$z = 100(y - x^2)^2 + (1 - x)^2$$

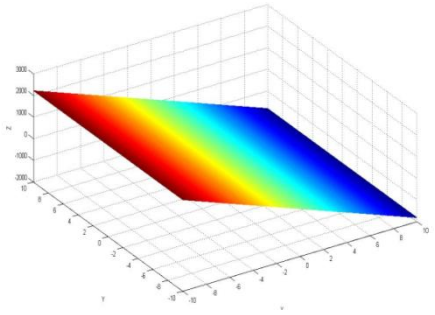

FIGURA 21: ROSENBROCK

4.1.5 FUNCIÓN TESTE 1

$$z = ((y-x)^2)^2 + (x-1)^2 + 4$$

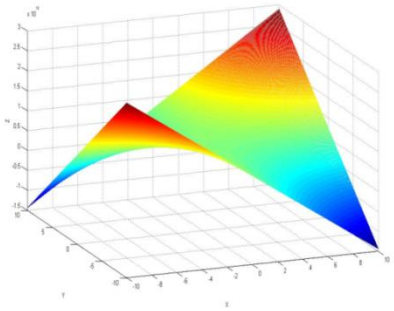

FIGURA 22: TESTE 1

4.1.6 FUNCIÓN RASTRIGIN

$$z = \sum_{i=1}^{10} {x_i}^2 - 10cos(2\pi x_i) + 10$$

4.2 CALCULOS REALIZADOS

En esta sección se muestran los resultados obtenidos de las pruebas y el análisis efectuado por los mismos.

4.2.1 PSO

Para cada figura a continuación (figura 23 - 27), se observa la ejecución del PSO en cada función, teniendo en cuenta que se mantiene $C_{ab} = 2$ y $w = 1$. La diferencia existente entre las graficas, es que las del lado izquierdo tienen solo 2 iteraciones y las del derecho tienen 35 iteraciones; esto con el fin de ver la diferencia que hace el aumentar la cantidad de iteraciones.

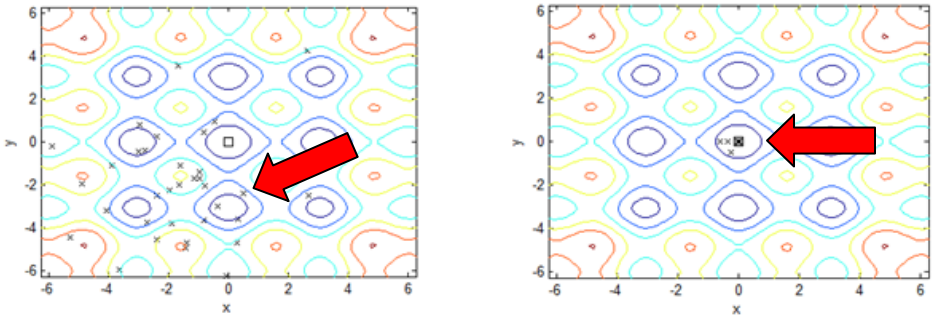

FIGURA 23: VENTER 2D Y PSO

En esta función específica (Figura 23) se observa como las partículas prueban máximos locales y globales demarcados en azul, pero se encuentra el máximo de la función a pesar de las pruebas previas.

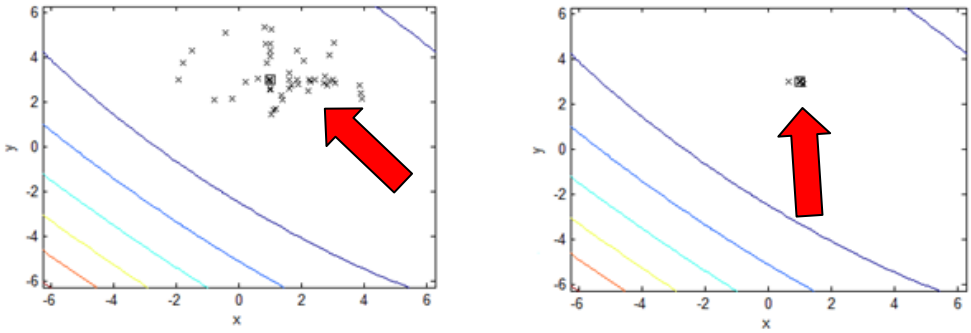

FIGURA 24: BOOTH Y PSO

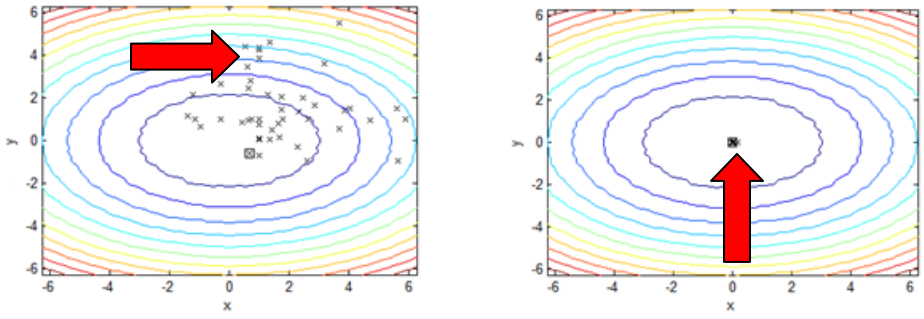

FIGURA 25: B2 Y PSO

Se puede ver la dispersión de las partículas en el plano para cada función.

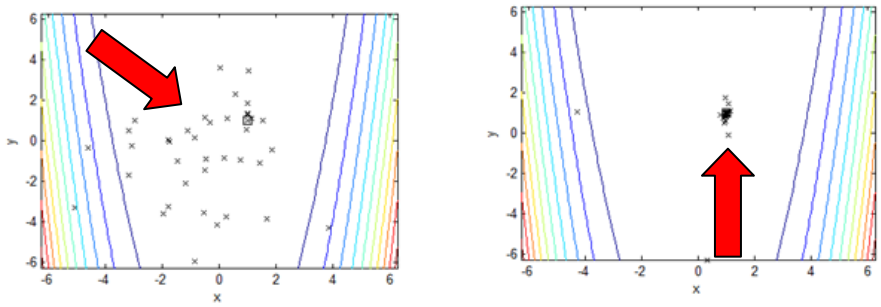

FIGURA 26: ROSENBROCK Y PSO

A pesar de haber realizado 35 iteraciones, algunas partículas tienen mejor ubicación que otras y esto es un riesgo que se ve minimizado por la cantidad de partículas de prueba. Debido a esto, la respuesta final del PSO es el mejor global de todas las iteraciones.

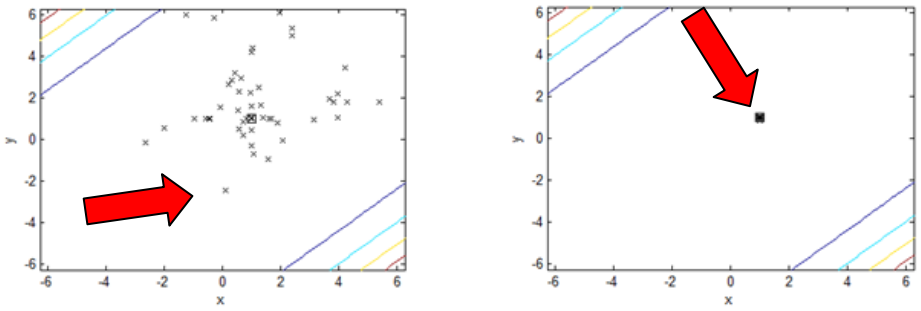

FIGURA 27: TESTE 1 Y PSO

En las figuras 23 a la 27 se puede observar como la comunidad o enjambre se dispersa por el espacio de la función, probando cada nueva ubicación en busca del óptimo del problema. Incluso se puede ver como en cada función, las partículas tienen un comportamiento diferente, lo cual es normal porque sus diferentes posiciones en el espacio obedecen uncialmente a ubicaciones aleatorias. Las graficas del lado derecho muestran como la inteligencia colectiva lleva al enjambre a encontrar una ubicación particular donde la función tiene su óptimo y el ver como éstas se agrupan, es señal que el método converge.

4.2.2 PSOCG

Para este caso, se aplicó una arquitectura levemente diferente llamada PSO de convergencia garantizada. Donde se mantuvo el valor de $C_{ab} = 2$ *y* $w = 1$ para todas las funciones, solo se variaron el número de iteraciones: 2 en las graficas de la izquierda y 35 a la derecha (figura 28-32).

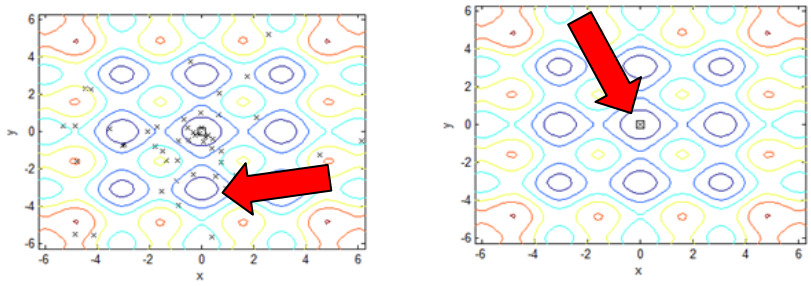

FIGURA 28: VENTER 2D Y PSOCG

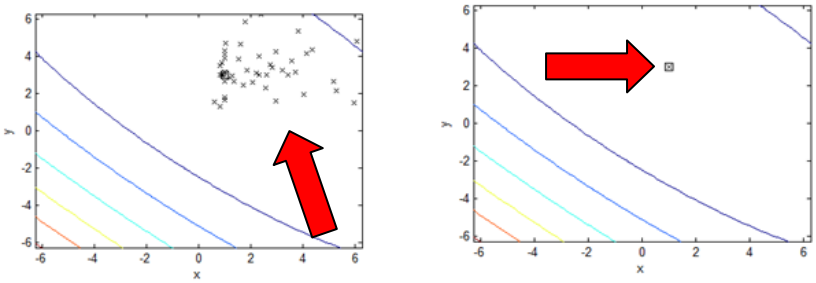

FIGURA 29: BOOTH Y PSOCG

Con tan solo 2 iteraciones, ya se puede ver un patrón de orden de las partículas y una tendencia hacia una región especifica del plano.

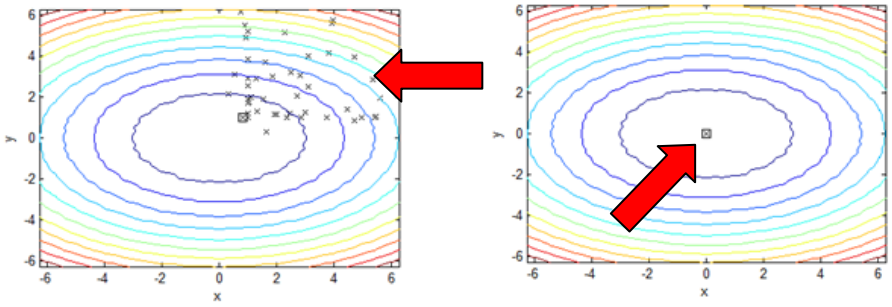

FIGURA 30: B2 Y PSOCG

La convergencia del método es apreciable en la Figura 30, pues todas las partículas convergieron en un mismo punto.

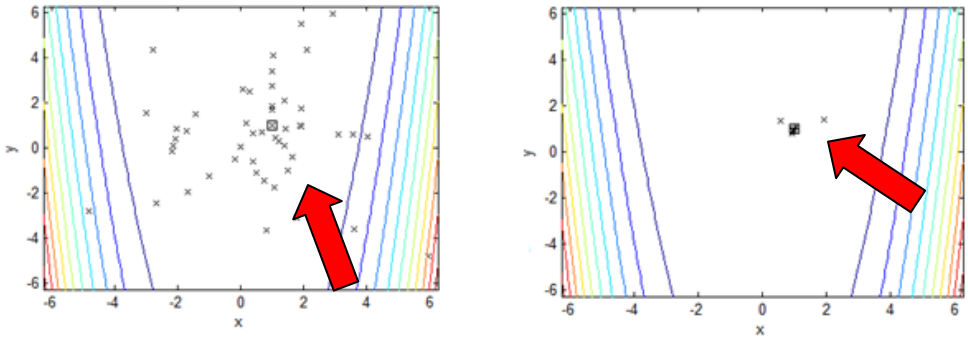

FIGURA 31: ROSENBROCK Y PSOCG

Al igual que en el PSO básico, la función Rosenbrock mostro que algunas partículas que no alcanzaron la mejor ubicación (Figura 31), lo que puede determinar que la función tiene un grado de dificultad superior a las demás. La diferencia es que para el caso de PSOCG el número de partículas "perdidas" son menores que en el PSO básico (Figura 26).

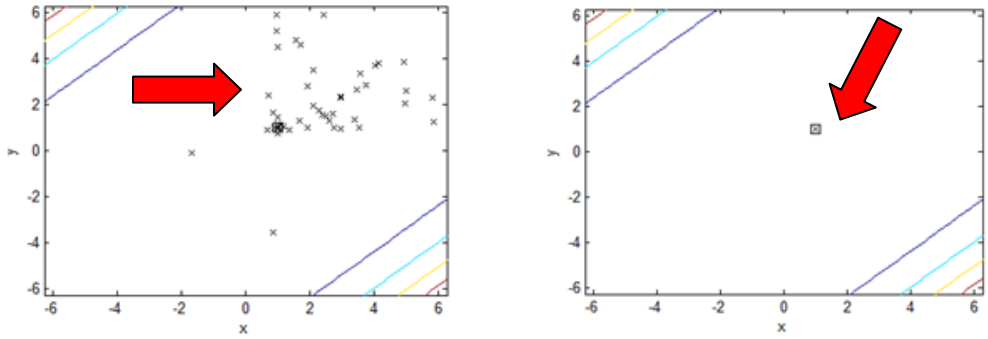

FIGURA 32: TESTE 1 Y PSOCG

Aunque inicialmente se puede captar el mismo tipo de dispersión de las partículas, característico del PSO básico, la respuesta, luego de 35 iteraciones muestra que todos los individuos encontraron una ubicación óptima dentro de la función (unos mejores que otros). Esta es la ventaja que tiene el PSO de convergencia garantizada sobre el PSO básico, lo cual garantiza que para cualquier función

utilizada, se encontrará al menos una respuesta óptima. Esta importante característica la aporta el factor 0.729 agregado a la ecuación principal del PSO.

4.2.3 ARQUITECTURAS

Para comprobar el funcionamiento de las dos arquitecturas, se hicieron las siguientes pruebas sobre las funciones previamente escogidas.

Como primera instancia se varió el número de iteraciones, dejando fijos $w\ y\ Cab$. En este caso se escogió $C_{ab} = 2\ y\ w = 1$. Luego, al igual que en el PSO y PSOCG, se empleó el mismo tipo de pruebas para los híbridos, comparando el funcionamiento de dichas arquitecturas, trabajando con 2 y 35 iteraciones. Obteniendo de esta forma los respectivos valores del error, desviación standard, media y su respectiva representación gráfica.

Para finalizar, se realizó una segunda prueba dejando inmóvil el número de iteraciones y variando $w\ y\ Cab$. Para este caso, se hicieron los mismos análisis de resultados, que en la primera parte.

# ITERACIONES	PSOSX (S)				PSOSX (PE)			
	TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
2	0.4571	0	0	1000	1.1498	0	0	1000
5	1.1659	0	0	1000	2.7788	0	0	1000
8	1.7693	0	0	1000	4.3927	0	0	1000
10	2.2374	0	0	1000	5.4676	0	0	1000
13	2.9699	0	0	1000	7.2291	0	0	1000
15	3.4069	0	0	1000	8.2747	0	0	1000
20	4.5453	0	0	1000	10.9502	0	0	1000
25	5.6339	0	0	1000	13.6619	0	0	1000
30	6.7152	0	0	1000	16.3485	0	0	1000
35	7.946	0	0	1000	19.1094	0	0	1000

TABLA 2: RESULTADOS DE VENTER VARIANDO ITERACIONES

Para el caso de la función Venter de la tabla 2, se puede destacar que la diferencia de los tiempos en las dos arquitecturas es apreciable, sobre todo porque su diferencia es incremental sin factor aparente, a medida que aumentan las iteraciones, aunque las dos arquitecturas llegan a una misma respuesta y se comprueba evaluando la respuesta en la función.

# ITERACIONES	PSOSX (S)				PSOSX (PE)			
	TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
2	0.46334	1	3	0	1.1979	1	3	0
5	1.1841	1	3	0	2.8856	1	3	0
8	1.8795	1	3	0	4.588	1	3	0
10	2.2715	1	3	0	5.6874	1	3	0
13	2.9987	1	3	0	7.4017	1	3	0
15	3.4007	1	3	0	8.5073	1	3	0
20	4.6204	1	3	0	11.1883	1	3	0
25	5.4877	1	3	0	13.8046	1	3	0
30	6.6111	1	3	0	16.5637	1	3	0
35	7.8377	1	3	0	19.5208	1	3	0

TABLA 3: RESULTADOS DE BOOTH VARIANDO ITERACIONES

Se puede apreciar que incluso para pocas iteraciones, el mejor global de cada arquitectura muestra la respuesta exacta de la función. Esto se debe a que el optimo de esta función es fácilmente diferenciable, además que no hay mínimos o máximos locales cerca al origen que interfieran en el proceso de búsqueda, porque la ubicación inicial de las partículas es cercana al origen.

# ITER	PSOSX (S)				PSOSX (PE)			
	TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
2	0.47176	0.61862	-2.6303e-005	0.41293	1.2248	-0.039404	0.1006	0.32153
5	1.1793	8.8823e-006	-8.8456e-006	3.7578e-009	2.7884	0.0056341	0.083143	0.21345
8	1.8799	2.3539e-005	2.3595e-005	2.6633e-008	4.5392	0.024171	-0.0019023	0.0084568
10	2.2866	-2.3508e-006	2.9773e-005	2.9849e-008	5.7071	-0.0068251	0.020625	0.014878
13	2.8555	0.0017562	0.00019111	4.5403e-005	7.1694	0.0032109	0.0067998	0.0016996
15	3.1706	3.0738e-005	8.6749e-006	1.6061e-008	8.2529	0.0029464	0.0017079	0.0002223
20	4.3101	2.6926e-005	-1.6651e-005	1.9696e-008	10.5483	0.0015645	0.00060974	4.7547e-005
25	5.3686	-0.00015889	3.336e-005	3.9899e-007	13.4602	-0.00026084	4.9861e-005	1.058e-006
30	6.5237	3.9557e-005	3.3261e-005	5.9567e-008	15.9807	-6.7007e-005	-3.3835e-005	1.0276e-007
35	7.3025	-7.6851e-006	-2.3671e-006	1.0341e-009	18.1774	2.4104e-005	1.6293e-005	1.7237e-008

Tabla 4: RESULTADOS DE B2 VARIANDO ITERACIONES

Para el caso particular de la función B2, en la tabla 4 se puede apreciar más fácilmente la forma en cómo las iteraciones hacen la diferencia en la respuesta final. Esto ocurre porque el grado de dificultad de esta función es mucho mayor que las demás. En la figura 20 se ve como hay una gran cantidad de máximos y mínimos locales, lo cual hace más difícil la búsqueda del óptimo.

La tabla 4 muestra que la calidad de la respuesta se incrementa a medida que las iteraciones aumentan, aunque siendo poco rigurosos, después de 5 iteraciones, la respuesta es prácticamente cero. Pero un análisis más minucioso demuestra que

la diferencia entre respuestas, para el caso del PSOSX(S), son considerables, aunque tengan valores muy pequeños:

$$\frac{2.6633e^{-8}}{3.5778e^{-9}} \approx 7.44\ veces; \quad \frac{4.5403e^{-5}}{1.6061e^{-8}} \approx 2.827e^{3} veces \quad \frac{5.9567e^{-8}}{1.0341e^{-9}} \approx 57.603\ veces$$

Como lo muestran los cálculos anteriores de la comparación de algunas respuestas del PSOSX(S), es evidente que el número de iteraciones es un factor crucial en la calidad de la respuesta.

En el PSOSX(PE) se presenta el mismo fenómeno, como lo indican los cálculos a continuación:

$$\frac{0.21345}{0.0084568} \approx 25.24\ veces; \quad \frac{0.0016996}{0.002223} \approx 7.645 veces; \quad \frac{1.0276e^{-7}}{1.7237e^{-8}} \approx 5.962\ veces$$

La diferencia entre las respuestas es grande aunque se trate de valores muy pequeños.

Comparando los resultados expuestos en la tabla 4, se puede concluir también que el PSOSX(S) puede arrojar respuestas de mejor calidad que el PSOSX(PE) en casi la mitad del tiempo, lo cual marca una diferencia cuando se trata de problemas complejos en donde el tiempo de computo y el uso de recursos es limitado.

# ITERACIONES	PSOSX (S)				PSOSX (PE)			
	TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
2	0.48321	1	1	0	1.2299	1	1	0
5	1.0179	1	1	0	2.7546	1	1	0
8	1.6582	1	1	0	4.2761	1	1	0
10	2.0318	1	1	0	5.3111	1	1	0
13	2.6729	1	1	0	6.7972	1	1	0
15	3.0748	1	1	0	7.8782	1	1	0
20	4.3114	1	1	0	10.4559	1	1	0
25	5.2027	1	1	0	13.0215	1	1	0
30	6.1511	1	1	0	15.7078	1	1	0
35	8.1402	1	1	0	19.4755	1	1	0

Tabla 5: RESULTADOS DE ROSENBROCK VARIANDO ITERACIONES

Los resultados planteados en la tabla 5, son similares a los encontrados en la tabla 3, debido a que el grado de dificultad de la función Rosenbrock es equivalente al de la función Booth, para lo cual, el comportamiento del las arquitecturas debe ser el mismo.

# ITERACIONES	PSOSX (S)				PSOSX (PE)			
	TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
2	0.41281	1	1	4	1.0918	1	1	4
5	1.0797	1	1	4	2.6583	1	1	4
8	1.6617	1	1	4	4.2242	1	1	4
10	2.0705	1	1	4	5.2724	1	1	4
13	2.691	1	1	4	6.8056	1	1	4
15	3.1139	1	1	4	7.8710	1	1	4
20	4.1452	1	1	4	10.4167	1	1	4
25	5.165	1	1	4	13.0815	1	1	4
30	6.1365	1	1	4	15.5534	1	1	4
35	7.2474	1	1	4	18.1663	1	1	4

Tabla 6: RESULTADOS DE TESTE VARIANDO ITERACIONES

Al igual que en la función Rosenbrock, la tabla 6 muestra un comportamiento bastante parecido de los híbridos, por las mismas razones expuestas anteriormente.

PSOSX (S)		PSOSX (PE)	
# ITERACIONES	ERROR	# ITERACIONES	ERROR
2	0	2	0
5	0	5	0
8	0	8	0
10	0	10	0
13	0	13	0
15	0	15	0
20	0	20	0
25	0	25	0
30	0	30	0
35	0	35	0
DESVIACIÓN	MEDIA	DESVIACIÓN	MEDIA
0 0	0 0	0 0	0 0

TABLA 7: VENTER Y OTROS RESULTADOS

Como ya se había comentado anteriormente y como era de esperarse en la función Venter, el error absoluto de los híbridos es cero, porque en todos los casos, se encontró la respuesta exacta al problema. Los resultados mostrados en

la desviación son cero en ambos casos, debido a que las respuestas no variaron en todas las pruebas expuestas en la tabla 2. Para el caso de la media, es cero por que todos los valores encontrados en las pruebas fueron cero. Cabe aclarar también que en las tablas 7 a la 11, los valores dobles que aparecen en los cuadros son dobles por que los cálculos fueron aplicados al par de respuestas X y Y entregadas por los híbridos.

PSOSX (S)		PSOSX (PE)	
# ITERACIONES	**ERROR**	**# ITERACIONES**	**ERROR**
2	0	2	0
5	0	5	0
8	0	8	0
10	0	10	0
13	0	13	0
15	0	15	0
20	0	20	0
25	0	25	0
30	0	30	0
35	0	35	0
DESVIACIÓN	**MEDIA**	**DESVIACIÓN**	**MEDIA**
0 0	1 3	0 0	1 3

TABLA 8: BOOTH Y OTROS RESULTADOS

Como se había mencionado antes, los valores dobles de la media y la desviación standard son debidos a que se están haciendo los cálculos para las respuestas X y Y de cada función. Para este caso la media tiene el valor de 1 para X y 3 para Y, que son las coordenadas donde se encuentra el optimo de la función.

PSOSX (S)		PSOSX (PE)	
# ITERACIONES	**ERROR**	**# ITERACIONES**	**ERROR**
2	0.61862	2	0.10804
5	1.2536e-005	5	0.083334
8	3.3329e-005	8	0.024246
10	2.9866e-005	10	0.021725
13	0.0017666	13	0.0075198
15	3.1939e-005	15	0.0034056
20	3.1659e-005	20	0.0016791
25	0.00016235	25	0.00026556
30	5.1682e-005	30	7.5065e-005
35	8.0414e-006	35	2.9094e-005

DESVIACIÓN	MEDIA	DESVIACIÓN	MEDIA
0.195571697e-005 6.1697e-005	0.0620346561e-005 2.6561e-005	0.015736 0.038055	-0.00090059 0.021162

TABLA 9: B2 Y OTROS RESULTADOS

Para la función B2, los resultados son más evidentes, por que como se ha expuesto anteriormente, el grado de dificultad de esta función permite que las respuestas varíen notablemente a medida que las iteraciones aumentan. Para los resultados del error expuestos en la tabla 9, se puede observar que el error absoluto es inversamente proporcional al número de iteraciones para ambos híbridos, pero es evidente que el error absoluto en el caso del PSOSX(PE) es mayor hasta 25 iteraciones, de ahí en adelante los valores son similares a los del PSOSX(S).

Los valores de desviación standard y media de cada arquitectura muestran valores cercanos a cero, debido a que el óptimo de la función se encuentra en el origen, aunque al igual que en análisis posteriores, los resultados del PSOSX(S) son superiores en calidad a los del PSOSX(PE).

PSOSX (S)		PSOSX (PE)	
# ITERACIONES	ERROR	# ITERACIONES	ERROR
2	0	2	0
5	0	5	0
8	0	8	0
10	0	10	0
13	0	13	0
15	0	15	0
20	0	20	0
25	0	25	0
30	0	30	0
35	0	35	0
DESVIACIÓN	MEDIA	DESVIACIÓN	MEDIA
0 0	1 1	0 0	1 1

TABLA 10: ROSENBROCK Y OTROS RESULTADOS

Los resultados de la tabla 10 son similares a los encontrados en la tabla 7, con la diferencia que para este caso, el óptimo de la función Rosembrock se encuentra en el punto (1,1), de ahí que los valores de la media en ambas arquitecturas tengan ese valor como resultado.

PSOSX (S)		PSOSX (PE)	
# ITERACIONES	**ERROR**	**# ITERACIONES**	**ERROR**
2	0	2	0
5	0	5	0
8	0	8	0
10	0	10	0
13	0	13	0
15	0	15	0
20	0	20	0
25	0	25	0
30	0	30	0
35	0	35	0
DESVIACIÓN	**MEDIA**	**DESVIACIÓN**	**MEDIA**
0 0	1 1	0 0	1 1

TABLA 11: TESTE Y OTROS RESULTADOS

Al no encontrar diferencia en la respuesta ante diferentes iteraciones, los valores de la desviación standard, media y error son los esperados (Tablas 7 - 11). A continuación se precedió a realizar otras pruebas manteniendo el número de iteraciones en 8 y variando los parámetros w y Cab.

W	Cab	PSOSX (S)				PSOSX (PE)			
		TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
0.1	2	1.866	0	0	1000	4.3908	0	0	1000
0.2	2	1.853	0	0	1000	4.5132	0	0	1000
0.3	2	1.8179	0	0	1000	4.3964	0	0	1000
0.4	2	1.8414	0	0	1000	4.3818	0	0	1000
0.5	2	1.859	0	0	1000	4.563	0	0	1000
0.6	2	1.8929	0	0	1000	4.483	0	0	1000
0.7	2	1.8143	0	0	1000	4.4324	0	0	1000
0.8	2	1.8188	0	0	1000	4.5319	0	0	1000
0.9	2	1.8419	0	0	1000	4.3789	0	0	1000
1	2	1.8268	0	0	1000	4.4442	0	0	1000
PSOSX (S)						PSOSX (PE)			
DESVIACIÓN		MEDIA				DESVIACIÓN		MEDIA	
0 0		0 0				0 0		0 0	

TABLA 12: RESULTADOS DE VENTER VARIANDO PARÁMETROS

Para este tipo de prueba, la tabla 12 se mantuvo el valor de $Cab = 2$ y varió el parámetro w, pero la respueta se encontró igual de rápido que en pruebas anteriores en la función Venter, aun así, se puede observar una pequeña variación en los tiempo de cálculo y para el caso del PSOSX(PE), el tiempo es notablemente diferente. Esto era de esperarse debido a que esta arquitectura necesita realizar una etapa más para hallar el resultado. Los valores de media y desviación standard no variaron en lo absoluto.

W	Cab	PSOSX (S)				PSOSX (PE)			
		TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
1	1.5	1.881	1	3	0	4.5753	1	3	0
1	1.6	1.8396	1	3	0	4.5898	1	3	0
1	1.7	1.8534	1	3	0	4.5282	1	3	0
1	1.8	1.8531	1	3	0	4.5013	1	3	0
1	1.9	1.8381	1	3	0	4.5615	1	3	0
1	2	1.8045	1	3	0	4.562	1	3	0
1	2.1	1.8722	1	3	0	4.5245	1	3	0
1	2.2	1.8689	1	3	0	4.6174	1	3	0
1	2.3	1.92	1	3	0	4.543	1	3	0
1	2.4	1.9003	1	3	0	4.5143	1	3	0

PSOSX (S)		PSOSX (PE)	
DESVIACIÓN	MEDIA	DESVIACIÓN	MEDIA
0 0	1 3	0 0	1 3

TABLA 13: RESULTADOS DE BOOTH VARIANDO PARÁMETROS

En la tabla 13, muestra las pruebas realizadas cambiando los valores Cab, mientras que $w = 1$ en todo momento. Al igual que en el análisis de la tabla 12, los valores de tiempo cambiaron muy poco, y de la misma forma, las respuestas X, Y y Z, no cambiaron.

W	Cab	PSOSX (S)				PSOSX (PE)			
		TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
0.1	1.5	1.7515	0.61858	-1.5466e-005	0.41293	4.2912	-0.00383	0.00050971	0.00021882
0.2	1.6	1.7562	2.3198e-005	-1.8325e-005	1.8985e-008	4.3631	0.0089566	0.020273	0.014881
0.3	1.7	1.7938	1.2505e-005	2.4923e-005	2.31e-008	4.2861	6.4902e-005	0.0001409	7.27e-007
0.4	1.8	1.8037	3.2306e-005	-2.1632e-005	3.0665e-008	4.3409	0.00063401	0.0069649	0.0016339
0.5	1.9	1.7825	5.2419e-005	2.3417e-005	5.7774e-008	4.3589	0.005419	0.0042509	0.0010273
0.6	2	1.8517	-4.705e-006	0.00018511	1.1511e-006	4.4859	0.001545	0.0017913	0.00014195
0.7	2.1	1.8764	-1.7132e-005	-9.262e-006	7.0852e-009	4.5543	-0.00084475	0.011096	0.0041386
0.8	2.2	1.8578	4.0882e-005	0.00039418	5.242e-006	4.5201	-0.0092745	0.014328	0.0081081
0.9	2.3	1.8917	-4.2655e-005	-2.4653e-007	2.6063e-008	4.5518	-0.019995	-0.0088223	0.0083223
1	2.4	1.8456	-2.945e-005	-1.4009e-005	1.9014e-008	4.4793	-0.0043664	-0.024189	0.01978

PSOSX (S)		PSOSX (PE)	
DESVIACION	MEDIA	DESVIACION	MEDIA
0.19561 0.00013418	0.061865 5.4869e-005	0.0080663 0.012486	-0.0021691 0.0026343

TABLA 14: RESULTADOS DE B2 VARIANDO PARAMETROS

Los datos plasmados en la tabla 14 muestran cómo afecta el resultado de las pruebas al variar ambos parámetros (W y Cab). Aunque los valores de Z son muy cercanos a cero, se puede ver que para algunos parámetros, la respuesta cambio notablemente en cada hibrido en formas diferentes.

En el PSOSX(S) se puede observar que para W=0.6 y Cab=2, el valor de Z cambió notablemente con respeto a los demás valores, al igual que cuando W=0.8, Cab=2.2. Estos registros son ±20 veces más grandes que los demás, lo que hace una diferencia bastante notable cuando la calidad de la respuesta es importante. Analizando ahora el PSOSX(PE), es bastante evidente que para W=0.3 y Cab=1.7, el valor de Z se hizo mucho más exacto (del orden de 10^{-7}).

Estos resultados muestran que para cada arquitectura, los parámetros de partida, no deben ser necesariamente los mismos, pues gracias a éstos, cada hibrido se hace más eficiente buscando respuesta al problema. Esta "eficiencia" se basa en el tiempo de computo.

W	Cab	PSOSX (S)				PSOSX (PE)			
		TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
0.1	2.5	1.7406	1	1	0	4.3424	1	1	0
0.2	2.4	1.7712	1	1	0	4.3137	1	1	0
0.3	2.3	1.8089	1	1	0	4.3392	1	1	0
0.4	2.2	1.8149	1	1	0	4.4054	1	1	0
0.5	2.1	1.7794	1	1	0	4.3941	1	1	0
0.6	2.0	1.8251	1	1	0	4.5361	1	1	0
0.7	1.9	1.8126	1	1	0	4.2817	1	1	0
0.8	1.8	1.8449	1	1	0	4.359	1	1	0
0.9	1.7	1.7433	1	1	0	4.3686	1	1	0
1	1.6	1.8065	1	1	0	4.3678	1	1	0

PSOSX (S)		PSOSX (PE)	
DESVIACIÓN	MEDIA	DESVIACIÓN	MEDIA
0 0	1 1	0 0	1 1

Tabla 15: RESULTADOS DE ROSENBROCK VARIANDO PARÁMETROS

La tabla 15 revela que para este tipo de función, la variación de los parámetros no muestra cambios en la respuesta, además, los tiempos de cálculo varían muy poco entre ellos, como para concluir algo adicional.

W	Cab	PSOSX (S)				PSOSX (PE)			
		TIEMPO	X	Y	Z	TIEMPO	X	Y	Z
0.2	2.5	1.7402	1	1	4	4.3761	1	1	4
0.4	2.4	1.8079	1	1	4	4.4131	1	1	4
0.6	2.3	1.8051	1	1	4	4.4288	1	1	4
0.8	2.2	1.8	1	1	4	4.4094	1	1	4
1	2.1	1.7973	1	1	4	4.3494	1	1	4
0.1	2.0	1.7842	1	1	4	4.4073	1	1	4
0.3	1.9	1.7019	1	1	4	4.3245	1	1	4
0.5	1.8	1.7601	1	1	4	4.5422	1	1	4
0.7	1.7	1.7836	1	1	4	4.4537	1	1	4
0.9	1.6	1.7544	1	1	4	4.486	1	1	4
PSOSX (S)						PSOSX (PE)			
DESVIACIÓN		MEDIA				DESVIACIÓN		MEDIA	
0 0		1 1				0 0		1 1	

Tabla 16: RESULTADOS DE TESTE VARIANDO PARÁMETROS

De la misma forma que en la tabla 15, la tabla 16 revela que para estos tipos de funciones, la respuesta no tiene variación alguna, variando los parámetros w y Cab.

Según los análisis realizados en las tablas anteriores, se puede observar que el tiempo de computo del PSOSX(PE) y del PSOSX(S) se mantienen constantes a pesar que los parámetros w y Cab cambiaron. Lo que sí es bastante evidente (tabla 14), es que las respuestas se ven afectadas por estas variaciones de parámetros. Lo que conlleva a que los parámetros w y Cab no son igualmente óptimos para ambos híbridos, esto demuestra que para cada arquitectura y problema, los valores de w y Cab iníciales son determinantes a la hora de encontrar una respuesta que cumpla los requerimientos necesarios.

FUNCIÓN VENTER 2D

Para cada función, las gráficas mostradas en el lado izquierdo son los resultados de evaluar cada función con el hibrido PSOSX(S) y a la derecha se muestran las correspondientes al PSOSX(PE).

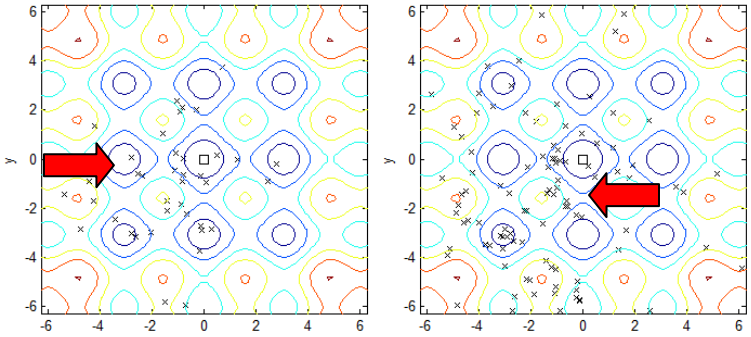

FIGURA 33: VENTER Y PSOSX CON 2 ITERACIONES

Se puede observar la dispersión inicial de las partículas aplicando en cada híbridos, 2 iteraciones. Otra observación importante a realizar, es que en el PSOSX(PE), parece que existieran mas partículas que en el PSOSX(S). Aunque en ambos casos se utilizaron el mismo número de individuos, para el PSOSX(PE), se dejó guardada la última posición de éstos individuos con el fin de ver como se acerca a la respuesta con nuevos parámetros.

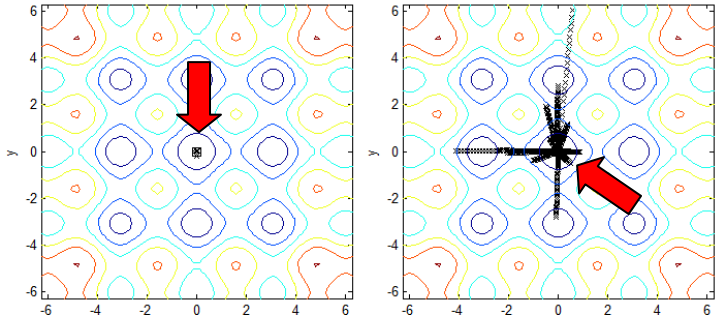

FIGURA 34: VENTER Y PSOSX CON 35 ITERACIONES

Ahora que se utilizaron 35 iteraciones es apreciable como cada arquitectura se acerca a la respuesta ajustando sus datos, tanto la respuesta en el PSOSX(S), como los parámetros iníciales en el PSOSX(PE). En este último, se puede ver el rastro de las partículas acercándose a la respuesta más directamente que al aplicar el PSO básico.

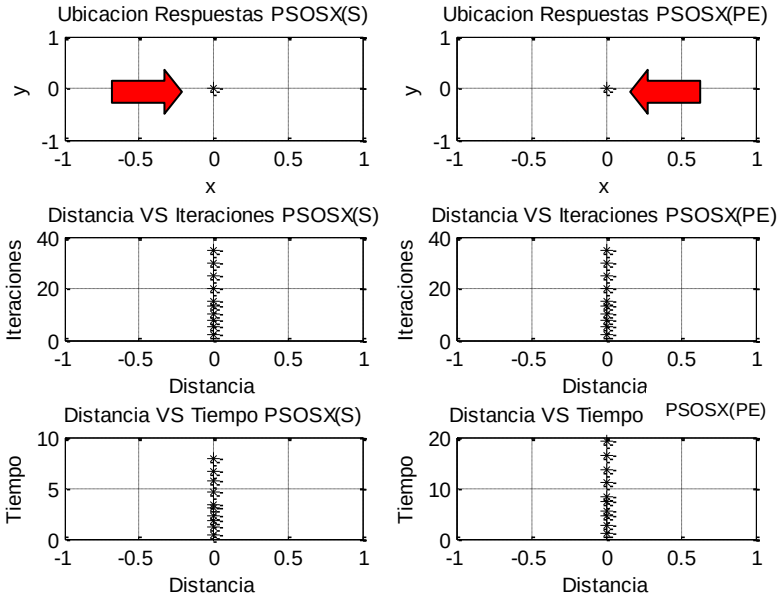

FIGURA 35: VENTER Y OTROS ANÁLISIS

En la Figura 35 se puede observar gráficas que compara las dos arquitecturas, empleadas en la función Venter. En ésta figura se puede destacar la ubicación de las respuestas encontradas por cada hibrido, las cuales están superpuestas todas en el origen, fenómeno explicado en análisis realizados anteriormente. La distancia de la respuesta experimental y la teórica, contra el número de iteraciones, las cuales también se encuentran a la misma distancia y no se visualizan cambios a través del cambio de iteraciones ni del tiempo de cálculo.

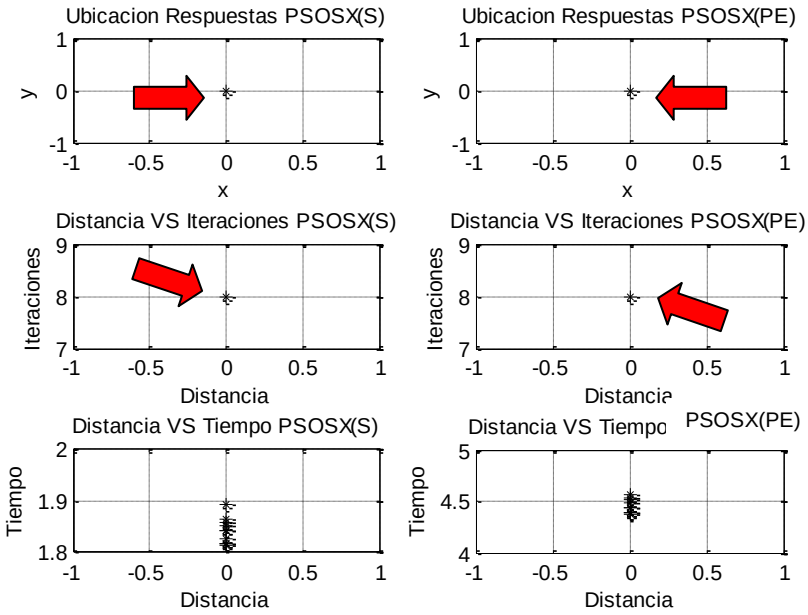

FIGURA 36: VENTER Y ANÁLISIS DE RESULTADOS VIARIANDO PARÁMETROS

Para la figura 36, se comparan los resultados encontrados variando los parámetros iníciales W y Cab, en donde se puede ver que las respuestas no varían mucho, en comparación con las de la figura 30, excepto que para este caso el valor de las iteraciones se estableció en 8, pero la distancia entre el valor teórico y el experimental sigue siendo el mismo, en consecuencia del tipo de función, como se había explicado en otros análisis.

FUNCIÓN BOOTH

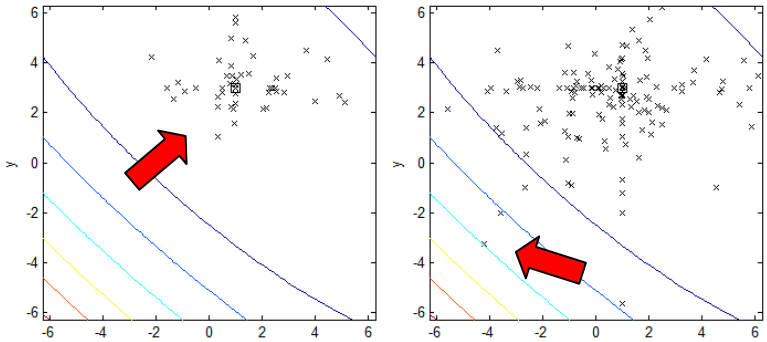

FIGURA 37: BOOTH Y PSOSX CON 2 ITERACIONES

En la figura 37 como en la figura 33, se observa la dispersión de las partículas por el plano, pero es importante revisar que algunas partículas perdieron, al menos momentáneamente el curso del enjambre.

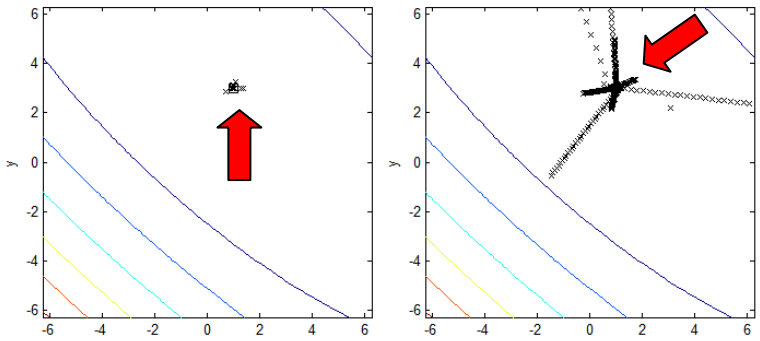

FIGURA 38: BOOTH Y PSOSX CON 35 ITERACIONES

Gracias a que se aumentó el número de iteraciones, en la figura 38 se puede apreciar como las partículas se desplazan hacia la respuesta de la función con mucha precisión. Como era de esperarse, la cantidad de iteraciones es un factor vital en ambas arquitecturas, para lograr así la respuesta acertada.

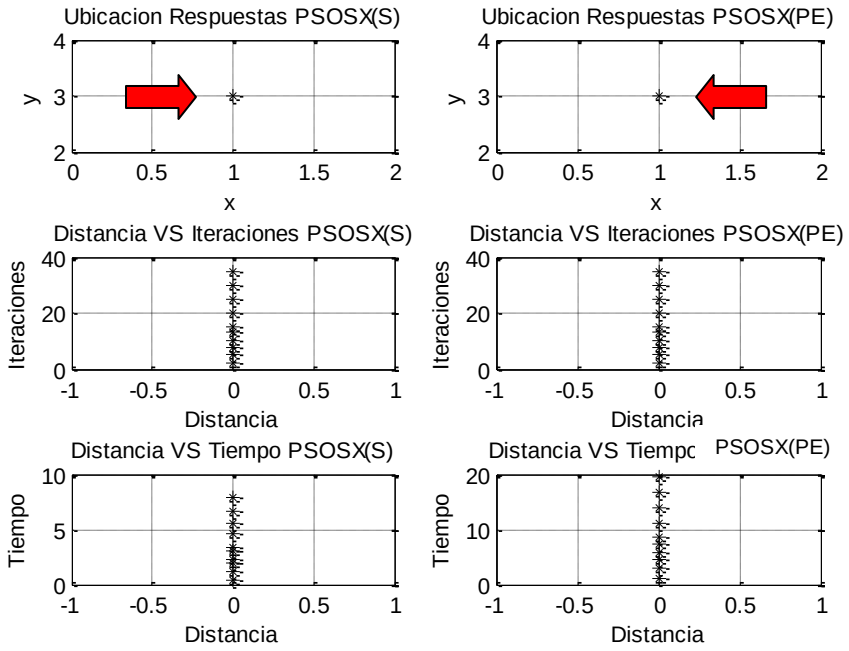

FIGURA 39: BOOTH Y ANÁLISIS DE RESULTADOS VIARIANDO ITERACIONES

En las graficas anteriores (figura 39), no es apreciable un cambio de los resultados, solo variando la cantidad de iteraciones, como se había revisando anteriormente con la función Booth.

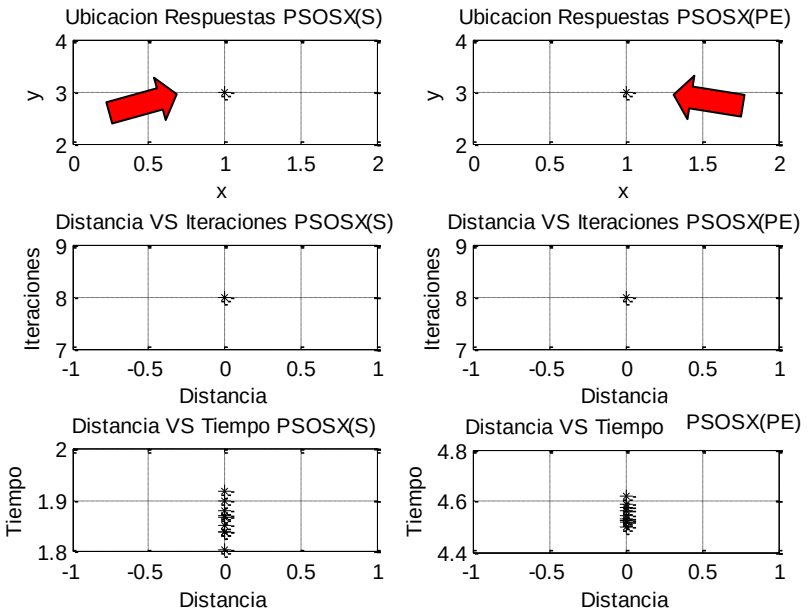

FIGURA 40: BOOTH Y ANÁLISIS DE RESULTADOS VIARIANDO PARÁMETROS

Para el caso en que se variaron los parámetros W y Cab con la función Booth, solo se puede observar que la respuesta se mantiene a lo largo del tiempo de cálculo, pero como se llegó a la respuesta exacta rápidamente, la distancia de todas las pruebas no cambió y se mantuvo en cero

FUNCIÓN B2

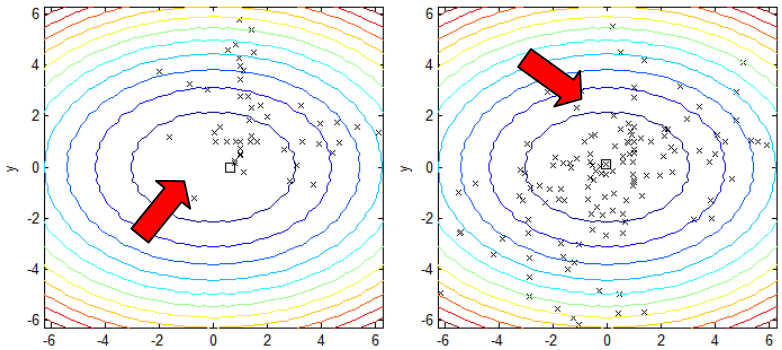

FIGURA 41: B2 Y PSOSX CON 2 ITERACIONES

En el caso de B2, la figura 41 revela que nuevamente la dispersión de los sujetos en el espacio de la respuesta.

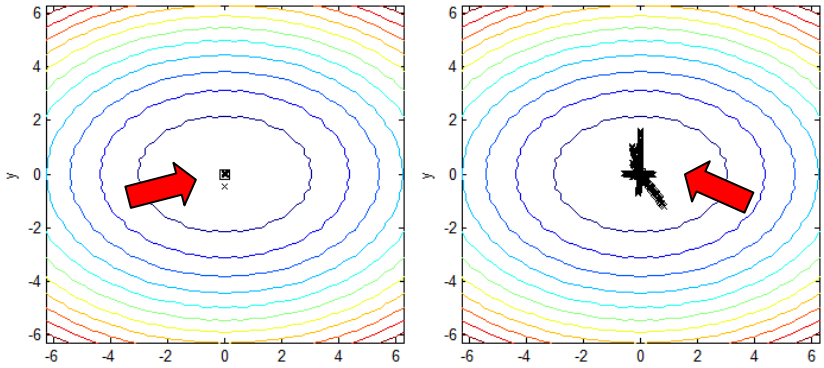

FIGURA 42: B2 Y PSOSX CON 35 ITERACIONES

Y en la figura 42, se encuentra que las partículas se ajustaron espléndidamente al óptimo de la función.

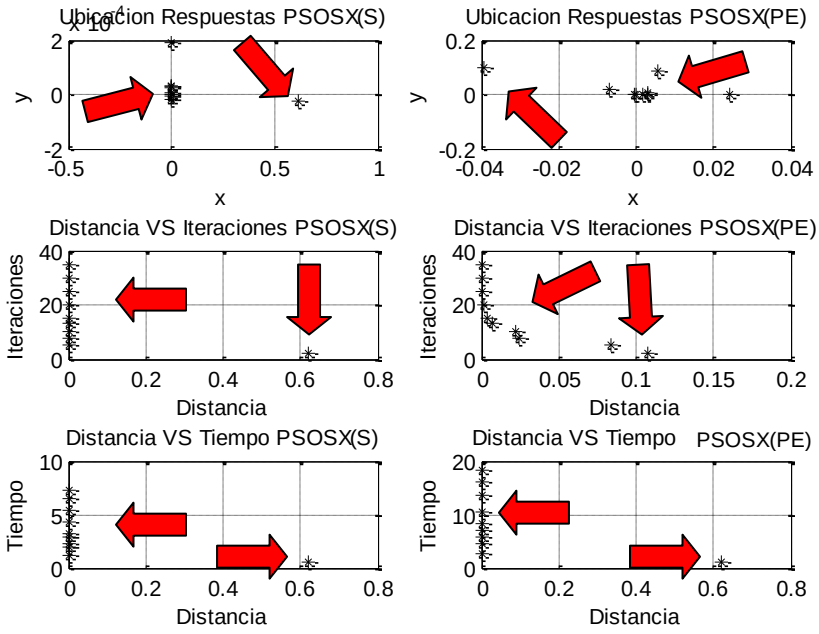

FIGURA 43: B2 Y ANÁLISIS DE RESULTADOS VIARIANDO ITERACIONES

Como se había estudiado anteriormente, la función B2 muestra ciertos cambios cuando se varían las iteraciones, cambios apreciables en la figura 43, que como se veía en el análisis de la tabla 4, las respuestas son muy cercanas a la respuesta, pero sus variaciones son suficientemente grandes para ser apreciadas en las gráficas. De esta manera se explican ciertos puntos separados de los demás, que a pequeña escala se ven más fácilmente.

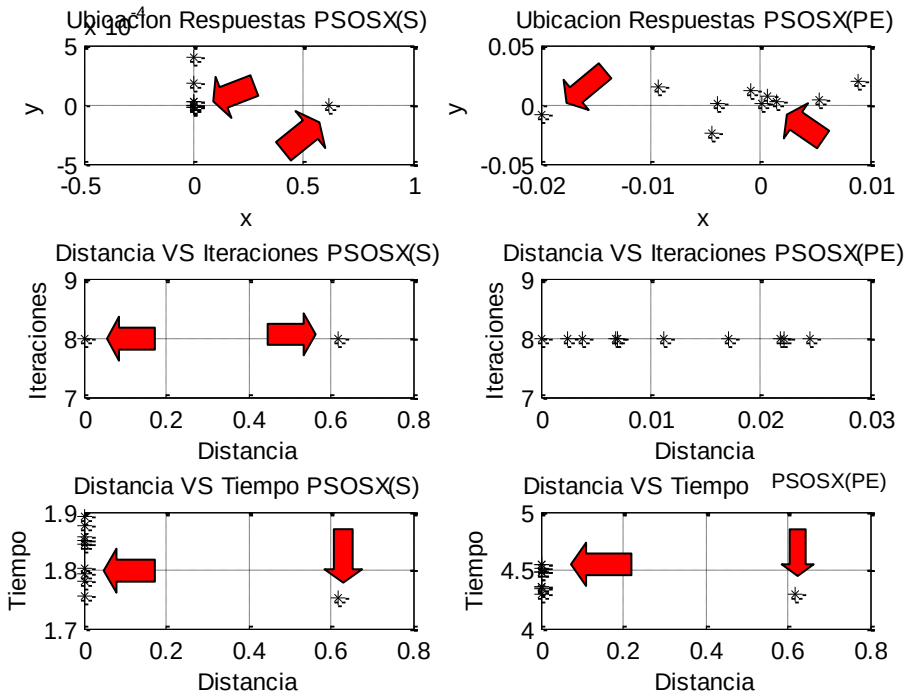

FIGURA 44: B2 Y ANÁLISIS DE RESULTADOS VIARIANDO PARÁMETROS

Ahora, para el caso de la figura 44, se puede corroborar los datos mostrados en la tabla 14, en donde se pudo apreciar ciertos cambios en el comportamiento de la respuesta, a medida que se realizaban cambios en los parámetros iníciales. Las ubicaciones de las respuestas ahora son mucho más apreciables gracias al factor de escala utilizado en la grafica, y vale destacar como en el PSOSX(PE) las respuestas mantienen cierta dispersión, menos apreciable en el PSOSX(S). De la misma forma, como las iteraciones se mantuvieron en 8, las distancias entre los

valores teórico y experimental, se hicieron evidentes a esta escala de muestreo, incluso comparándolas con el tiempo de cálculo.

FUNCIÓN ROSENBROCK

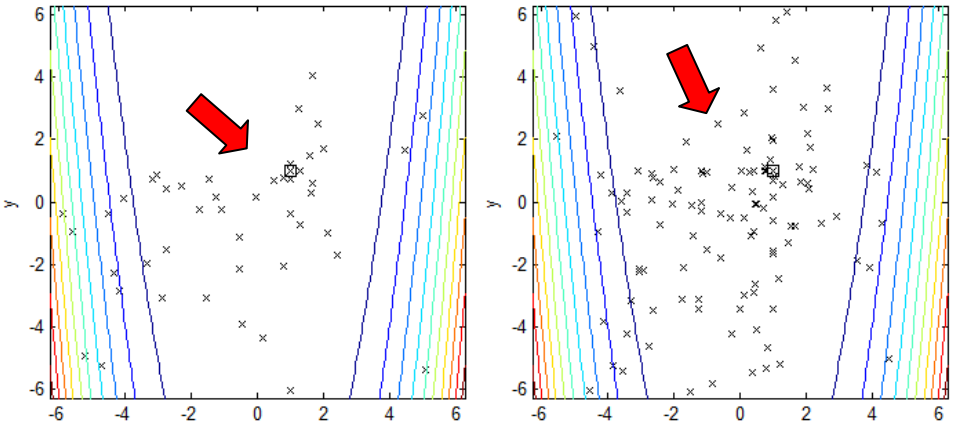

FIGURA 45: ROSENBROCK Y PSOSX CON 2 ITERACIONES

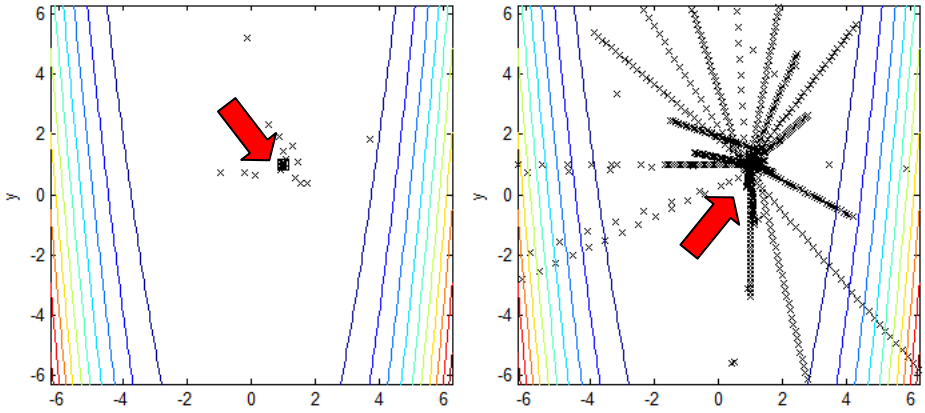

FIGURA 46: ROSENBROCK Y PSOSX CON 35 ITERACIONES

En las figuras 45 y 46 se muestra la operación del PSOSX(S) y PSOSX(PE), tanto con 2 iteraciones, como con 35. Para el último caso, el PSOSX(PE) muestra como después de ajustar los parámetros Cab y W, las partículas se dirigieron más eficientemente hacia el optimo de la función, aunque aplicando mas pasos en el proceso.

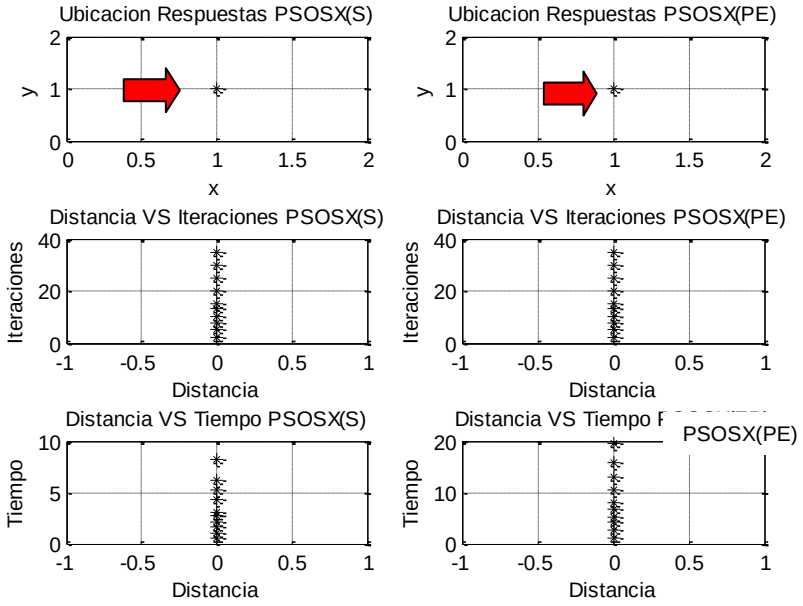

FIGURA 47: ROSENBROCK Y ANÁLISIS DE RESULTADOS VIARIANDO ITERACIONES

Como en la figura 39 con la función Booth, la figura 47 no muestra cambios por el tipo de función utilizado, ocurre el mismo fenómeno antes explicado.

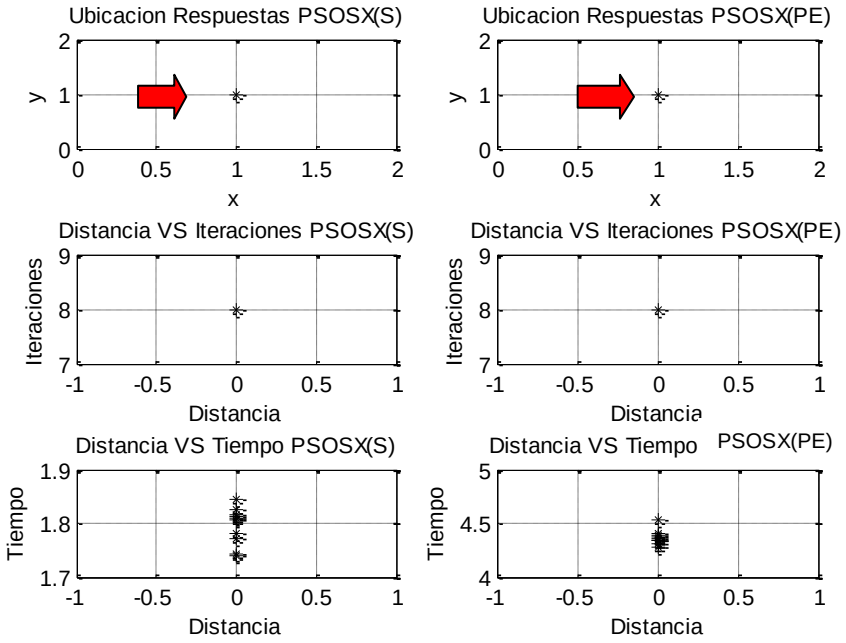

FIGURA 48: ROSENBROCK Y ANÁLISIS DE RESULTADOS VIARIANDO PARÁMETROS

La figura 48 no muestra cambios apreciables cambiando parámetros iníciales y estableciendo las iteraciones en 8. En ambos híbridos la respuesta se halla casi de inmediato.

FUNCIÓN TESTE 1

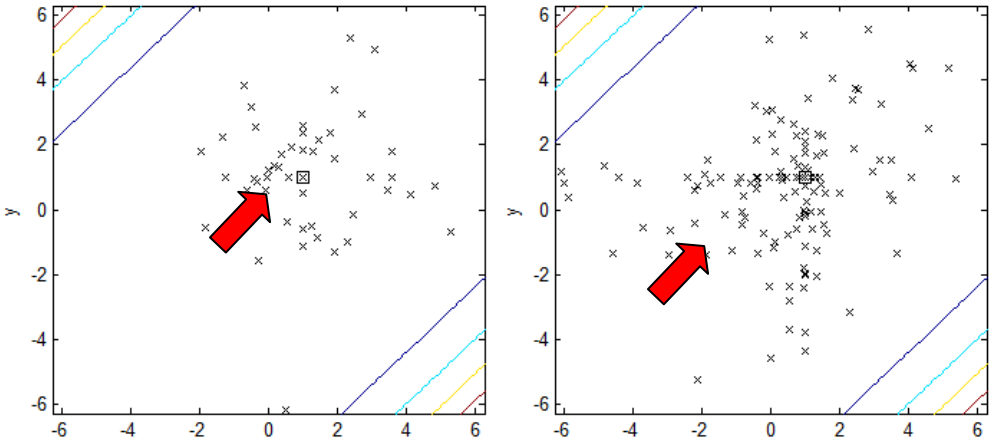

FIGURA 49: TESTE Y PSOSX CON 2 ITERACIONES

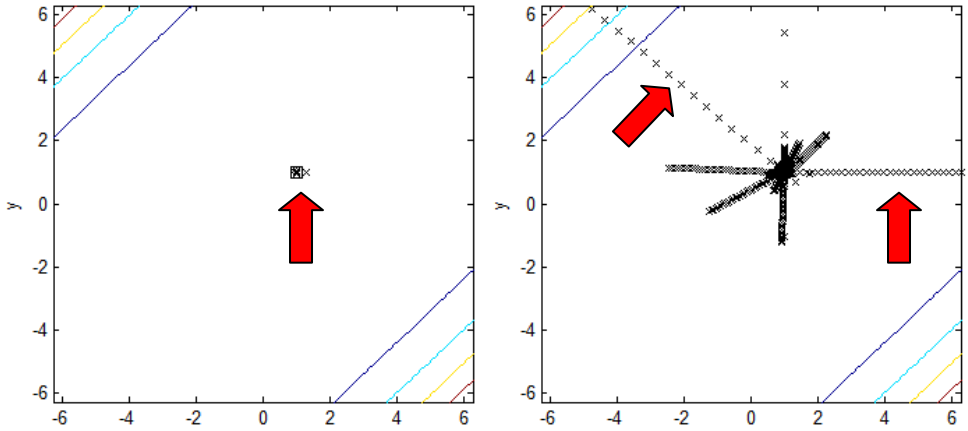

FIGURA 50: TESTE Y PSOSX CON 35 ITERACIONES

Al revisar la figura 50, se puede observar que las partículas que se encontraban lejos del optimo (para PSOSX(PE)), se ajustaron rápidamente rumbo al optimo de la función.

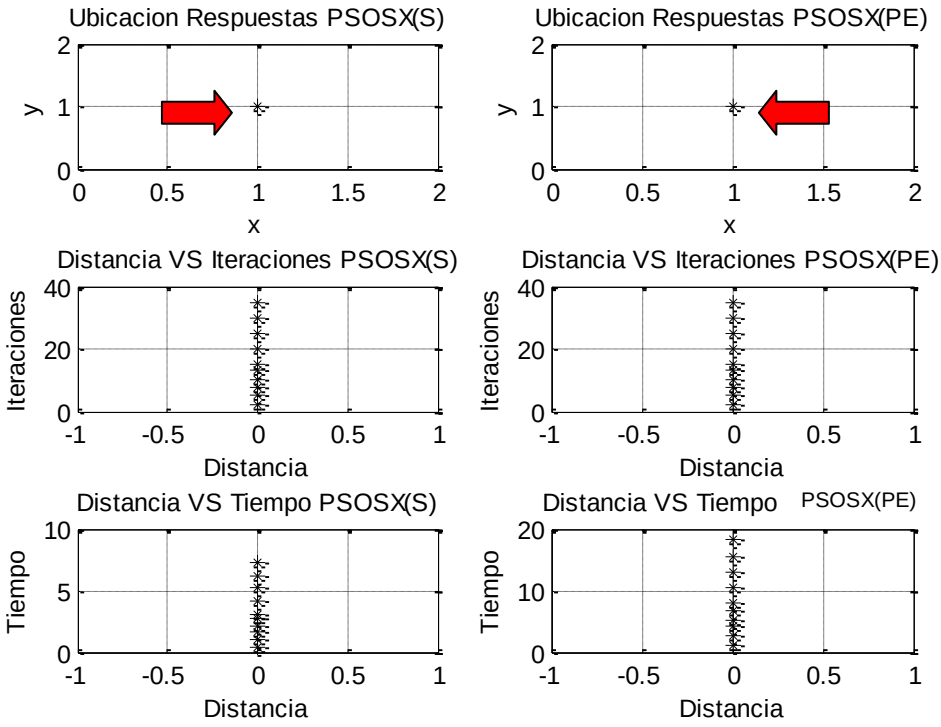

FIGURA 51: TESTE Y ANÁLISIS DE RESULTADOS VIARIANDO ITERACIONES

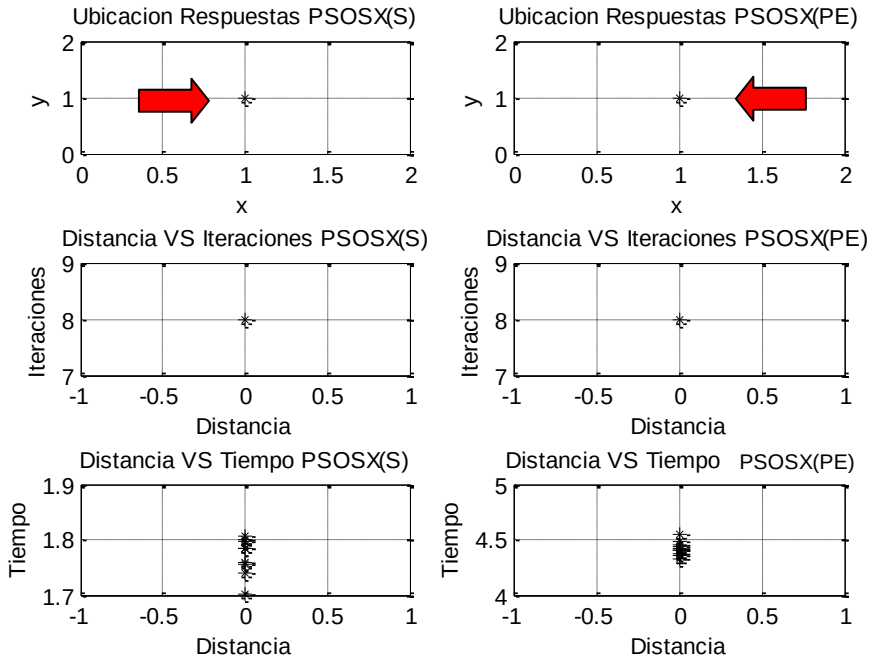

FIGURA 52: TESTE Y ANÁLISIS DE RESULTADOS VIARIANDO PARÁMETROS

Las figuras 51 y 52 muestran el mismo comportamiento de las figuras 37 y 39, para lo cual, las respuestas no cambiaron a pesar de las variaciones aplicadas a las iteraciones y parámetros de inicio, para la función Teste.

Luego de aplicar las pruebas planteadas, se pudo observar diferentes comportamientos concluyentes, que permiten mejorar aun más el comportamiento de cada hibrido. Como es el caso de los parámetros iníciales C_{ab} y w, los cuales son vitales para correcta operación de los métodos. Gracias a la experiencia y los resultados de las pruebas, se escogieron los valores $C_{ab} = 2$ y $w = 1$, para los cuales los algoritmos mostraron cierta afinidad y exactitud.

Si se habla ahora de la calidad de las respuestas finales, cabe resaltar que el PSOSX(S) mostró una gran diferencia, en comparación con el PSOSX(PE). En el PSOSX(S) se puede alcanzar una exactitud mayor en el óptimo de la función utilizada, usando menos procesos, pues solo hay que utilizar una etapa de PSO básico y una de Simplex, a diferencia del PSOSX(PE), que tiene que ejecutar una etapa de PSO básico, una de Simplex para ajustar w y C, y luego volver a operar el PSO básico de nuevo. Lo que se traduce en menor tiempo de cómputo y menos recursos utilizados por el sistema. El nivel de dificultad de cada función hace una diferencia apreciable en el tiempo de computo y cantidad de iteraciones necesarias para alcanzar la calidad deseada en el respuesta, además que no puede haber un único algoritmo que sirva en todos los casos a evaluar, por eso vale la pena constatar la respuesta encontrada con los híbridos, incluso empleando el PSOCG.

5. CONCLUSIONES

Los resultados obtenidos de las pruebas realizadas con la herramienta implementada, revelan que los algoritmos propuestos mejoran el funcionamiento del PSO básico, generando soluciones con una calidad admisible.

La distribución inicial de las partículas es un factor que tiene gran influencia en el tiempo de cómputo en Matlab, sobre todo cuando hay más de 3 dimensiones en estudio. Luego de muchas pruebas se concluyó que era más eficiente ubicar las partículas a lo largo de cada eje, lo cual dio como resultado un menor tiempo de cálculo, en vez de utilizar el tradicional orden en forma de matriz.

El número de iteraciones cumple un papel importante a la hora de buscar exactitud en la respuesta y de esta forma se concluye que a medida que crecen las iteraciones, se obtiene mayor precisión, pero también mayor tiempo de computo; hay que tener muy en cuenta que para ambas arquitecturas, la duración de cada iteración es diferente debido a la cantidad de procesos necesarios en cada una.

Partiendo de las pruebas realizadas, se pudo observar que los parámetros iníciales ($w\ y\ Cab$), influyen notablemente en el proceso de búsqueda del óptimo de la función; como en cualquier método numérico, pero también se aclaró que existen mejores parámetros para cada hibrido por separado, es decir, aunque al utilizar cualquier parámetro dentro de los rangos establecidos funcione, hay mejores valores para cada hibrido y para cada problema.

El método Simplex depende directamente de las iteraciones del PSO, debido a que el fminsearch necesita puntos de partida cercanos a la respuesta; por

consiguiente, si las partículas no se acercan lo suficiente al óptimo, el método simplex no converge, sea para el PSOSX(S) como para el PSOSX(PE).

Según los experimentos realizados, se pudo observar que el PSOSX(S) puede encontrar más rápidamente respuestas con mejor exactitud que el PSOSX(PE), siempre y cuando se realicen las iteraciones necesarias para que el simplex converja eficientemente.

El aumento excesivo del número de partículas por cada dimensión, trae como consecuencia el aumento de tiempo de cálculo de las posiciones iníciales; haciendo ineficiente al programa, es decir, se gasta más tiempo ubicando las partículas en el espacio definido, que realizando los respectivos análisis. Esto puede ocurrir debido a que la posibilidad de encontrar la respuesta rápidamente mejora si hay más partículas en el espacio y se necesitarían menos iteraciones. La idea es no abusar de este parámetro.

El PSOCG, muestra una diferencia considerable respecto al PSO en la búsqueda de la respuesta, ya que en éste convergen rápidamente la mayoría de las partículas, aunque el valor de la respuesta final no sea muy diferente de la entregada por el PSO básico. Cabe aclarar que esta prueba no estaba dentro de los alcances del proyecto, pero se realizó ya que su implementación permite observar diferentes comportamientos del PSO.
Mediante el planteamiento de la nueva arquitectura PSOSX de evolución paramétrica, se concluyó que es una herramienta efectiva y aplicable a proyectos de ingeniería.

El PSOSX(PE) tiene una gran ventaja sobre el PSOSX(S), en el sentido que si se introducen parámetros iníciales inapropiados, el simplex puede corregir ese error y así aumentar la eficiencia del PSO básico, evitando que diverja. Por supuesto que esto trae consecuencias como respuestas ineficientes, o tiempos de cómputo más prolongados.

Aunque teóricamente las dos arquitecturas funcionan en todos los problemas, hay casos para los cuales hay que tener en cuenta otros factores para su uso. 1. Problemas donde la respuesta no es un punto, si no un área, para lo cual una búsqueda puntual no es efectiva. 2. Funciones discontinuas compuestas las cuales están definidas por regiones. 3. Casos donde el problema tiene más de un máximo o mínimo con el mismo valor y el usuario debe elegir la respuesta más conveniente. Para todos estos problemas en mención, el usuario debe integrar condiciones extra de operación para encontrar una respuesta satisfactoria.

En base a las múltiples pruebas realizadas se concluyó que el criterio de parada más indicado para la herramienta fue la comparación entre los global best consecutivos, en donde la tolerancia entre ellos debe ser menor del 10%. Esto debido a que la respuesta no se ve afectada considerablemente. Aunque existen otros criterios que se descartaron, como el número de iteraciones el cual malgastaba tiempo, o como asignar una velocidad final, que no sirve de nada si no se tiene una idea de la respuesta objetivo, entre otros.

Con la implementación del trabajo de grado se contribuyó resolviendo dos problemas comunes en la búsqueda de soluciones a problemas complejos, situaciones en que se necesitan respuestas inmediatas con tolerancias de errores admisibles o en casos donde “el tiempo no es un factor primordial”, como en los proyectos civiles donde la respuesta no tiene que ser inmediata, pero la exactitud es una prioridad, o en aéreas donde se pueden sacrificar algunos segundos con tal de encontrar mayor exactitud en la respuesta.

6. RECOMENDACIONES

Implementar nuevos híbridos de múltiples etapas, con el fin de comparar la eficiencia obtenida con diferentes condiciones de operación y de esta manera buscar nuevas alternativas de trabajo ante problemas complejos.

Aplicar los híbridos imprentados en problemas reales de ingeniería.

Para casos generales, se recomienda emplear alrededor de 10 partículas por dimensión para no malgastar tiempo de cómputo ubicando posiciones innecesariamente.

Se recomienda reducir la función a su mínima expresión, para no introducir más vectores de estado sin necesidad, lo que traduce un mayor tiempo de cálculo y recursos empleados del sistema.

Aunque Matlab es una herramienta muy eficiente y practica, está ligada directamente a los recursos dispuestos por Windows. Se recomienda hacer nuevas pruebas en Matlab sobre Linux para comparar tiempos de cálculo.

Comprobar siempre la veracidad de la respuesta con otros métodos, para tener cuenta el teorema del “No free launch”; no puede haber un método único que funcione para todas las funciones.

7. BIBLIOGRAFÍA

1. **HERRERA, Francisco.** "Introducción a los Algoritmos", ESPAÑA. Disponible en: *http://sci2s.ugr.es/seminars/5/Int-Metaheuristicas-UIA-Baeza-2006.pdf* (Fecha de revisión: Nov 20/08).

2. "Algorítmica", 2006 – 2007. Disponible en: *http://sci2s.ugr.es/docencia/algoritmica/Tema01-Metaheuristicas-06-07.pdf* (Fecha de revisión: Nov 20/08).

3. **BATISTA MELIÁN, Belén y Moreno Pérez, José Andrés. "**Metaheurísticas para la planificación logística", Febrero de 2005, Grupo de Computación Inteligente. Universidad de La Laguna Disponible en: ***http://webpages.ull.es/users/jamoreno/www/talks/TRANSNOVA04M.pdf*** (Fecha de revisión: Nov 22/08).

4. "Algoritmos evolutivos", 2007. Centro de Cálculo, Instituto de Computación,Facultad de Ingeniería, Universidad de la República, Uruguay. Disponible en: ***http://www.fing.edu.uy/inco/cursos/geneticos/ae/2007/Clases/clase1.pdf*** (Fecha de revisión: Dic 4/08).

5. **COELLO COELLO, Carlos.** "Introducción a la Computación Evolutiva", Departamento de Computación, Cinvestav-ipn, Col. San Pedro Zacatenco, México. Disponible en: ***http://delta.cs.cinvestav.mx/~ccoello/compevol/clase16-2008.pdf*** (Fecha de revisión: Nov 26/08).

6. **COTTA PORRAS, Carlos. "**Teoría de la Computación Evolutiva", 8 de Agosto, 2001. Dpto. de Lenguajes y Ciencias de la Computación. Disponible en: ***http://www.lcc.uma.es/~ccottap/slides/El_Escorial_2001/theory_archivos/frame.htm*** (Fecha de revisión: Dic 3/08).

7. **ROMANO, Carlos Andrés; FRAMIÑÁN TORRES, José Manuel y PASTOR MORENO, Rafael.** "Optimización mediante Cúmulos de Partículas del problema de secuenciación CONWIP", Octubre 2004. Disponible en: http://personales.upv.es/~candres/Presentacion%20CONWIP%20PSO%20SEIO.pdf (Fecha de revisión: Dic 3/08).

8. "Bioinformática" 2007-2008. Tema 4. Optimización Basada en Nubes de Partículas (Particle Swarm) Disponible en: http://sci2s.ugr.es/docencia/bioinformatica/Tema%204%20PSO%2007-08.pdf (Fecha de revisión: Dic 1/08).

9. **ARAGÓNY, Victoria; CAGNINAY, Leticia y ESQUIVELY, Susana. "**Metaheurísticas basadas en Inteligencia Computacional Aplicadas a la Resolución de Problemas de Optimización Restringidos". Universidad Nacional de San Luis, Argentina. Disponible en: http://ficcte.unimoron.edu.ar/wicc/Trabajos/I%20-%20asi/639-wicc06_lop.pdf (Fecha de revisión: Dic 3/08).

10. **GAO , Liang; ZHOU, Chi y ZAN, Kun. "**Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. Particle Swarm Optimization forSimultaneous Optimization of Design and Machining Tolerances. pp. 321 – 330.

11. **HUANG, Tony y SANAGAVARAPU MOHAN, Ananda. "**A Microparticle Swarm Optimizer for the Reconstruction of Microwave Images", Marzo 2007. IEEE transactions on antennas and propagation, vol. 55, no. 3.

12. **WROBEL, Rafal y MELLOR, Phil. "**Particle Swarm Optimisation for the Design of Brushless Permanent Magnet Machines". Departamento de Ingeniería Eléctrica y Electrónica, Universidad de Bristol.

13. **HSU, Chao-Hsin y JUANG, Chia-Feng. "**Temperature Control by Chip-Implemented Adaptive Recurrent Fuzzy Controller Designed by Evolutionary Algorithm", Noviembre 2005. IEEE transactions on circuits and systems—i: regular papers, vol. 52, no. 11.

14. **ZOU, Qi; BAI, Yuebin; JU Yanwen y WU Wei. "**PSOS: A Novel Protocol Simulation Platform for Satellite Networks".

15. **SAATCHI, Sara y HUNG, Chih-Cheng. "**Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. Swarm Intelligence and Image Segmentation. pp. 163 – 178.

16. **CASTRO, Emiliano y TSUZUKI, Marcos. "**Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. Simulation Optimization Using Swarm Intelligence as Tool for Cooperation Strategy Design in 3D Predator-Prey Game. pp. 87 – 100.

17. **Ali T, Al-Awami; Mohammed A, Abido y Youssef L, Abdel-Magid.** Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. Application of PSO to design UPFC-based stabilizers. pp. 235 – 262.

18. **CHEN, Bing-Rui y FEN, Xia-Ting. "**Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. CSV-PSO and Its Application in Geotechnical Engineering. pp. 263 – 288.

19. **PEREZ, Ruben y BEHDINAN, Kamran. "**Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. Particle Swarm Optimization in Structural Design. pp. 373 – 394.

20. **SAHIN, Ferat y DEVASIA, Archana. "**Swarm Intelligence" Focus on Ant and Particle Swarm Optimization. Distributed Particle Swarm Optimization for Structural Bayesian Network Learning. pp. 505 – 530.

21. **HO, S. L; YANG, NI, Shiyou Guangzheng y WONG H. C.** "A Particle Swarm Optimization Method With Enhanced Global Search Ability for Design Optimizations of Electromagnetic Devices", Abril 2006. IEEE TRANSACTIONS ON MAGNETICS, VOL. 42, NO. 4.

22. **LIMA, Joaquín y BARÁN, Benjamín. "**Optimización de Enjambre de Partículas aplicada al Problema del Cajero Viajante Bi-objetivo". Universidad Nacional de Asunción Universidad Nacional de Asunción**,** Campus Universitario, San Lorenzo – Paraguay. Disponible en: http://cabrillo.lsi.uned.es:8080/aepia/Uploads/32/345.pdf (Fecha de revisión: Dic 10/08).

23. **TORO, Eliana; RESTREPO, Yov Steven y GRANADA, Mauricio. "**Adaptación de la técnica de particle swarm al problema de secuenciamiento de tareas", Diciembre de 2006. Disponible en: http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/161217revista1111111.pdf (Fecha de revisión: Dic 2/08).

24. **DAS, Sanjoy; KODURU, Praveen; GUI, Min; COCHRAN, Michael; WAREING, Austin; WELCH, Stephen y BABIN, Bruce. "**Adding Local Search to Particle Swarm Optimization**",** 16 – 21 de Julio 2006. IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada.

25. **JUANG, Chia-Feng; HSU, Chao-Hsin y HSIEH, Cheng-Da. "**Temperature Control by Recurrent Fuzzy Network Designed Via Clustering Aided Swarm Intelligence**"**, 8 – 11 de Octubre 2006. IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan.

26. **MARTÍNEZ, José Luis.** "Optimizacion y ajuste de parametros mediante el metodo simplex (Nelder-Mead)", 3-4 de Mayo 2001. 1ª Reunión de Usuarios de EcosimPro, UNED, Madrid**.** Disponible en: http://www.ecosimpro.com/download/articles/C01_19_es.pdf (Fecha de revisión: Mar 1/08).

8. ANEXO I

TUTORIAL

La herramienta consta de:

- Una interfaz (figura 53), que le permite al usuario escoger la función con que desea trabajar.

- Un simulador que muestra los resultados obtenidos por el PSO (figura 55).

- Un simulador que muestra los resultados obtenidos por el PSOCG (figura 64).

- Un simulador donde se compara las dos arquitecturas planteadas (figura 66).

- Una interfaz, utilizada para encontrar la desviación, el error y la media (figura 69).

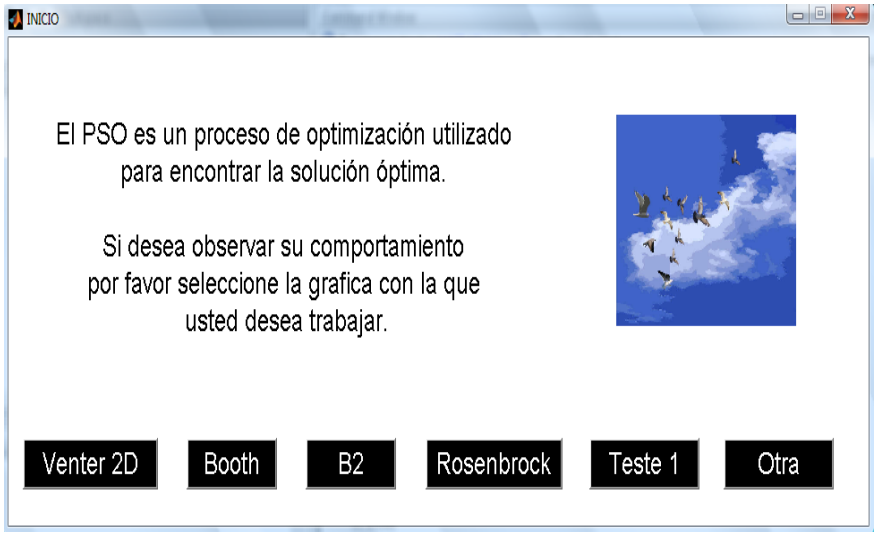

FIGURA 53: VENTANA DE INICIO

Para comenzar con el programa, se debe dar clic en la opción “Venter”, “Booth”, “B2”, “Rosenbrock” o “Teste 1” (figura 54), si se desea trabajar con funciones de dos dimensiones, de lo contrario presionar Otra.

FIGURA 54: FUNCIÓN A GRAFICAR

La ventaja de trabajar con dos dimensiones, es los resultados de los análisis se pueden observar gráficamente, lo que no ocurre con una función multidimensional.

Al dar clic, se abre la ventana del simulador mostrado en la figura 55.

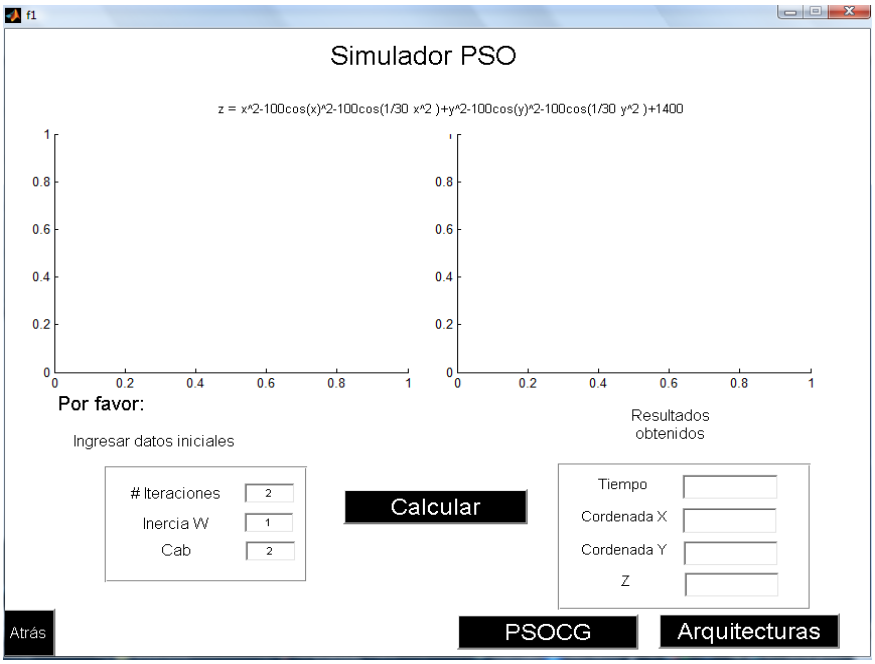

FIGURA 55: SIMULADOR DEL PSO

En esta ventana, se deben ingresar los valores como lo indica la figura 56. El # de iteraciones que se desean realizar (entre mas iteraciones se escojan más exacta es la respuesta, pero la rapidez de la arquitectura disminuye), el valor de la inercia w que por experiencia, varía entre 0 y 1; y para finalizar, se debe ingresar el valor de C_1 y C_2, que habitualmente están en el rango de 1.5 a 2.5. En este

documento $C_1 = C_2 = C_{ab}$, esto debido a que generalmente C_1 y C_2 son iguales o muy cercanos.

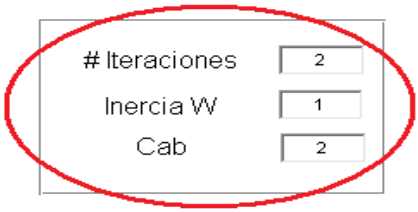

FIGURA 56: DATOS INICIALES DEL PSO

Luego se prosigue a ejecutar el algoritmo PSO dando clic en el botón “Calcular” que se muestra en la figura 57.

FIGURA 57: ITERACION DEL PSO

Los resultados del tiempo que demora el proceso y las coordenadas de la respuesta óptima; son mostrados como indica la figura 58.

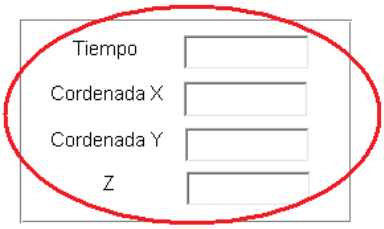

FIGURA 58: RESULTADOS DEL PSO

La gráfica mostrada a la izquierda del simulador PSO, es la función que se está optimizando y al lado derecho del mismo, se encuentra graficada la respuesta óptima, encontrada por el PSO.

En la parte inferior de la figura 55, se encuentran tres opciones. La primera opción (figura 59), es empleada para trabajar con el algoritmo PSOCG. La segunda (figura 60), se utiliza para comparar el funcionamiento del PSOSX(S) y el PSOSX (PE).

La tercera (figura 61), es necesaria cuando se desea volver, a la ventana anterior.

FIGURA 59: PSOCG

FIGURA 60: ARQUITECTURAS DEL PSO

FIGURA 61: RETROCEDER

Si en vez de escoger una función de dos dimensiones se oprimió Otra (figura 62), aparece el simulador de la figura 63.

FIGURA 62: MULTIDIMENSIONAL

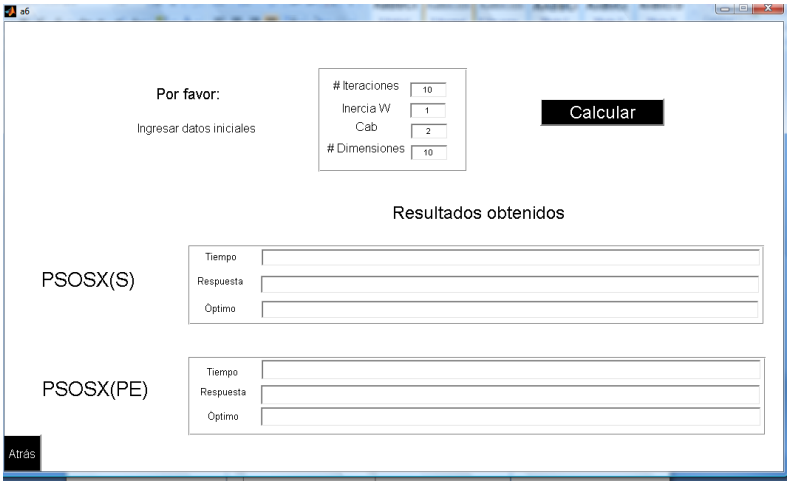

FIGURA 63: SIMULADOR PARA FUNCIONES MULTIDIMENCIONALES

Al igual que en los anteriores simuladores, el multidimensional le permite al usuario ingresar unos datos iníciales como se indica en la figura 64. En este simulador se pide un dato extra y es el número de dimensiones con los cuales se desea trabajar, cabe aclarar que el software parte de unos datos predeterminados que pueden ser manipulados por el usuario.

FIGURA 64: DATOS INICIALES DEL MULTIDIMENCIONAL

Es importante destacar que para introducir la función a optimizar, basta con introducirla en el código de esta interfaz, modificando el archivo a6.m e introduciendo la función en los campos demarcados con: "%FUNCION A UTILIZAR (cámbiela aquí x=x(1), y=x(2)...)", y la forma de introducirla es remplazando las variables X, Y, Z… por $X(1), X(2), \ldots$ Luego solo hay que seguir los pasos ya mencionados para la introducción de parámetros y ejecución del programa.

Al presionar el botón "PSOCG", se muestra el simulador indicado en la figura 65. En éste, se debe seguir los mismos pasos realizados en el simulador del PSO: ingresar los datos iníciales, escoger la función e iterar.

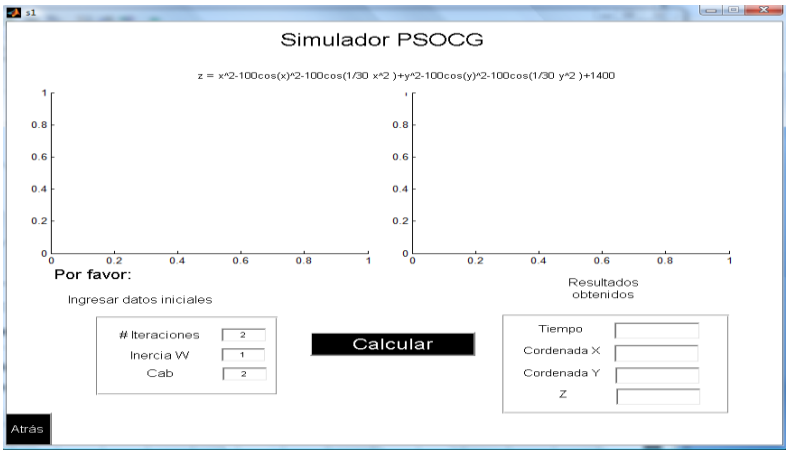

FIGURA 65: SIMULADOR PSOCG

Al dar clic en “Arquitecturas”, se observa el simulador indicado en la figura 66. El funcionamiento de éste simulador es similar a los del PSO y el PSOCG. Al lado izquierdo y derecho, se observan los resultados obtenidos (tiempo y posición de la respuesta óptima) por el PSOSX(S) y el PSOSX(PE) respectivamente.

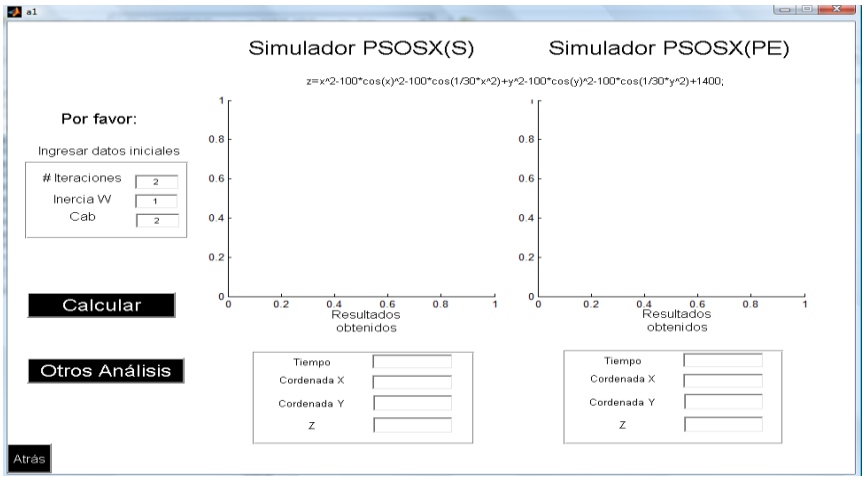

FIGURA 66: SIMULADOR DE LAS ARQUITECTURAS

Para conocer otros cálculos como la exactitud, la desviación estándar y la media de las dos arquitecturas planteadas, es necesario presionar “Otros Análisis” como lo indica la figura 66.

FIGURA 67: ANÁLISIS DEL PSOSX(S) Y PSOSX (PE)

La interfaz mostrada en la figura 69, es utilizada como complemento para comparar datos obtenidos de la herramienta. El usuario debe ingresar los resultados calculados previamente, para luego presionar el botón “calcular” mostrado en la figura 68.

FIGURA 68: CALCULAR

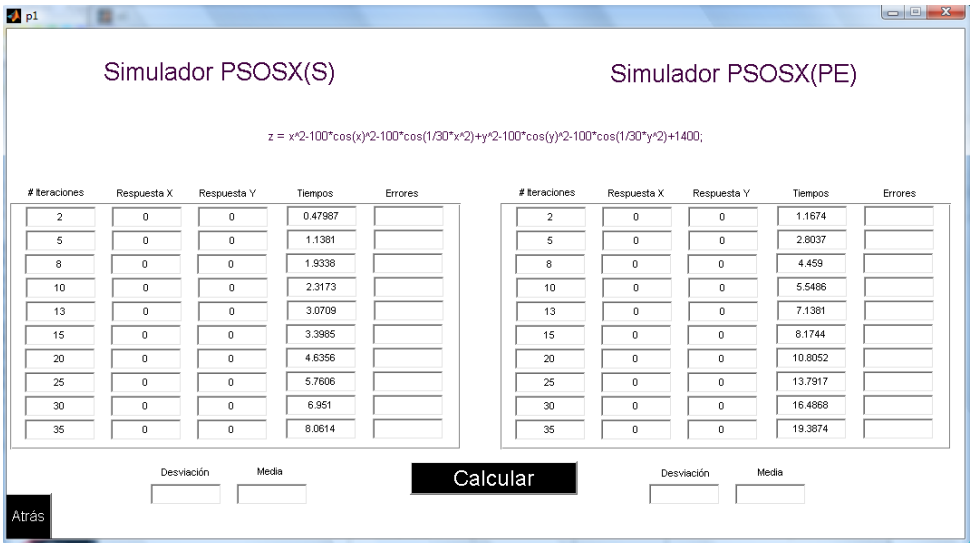

FIGURA 69: PRUEBAS

La figura 69, muestra el error absoluto de cada respuesta, con respecto al valor teórico óptimo de la función; la desviación standard y la media de las posibles respuestas. Además, gráficamente compara los resultados del desempeño de cada arquitectura, para las ecuaciones utilizadas.