LOCALIZACIÓN DE OBJETOS A PARTIR DE TÉCNICAS BIO-INSPIRADAS USANDO MÚLTIPLES AGENTES

MANUEL CAMILO OSORIO DÍAZ
RODRIGO BARROS ROMERO BARRIOS

UNIVERSIDAD SANTO TOMÁS
FACULTAD INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C.
2017

LOCALIZACIÓN DE OBJETOS A PARTIR DE TÉCNICAS BIO-INSPIRADAS

USANDO MÚLTIPLES AGENTES

MANUEL CAMILO OSORIO DIAZ
RODRIGO BARROS ROMERO BARRIOS

Director: Ing. CARLOS QUINTERO PEÑA MSc
CO- director. Ing. CARLOS SAITH RODRÍGUEZ MSc

UNIVERSIDAD SANTO TOMÁS
FACULTAD INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C.
2017

Dedicatoria y agradecimientos

Para comenzar, queremos agradecer a nuestros padres, hermanos y amigos, por darnos siempre su apoyo incondicional en esta etapa tan importante de nuestras vidas, por creer en nosotros y estar a nuestro lado.

Nos gustaría hacer una mención especial a los docentes Carlos Andrés Quintero Peña y Carlos Saith Rodríguez Rojas que fueron las dos personas que más nos acompañaron en este proceso, porque sin ellos no hubiera sido posible cumplir esta meta tan importante que nos propusimos ya hace un año.

TABLA DE CONTENIDOS

pág.

1. PROBLEMA 6
2. ANTECEDENTES 8
3. JUSTIFICACIÓN 10
4. OBJETIVOS 11
4.1. OBJETIVO GENERAL 11
4.2. OBJETIVOS ESPECÍFICOS 11
6. DISEÑO METODOLÓGICO 17
6.1. REVISIÓN DEL ESTADO DEL ARTE DE ALGORITMOS BIO-INSPIRADOS PARA LA PLANEACIÓN DE RUTAS DE MÚLTIPLES AGENTES 17
6.2. SIMULACIÓN DE UN CONJUNTO DE TÉCNICAS BIO-INSPIRADAS QUE PERMITAN CARACTERIZAR SU DESEMPEÑO EN LA LOCALIZACIÓN DE OBJETOS EN AMBIENTES DESCONOCIDOS 18
6.3. DISEÑO DE ESCENARIOS DE PRUEBA 18
6.4. IMPLEMENTACIÓN DE TÉCNICAS DE LOCALIZACIÓN EN UN SISTEMA CON MÚLTIPLES AGENTES Y SU INTEGRACIÓN CON LAS TÉCNICAS BIO-INSPIRADAS SELECCIONADAS. 18
6.5. EVALUACIÓN DEL DESEMPEÑO DEL SISTEMA DE LOCALIZACIÓN USANDO MÚLTIPLES AGENTES EN DISTINTOS ESCENARIOS 18
7. DESARROLLO DEL PROYECTO 19
7.1. DISEÑO DE LA SOLUCIÓN 19
7.2. REVISIÓN DEL ESTADO DEL ARTE 19
7.3. SIMULACIÓN DE UN CONJUNTO DE TÉCNICAS BIO-INSPIRADAS 21
7.3.1. PROBLEMA DE PLANEACIÓN DE RUTAS 21
7.3.2. DISEÑO DE EXPERIMENTOS 25
7.4. DISEÑO DE ESCENARIOS DE PRUEBA 36
7.5. IMPLEMENTACIÓN DE TÉCNICAS DE LOCALIZACIÓN EN UN SISTEMA CON MÚLTIPLES AGENTES 41
7.6. DISEÑO DE LA APLICACIÓN 43
7.6.2. LISTA DE REQUERIMIENTOS FUNCIONALES 47
7.7. VALIDACIÓN DE EXPERIMENTOS 47
7.7.1. VALIDACIÓN REQUERIMIENTOS FUNCIONALES 48
7.8. RESULTADOS DEL PROYECTO 50

8. CONCLUSIONES 54
9. BIBLIOGRAFÍA.. 55

1. PROBLEMA

La robótica es una rama de la tecnología que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el hombre [1]. Se han alcanzado grandes resultados gracias al avance tecnológico que se ha desarrollado a nivel global. En este proceso de desarrollo, compañías como Google, iRobot, Nothrop Grumman y Rethink Robotics, entre otros han comenzado una carrera para la implementación de soluciones a problemáticas que la humanidad presenta, desde el reemplazo de personal en trabajos de alto riesgos y asistencia robótica hasta procesos industriales de automatización.

Un campo específico de la robótica es en trabajo de búsqueda, como en áreas de difícil acceso o lugares de alto riesgo, (zonas contaminadas biológicamente, radiación, zonas de conflicto, incendios, etc), tareas que en la mayoría de países han sido asignadas a personas de rescate y de búsqueda, asumiendo un riesgo hacia ellos mismos en la realización de su trabajo.

Diversas organizaciones como DARPA [2], CRASAR [3] y RoboCup [4], realizan concursos para el diseño de robots ayudantes en zonas de desastres. Por ejemplo, DARPA Robotics Challenge consiste en una competencia que promueve el desarrollo de robots semi-autónomos que puedan realizar tareas complejas en ambientes con humanos [5].

Adicionalmente, los científicos han desarrollado en las últimas décadas técnicas computacionales basadas en el comportamiento de animales y de otros sistemas biológicos que les puedan brindar mayores capacidades de toma de decisión a robots. Estas técnicas han sido exitosamente implementadas en la creación de los sistemas robóticos semi-autónomos para diversos usos.

En este proyecto se propone coordinar múltiples robots implementando técnicas bio-inspiradas con el fin de localizar objetos de interés en ambientes no estructurados y desconocidos a partir de información del ambiente adquirida con sensores. Además, se medirá el desempeño de estas técnicas en distintos escenarios de prueba con el fin de realizar una comparación cuantitativa.

Para esta tarea es importante que los robots desarrollados posean una buena representación del ambiente en el que se mueven con el fin de que puedan llevar a cabo las acciones de búsqueda que les fueron encomendadas. Los procesos de toma de decisiones en planeación de rutas, auto localización, detección de obstáculos y mapeo, son algunas de las necesidades de un robot para tener un buen desempeño autónomo en ambiente desconocidos.

2. ANTECEDENTES

En primer lugar, se tiene que en enero de 2005 fue presentado en la Facultad de Ingeniería de la Universidad de Cantabria, España, comisión de Post Grado, el trabajo de ponencia, **Un algoritmo Híbrido basado en colonias de hormigas para la resolución de problemas de distribución en planta orientados a procesos** [6], por Ángel Cobo y Ana M. Serrano. Se presenta un algoritmo que usa 2 técnicas meta heurísticas, que consisten en algoritmos basados en colonias de hormigas (ACO), y un algoritmo genético que permite mejorar el conjunto de soluciones obtenidas por hormigas artificiales. En el ACO, un conjunto de hormigas artificiales construye de forma concurrente un conjunto de posibles soluciones del problema por medio de asignaciones parciales de secciones o talleres a áreas de la planta.

También se consultó el trabajo **Contribución a la auto localización de robots móviles basada en la fusión de información Multisensorial** [7] por Danilo Navarro García, el cual fue presentado en octubre de 2010 como Tesis Doctoral del Departamento de Informática de Sistemas y Computadores de la Universidad Politécnica de Valencia. Mediante la fusión de datos provenientes de sensores de bajo costo, se logra que un robot móvil se auto localice adecuadamente de forma que pueda navegar confiablemente en entornos estructurados. Se probaron y evaluaron off-line los distintos mecanismos de fusión y filtrado propuestos con el desarrollo de modelos sensoriales y se usaron pseudo códigos para la simulación de la operación de estos sensores en un robot real.

En el año 2006 fue presentado en la Facultad de Ingeniería Informática de la Universidad Rovira i Virgilio, como proyecto de final de carrera el trabajo **Auto localización de robots móviles y modelado del entorno mediante visión estereoscópica** [8] por Carlos Ezquerra Cerdán. En este proyecto, se usó un robot Pioneer P2AT, el cual posee una cámara que dispone de 3 objetivos, por lo

que se pudieron realizar imágenes en 3 dimensiones en la que se puede identificar la posición exacta de un punto de la imagen respecto a otro punto de referencia. Para la auto localización se usó un algoritmo mediante la comparación de imágenes de una cámara de visión estéreo.bMurphy, directora del Centro para la **Búsqueda y Rescate Asistido por Robots** (CRASAR) de la Universidad de Texas A&M, viaja acompañada de sus robots animales (Halcón, Delfín, Colibrí y Suricata) para fomentar su uso, ya sea en zonas catastróficas o en conferencias como la de TED Women [3]. Sus robots han sido usados en 46 desastres en 15 países, incluyendo el tsunami de Fukushima, el huracán Katrina, el derrame de petróleo del Golfo de México y el terremoto de Haití. En Europa se usaron en dos terremotos en Italia y tras la explosión de una central de energía en Chipre [9].

El último concurso del DARPA Robotics Challenge fue en junio del 2015, donde hubo 3 equipos ganadores (Team KAIST, Team IHMC Robotics y Tartan Rescue), quienes lograron terminar los objetivos propuestos por la competencia (Manejar un vehículo, caminar en terrenos difíciles, subir escaleras, retirar escombros, abrir puertas, romper muros con herramientas, abrir válvulas, utilizar mangueras) tareas en las cuales serían de asistencia en un equipo de rescate [10].

3. JUSTIFICACIÓN

La tarea de la localización de un objeto es sencilla para una persona, en términos de que el objeto sea plenamente visible, el área de búsqueda sea pequeña, y segura para la persona. Si se cambian algunos parámetros, como que el objeto sea un gas imperceptible y tóxico o el acceso a esta área sea demasiado restringido, una tarea tan sencilla como ésta puede convertirse en algo tedioso y peligroso para una persona común.

Dicha tarea puede ser relevada a un robot con el fin de evitar poner en peligro vidas humanas en zonas riesgosas (tóxicos, radiación, zonas de conflicto), o minimizar el tiempo de búsqueda utilizando múltiples agentes coordinados para la localización de este objeto. Las técnicas bio-inspiradas constituyen una herramienta importante para la coordinación de dos o más robots para realizar dichas tareas, ya que se basan en el comportamiento colaborativo de múltiples individuos en la naturaleza y son ampliamente utilizadas en sistemas que pueden ser representados como procesos de optimización, con bajas demandas computacionales y altas capacidades de adaptación [11].

4. OBJETIVOS

4.1. OBJETIVO GENERAL

- Localizar objetos de interés en ambientes no estructurados a través de múltiples agentes usando técnicas bio-inspiradas.

4.2. OBJETIVOS ESPECÍFICOS

- Simular un conjunto de técnicas bio-inspiradas (colonia de hormigas, colonia de abejas, murciélago, enjambre de partículas, entre otras) que permitan evaluar su desempeño en la localización de objetos en ambientes no estructurados.

- Diseñar un conjunto de escenarios de prueba donde los múltiples agentes llevarán a cabo la localización de los objetos de interés.

- Implementar las técnicas de localización seleccionadas en un sistema con múltiples agentes homogéneos para cada escenario diseñado.

- Evaluar el desempeño del sistema de localización usando múltiples agentes.

5. MARCO TEÓRICO

La localización es la capacidad que tiene un robot de reconocer el ambiente en el que se encuentra, para esto el robot se dota de herramientas sensoriales para captar lo que existen en su alrededor. Dependiendo de estas herramientas se utiliza un algoritmo para procesar esta información; un robot con múltiples herramientas es capaz de reconocer en un punto lo que se encuentra a su alrededor. La manera en la cual se localiza un objeto es determinada por el comportamiento del robot.

En la actualidad, se han comenzado a implementar innumerables algoritmos para la resolución y optimización de problemas en la industria. Dentro de estos, se pueden encontrar los algoritmos bio-inspirados, los cuales consisten en simular el comportamiento y forma de pensar de sistemas biológicos (hormigas, libélulas y peces entre otros).

Muchos ingenieros han desarrollado múltiples algoritmos bio-inspirados basado en los sistemas biológicos, quienes tienen una manera muy peculiar de optimizar las cosas, por ejemplo, las hormigas cuando van a buscar alimento, crean múltiples caminos y van dejando feromonas que las otras puedan reconocer para usar al momento de encontrar alimento, el recorrido más corto y con menos obstáculos a su meta. Este algoritmo basado en hormigas es muy usado en los barcos de pesca, turismo y otros, con el propósito de buscar el camino con menos contratiempos del viaje, evitando tormentas, fuertes oleajes, terremotos, submarinos y así llegar de manera más segura a su destino.

La ingeniería de sistemas complejos centra sus esfuerzos en el trabajo y construcción de estos sistemas, que son sistemas compuestos de múltiples individuos interactuando de forma no lineal con reglas simples y sin control global [12].

El PSO (Particle Swarm Optimization) es una técnica bio-inspirada basada en el comportamiento que tienen las aves, los bancos de peces y los enjambres de abejas, entre otros, para buscar alimentos.

El modelo en el cual está basado el PSO, es el Global Best Model , que explica que cada partícula de un grupo de estas posee una posición personal y se va buscando en una función de costo los mínimos que están alrededor de cada partícula con el propósito de buscar el objetivo que es el mínimo global de la función. La manera en que se comporta el grupo está basada en el mejor global, donde un grupo de partículas se ayuda entre ellas para hacer más eficiente la búsqueda. El personal best de cada partícula se actualizará constantemente que está se va moviendo, y una de ellas tendrá un mejor personal que el resto, el cual se convierte en el global best, y el resto de partículas sigue este global con el propósito de llegar al objetivo.

PSO ORIGNAL

Inicialización de partículas en posiciones cercanas.

Inicialización de velocidades aleatorias.

While No alcance el máximo de iteración o No esté cerca del Objetivo **do**

For Cada partícula hasta el Máximo de partículas

Actualización de la velocidad a partir PBest y GBest.

Actualización de Posiciones

Evaluación de la posición de la partícula en la función costo

Actualización del Personal Best

Actualización del Gobal Best

Condición de distancia entre la partícula y el objetivo

End_if

Actualización del peso de Inercia

End_condition

Figura 1. Pseudo-código PSO.

Inicialmente se hace un arreglo de partículas, donde el grupo de estas están ubicadas aleatoriamente en el plano de la función, y estas tienen una posición inicial y una velocidad con la cual comienzan a moverse en busca del mínimo más próximo a ellas con una velocidad que va disminuyendo con el paso del tiempo.

$$V_{(x,y)}(t+1) = W * V_{(x,y)} + C_1 r_1 \left(b_{(x,y)} - P_{(x,y)}\right) + C_1 r_1 (g_{(x,y)} - P_{(x,y)}) \qquad (1)$$

De la ecuación (1) se puede ver que hay varios aspectos a considerar al momento de actualizar la velocidad, y estos son: el w, c1, c2, b(x,y), p(x,y), g(x,y) , que vienen siendo coeficiente de inercia, constante de reconocimiento personal, constante de reconocimiento global, mejor posición donde ha estado la partícula, posición de la partículas en el eje “xy” y mejor posición global de las partículas respectivamente. En la figura 2 se puede observar un campo potencial donde se encuentran distribuidas muchas partículas.

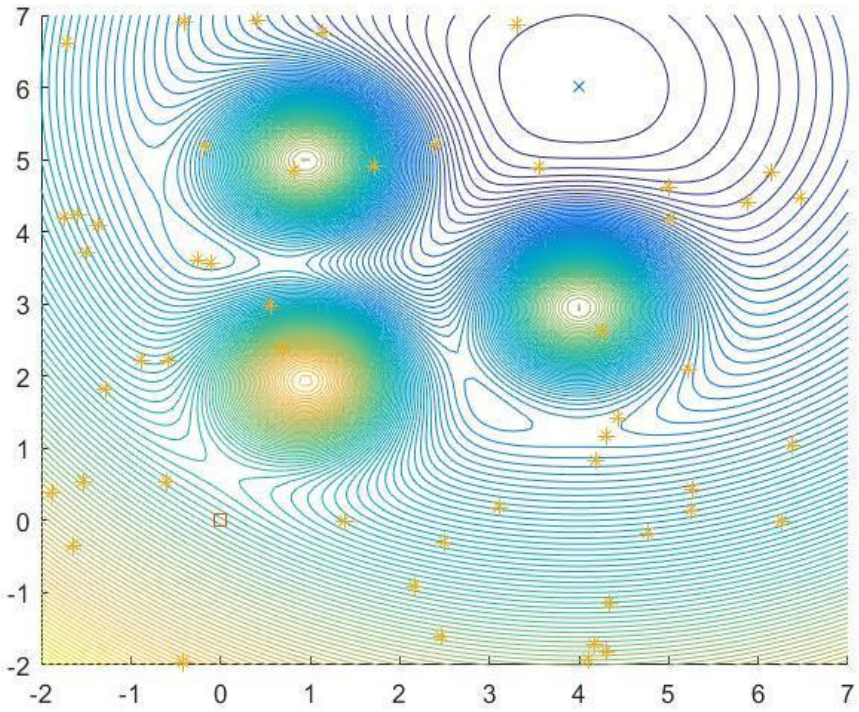

Figura 2. Distribución de las partículas en el campo potencial.

Por otro lado, se tiene el ACO (Ant Colony Optimization), el cual es un algoritmo que fue propuesto por primera vez por Marco Dorigo. El algoritmo trata de una técnica probabilística, usada para resolver problemas complejos que pueden ser menores usando un camino de grafos. Este algoritmo como bien dice su nombre, está basado en las colonias de hormigas, basado en su comportamiento cuando están buscando un camino entre la colonia y un alimento.

```
ACO
Inicialización de feromonas
While No alcance el máximo de iteración do
    Cada hormiga es inicializada en el nodo de Inicio.
    For Cada Hormiga hasta el Máximo de hormigas
        If Si la hormiga no llega a su destino
            For Cada nodo hasta máximo de nodos
                Se calcula las reglas de estado de transición para ACS.
            End for
        End_If
        Calculo del costo para la ruta de la hormiga
        Actualización de Mejor solución.
    End_for
    Actualización de feromona
End_condition
```

Figura 3. Pseudo-código ACO.

Este algoritmo, es usado principalmente en un caso particular, TSP (Travelling Salesman Problem), el cual es un problema muy conocido de optimización combinatoria computacionales, el cual, aunque parezca sencillo es bastante complejo. En este caso, cada nodo viene siendo una ciudad.

Para el algoritmo, una hormiga se ubica sobre un nodo n del grafo y entre más número de nodos, más rutas habrá, y, por ende, el recorrido será mayor, se puede entender fácilmente que entre N nodos, las rutas posibles vienen siendo N! , lo que hace complejo el problema y se busca la mejor solución posible.

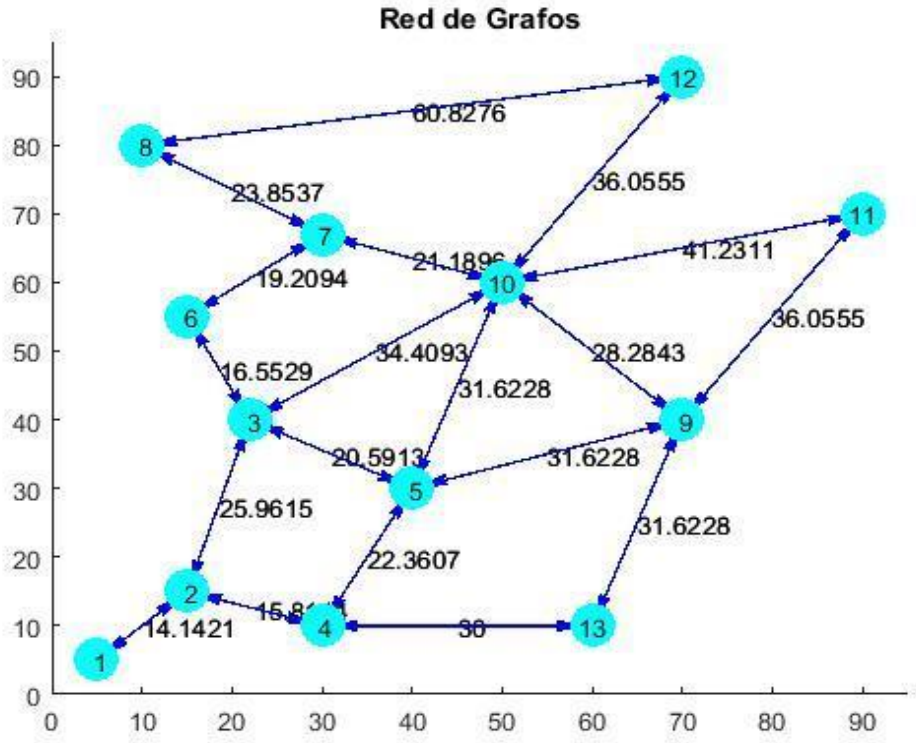

Figura 4. Nodos conectados en una red de grafos

6. DISEÑO METODOLÓGICO

El proyecto se dividirá en 5 etapas:

1) Revisión del estado del arte de algoritmos bio-inspirados (colonia de hormigas, colonia de abejas, murciélago, enjambre de partículas, entre otras) para la planeación de rutas de múltiples agentes

2) Simulación de un conjunto de técnicas bio-inspiradas que permitan caracterizar su desempeño en la localización de objetos en ambientes desconocidos.

3) Diseño de escenarios de prueba.

4) Implementación de técnicas de localización en un sistema con múltiples agentes y su integración con las técnicas bio-inspiradas seleccionadas.

5) Evaluación del desempeño del sistema de localización usando múltiples agentes en distintos escenarios.

Se mencionan en las secciones 6.1., 6.2., 6.3., 6.4., 6.5.

6.1. REVISIÓN DEL ESTADO DEL ARTE DE ALGORITMOS BIO-INSPIRADOS PARA LA PLANEACIÓN DE RUTAS DE MÚLTIPLES AGENTES

Se realiza una revisión del estado del arte de los distintos tipos de algoritmos bio-inspirados y sus aplicaciones en los problemas de planeación de rutas y

evasión de obstáculos. Esta revisión proveerá información sobre las distintas topologías y formas de comunicación de cada algoritmo.

6.2. SIMULACIÓN DE UN CONJUNTO DE TÉCNICAS BIO-INSPIRADAS QUE PERMITAN CARACTERIZAR SU DESEMPEÑO EN LA LOCALIZACIÓN DE OBJETOS EN AMBIENTES DESCONOCIDOS

Con los algoritmos bio-inspirados previamente investigados, se implementarán en simulación (MATLAB) para caracterizar el desempeño de cada uno y en conjunto en la tarea de la localización de objetos.

6.3. DISEÑO DE ESCENARIOS DE PRUEBA

Luego que se hayan implementado en simulación las técnicas bio-inspiradas, se diseñan los escenarios físicos en donde se validaron las técnicas seleccionadas.

6.4. IMPLEMENTACIÓN DE TÉCNICAS DE LOCALIZACIÓN EN UN SISTEMA CON MÚLTIPLES AGENTES Y SU INTEGRACIÓN CON LAS TÉCNICAS BIO-INSPIRADAS SELECCIONADAS.

Se implementarán las técnicas bio-inspiradas seleccionadas y se integrarán con un algoritmo de localización de objetos a través del mapeo y navegación de los agentes en los escenarios diseñados.

6.5. EVALUACIÓN DEL DESEMPEÑO DEL SISTEMA DE LOCALIZACIÓN USANDO MÚLTIPLES AGENTES EN DISTINTOS ESCENARIOS

Por último, se llevarán a cabo experimentos en los escenarios diseñados con el fin de evaluar el desempeño de cada una de las técnicas bio-inspiradas.

7. DESARROLLO DEL PROYECTO

7.1. DISEÑO DE LA SOLUCIÓN

En el siguiente capítulo, se explica el proceso que se hace para el desarrollo del proyecto y se divide en las etapas 7.2., 7.3., 7.4., 7.5., 7.6., donde se explica cada una de estas.

1. Revisión del Estado del arte
2. Simulación de conjunto de técnicas bio-inspiradas:
3. Diseño de escenarios
4. Implementación de los algoritmos seleccionados
5. Validación de los algoritmos seleccionados.
6. Resultados

7.2. REVISIÓN DEL ESTADO DEL ARTE

El primer paso fue revisar el estado del arte, el cual se basa en la investigación de artículos y libros relacionados con las técnicas bio-inspirados. En esta investigación se encuentran y se estudian varias técnicas bio-inspiradas. Algunas de estas técnicas se muestran en la tabla 1.

Técnicas bio-inspiradas	Descripción	Artículos relacionados
PSO (Particle Swarm Optimization)	Es un algoritmo de optimización y búsqueda de alimento de ciertos animales en la naturaleza. Está basado en el modelo del GBest (Global Best Model) explicado en el marco teórico.	1. PSO-based Search mechanism in dynamic environments: Swarms in Vector Fields. [13] 2. A PSO-based hyper-heuristic for evolving dispatching rules in job shop scheduling. [14]
ACO (Ant Colony Optimization)	Es un algoritmo probabilístico, que se usa para resolver problemas complejos en una red de grafos.	1. Estimation of cost rate percentage for travelling salesman problem using hybridized ACO algorithm. [15] 2. A Taguchi-Based Heterogeneous Parallel Metaheuristic ACO-PSO and Its FPGA Realization to Optimal Polar-Space Locomotion Control of Four-Wheeled Redundant Mobile Robots. [16]
GA (Genetic Algorithm)	Son algoritmos basados en procesos genéticos de los seres vivos, usado principalmente para dar solución a problemas de búsqueda y optimización.	1. Parameter estimation of nonlinear nitrate prediction model using genetic algorithm. [17] 2. A multi-agent genetic algorithm for multi-period emergency resource scheduling problems in uncertain traffic network [18]

Tabla 1. Descripción de algunos tipos de algoritmos bio inspirados y ejemplos

7.3. SIMULACIÓN DE UN CONJUNTO DE TÉCNICAS BIO-INSPIRADAS

En esta sección, se explican las técnicas y prueban las técnicas bio inspiradas seleccionadas para escoger cuál será la que ayude a resolver el objetivo general.

7.3.1. PROBLEMA DE PLANEACIÓN DE RUTAS

Para comenzar, se implementó en MATLAB un algoritmo de planeación de ruta, donde se buscaba hacer una ruta entre dos puntos en una función de costo que es un campo potencial mostrado en la figura #. Tenemos una partícula que recorrerá el campo potencial que es descrito por la ecuación (2):

$$J_{(x,y)} = W_1 * J_{o(x,y)} + W_2 * J_{g(x,y)} \qquad (2)\ [11]$$

Donde el primer término de la suma describe la función de los obstáculos y el segundo término de la ecuación la función objetivo. La partícula va recorriendo dicho campo potencial esquivando los obstáculos usando un movimiento de aproximación que la mueve de la posición (x(k), y(k)) en un ángulo Θ a una distancia λ como describe la ecuación (3):

$$\begin{pmatrix} x(k+1) \\ y(k+1) \end{pmatrix} = \begin{pmatrix} x(k) \\ y(k) \end{pmatrix} + \lambda \begin{pmatrix} \cos(\Theta) \\ \sin(\Theta) \end{pmatrix} + \Delta\lambda \begin{pmatrix} \cos(\Delta\Theta) \\ \sin(\Delta\Theta) \end{pmatrix} \qquad (3)\ [11]$$

Donde los 2 primeros términos de la suma representan la posición deseada donde llegará la partícula. El tercer término representa el ruido que indica la incertidumbre de la partícula. El $\Delta\lambda$ es un número aleatorio escogido en cada iteración entre [-0,1λ, 0,1λ] que es el 10% de incertidumbre para el movimiento. También se tiene un $\Delta\Theta$ que se asume entre [-π, π]. Con se garantiza que el movimiento de la partícula no sea perfecto, pero se asegura que terminará en

cualquier punto de la región circular con radio 0,1λ alrededor de los puntos y se hace para describir el movimiento inexacto de un objeto en la vida real.

Como se puede observar en la figura #, se muestra como la partícula hace una ruta a través de los obstáculos para llegar a la meta.

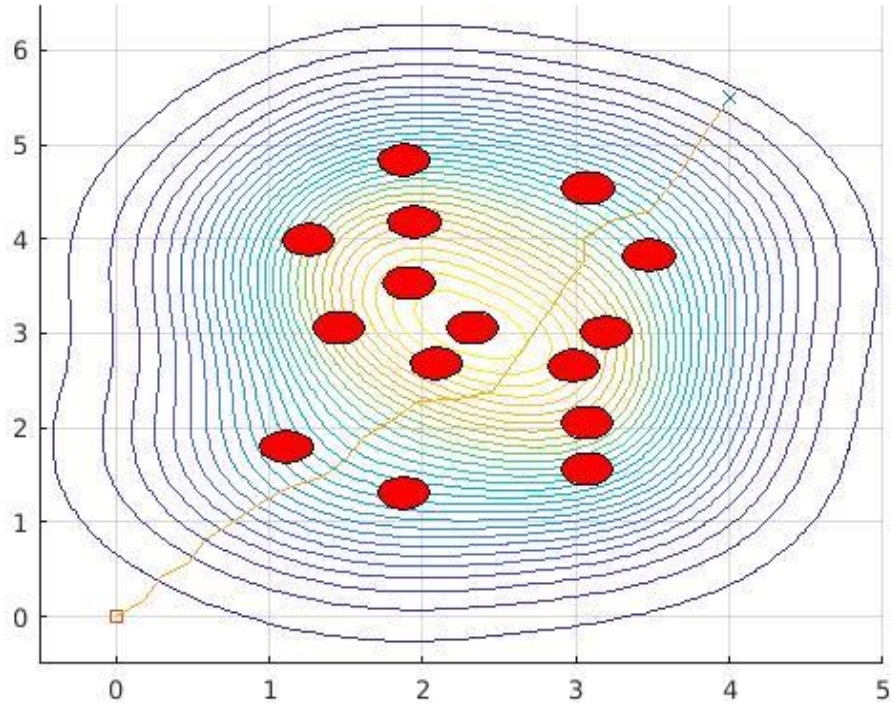

Figura 5. Algoritmo de planeación de ruta donde la partícula alcanza la meta

Por el contrario, en la figura 5, la partícula no es capaz de hacer la ruta a través de los obstáculos y ocurre un estancamiento por lo cual no puede llegar a la meta.

Figura 6. Algoritmo de planeación de ruta donde la partícula se estanca.

7.3.1.1. PLANEACIÓN DE RUTAS USANDO PARTICLE SWARM OPTIMIZATION

En esta parte se pasa a la implementación del PSO en MATLAB. Lo primero que se hizo fue adaptar la función de costo usada en el algoritmo de planeación de ruta a la de PSO, para tener una función donde se evalúan las partículas. Luego de esto se ponen los parámetros que consisten en el máximo número de iteraciones a realizar, el número de obstáculos, la población de las partículas, los coeficientes de aprendizaje explicados en el marco teórico y el peso inercial. Para usos de simulación se implementan límites de velocidad y del campo. Se crean múltiples arreglos que se usan para almacenar los datos de la posición, velocidad, costo, mejor posición, mejor costo, y mejor solución.

Luego de esto, las partículas se ubican en una posición aleatoria x(k), y(k), la cual es donde están fueron ubicadas inicialmente en el campo. También se evalúan en el campo, y se obtiene el global best. Como bien se explica en el marco teórico, la ecuación (1) describe como la velocidad se actualiza y con esto se puede actualizar la posición que está descrita por la ecuación (4):

$$P_{(x,y)} = P_{(x,y)+}V_{(x,y)} \qquad (4)$$

De la ecuación 4, la posición siguiente de la partícula está descrita por la posición anterior más la velocidad. Ya con esto, se pueden evaluar las partículas en la función de costo donde se actualiza el personal best de cada partícula y se encuentra el global best. Con cada iteración las partículas se van organizando en su camino al mínimo de la función y evitando los obstáculos que son los máximos de la función.

Como todo algoritmo, este tiene sus ventajas y desventajas, dichas ventajas ya se mencionaron antes y las desventajas se mencionan ahora. El algoritmo PSO siempre se dirige al punto global mínimo, pero al encontrarse con un muro de

obstáculos se crea un mínimo local, estas partículas verán ese lugar como un mínimo global y se quedaran estancadas, modificando los parámetros de búsqueda en el algoritmo de PSO en los enjambres se pueden liberar solo si una partícula encuentra otro mínimo, eso quiere decir que una partícula pudo encontrar una ruta en la cual pueden salir del muro o estructura que las detiene, pero está manera de evitar los mínimos locales tiene el inconveniente que las partículas ignoren los obstáculos y las choquen bruscamente, así se descarta la modificación de los parámetros. La técnica que se implementa para evitar el estancamiento de las partículas es llamada “Wall Follower”, donde las partículas bordean los obstáculos y así evitan el estancamiento y de esta manera las partículas evitan chocar con los obstáculos y logran salir del estancamiento.

7.3.1.2. PROBLEMA DE LA RUTA MÁS CORTA USANDO ANT COLONY SYSTEM

Para resolver el problema de la ruta más corta se decide seleccionar el algoritmo bio inspirado **A**nt **C**olony **S**ystem, cuyo algoritmo de optimización es el último publicado por Marco Dorigo y Luca [19] creadores de **A**nt **C**olony **O**ptimization. El ACS es un algoritmo diseñado para resolver el problema de **T**raveling **S**alesman **P**roblem por sus siglas TSP, mencionado en el marco teórico, por lo tanto lo que se hace es implementar ACS para resolver el problema de TSP. Cabe mencionar que el problema de ruta más corta es un problema similar al TSP, así que, se implementa ACS en MATLAB, con el objetivo de modificarlo para resolver el problema de la ruta más corta.

La Red de grafos es la manera de representar el modelo donde las hormigas se desplazan; este modelo contiene los siguientes objetos: Los ***vértices*** (o nodos) están representadas con ID’s unidos por ***arcos*** (conexiones) que representan la distancia entre los vértices.

El problema de ruta más corta está propuesto como la búsqueda de una ruta entre dos vértices donde la suma de todos los arcos que la constituyen sea la mínima.

Para dar solución al problema de ruta más corta se realizan las siguientes modificaciones en el algoritmo ACS. Primero, las hormigas solamente pueden acceder desde el vértice actual donde se encuentran a los vértices relacionados por arcos, lo cual limita el acceso a los vértices desde los nodos que no tienen relación. La segunda modificación consiste en delimitar la ruta de la hormiga donde estas inician su recorrido desde un vértice inicial, y lo finalizan hasta encontrar el vértice final, Solamente cuyas hormigas hayan completado exitosamente los recorridos podrán actualizar las feromonas.

7.3.2. DISEÑO DE EXPERIMENTOS

Para esta sección, se realizan una serie de experimentos para la caracterización de los algoritmos implementados en MATLAB.

- ✓ PLANEACIÓN DE RUTA

Para este experimento se simula 500 veces el algoritmo de planeación de ruta en ambientes aleatorios, variando el número de obstáculos, se saca la varianza, el promedio de iteraciones y el número de veces que el algoritmo se estanca como se muestra en la Figura 7.

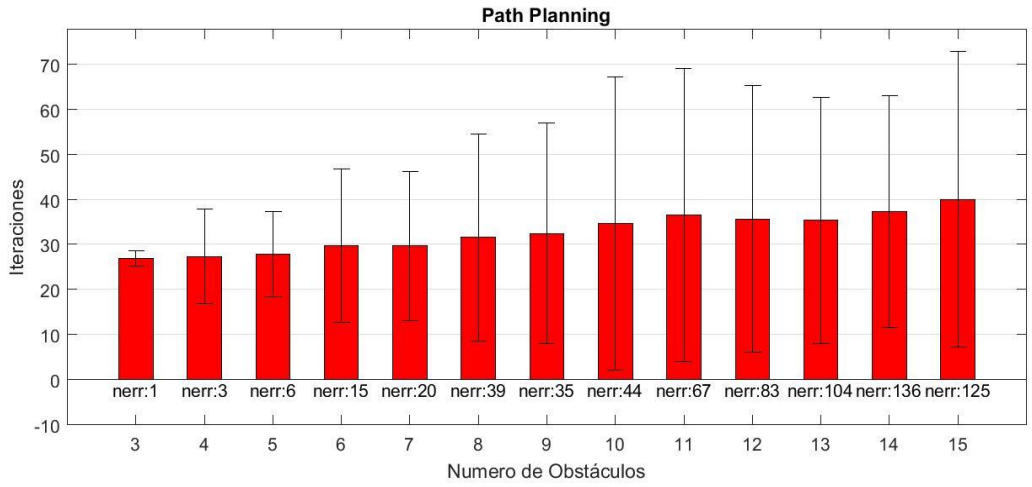

Figura 7. Datos obtenidos del experimento de planeación de ruta

✓ PARTICLE SWARM OPTIMIZATION

Para este experimento, se simula 500 veces el algoritmo en ambientes aleatorios, al igual que en planeación de ruta, se varía la cantidad de obstáculos, pero con la excepción que también se varía el número de partículas y se le saca la varianza, el promedio de iteraciones y el número de veces que se estanca el algoritmo como se muestra en las Figuras 8, 9, 10, 11, 12, 13.

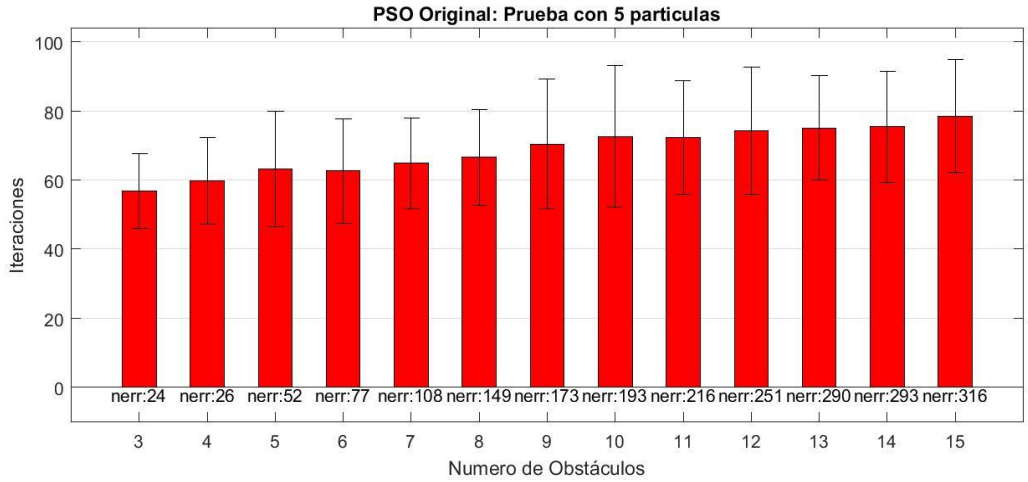

Figura 8. Datos obtenidos del experimento con 5 partículas

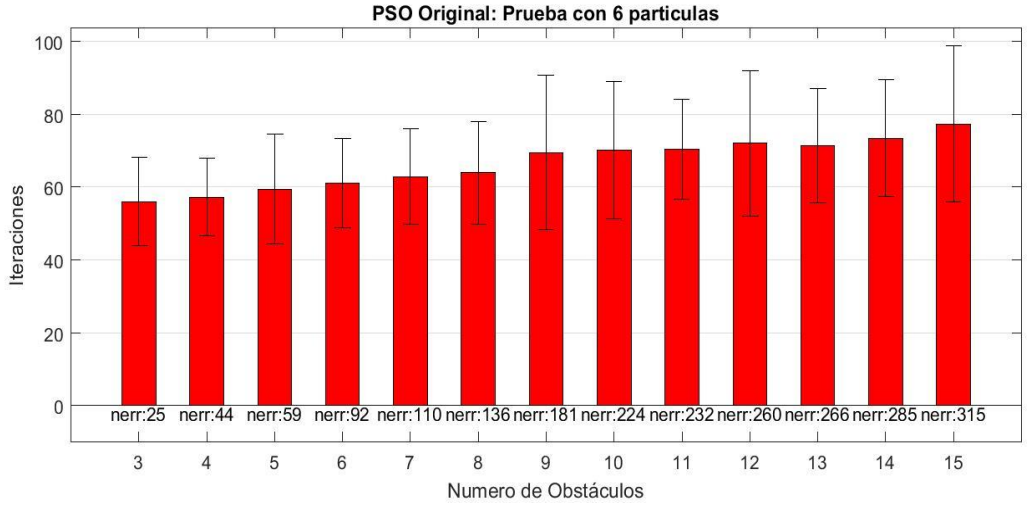

Figura 9. Datos obtenidos del experimento con 6 partículas

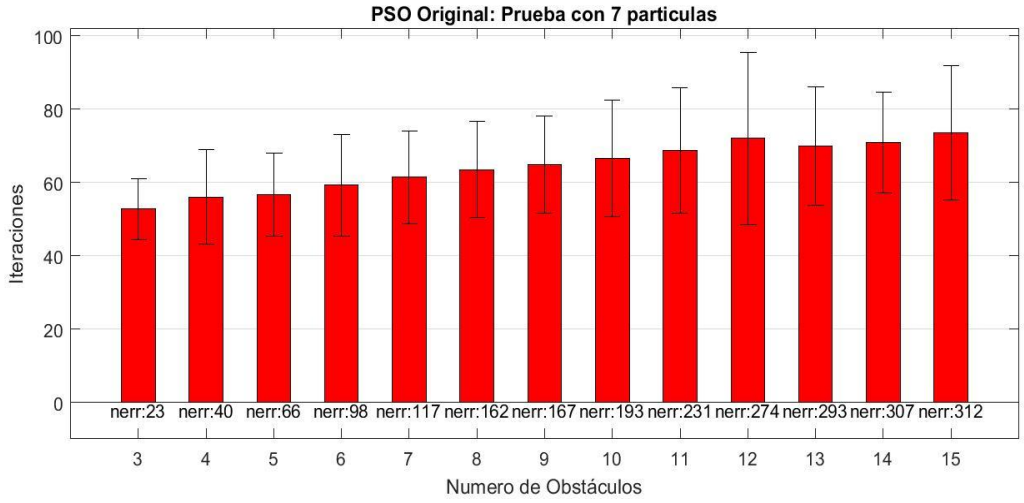

Figura 10. Datos obtenidos del experimento con 7 partículas

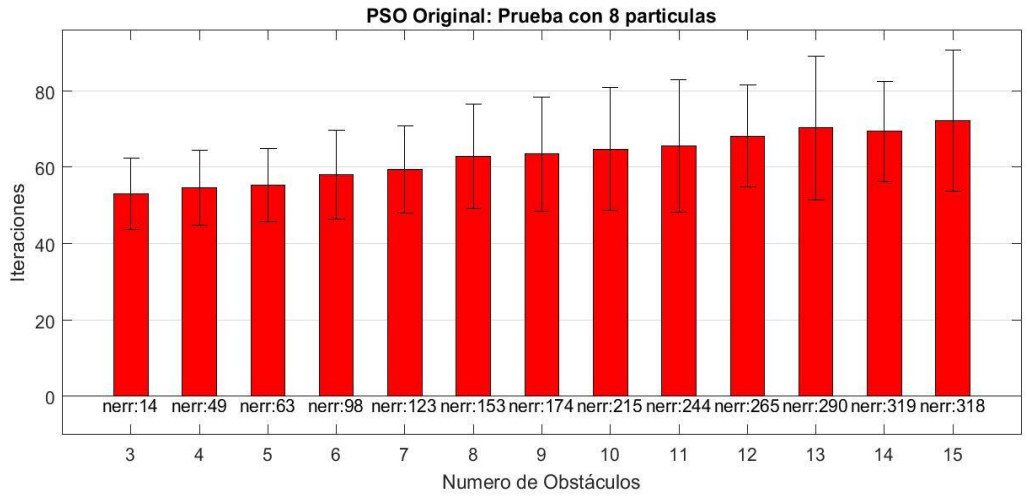

Figura 11. Datos obtenidos del experimento con 8 partículas

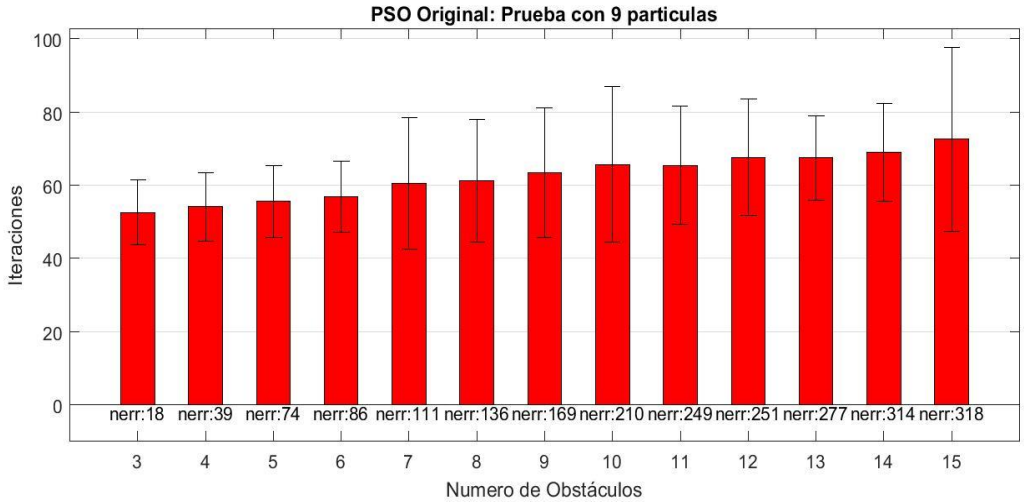

Figura 12. Datos obtenidos del experimento con 9 partículas

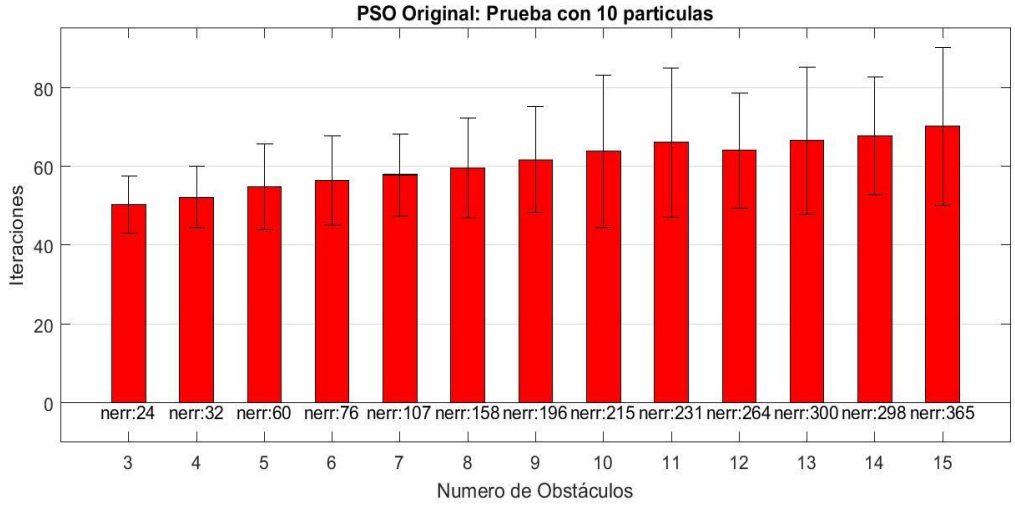

Figura 13. Datos obtenidos del experimento con 10 partículas

- ✓ PARTICLE SWARM OPTIMIZATION MODIFICADO

Para este experimento, se hace como en los 2 anteriores, se simula 500 veces el algoritmo en ambientes aleatorios, se varía el número de los obstáculos y de las partículas y se le saca la varianza, el promedio de iteraciones. Como este es un algoritmo diseñado para evitar el estancamiento de las partículas, no ocurren errores y las partículas siempre localizan el objetivo como se muestra en las Figuras 14,15,16,17,18,19.

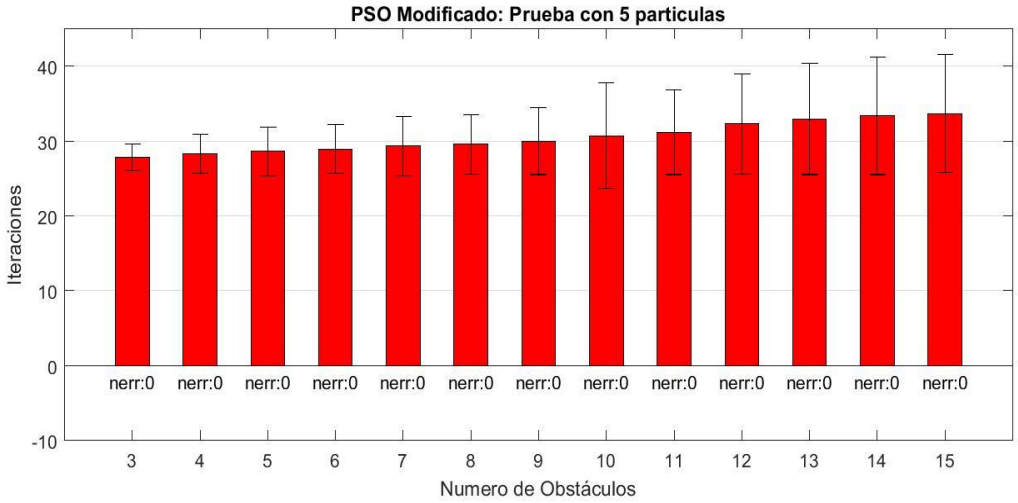

Figura 14. Datos obtenidos con 5 partículas

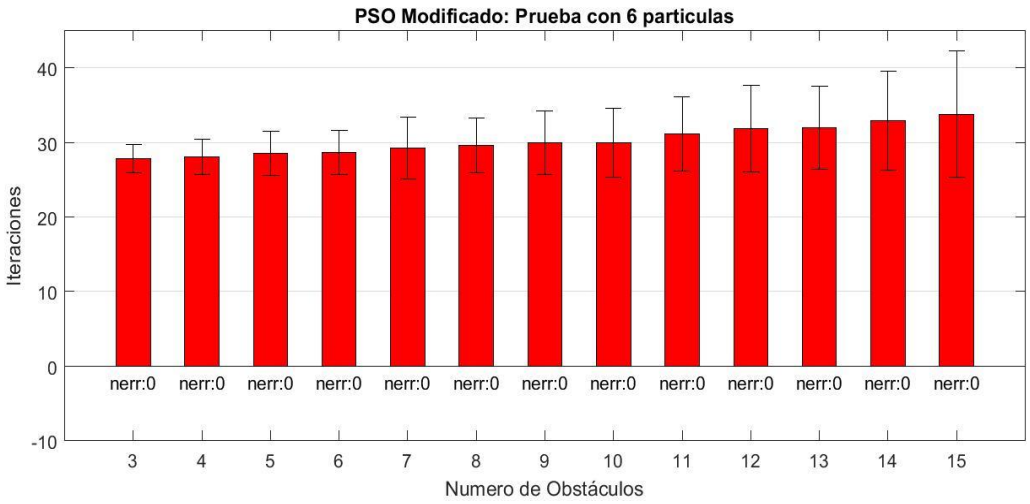

Figura 15. Datos obtenidos con 6 partículas

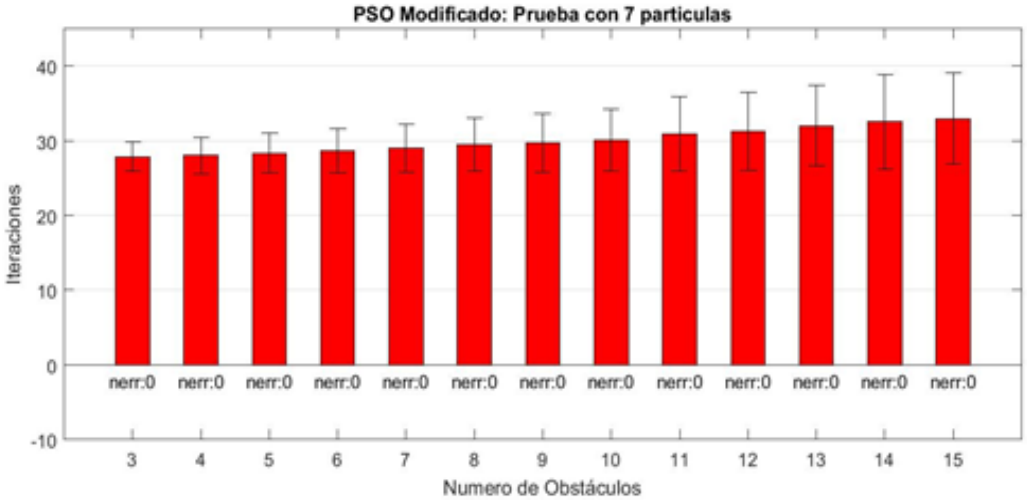

Figura 16. Datos obtenidos con 7 partículas

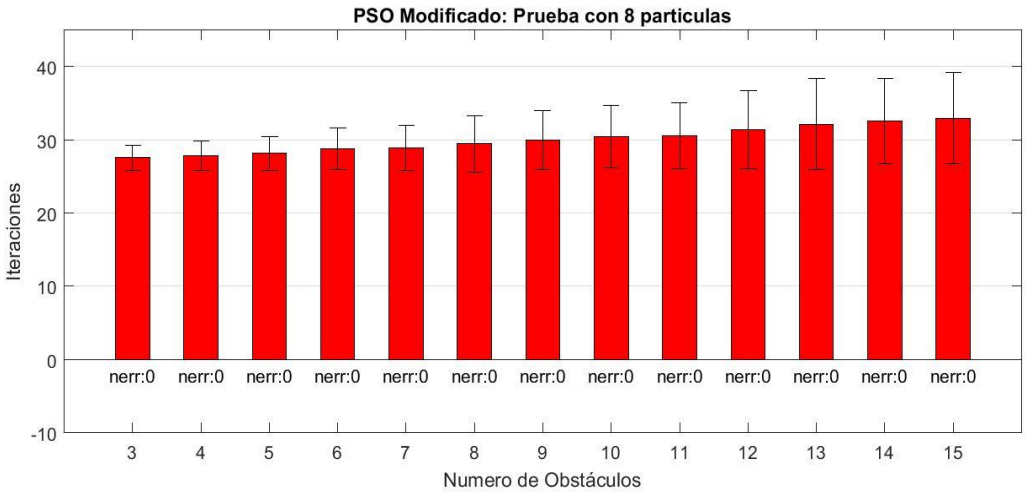

Figura 17. Datos obtenidos con 8 partículas

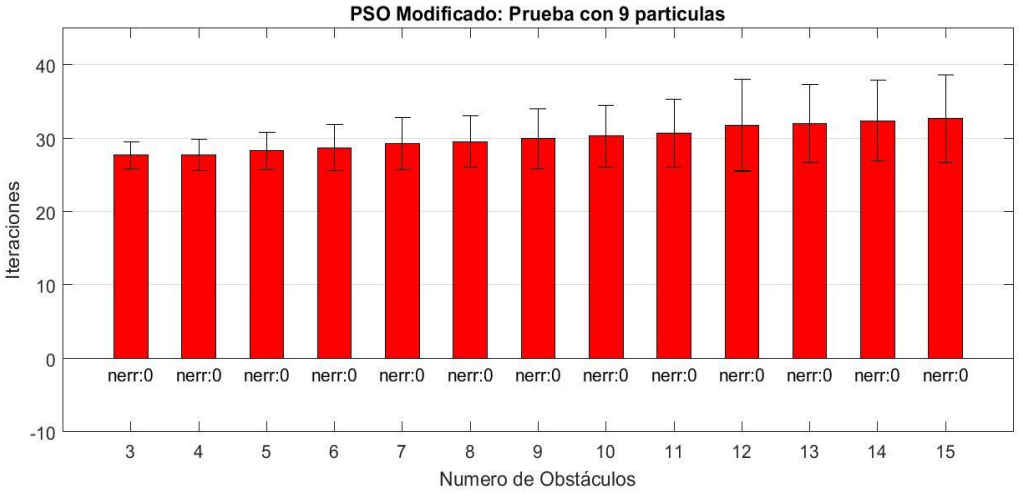

Figura 18. Datos obtenidos con 9 partículas

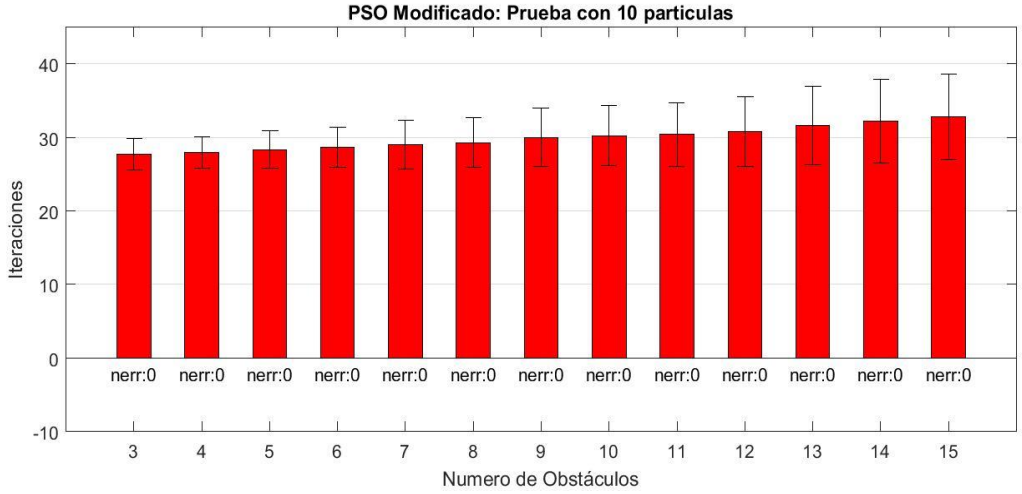

Figura 19. Datos obtenidos con 10 partículas

- ✓ ANT COLONY SYSTEM

En el caso de ACS, se realiza una comparación con un algoritmo ya implementado en MATLAB, conocido como Dikjstra, el cual es un algoritmo diseñado para determinar la trayectoria más corta en una red de grafos, entre el vértice de origen y los otros vértices. Lo que se busca es observar si ACS arroja resultados iguales o aproximados al algoritmo de Dijkstra.

El algoritmo se ejecuta variando la cantidad de hormigas y de vértices, para comprobar si a mayor población mejor será la aproximación a la ruta más corta en una red de grafos.

Las figuras 20,21,22,23,24,25 muestran que el algoritmo de ACS, se aproxima mucho al Dikjstra y en algunos casos encuentra la misma distancia, así como también que, a mayor cantidad de hormigas, el algoritmo se hace más eficiente. Los valores que se encuentran debajo de las barras de las figuras, indican las desviaciones estándar que hay entre la solución encontrada por ACS y la encontrada por Dikjstra.

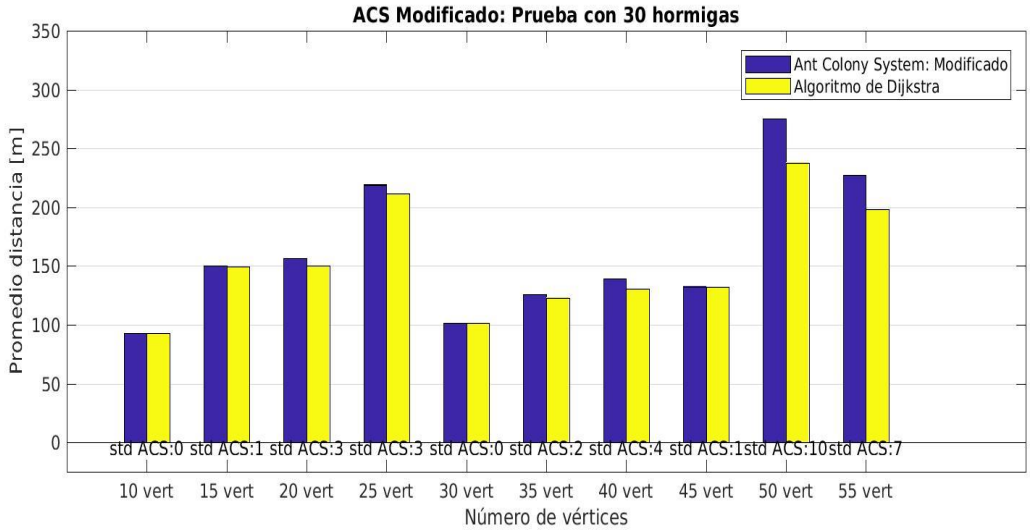

Figura 20. Diagrama de barras que muestra la compara los algoritmos usando 30 hormigas

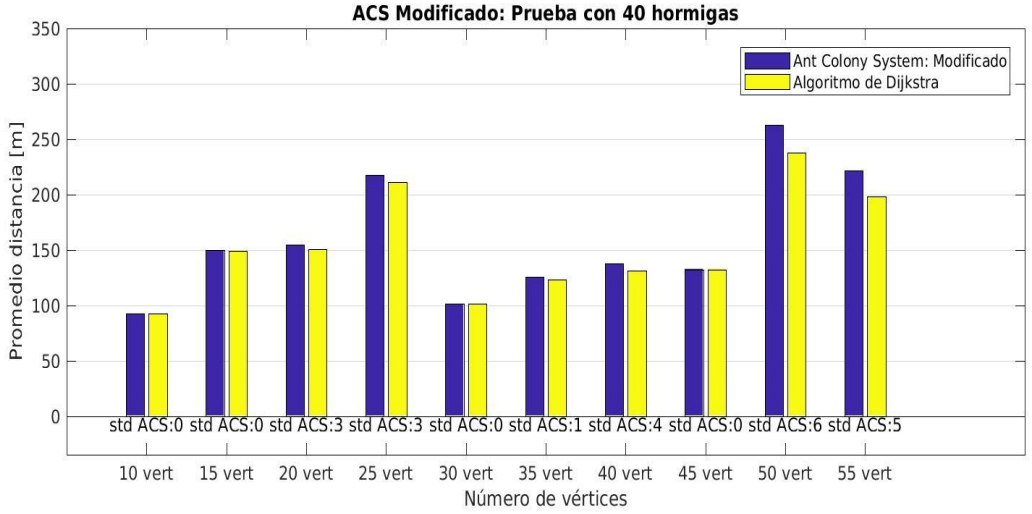

Figura 21. Diagrama de barras que muestra la compara los algoritmos usando 40 hormigas

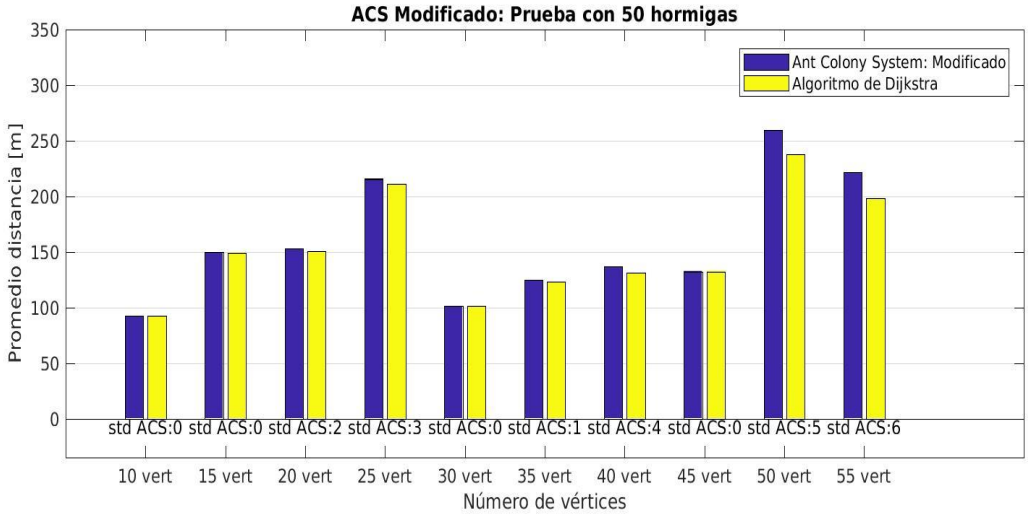

Figura 22. Diagrama de barras que muestra la compara los algoritmos usando 50 hormigas

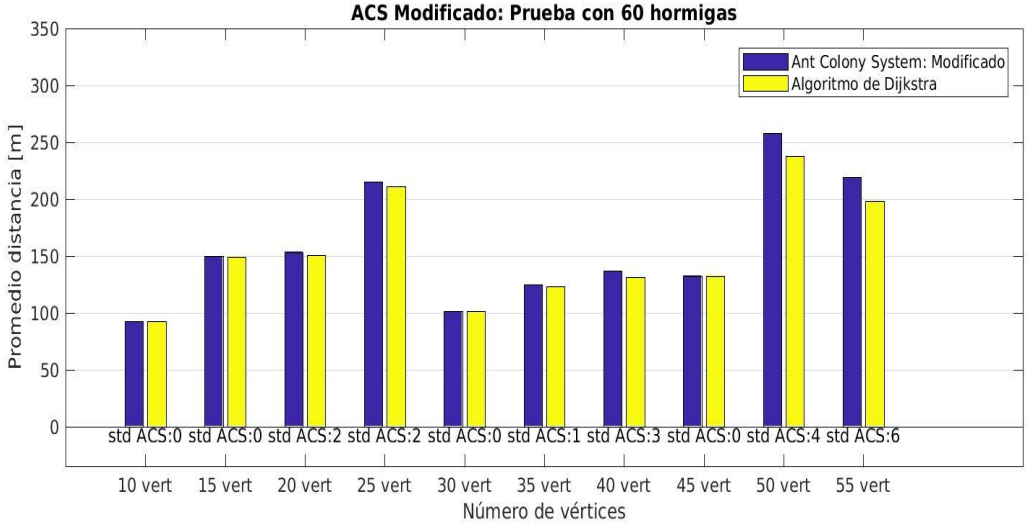

Figura 23. Diagrama de barras que muestra la compara los algoritmos usando 60 hormigas

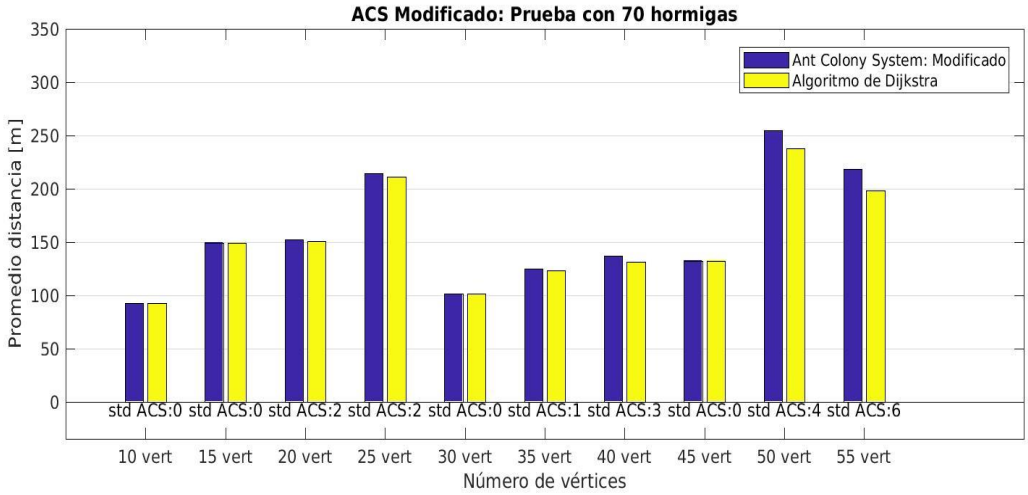

Figura 24. Diagrama de barras que muestra la compara los algoritmos usando 70 hormigas

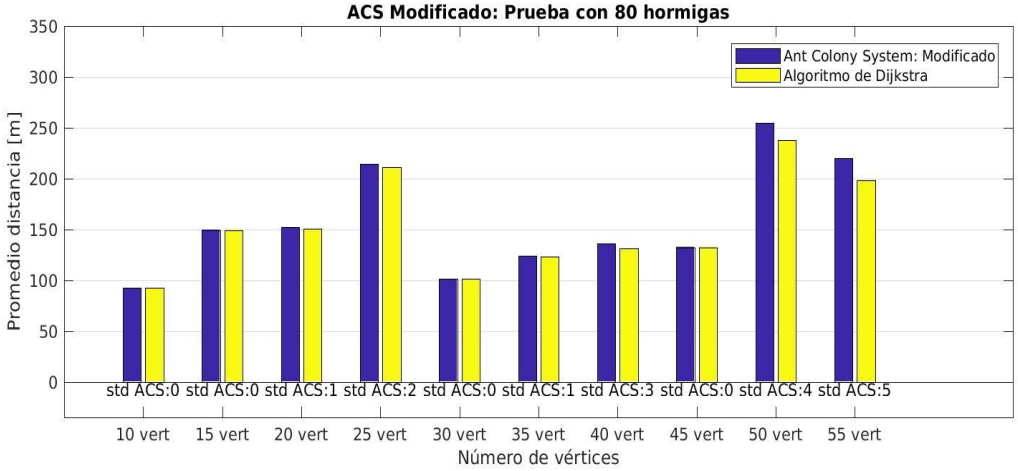

Figura 25. Diagrama de barras que muestra la compara los algoritmos usando 30 hormigas

7.4. DISEÑO DE ESCENARIOS DE PRUEBA

Se diseñan 5 escenarios de prueba para evaluar el desempeño de las partículas en diferentes ambientes.

- Primero escenario

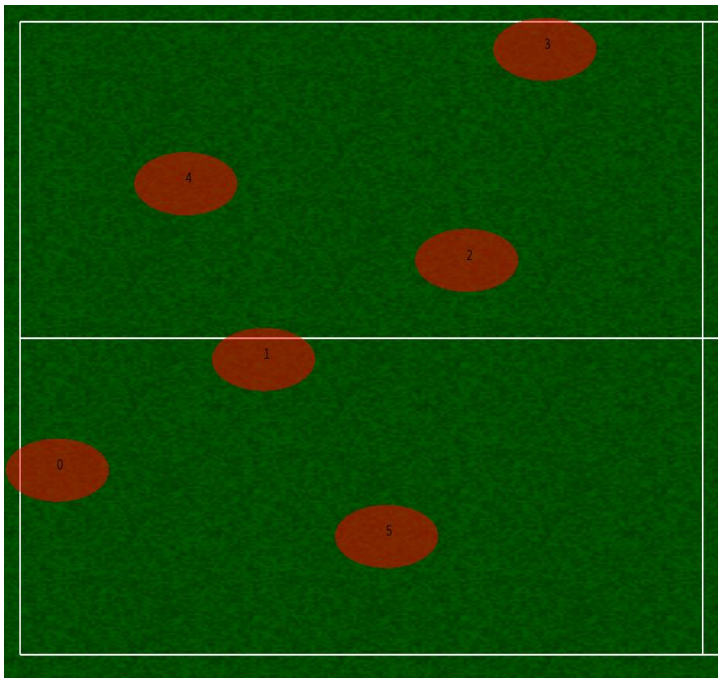

Figura 26 Primer escenario de prueba

Para el primer escenario, como se muestra en la figura 26 se distribuyen los obstáculos de tal manera que se estuvieran interponiendo en la trayectoria de las partículas y demostrar que las partículas pueden adaptarse a estos inconvenientes y cumplir con la trayectoria establecida.

- Segundo escenario

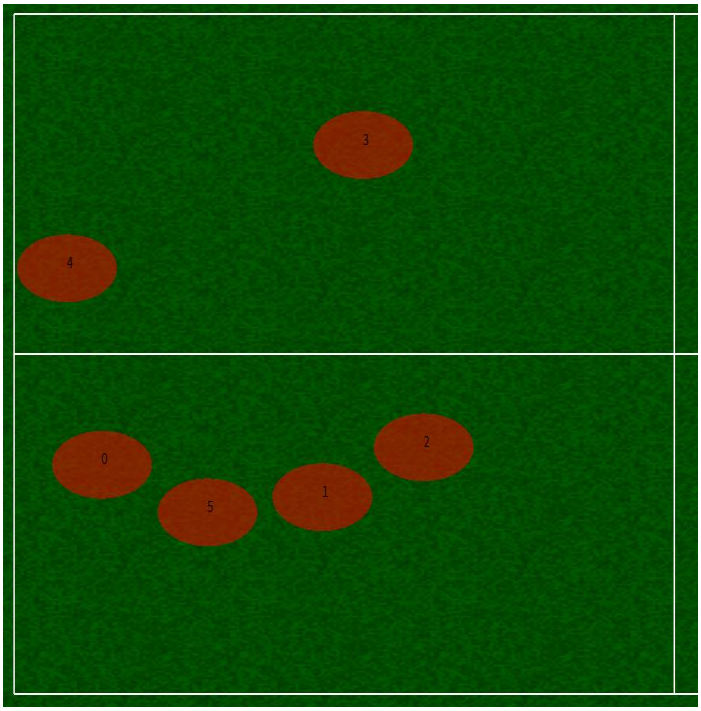

Figura 27 Primer escenario de prueba

Para el segundo escenario, como se muestra en la figura 27 se ubican los obstáculos de tal manera que se crea una especie de barrera en la mitad que sólo pueda ser atravesada por los costados. Esto se piensa luego de observar los resultados de las pruebas realizadas, donde se puede observar que con el PSO original las partículas casi siempre se estancan cuando tenían una gran barrera de obstáculos al frente y poder probar el PSO modificado donde las partículas bordean a los obstáculos para evitar el estancamiento..

- Tercer Escenario

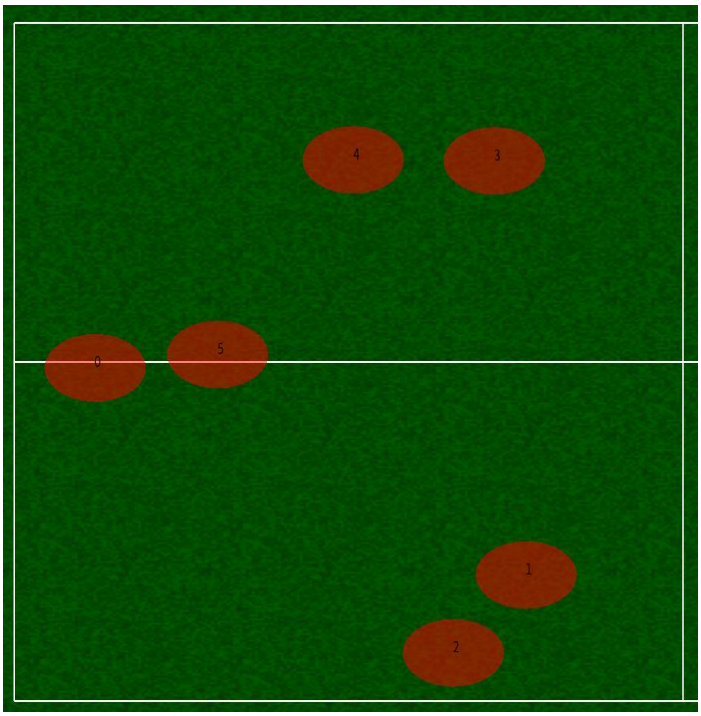

Figura 28 Primer escenario de prueba

Para este tercer escenario, la figura 28 muestra como tres mini barreras obstaculizan a las partículas cuando van a cumplir cierto tramo de la trayectoria, y las obliga a realizar la función de bordeado de los obstáculos para alcanzar a realizar la trayectoria como en el escenario 2. A diferencia del escenario 2, la acción de bordeado se realiza tres veces en este mapa para observar que el algoritmo siempre está en correcto funcionamiento.

- Cuarto Escenario

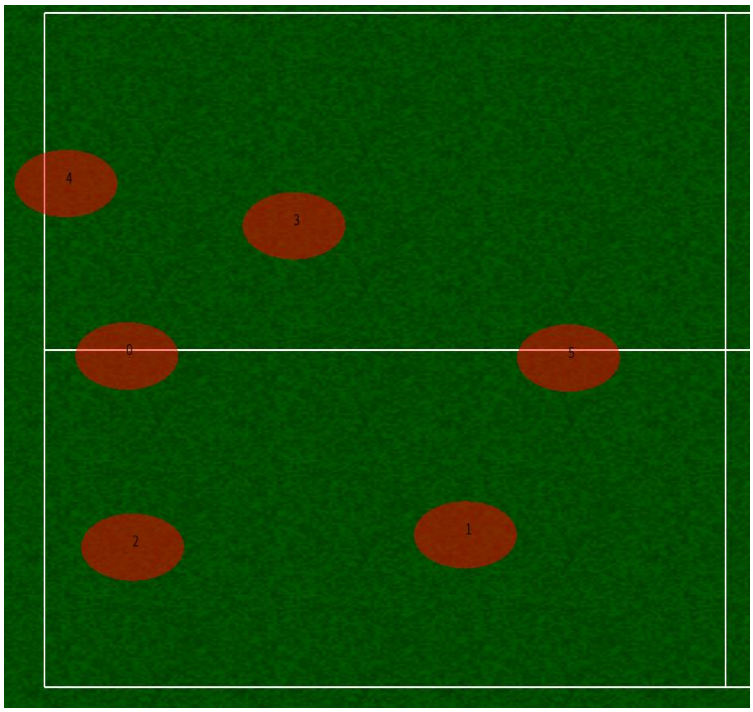

Figura 29. Cuarto escenario de prueba

Para este escenario, como se muestra en la Figura 29, se ubican los obstáculos de tal manera que las partículas tuvieran un pequeño espacio por donde pasar y poder demostrar que las partículas no se choquen entre ellas a pesar del espacio reducido que atraviesan en ciertos tramos.

- Quinto escenario

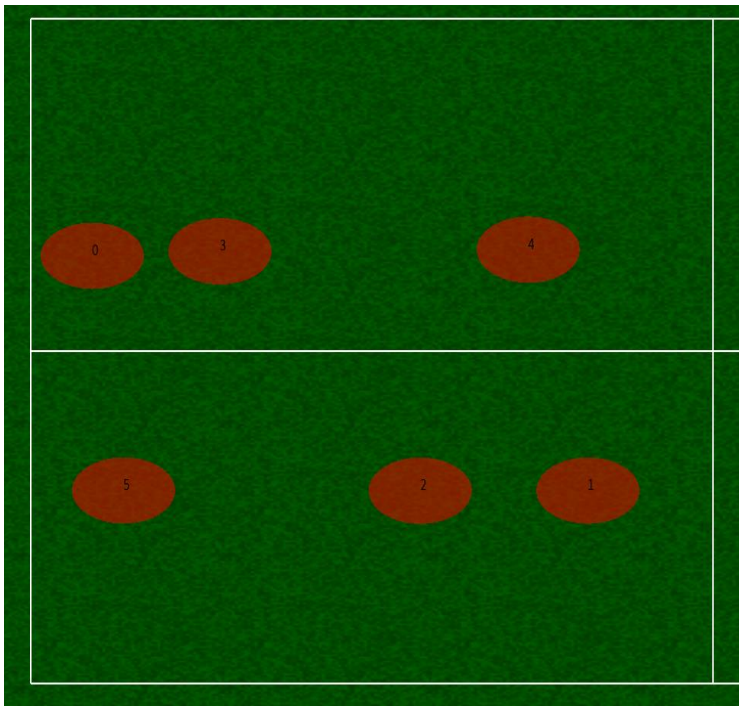

Figura 30. Cuarto escenario de prueba

El último escenario, como se muestra en la Figura 30 se piensa para probar cómo se desenvuelven las partículas cuando tienen barreras en frente y al mismo tiempo para cumplir con la trayectoria deben pasar por espacios reducidos y observar que los algoritmos funcionan correctamente en cualquier circunstancia sea por tener una barrera en frente de ellas o puedan pasar a través de espacios reducidos sin chocarse.

7.5. IMPLEMENTACIÓN DE TÉCNICAS DE LOCALIZACIÓN EN UN SISTEMA CON MÚLTIPLES AGENTES

El diseño de la aplicación se modela en base al problema de localizar un objeto en un escenario no estructurado, para ello se acota el problema en un ambiente controlado, con medidas de un área conocida, con superficie plana y lisa, además las partículas se modelan como agentes homogéneos. Todos estos requisitos se cumplen la la infraestructura que se encuentra en el laboratorio de Robótica de la Universidad Santo Tomás, además se dispone de herramientas de software en el cual se puede recrear este espacio para realizar las simulaciones de implementación física. Esta infraestructura se muestra en la figura 31.

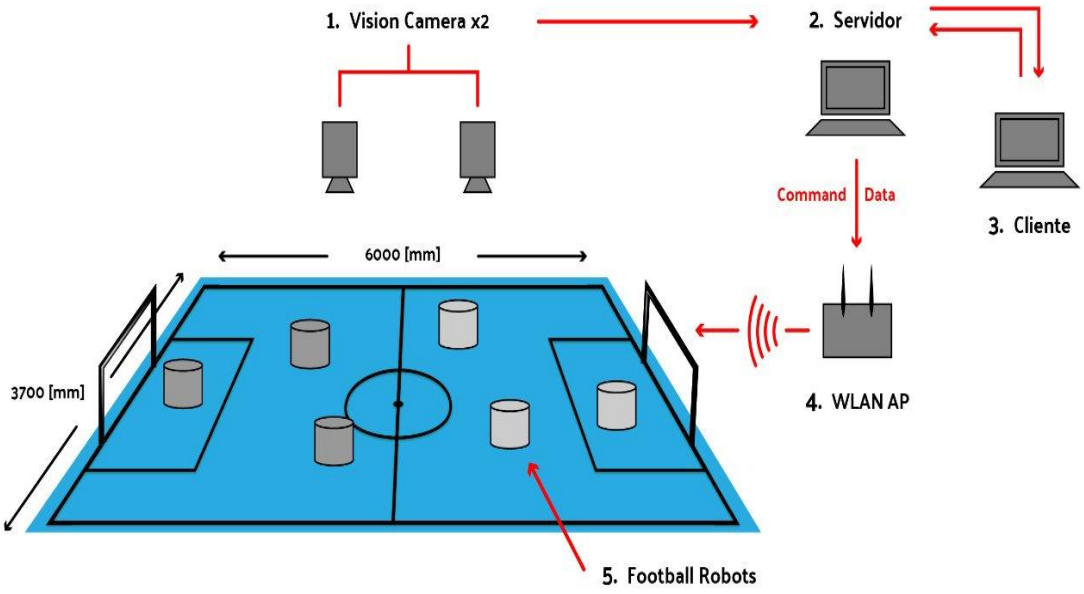

Figura 31. Representación gráfica del campo de juego y la manera en que los robots recopilan información.

7.5.1. ARQUITECTURA DE LA SOLUCIÓN

El escenario controlado es modelado para una Liga Small Size de Robocup, cuenta con una cancha de fútbol de medidas de 3700 milímetros de fondo por 3000 milímetros de largo, dos cámaras ubicadas en la parte superior de la cancha capturan la posición de los agentes como se muestra en la figura 31, donde esta

información es recibida por el servidor que se muestra en la figura 32, y así mismo el servidor tiene el control para enviar los comandos a los agentes para su movimiento por medio de un WLAN AP.

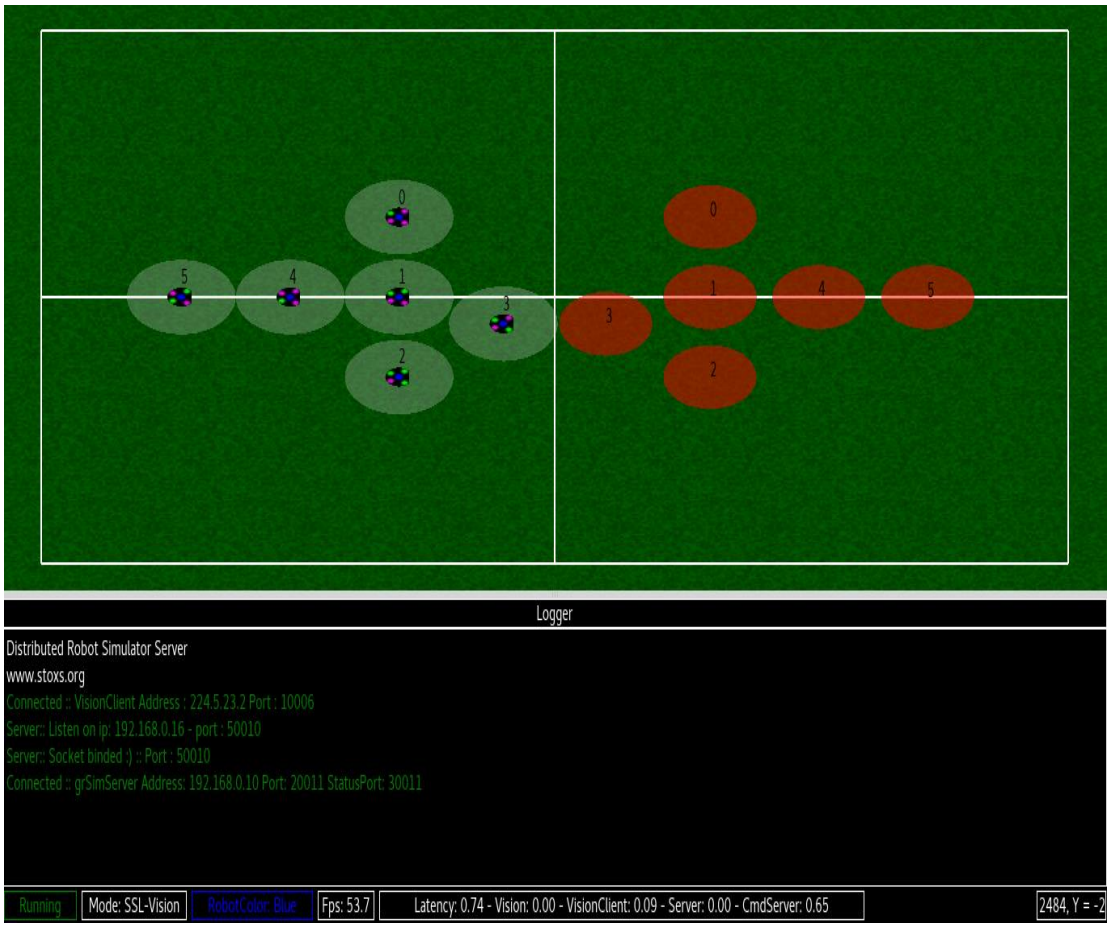

Figura 32. Servidor

Este mismo escenario es representado por el software de simulación grSim que se puede ver en la figura 33, que cuenta con 12 agentes divididos en 2 equipos ubicados en un escenario con estas mismas medidas y con este programa un computador puede convertirse en la representación física del escenario de la Liga Small Size. Así, el servidor puede adquirir datos desde el escenario real, o desde la aplicación grSim.

El diseño de la aplicación se hace en el cliente como está en la 34 figura, donde este se comunica con el servidor adquiriendo la información, para procesarlas y luego enviarlas de regreso al servidor para su ejecución.

Figura 33. Software de simulación grSim

7.6. DISEÑO DE LA APLICACIÓN

Partiendo en la sección 7.5. se realiza una lista de requerimientos funcionales 7.6.2. en las cuales debe tener el cliente, y debe cumplir con los requerimientos funcionales que citados en las tablas 7.7.1

7.6.1. ARQUITECTURA DE LA APLICACIÓN

La arquitectura de la aplicación se desarrolla en programación orientada a objetos, un paradigma de programación donde los objetos procesan los datos de entrada para obtener datos de salida. Cómo lo muestra la figura 34.

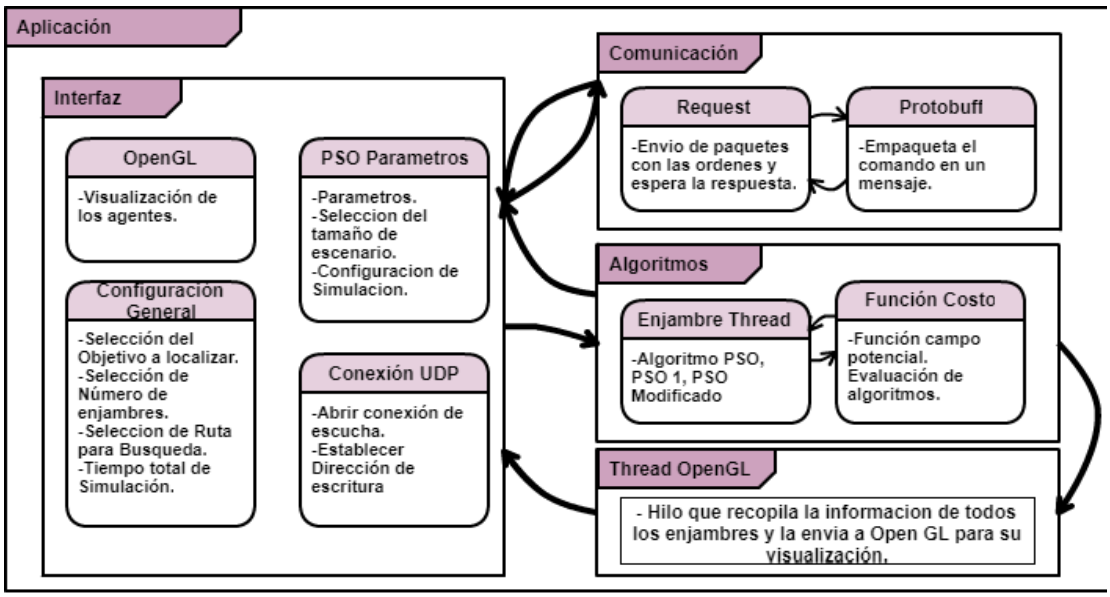

Figura 34. Diagrama de bloques de la arquitectura de la aplicación,

La aplicación se compone de 4 módulos principales, el *Interfaz*, la *Comunicación*, los Algoritmos, y el *Thread OpenGL*. El *interfaz* es la parte visual de la aplicación quien interactúa con el usuario y trasmite la información a las demás clases. El módulo de *Comunicación* es el objeto que se comunica con el Servidor para su simulación, este puede adquirir datos, cómo enviar comandos. El módulo de *Algoritmos* son los que maneja la información brindada para su ejecución, con los algoritmos de PSO, PSO Modificado, por último, el módulo Thread OpenGL es el recolector de información quien integra la información de todos los enjambres las envía al módulo de la interface del usuario OpenGL donde at través de una API se representan los agentes y los obstáculos en 3D.

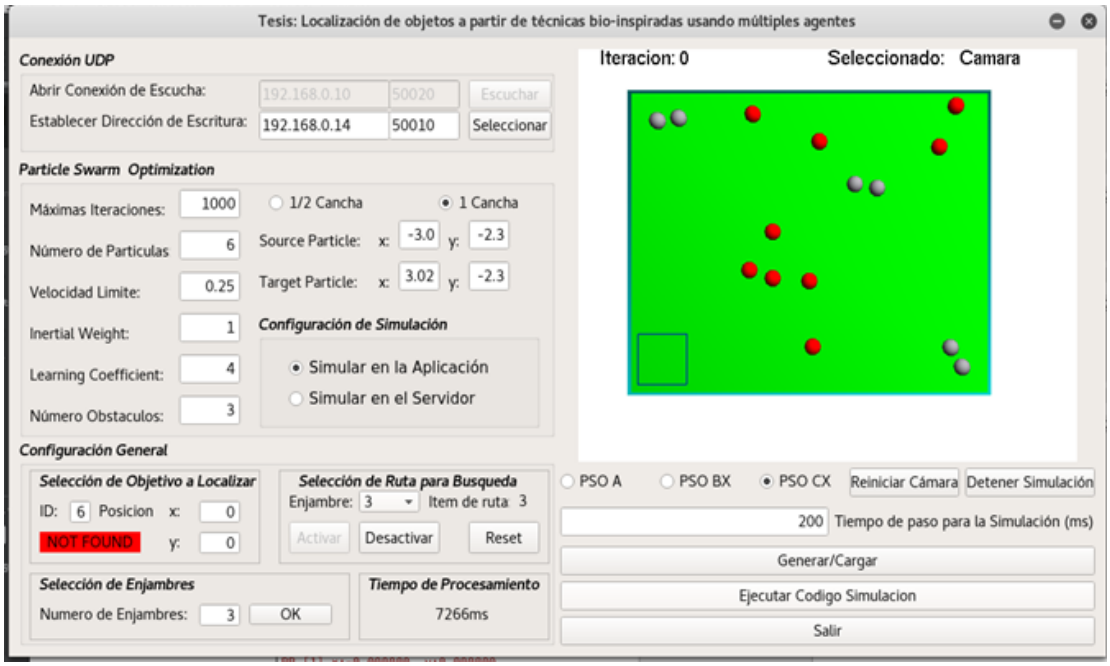

Figura 35. Interfaz gráfica de la aplicación

El módulo del interfaz al usuario recibe todos los parámetros necesarios para el funcionamiento, primero recibe la configuración de la conexión UDP, se reciben dos direcciones IP, y dos puertos, la primera dirección indica el canal en donde se escuchan los datos, y la segunda es la dirección corresponde al servidor donde se enviarán los datos. Esta información una vez seleccionada se envía a un método en el cual se comunica con el servidor haciendo uso de protobuff. La segunda sección es la configuración de los parámetros del algoritmo; aquí se pueden modificar todos los parámetros que modelan el algoritmo, tanto número de partículas hasta los coeficientes de aprendizaje, Está configuración es general, permite seleccionar el tamaño del escenario (½ cancha , 1 chancha) , Visualiza la zona de inicio de las partículas y la zona final de las partículas, al final se decide si la simulación se quiere realizar desde la Aplicación Implementada o realizando el enlace de comunicación con el servidor y dependiendo de la selección se tienen en cuenta parámetros como el número de obstáculos y el número de partículas. La 3ra parte consiste en la configuración general de la búsqueda del objeto a

localizar, Está contiene 4 secciones, La primera *Selección de Objeto a localizar*, configura la ID del obstáculo que se desea localizar , cuando algún agente lo encuentre la ubicación quedara guardada en esta selección en la posición x, y, la siguiente selección es la ruta de búsqueda, para cada enjambre creado, se puede crear una ruta, esta ruta se puede activar, desactivar y reiniciar, los parámetros de la ruta se configura a través de la comunicación entre el módulo de OpenGL y esté módulo, por ultimo esta la sección de tiempo de procesamiento que indica el tiempo cronometrado desde que inicia el recorrido hasta que termine el recorrido o hasta cuando encuentre el objeto a localizar. La última sección son los botones de la simulación, en primera instancia se encuentra una sección donde se selecciona un algoritmo, botones para reiniciar la cámara del simulador, y un botón para detener la simulación. Luego se encuentra un parámetro para el tiempo de la simulación, este tiempo determina la velocidad en que se comunica el servidor y el cliente. Para finalizar se encuentran los botones Generar/Cargar, en el modo de simular en la aplicación se genera los valores iniciales, como posición de las partículas y de los obstáculos, y en el modo simular en el servidor, se comunica con el servidor para ubicar los agentes en el simulador de la aplicación.

El módulo de interfaz tiene cómo clase al API Open GL para la visualización de los algoritmos tanto en forma ejecutada desde la aplicación cómo en forma ejecutada en el servidor. Aquí el escenario es modelado como una chanca medidas de una cancha de futbol de Small Size Robocup, los agentes son modelados como esferas con diámetro de 17cm de color gris, y los obstáculos son modelados con las mismas medidas pero de color rojo, la posición inicial de los robots está marcada con un borde de rectángulo azul y los puntos de las rutas están marcadas cómo un borde rectángulo rojo.

El siguiente modulo es la *Comunicación*, este módulo tendrá dos métodos, envió de comandos o envió de peticiones, el método de envió de comandos se encarga de enviarle el mensaje al servidor sin esperar respuesta, el método de envió de

peticiones se encarga de solicitar posición de un robot y las lecturas de sus sensores. Esté modulo es llamado por todos los enjambres generados.

El módulo *Algoritmos* se encarga de generar los enjambres y de lanzarlos como hilos, para cada enjambre se selecciona una ruta diferente, y unos agentes diferentes, así cada enjambre tiene una Clase de función de costo diferente. Y solamente su comportamiento se verá afectado por los agentes integrantes de ese enjambre.

Finalmente, el módulo de Thread OpenGL, integra toda la información generada por los enjambres donde cada uno de estos necesita conocer la ubicación de los otros agentes de otros enjambres esto con el fin de evitar colisiones entre diferentes enjambres, está información generada es enviada al módulo de OpenGL que se encarga de tomar esta información y graficar las partículas y enjambres. Se lanza un hilo de este módulo para la permanente actualización de la información de posiciones, partículas, y la actualización simuladora de OpenGl.

7.6.2. LISTA DE REQUERIMIENTOS FUNCIONALES

- ❏ R1. Establecer conexión UDP.
- ❏ R2. Ingresar parámetros de simulación.
- ❏ R3. Establecer ruta de búsqueda.
- ❏ R4. Reiniciar simulación.
- ❏ R5. Cargar partículas en el escenario.
- ❏ R6. Dar tiempo de simulación.
- ❏ R7. Indicar cuando las partículas encuentran el objetivo.

7.7. VALIDACIÓN DE EXPERIMENTOS

Se realizan las pruebas en la aplicación del cliente, donde se simula el algoritmo quinientas veces en búsqueda de un objetivo desconocido en cada uno de los cinco escenarios y teniendo en cuenta esto, se observa en cuál de estos escenarios se desenvuelven mejor las partículas.

Para el caso de los enjambres, se hace primero con dos enjambres de tres partículas, luego de tres enjambres de dos partículas y para los dos casos anteriores se realizan las mismas pruebas que el experimento anterior.

7.7.1. VALIDACIÓN REQUERIMIENTOS FUNCIONALES

Se validan los requerimientos funcionales de la aplicación donde se evidencia en las siguientes tablas:

Nombre	**R1. Establecer conexión UDP**
Resumen	**Se establece una conexión UDP entre un cliente y un servidor.**
Entradas	**IP del cliente y del servidor**
Resultados	**Conexión exitosa**

Nombre	**R2. Ingresar parámetros de simulación**
Resumen	**Se indican todos los parámetros con los cuales se harán la simulación.**
Entradas	**Máximas iteraciones, número de partículas, velocidad límite, peso inercial, el coeficiente global, ID del objetivo y el número de enjambres .**

Resultados	**Los parámetros ingresan correctamente y la simulación se hace con normalidad.**

Nombre	**R3. Establecer ruta de búsqueda**
Resumen	**Se establece una ruta que seguirán los enjambres en la búsqueda del objetivo.**
Entradas	**Puntos de la ruta que se lleva a cabo..**
Resultados	**Los enjambres realizan de manera exitosa la ruta..**

Nombre	**R4. Reiniciar simulación**
Resumen	**Se reinicia la simulación con los valores guardados anteriormente.**
Entradas	**Ninguna..**
Resultados	**Se reinicia exitosamente la simulación**

Nombre	**R5. Cargar partículas en el escenario**
Resumen	**Se carga la posición de las partículas del servidor en la interfaz gráfica.**
Entradas	**Posición “x” y “y” de las partículas**

Resultados	**Las partículas se cargan exitosamente.**

Nombre	**R6. Dar tiempo de simulación**
Resumen	**Indica cuál es el tiempo que se demoraron las partículas en encontrar el objetivo**
Entradas	**Ninguna**
Resultados	**Se muestra el tiempo de la simulación**

Nombre	**R7. Indicar cuando las partículas encuentran el objetivo**
Resumen	**Se muestra una bola azul en la simulación cuando todas las partículas encuentran el objetivo.** **Si los robots no encuentran el objetivo en el número de iteraciones establecido, se detendrá la simulación**
Entradas	**Ninguna**
Resultados	**Las partículas encuentran siempre el objetivo.**

7.8. RESULTADOS DEL PROYECTO

En las figuras ####, se muestran los resultados obtenidos de las simulaciones realizadas en cada escenario, variando el número de partículas y el número de enjambres como se explica anteriormente.

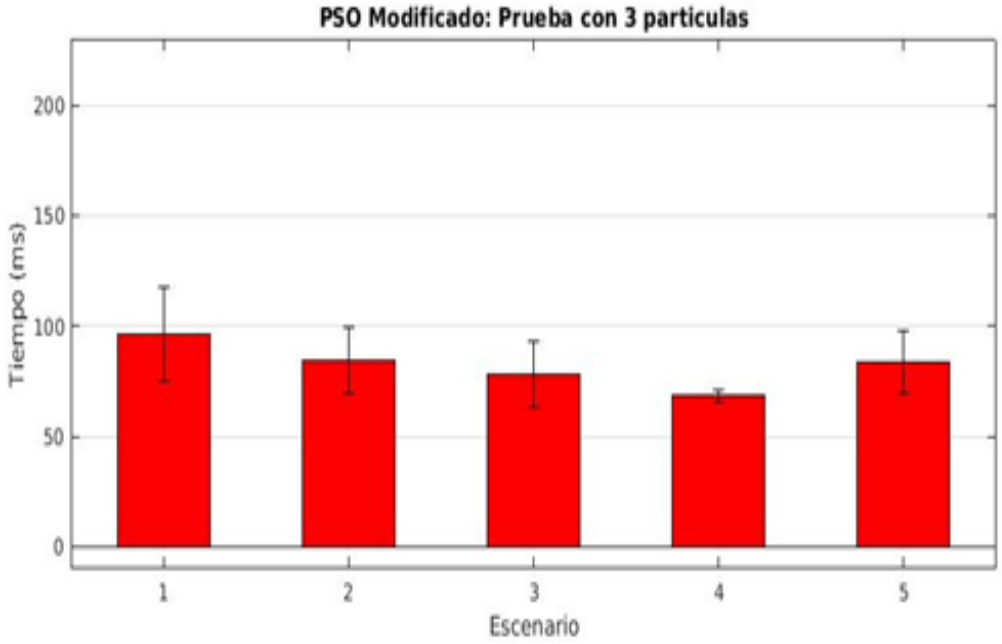

Figura 36. Datos obtenidos de las simulaciones con 3 partículas

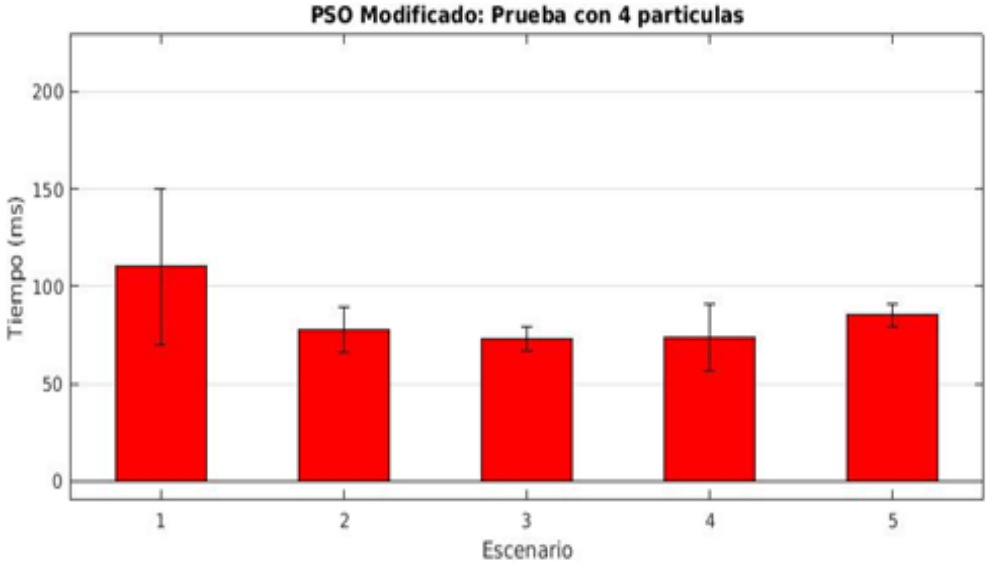

Figura 37. Datos obtenidos de las simulaciones con 4 partículas

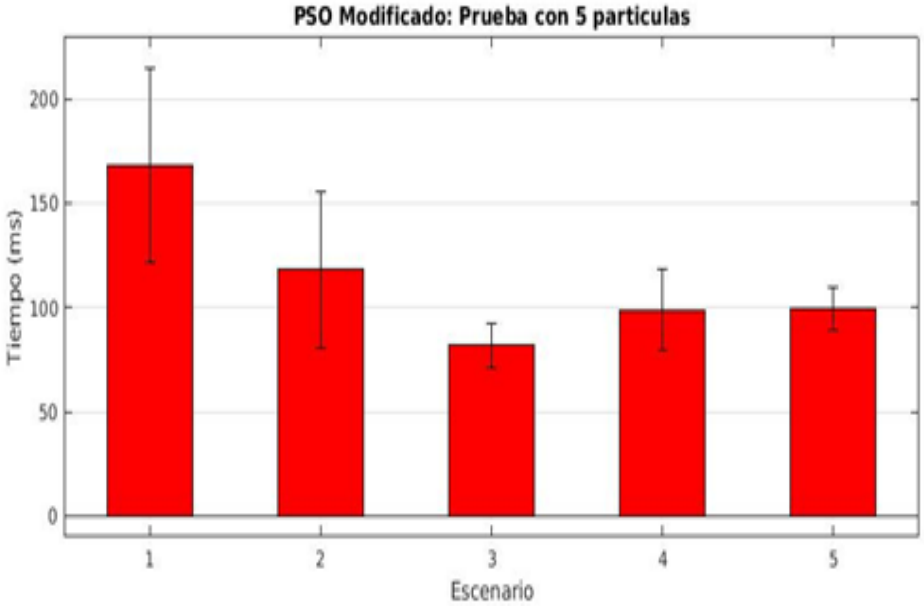

Figura 38. Datos obtenidos de las simulaciones con 5 partículas

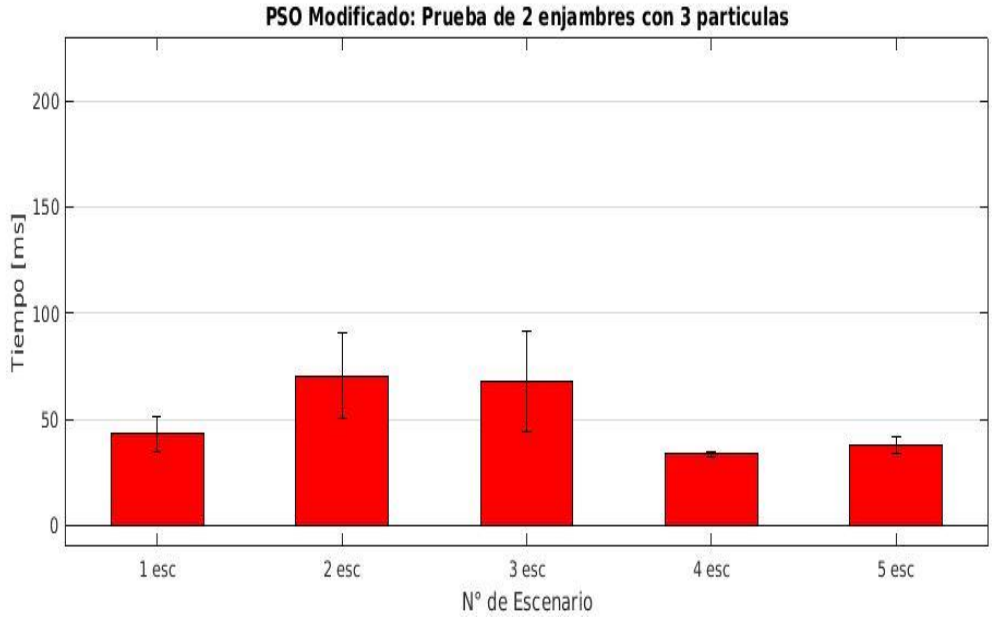

Figura 39. Promedio de datos obtenidos en las pruebas de 2 enjambres de 3 partículas en cada uno de los escenarios

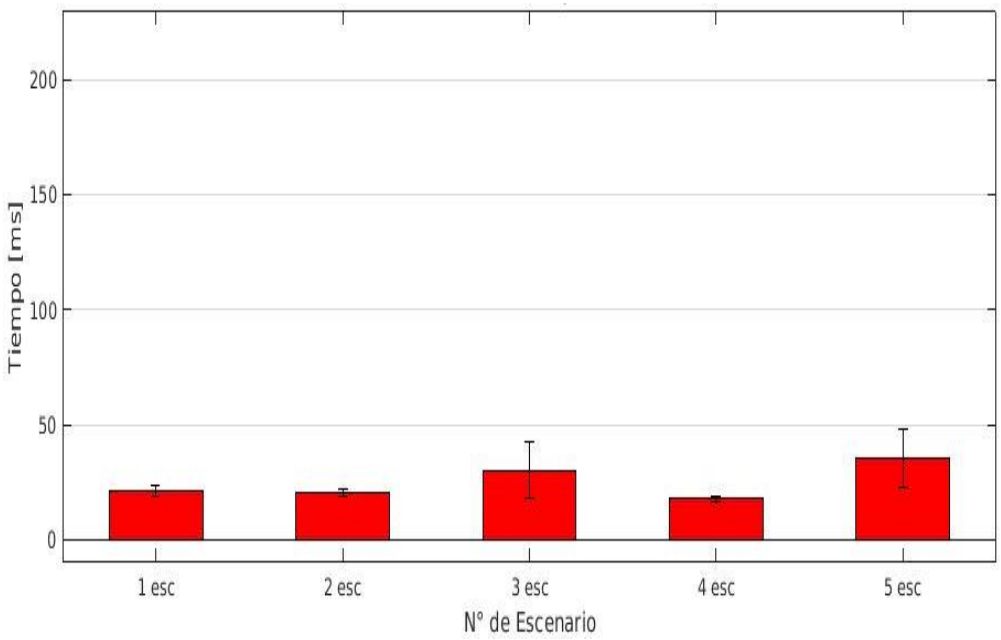

Figura 40. Promedio de datos obtenidos en las pruebas de 3 enjambres de 2 partículas en cada uno de los escenarios.

Como se puede observar, el tiempo que le toma a las partículas en localizar el objetivo se ve reducido considerablemente cuando se divide por enjambres. Esto se debe a que, al ser los enjambres independientes de los otros, poseen mayor campo de búsqueda, lo cual, reduce considerablemente el tiempo de localización del objetivo.

8. CONCLUSIONES

Se implementó una técnica bio inspirada con una modificación para que fuera más eficaz y se comparó mediante unas pruebas, qué técnicas era mejor para cumplir con los objetivos del proyecto.

Como se pudo observar, la técnica de PSO Original es de por sí una herramienta muy poderosa que permite la búsqueda del mínimo global de la función mediante el uso de la comunicación entre el enjambre de partículas, donde estas poseen dos tipos de comportamiento: individual y colectivo.

Dichos comportamientos están regidos por dos coeficientes, donde las partículas van en busca del mejor mínimo individual, y luego de uno global. Como las pruebas lo demuestran, este algoritmo tiene un gran problema cuando se encuentra en escenarios con múltiples obstáculos que ocasiona que las partículas se estanquen y no puedan seguir avanzando. Por esta razón, fue necesario implementar una heurística para el mejoramiento del algoritmo y para que este no volviera a ocurrir. La fusión de estos dos algoritmos generó uno más inteligente que el PSO Original, que le permitió desenvolverse de manera favorable en los escenarios generados con diferentes características para la evaluación de dicho algoritmo.

Si bien existen muchas técnicas bio inspiradas que hubiesen servido para cumplir los objetivos del proyecto, el PSO en especial fue considerado el más adecuado para el propósito de localización de objetos en campos no estructurados con múltiples obstáculos.

9. BIBLIOGRAFÍA

[1] Blog de robótica [en línea]: https://robotica.wordpress.com/about/

[2] Página oficial de DARPA [en línea]: http://www.darpa.mil/

[3] Página oficial de CRASAR [en línea]: http://crasar.org/

[4] Página oficial de Robocup Rescue [en línea]: http://www.robocup2016.org/en/leagues/robocup-rescue/

[5] Página oficial del concurso DARPA Robotics Challenge [en línea]: http://www.darpa.mil/program/darpa-robotics-challenge

[6] Cobo Ortega, A. & Serrano Bedia, A. (2005). Un algoritmo Híbrido basado en colonias de hormigas para la resolución de problemas de distribución en planta orientados a procesos (Ponencia). Universidad de Cantabria, Santander, España.

[7] Navarro García, D. (2009). Contribución a la auto localización de robots móviles basada en la fusión de información Multisensorial. (Tesis doctorado). Universidad Politécnica de Valencia, Valencia, España.

[8] Ezquerra Cerdán, C. (2006). Auto Localización de robots móviles y modelado del entorno mediante visión estereoscópica (Tesis de pregrado). Universidad Rovira i Virgili, Tarragona, España.

[9] Isaac Hernández (2015) Robots al rescate (Noticia el mundo.es)[en línea]: http://www.elmundo.es/ciencia/2015/10/09/5616a62e268e3e15768b463b.html.

[10] DARPA Robotics Challenge Finals 2015, Archivos de equipos concursantes [en línea]: http://archive.darpa.mil/roboticschallenge/index.htm

[11] K.M.Passino (2015).Biomimicry for Optimization, Contro, .and Automation.

[12] Gómez, Nelson & Maldonado, Carlos (2011). Sistemas bio-inspirados: Un marco teórico para la ingeniería de sistemas complejos [en línea]: http://pasaporte.urosario.edu.co/Administracion/documentos/Documentos-de-Investigacion/bi112web.pdf

[13] P. Bartashevich, L. Grimaldi and S. Mostaghim *IEEE Congress on*

Evolutionary Computation (CEC), Donostia, San Sebastián, Spain, 2017 pp. 882-889.

[14] S. Nguyen and M. Zhang, *IEEE Congress on Evolutionary Computation (CEC)*, Donostia, San Sebastián, Spain, 2017, pp. 882-889.

[15] K. Jaiswal and S. Nagar, *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, 2016, pp. 1-6.

[16] H. C. Huang, in *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 915-922, Aug. 2015.

[17] Rui Wu *et al*., *IEEE Congress on Evolutionary Computation (CEC)*, Donostia, San Sebastián, Spain, 2017, pp. 1893-1899.

[18] Yawen Zhou and J. Liu, *IEEE Congress on Evolutionary Computation (CEC)*, Donostia, San Sebastián, Spain, 2017, pp. 43-50.

[19] Marco Dorigo, Senior Member, IEEE, and Luca Maria Gambardella, Member, IEEE: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem