UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Aplicaciones Prácticas para Algoritmos de Inteligencia y Robótica de Enjambre

Trabajo de graduación presentado por Daniela María Baldizón García
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

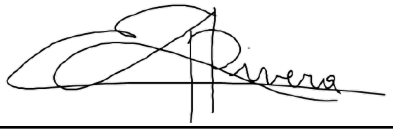
Aplicaciones Prácticas para Algoritmos de Inteligencia y Robótica de Enjambre

Trabajo de graduación presentado por Daniela María Baldizón García
para optar al grado académico de Licenciada en Ingeniería Mecatrónica

Guatemala,

2022

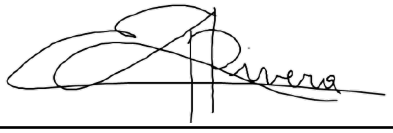
Vo.Bo.:



(f) _____

Dr. Luis Alberto Rivera Estrada

Tribunal Examinador:



(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

M.Sc. Miguel Enrique Zea Arenales

(f) _____

M.Sc. Pablo Roberto Oliva Fonseca

Fecha de aprobación: Guatemala, 5 de enero de 2022.

Prefacio

La elaboración de este trabajo requirió utilizar conocimientos que abarcan varias áreas de investigación, tales como robótica, programación de algoritmos y sistemas de control. Fue posible la realización de este trabajo gracias a todos los conocimientos adquiridos a lo largo de la carrera. En este trabajo se presentan aplicaciones para algoritmos de inteligencia computacional e inteligencia de enjambre. Una de las motivaciones de realizar este trabajo es brindar herramientas que eviten poner en riesgo la vida de las personas en exploraciones; y brindar herramientas que puedan dar resultados más exactos para el procesamiento de imágenes médicas, y que no son sencillos de obtener mediante el ojo humano.

Agradezco a Dios, a mis padres por brindarme la oportunidad de estudiar en la Universidad del Valle de Guatemala. Le agradezco a mi asesor de tesis Dr. Luis Alberto Rivera por compartir sus conocimientos conmigo y ser un excelente catedrático. De igual manera, agradezco a todas las personas que conocí durante esta etapa de estudio, ya que tengo buenas memorias de las experiencias compartidas con ellas.

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo principal de este proyecto es la implementación de algoritmos de inteligencia computacional y robótica de enjambre, y desarrollar aplicaciones para dichos algoritmos. Para ello se implementó el algoritmo *Ant Colony Optimization* modificado, para que este resuelva problemas de exploración de terrenos, planificación de trayectorias y evasión de obstáculos, y problemas de procesamiento de imágenes biomédicas. Asimismo, para estos algoritmos se realizaron simulaciones realistas y, en algunos casos, se tomó en cuenta restricciones físicas que afectan el funcionamiento del algoritmo, como lo son los obstáculos y velocidad.

Para los problemas de exploración de terrenos, planificación de trayectorias y evasión de obstáculos se propuso dos algoritmos. El primero se enfoca en la planificación de trayectorias y evasión de obstáculos, y el segundo en explorar terrenos desconocidos y realizar mapeos de estos. Ambos algoritmos se implementaron en Matlab y se tomó como base la implementación del *Ant Colony Optimization* desarrollada en un trabajo anterior. Los algoritmos se validaron utilizando tres mapas. Se logró planificar trayectorias que evaden obstáculos, y replicar mapas explorados.

Se validó el funcionamiento del algoritmo de planificación de trayectorias en la plataforma *Webots* utilizando robots diferenciales. Para esto, se tomaron las trayectorias generadas por el algoritmo. Se implementaron los controladores LQI, de pose y de pose de Lyapunov.

Para los problemas de procesamiento de imágenes médicas se propuso un algoritmo que realiza registro de imágenes médicas. La implementación de dicho algoritmo se realizó en Matlab y se tomó como base la implementación del *Ant Colony Optimization* desarrollada en un trabajo anterior. La finalidad de este algoritmo es transformar una imagen distorsionada en una imagen de referencia. Se validó la implementación utilizando imágenes de prueba e imágenes médicas. Se logró transformar las imágenes distorsionadas en las imágenes de referencia y obtener la transformación que se debe emplear a la imagen distorsionada para obtener la imagen registrada.

Abstract

The main objective of this research thesis is the implementation of artificial intelligence algorithms and swarm robotics, and develop applications for those algorithms. To achieve this goal, the Ant Colony Optimization algorithm was modified and implemented to resolve environment exploration, path planning, obstacle evasion and medical image processing problems. Realistic simulations were made in order to test if the algorithm achieved its designated goal. In some cases, physical restrictions were considered, like obstacles and velocity.

Two algorithms were implemented in order to solve the environment exploration, path planning and obstacle avoidance problems. The first one was implemented in order to plan a trajectory on an obstacle course and the other algorithm was implemented to explore unknown environments and map them. Both of these algorithms were programmed on Matlab using the previously developed Ant Colony Optimization algorithm. This algorithm was developed prior this investigation and has been modified to meet the desired goals. This algorithms were validated in three maps. It was possible to plan trajectories that evade obstacles, and replicate explored maps.

The Webots tool was used in order to validate the path planning and obstacle avoidance algorithm. This algorithm was validated on a differential drive. For this, the trajectories generated for the algorithm were used. The controllers implemented in the robot were: Linear Quadratic Integrator (LQI), pose controller and Lyapunov's pose controller.

For problems of medical image processing, an algorithm of medical image registration was proposed. The implementation of this algorithm was made in Matlab using the previously developed Ant Colony Optimization algorithm. The purpose of this algorithm is to transform a distorted image into a reference image. The implementation was validated using test images and medical images. It was possible to transform the distorted images into the reference images, and obtain the transformation that has to be applied in the distorted image to get a registered image.

CAPÍTULO 1

Introducción

La inteligencia de enjambre se centra en los comportamientos colectivos de muchos individuos que interactúan entre sí y con su entorno, de la misma forma que lo hacen los enjambres en la naturaleza. Algunas de las especies estudiadas por la inteligencia de enjambre son: abejas, hormigas, peces, termitas o animales terrestres. Esto puede utilizarse en programas de computadora donde se requiere optimización, o en sistemas de múltiples robots.

El *Ant Colony Optimization* (ACO) se basa en el comportamiento de las hormigas en buscar alimento. Existen variantes para este algoritmo, como lo es *Ant System* (AS). La idea de estos algoritmos surgió por un experimento, en el que las hormigas escogían el camino más corto desde la colonia hasta alguna fuente de comida. Algunas aplicaciones del ACO son: redes neuronales, inteligencia artificial, procesamiento de imágenes y enrutamiento de vehículos.

En este trabajo se presentan aplicaciones para algoritmos de inteligencia computacional e inteligencia de enjambre. Las aplicaciones propuestas se basan en el ACO. Se tomó como base para las aplicaciones la implementación de los algoritmos *Simple Ant Colony* y *Ant System*, desarrollada en un trabajo de graduación anterior. La primera aplicación se enfoca en planificación de trayectorias y evasión de obstáculos, donde las hormigas evitan las trayectorias con pasillos estrechos, donde físicamente es imposible que pase un robot. Esta aplicación se validó en Webots y así considerar en las trayectorias las restricciones físicas de los robots.

La segunda aplicación se enfoca en la exploración de terrenos. Para esto se modificó la aplicación de planificación de trayectorias y evasión de obstáculos, para que las hormigas pudieran almacenar los nodos en los que detectaban obstáculos, para luego reconstruir el mapa explorado con la información obtenida.

La tercera y última aplicación de este trabajo se enfoca en la implementación de aplicaciones biomédicas. Para esto, se implementó un algoritmo para registro de imágenes médicas. Con este algoritmo se obtiene la transformación necesaria para llevar una imagen distorsio-

nada a una imagen objetivo, y con esto obtener una imagen registrada.

CAPÍTULO 2

Antecedentes

2.1. Robotarium de Georgia Tech

El proyecto de Robotarium del Tecnológico de Georgia en Estados Unidos proporciona una plataforma de investigación de robótica de enjambre accesible de forma remota de libre acceso. El objetivo de este proyecto es que cualquier persona pueda cargar y probar sus ideas en un hardware robótico real, y así librar las inversiones significativas en mano de obra. Para utilizar el Robotarium solo se necesita descargar el simulador de MATLAB o Python y registrarse en la página para recibir la aprobación de la administración del Robotarium [1].

2.2. Algoritmo D*

Es un algoritmo popular para planificación de trayectorias de robots. Este algoritmo encuentra la mejor trayectoria a través de un grafo, para esto primero calcula el grafo, el cual corresponde a la cuadrícula de ocupación de entrada [2].

Primero, el algoritmo generaliza la cuadrícula de ocupación a un mapa de costo que representa el costo de moverse entre cada celda en dirección horizontal o vertical. Segundo, el algoritmo permite la replanificación incremental. Esto último es importante si mientras se están moviendo los robots, se descubre que el lugar es diferente al mapa que se tiene. Si se descubre que una ruta tiene más costo del esperado o está bloqueada, se puede replanificar para encontrar una mejor trayectoria [2].

D* encuentra el camino que minimiza el costo total de moverse. Si se está interesado en el tiempo mínimo que para alcanzar una meta entonces el costo es el tiempo que toma cruzar una celda [2].

2.3. Aplicaciones de Ant Colony Optimization (ACO)

En [3] se encuentra un trabajo en el que se utiliza el ACO para registro de imágenes médicas. Toman una imagen a la que nombran imagen plantilla y el algoritmo se encarga de realizar transformaciones en dicha imagen para que sea lo más parecida posible a una imagen de referencia. La imagen generada por el algoritmo es la imagen registrada, y se puede ver de qué forma se realizó la transformación de la imagen por medio de un mapa de cuadrícula de deformación generado por el algoritmo. Este algoritmo es utilizado para el registro de imágenes médicas.

En [4] proponen un algoritmo basado en el ACO que garantiza que los robots encuentran un camino satisfactorio en presencia de pasillos estrechos. Este trabajo se centra específicamente en un caso de dos dimensiones, y consiste en crear el ambiente en el que se desea que los robots encuentren la trayectoria hasta el punto especificado, y si dentro de esta trayectoria se encuentra un pasillo estrecho, este pueda ser tomado como parte de la trayectoria o esquivado. Además dentro del ambiente se encuentran obstáculos que son evadidos de forma satisfactoria. En este trabajo comparan este algoritmo con algoritmos similares dedicados a la generación de trayectorias con ACO, y se obtuvo que este converge más rápido.

2.4. Aplicaciones de Particle Swarm Optimization (PSO)

En [5] se propone un algoritmo que mezcla el PSO con campos de Markov ocultos (HMRF por sus siglas en inglés). El objetivo de este algoritmo es implementar un método para la segmentación de imágenes médicas, pues este es uno de los problemas fundamentales en el campo de la segmentación de imágenes. Por medio de los HMRF se encuentra la función de costo a optimizar por cada segmento de la imagen, y luego se optimiza la función encontrada mediante el PSO. En este artículo se realiza la segmentación específicamente para imágenes cerebrales.

2.5. Implementación de Algoritmos de Inteligencia de Enjambre en UVG

En la fase anterior a este trabajo [6] se implementó el algoritmo *Ant System* (AS) (o *Ant Colony* (ACO)) como planificador de trayectorias. Dicho algoritmo se basa en el comportamiento de las hormigas para buscar y hallar alimento. Con esta implementación se obtuvo una alternativa de planificación de trayectorias a parte de la del *Modified Particle Swarm Optmimization* (MPSO), desarrollada en el proyecto Robotat [7]. Estos algoritmos de inteligencia de enjambre tienen diferentes enfoques, el de AS es basado en teoría de grafos y el del MPSO es en funciones de costo.

Se encontraron los parámetros para el correcto funcionamiento del AS y se validaron por medio de simulaciones computarizadas que permiten visualizar el comportamiento de las feromonas depositadas por la colonia. Se adaptaron los modelos de movimiento y de cinemática de robots E-Puck al ACO, y así se logró realizar la implementación en Webots

para comparar su desempeño con el del MPSO.

Se implementó distintos controladores para considerar que los motores que controlan al robot tienen un límite de velocidad y garantizar que este haga lo deseado. Entre los controladores que se implementaron están: Transformación de uniciclo (TUC), PID de velocidad lineal y angular, PID de acercamiento exponencial, de pose, de pose de Lyapunov, Closed-loop steering, Regulador cuadrático lienal (LQR) y Integrador cuadrático linal (LQI). Se concluyó que los modelos de movimiento de los Bitbots desarrollados en el proyecto Robotat [7] funcionan tanto en AS como en MPSO.

Se realizaron pruebas con computación paralela para agilizar la planeación de trayectorias de los robots con el Ant System. También se realizaron pruebas con algoritmos genéticos para explorar si estos pueden ser alternativa al AS y MPSO, pero los resultados no fueron satisfactorios.

En los trabajos anteriores no se especificó ninguna tarea concreta para llevar a cabo durante la ejecución de los algoritmos, es decir, únicamente se implementaron los algoritmos como tal, sin asignarles alguna aplicación específica. Algunas de las simulaciones de estos trabajos tampoco toman en cuenta obstáculos que puedan presentarse en las trayectorias de los robots.

CAPÍTULO 3

Justificación

En los trabajos previos se implementaron los algoritmos *Ant Colony* y *Modified Particle Swarm Optimization* como planificadores de trayectorias. También se realizaron simulaciones utilizando la plataforma de Webots para validar el funcionamiento de dichos algoritmos tomando en cuenta las restricciones físicas de los robots. Sin embargo, en estas simulaciones únicamente se validó el funcionamiento de los algoritmos sin tomar en cuenta obstáculos y restricciones que pudiera presentar una aplicación real. Debido a que no se tomaron en cuenta estas restricciones, se desearía implementar los algoritmos de tal forma que se puedan lograr simulaciones más realistas, además se quisiera orientar la implementación de los algoritmos a aplicaciones prácticas.

Uno de los campos en los que se quiere explorar aplicaciones es el de biomédica. Esto debido a que en la actualidad algunos métodos utilizados en el campo de la medicina aún son muy rudimentarios y requieren de la intervención de médicos o especialistas, por lo que al explorar aplicaciones biomédicas se puede obtener una herramienta que facilite el trabajo en el área de la medicina y brinde resultados más exactos.

Otro campo en el que se quiere implementar aplicaciones es el de exploración de terrenos, planificación de trayectorias y evasión de obstáculos. Esto se debe a que el trabajo de exploración y planificación puede llegar a ser muy riesgoso para las personas y se exponen vidas humanas al realizar estos trabajos, ya que en algunos casos puede llegar a ser muy peligroso. Entonces, es más práctico que estos problemas sea resueltos por un algoritmo que puede brindar resultados óptimos y sin exponer vidas.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Implementar algoritmos de inteligencia computacional y robótica de enjambre, y desarrollar escenarios prácticos simulados y aplicaciones para dichos algoritmos.

4.2. Objetivos específicos

- Implementar algoritmos para exploración de terrenos, planificación de trayectorias, y evasión de obstáculos.
- Implementar algoritmos para aplicaciones biomédicas.
- Validar la implementación de los algoritmos mediante simulaciones realistas y restricciones físicas.

CAPÍTULO 5

Alcance

El alcance de este trabajo abarcó la implementación simulada de aplicaciones para algoritmos de inteligencia computacional y robótica de enjambre. Para esto, se tomó como base los algoritmos implementados en la fase anterior a este proyecto, y se modificaron para obtener la implementación de aplicaciones para estos algoritmos. Se implementaron algoritmos modificados que tenían como finalidad la planificación de trayectorias, evasión de obstáculos y exploración de terrenos.

Para la planificación de trayectorias, el algoritmo es capaz de encontrar la trayectoria entre dos puntos en un mapa representado como una cuadrícula. En la evasión de obstáculos se evita que la trayectoria encontrada pase por una cuadrícula que contiene un obstáculo, y que se eviten los caminos que tienen pasillos estrechos, por donde un robot no podría pasar debido a las restricciones físicas de este. En la exploración de terrenos se encuentra la trayectoria entre dos puntos, y se reconstruye el mapa explorado con la información que guardaron los agentes del algoritmo.

En primer lugar, se implementaron los algoritmos antes mencionados en Matlab. La implementación toma a los robots como partículas sin masa y sin restricciones de actuadores. Dentro de la cuadrícula los robots pueden moverse entre los centros de los cuadros, por lo que las trayectorias tienen únicamente ángulos de 0°, 45° y 90° respecto al movimiento anterior.

Para tomar en cuenta las restricciones físicas de los robots, se utilizó Webots para simular las trayectorias. Se validó la generación de trayectorias utilizando al robot e-puck. Se implementaron controladores para la velocidad de las ruedas de los robots, y así considerar restricciones de velocidad. En esta parte se utilizó la trayectoria ya generada y con controladores punto a punto se llevó al robot desde el inicio a la meta.

Para futuros trabajos se podrán implementar estos algoritmos en los Bitbots físicos de UVG. También se podrían migrar los algoritmos a otro lenguaje de programación y así comparar o mejorar su desempeño.

CAPÍTULO 6

Marco teórico

6.1. Particle Swarm Optimization (PSO)

Es un algoritmo de optimización meta-heurístico ya que utiliza analogías con otros procesos para resolver un problema. El algoritmo se inspira en la evolución del comportamiento colectivo, trata de imitar el comportamiento social de grupos de animales como manadas, cardúmenes, parvadas, etc. Los métodos meta-heurísticos no se enfocan en resolver un problema en particular, por lo que estos pueden emplearse en cualquier problema y obtener un resultado aceptable. El PSO no es determinista, lo que significa que los resultados obtenidos no siempre serán los mismos aunque se trate de una misma función. Los algoritmos de enjambre de partículas se caracterizan por ser eficientes y de bajo costo computacional [8].

A lo largo del tiempo se han propuesto variaciones para el PSO, esta sección se enfoca en la versión inercial o clásica. El objetivo de un problema de optimización es encontrar un vector $X = [x_1, x_2, x_3, \ldots, x_n]$ que minimice o maximice cierta función $f(X)$. El vector variable X es el vector de posición que representa un modelo variable y es de dimensión n, donde n es la cantidad de variables que hay que encontrar en el problema. La función $f(X)$ es la función peso, esta evalúa que tan bien o mal es la posición X [8].

Tomando un enjambre con P cantidad de partículas, hay un vector posición $X_i^t = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in})^T$ y un vector velocidad $V_i^t = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in})^T$ a una t-iteración para cada una de las partículas que compone al enjambre. Estos vectores son actualizados a través de una dimensión j de acuerdo a las siguientes ecuaciones [8]:

$$V_{ij}^{t+1} = wV_{ij}^t + c_1 r_1^t \left(pbest_{ij} - X_{ij}^t\right) + c_2 r_2^t \left(gbest_j - X_{ij}^t\right) \tag{1}$$

$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1} \tag{2}$$

Donde $i = 1, 2, \ldots, P$ y $j = 1, 2, \ldots, n$.

La ecuación (1) actualiza la velocidad y (2) actualiza la posición de las partículas. w es la constante de inercia peso. El segundo término representa la cognición individual y es calculado por medio de la diferencia entre la mejor posición de la propia partícula (*pbest*) y su posición actual. La idea detrás de este término es que mientras la partícula está más lejos de su mejor posición la diferencia aumenta; por consecuencia, el término aumenta atrayendo a la partícula a su mejor posición. El parámetro c_1 multiplica al término como una constante positiva y es un parámetro individual-cognición, y marca la importancia de las experiencias anteriores de cada partícula. El parámetro r_1, es de valor aleatorio con rango [0,1] y evita las convergencias prematuras. [8].

El tercer término de (1) es el de aprendizaje social. Esto es así debido a que todas las partículas del enjambre pueden intercambiar información sobre el mejor punto alcanzado sin importar cual lo encontró (*gbest*). La diferencia que se muestra en este término actúa como atractor para las partículas al mejor punto global encontrado en una iteración t. Similarmente, c_2 es el parámetro de aprendizaje social, y su valor representa la importancia del aprendizaje global del enjambre. Y r_2 tiene el mismo rol que r_1 [8].

6.2. Ant Colony Optimization (ACO)

Uno de los comportamientos que han estudiado los entomólogos es la habilidad que tienen las hormigas de encontrar el camino más corto entre su hormiguero y las fuentes de alimento. A partir de estos estudios y observaciones, Marco Dorigo desarrolló el primer algoritmo que modela el comportamiento de las hormigas al buscar alimento, dicho algoritmo fue llamado como Ant Colony Optimization Meta-Heuristic (ACO-MH). Después de esto, se han desarrollado numerosas variantes para dicho algoritmo, como lo son el Ant System (AS), Ant Colony System (ACS), Max-Min Ant System (MMAS), Ant-Q, Fast Ant System, Antabu, AS-Rank y ANTS [9].

Las hormigas reales empiezan la búsqueda de alimento comportándose inicialmente de forma aleatoria o con un patrón de actividad caótica. Al encontrar una fuente de alimento, los patrones de actividad se vuelven más organizados con más hormigas siguiendo el mismo camino hacia la fuente de alimento. Lo que ocurre es que las hormigas que encuentran el alimento influencian a las demás hacia la comida, este comportamiento es resultado del mecanismo de reclutamiento de las hormigas. La mayoría de las especies de hormigas utilizan como forma de reclutamiento la comunicación indirecta, la cual hacen por medio de rastros de feromonas y se denomina estigmergia. Cuando una hormiga encuentra una fuente de alimento, esta regresa al hormiguero por el mismo camino llevando alimento. Al hacer esto deja rastros de feromonas en el camino para indicar a las demás hormigas qué camino seguir. Las hormigas recolectoras deciden qué camino seguir basándose en las concentraciones de feromonas de los diferentes caminos. Los caminos con mayor concentración de feromonas tienen una mayor probabilidad de ser elegidos. Mientras más hormigas sigan el mismo camino, aumenta la concentración de feromonas, por lo que este camino es el que más será utilizado por el conjunto de hormigas. El comportamiento colectivo resultante es una forma de comportamiento auto-catalítica, donde la retroalimentación positiva de un camino hacia el alimento causa que este camino sea seguido por las demás hormigas [9].

Deneubourg estudió el comportamiento de las hormigas por medio de un experimento, en

el que el hormiguero está separado de la fuente de alimento por un puente con dos caminos de la misma longitud. Al pasar el tiempo, las hormigas eligieron de manera aleatoria uno de los dos caminos para que fuera la ruta hacia la fuente de alimento. Este experimento es conocido como "El puente binario". Goss extendió el experimento del puente binario haciendo que uno de los caminos fuera más largo que el otro. Las hormigas seleccionaron el camino más corto como su ruta hacia el alimento. Como las hormigas que van por el camino corto regresan antes al hormiguero, la cantidad de feromonas en este camino es reforzada antes que en el camino largo. La probabilidad de las hormigas de escoger el camino más corto aumenta al incrementar la diferencia del largo de los caminos, este efecto se conoce como "largo diferencial" [9].

Los caminos más cortos tendrán una mayor concentración de feromonas al transcurrir el tiempo, esto se debe a que las hormigas regresan antes por esos caminos. Las feromonas se evaporan con el tiempo, y como consecuencia de esto las concentraciones de feromonas decrecen más rápido en los caminos largos. La feromona artificial imita las características de la feromona real, indica la popularidad de una solución del problema de optimización que se está resolviendo. De hecho, la feromona artificial funciona como una memoria a largo plazo del proceso completo de la búsqueda [9].

6.2.1. Simple Ant Colony Optimization (SACO)

El SACO es una implementación algorítmica del experimento del puente binario de Deneubourg. Considerando el problema general de encontrar el camino más corto entre dos nodos en un grafo $G = (V, E)$, donde V es el conjunto de vértices o nodos y E es la matriz que representa la conexión entre los nodos. El largo L^k del camino construido por la hormiga k es calculado como la cantidad de saltos en el camino desde el nodo que representa al nido hasta el que representa el destino con la comida. En la Figura 1 se muestra un ejemplo de un grafo y el camino seleccionado. El camino elegido está marcado con flechas continuas y su longitud es 2. Cada arista (i, j) del grafo tiene una concentración $\tau_{i,j}$ asociada [9].

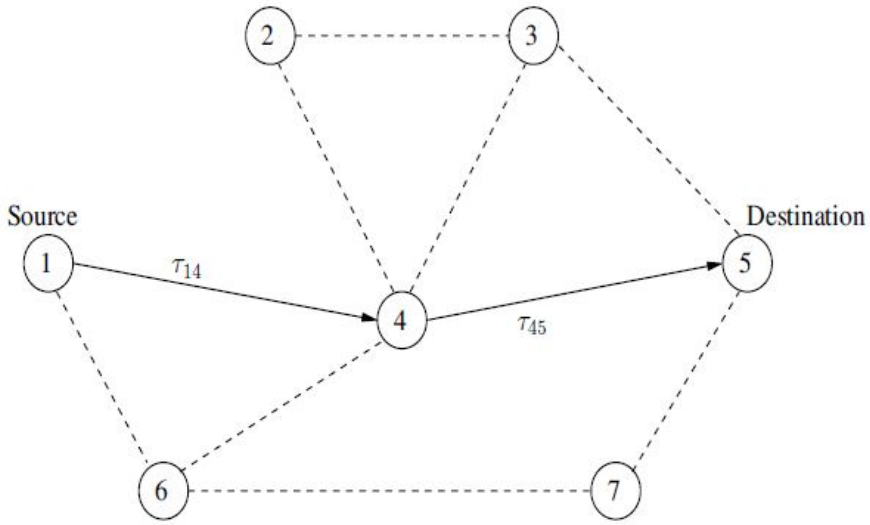


Figura 1: Ejemplo de grafo para el camino más corto [9].

En el algoritmo, a cada arista se le asigna un valor aleatorio pequeño de feromona ($\tau_{ij}(0)$) (aunque en la realidad esto debería ser cero). Luego, las hormigas deciden de forma aleatoria que camino/arista seguir. k hormigas se colocan en el nodo fuente. En cada iteración del algoritmo cada hormiga construye un camino (solución) hacia el nodo de destino. En cada nodo i, cada hormiga k selecciona el siguiente nodo j, basándose en la probabilidad de transición,

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)}{\sum_{j \in \mathcal{N}_i^k} \tau_{ij}^\alpha(t)} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases} \tag{3}$$

donde $\mathcal{N}_i^k$ es el conjunto de nodos viables conectados al nodo i, respecto a la hormiga k. Si se llega a dar el caso $\mathcal{N}_i^k = 0$, entonces el nodo anterior se incluye al conjunto. α es una constante positiva que amplifica la influencia de las concentraciones de feromona, es decir, para valores grandes de α la importancia de la feromona es grande, especialmente para los valores iniciales aleatorios de feromona, lo cual puede hacer que la convergencia sea rápida y resulte en caminos no óptimos [9].

Una vez todas las hormigas han construido un camino completo desde el nodo de origen al nodo de destino, cada hormiga vuelve a recorrer su camino hacia el nodo fuente, y deposita una cantidad de feromona,

$$\Delta\tau_{ij}^k(t) \propto \frac{1}{L^k(t)} \tag{4}$$

en cada arista (i, j) del correspondiente camino. $L^k(t)$ es el largo del camino construido por una hormiga k en el tiempo t. De esta forma, considerando que la feromona puede evaporarse y así evitar la convergencia temprana y no óptima, la cantidad de feromona en cada arista está dada por

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^k(t) \tag{5}$$

donde n_k es la cantidad de hormigas. La constante $\rho \in [0,1]$ y especifica la tasa a la que las feromonas se evaporan, causando que las hormigas "olviden" las decisiones anteriores. Para valores grandes de ρ la feromona se evapora más rápido. Mientras más feromonas se evaporen, la búsqueda se vuelve más aleatoria [9].

Se sabe que

$$\Delta\tau_{ij}^k(t) = \frac{Q}{L_k}(t) \tag{6}$$

donde Q es una constante y L es el costo del trayecto, como el largo de este. El resultado de esto representa el cambio de feromona entre el nodo i y j que la hormiga visitó en la iteración t [10].

6.2.2. Ant System (AS)

Este algoritmo es una mejora del SACO (A pesar que el AS se desarrolló antes). La probabilidad de transición cambia para incluir la información heurística y agregar capacidad de memoria con una lista tabú, la cual previene que una hormiga visite el mismo nodo dos veces [10]. La ecuación de probabilidad de transición es,

$$p^k_{(i,j)}(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij}(t))^\beta}{\sum_{k \in J_k} (\tau_{ij}(t))^\alpha \cdot (\eta_{ij}(t))^\beta} & \text{si j} \in N^k_i \\ 0 & \text{si j} \notin N^k_i \end{cases} \tag{7}$$

donde η_{ij} representa la efectividad *a priori* del movimiento desde i a j, o el inverso del costo de la arista [9].

Por lo tanto,

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{8}$$

donde d_{ij} es la distancia (o costo) entre los nodos i y j [9].

6.3. Aplicaciones del Ant Colony Optimization

Los algoritmos de ACO se utilizan normalmente para resolver problemas no determinísticos, los cuales son aquellos que no se pueden solucionar por algoritmos con estructura polinomial, por lo que requieren del tipo exponencial. Para la solución, se realizan iteraciones con las que se pueden encontrar respuestas globales y particulares [11].

Algunos campos de aplicación de los algoritmos por optimización por colonia de hormigas son: redes neuronales, inteligencia artificial, optimización de funciones numéricas, sistemas difusos, procesamiento de imágenes, control de sistemas, problemas del hombre viajero, enrutamiento de vehículos y líneas de producción de carros [11].

6.3.1. Planificación de trayectorias para robots y evasión de obstáculos

En [4] se muestra una mejora al ACO que tiene como fin garantizar que un grupo de robots encuentren una trayectoria satisfactoria en un ambiente que contiene pasillos estrechos. En este documento, se presenta una nueva matriz de adyacencia ponderada para determinar la dirección de paso y, por lo tanto, los pasillos estrechos se evitan rediseñando las reglas de paso. También, se introducen la mejor y la peor hormiga para el ajuste de feromonas para facilitar el proceso de búsqueda. En la Figura 2 se muestra un ejemplo de un pasillo estrecho.

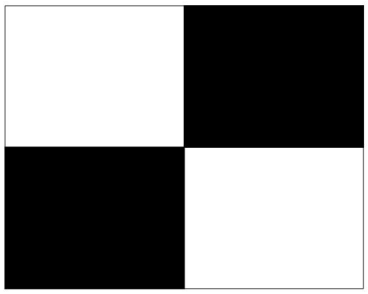


Figura 2: Pasillo estrecho.

Reglas para caminar

Se considera que los robots pueden moverse en ocho direcciones: superior, inferior, izquierda, derecha, superior izquierda, superior derecha, inferior izquierda e inferior derecha [4].

Cuando la condición (9) se cumple, la dirección del siguiente movimiento será superior, inferior, izquierda o derecha. Cuando la condición (10) se cumple, la dirección del movimiento será superior izquierda, superior derecha, inferior izquierda o inferior derecha. En estas condiciones i representa las coordenadas de la cuadrícula actual en el eje X, y j representa las coordenadas de la cuadrícula actual en el eje Y. m representa a las coordenadas de las cuadrículas vecinas en el eje X, y n representa las coordenadas de las cuadrículas vecinas en el eje Y [4].

$$i_m + j_n = 1 \tag{9}$$

$$\begin{aligned} i_m &= 1 \\ j_n &= 1 \end{aligned} \tag{10}$$

En la Figura 3 se muestran las ocho direcciones a las que puede moverse una hormiga. Según esta nomenclatura, se pueden distinguir los caminos estrechos en el mapa de acuerdo a las ecuaciones (11), (12), (13) y (14). Por lo tanto, se debe calcular la distancia para cada dirección [4].

(m-1,n-1)	(m-1,n)	(m-1,n+1)
(m,n-1)	(m,n)	(m,n+1)
(m+1,n-1)	(m+1,n)	(m+1,n+1)

Figura 3: Posición actual de la hormiga con 8 posibles próximas direcciones [4].

Las siguientes ecuaciones establecen los criterios de la existencia de un camino estrecho:

$$G(m, n-1) \neq 1 || G(m-1, n) \neq 1 \tag{11}$$

$$G(m, n-1) \neq 1 \| G(m+1, n) \neq 1 \tag{12}$$

$$G(m, n+1) \neq 1 \| G(m+1, n) \neq 1 \tag{13}$$

$$G(m, n+1) \neq 1 \| G(m-1, n) \neq 1 \tag{14}$$

Donde $G(m, n)$ es un escalar $\in R$ que denota un indicador de obstáculos, que también es un elemento de la matriz de entorno. Cuando su valor es 1 indica un obstáculo y cuando es 0 indica lo contrario [4].

Cuando la condición (11) se cumple, se conoce que hay un camino estrecho en la dirección superior derecha. Asimismo, la condición (12) indica que el camino estrecho está en la dirección inferior izquierda; la condición (13) corresponde a la dirección inferior derecha y (14) a la dirección superior derecha [4].

La dirección de movimiento global

Los pasillos estrechos aparecen en las direcciones superior derecha, superior izquierda, inferior derecha o inferior izquierda. Cuando se cumplen las condiciones (11)-(14), quiere decir que no hay pasillos estrechos alrededor; de lo contrario si hay [4].

Se introduce la matriz de adyacencia ponderada (D), la cual almacena la distancia entre las posiciones en el mapa. Esta matriz se calcula de la siguiente manera:

$$D((i-1) * l + j, (m-1) * l + n) = \left({i_m}^2 + {j_n}^2\right)^{0.5} \tag{15}$$

Donde l es el tamaño del mapa. El objetivo es que por cada nodo nuevo que se visite, la distancia entre este nodo y los demás solo sea calculada una vez [4].

Diseño de la función heurística

La probabilidad de la hormiga k de ir de la posición i a la posición j se obtiene con:

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}[\lambda_{jo}(t)]^{\gamma}}{\sum_{k\in\{N-Tabu_k\}}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}[\lambda_{jo}(t)]^{\gamma}}, & j \in \{N - Tabu_k\} \\ 0, & \text{otro} \end{cases} \tag{16}$$

Donde $\tau_{ij}(t)$ denota la concentración de feromona en el camino (i, j) al tiempo t; η_{ij} denota la visibilidad desde la posición i a la j; λ_{jo} indica que se avanza desde la posición j hasta el punto deseado o; α denota el coeficiente del factor heurístico de la información; β es el coeficiente de factor heurístico esperado y γ es el coeficiente de factor heurístico esperado. Se introduce el factor de orientación objetiva λ en la función heurística, el cual se obtiene como:

$$\lambda_{jo} = \frac{1}{S_{jo}} \tag{17}$$

Donde S_{jo} representa la distancia segura más corta entre la posición actual j y la posición objetivo o. Esta distancia se obtiene por medio de la matriz D. Mientras más larga sea la distancia más corta segura, el factor de guía es más pequeño; mientras más corta sea la distancia más corta segura, el factor de guía es más grande, y por lo tanto tiene mayor posibilidad de selección [4].

Diseño de la función de actualización óptima de las peores feromonas.

El algoritmo propuesto en [4] actualiza las feromonas en los caminos utilizando la ecuación (18). Donde ψ se calcula por medio de la ecuación (19). Esta ecuación es bastante similar a la planteada por Dorigo [9].

$$\tau(i, j) = (1 - \rho(\tau(i, j))) * \tau(i, j) + \psi(i, j) * \Delta\tau(i, j) \tag{18}$$

$$\psi(i, j) = \begin{cases} Q, & (i, j) \in \text{ Óptimo global} \\ -Q, & (i, j) \in \text{ Peor global} \\ 0, & (i, j) \in \text{ Otro} \end{cases} \tag{19}$$

Algoritmo propuesto

Los pasos de este algoritmo modificado se muestran a continuación [4]:

Paso 1. Recopilar la información del entorno y obtener el mapa de cuadrícula del entorno. Luego, establecer el punto de inicio y final. Inicializar los parámetros de la colonia de hormigas.

Paso 2. Determinar la ubicación actual y si hay un camino estrecho alrededor de la cuadrícula.

Paso 3. Calcular la distancia entre todas las cuadrículas y las cuadrículas circundantes de acuerdo a las reglas de caminar y construir una nueva matriz de adyacencia ponderada.

Paso 4. Colocar las m hormigas en el punto de inicio y elegir el siguiente nodo-j basándose en la distancia de la nueva matriz de adyacencia ponderada con la distribución de feromonas en diferentes posiciones.

Paso 5. Añadir el nodo-j a la lista tabú de la k-hormiga.

Paso 6. Repetir los pasos 4 y 5. La hormiga alcanza el punto objetivo, y guarda la ruta y la longitud de ruta de cada hormiga, de las cuales se selecciona la ruta óptima y la longitud de ruta óptima. La longitud de ruta promedio de las m hormigas se calcula en este ciclo.

Paso 7. Actualizar las feromonas de las rutas.

Paso 8. Determinar si el algoritmo alcanza el máximo número de iteraciones, luego terminar el algoritmo y generar la ruta óptima. Si esto no se cumple, repetir los pasos 3 al 7.

6.3.2. Registro de imágenes médicas

El registro de imágenes es un proceso que superpone dos o más imágenes de varios equipos de imágenes o sensores tomadas en diferentes momentos y ángulos, o de la misma escena para alinear geométricamente las imágenes para su análisis [12]. Las aplicaciones de registro de imágenes incluyen la combinación de imágenes del mismo sujeto de diferentes modalidades, la alineación de secuencias temporales de imágenes para compensar el movimiento del sujeto entre exploraciones, la guía de imágenes durante las intervenciones y la alineación de imágenes de múltiples sujetos en estudios [13]

El objetivo de registrar imágenes es encontrar una función de correspondencia de coordenadas de mapeo desde una imagen de origen, a coordenadas de puntos homólogos en una imagen de destino. En aplicaciones clínicas, esto se puede utilizar para hacer coincidir imágenes del mismo paciente en un período de tiempo [14].

El registro de imágenes médicas podría implicar reunir toda la información de un paciente determinado. Los desarrollos recientes en el registro de imágenes médicas han sido impulsados por unificar la información de imágenes con el fin de hacer un mejor uso de ciertos tipos de información de imágenes para aplicaciones clínicas específicas o en la investigación médica [13].

En [3] se plantea un algoritmo basado en el ACO que realiza registro de imágenes. El algoritmo propuesto toma hormigas que se mueven en una imagen 2D para construir una matriz de feromonas, que representa la información de la ruta en cada ubicación de píxel de la imagen. Los movimientos de las hormigas están guiados por la variación local de los valores de intensidad de la imagen.

En el algoritmo desarrollado se toman los píxeles de las imágenes como el nido de un

enjambre de hormigas, y las hormigas son capaces de buscar la "comida" en su memoria. Luego, las hormigas depositan feromonas en los píxeles, que afectan el movimiento de las hormigas. El proceso de registro de actualización de la feromona, la dirección y la distancia de avance se repite hasta que el coeficiente de correlación entre las imágenes registradas y de referencia alcanza un máximo. Este algoritmo puede ser utilizado en diversas áreas, una de ellas es el registro de imágenes médicas.

Inicialización

Inicialmente hay $M \times N$ hormigas uniformemente asignadas a la imagen de entrada, la cual es de dimensión $M \times N$. Cada píxel es visto como un nodo y el valor inicial de la matriz de feromonas es establecido como una constante $\tau_{inicial}$. El desafío es definir qué tipo de comida necesita cada hormiga mediante el uso del *Ant Colony System* para registro de imágenes. En este enfoque, la comida es representada como la diferencia de intensidad entre dos imágenes médicas utilizando:

$$I_{\text{diff}}^{(n)}(l,m) = I_R(l,m) - I_T^{(n)}(l,m) \tag{20}$$

Donde $I_R(l,m)$ es la intensidad del nodo (l,m) en la imagen de referencia, $I_T^{(n)}(l,m)$ es la intensidad en el nodo (l,m) en la imagen de *template* o plantilla en la enésima iteración, $I_{\text{diff}}^{(n)}$ es la diferencia entre las imágenes de referencia y plantilla en la enésima iteración. Si $I_{\text{diff}}^{(n)} < 0$, la intensidad en el nodo (l,m) en la imagen plantilla es más grande que en la imagen de referencia; por lo tanto, la hormiga en ese nodo buscará intensidades más bajas en los vecinos como comida. Por otro lado, si $I_{\text{diff}}^{(n)} \geq 0$, la hormiga buscará nodos con mayores intensidades en la proximidad a (l,m).

Construcción

Luego de definir la comida de las hormigas, lo que sigue es determinar una función de probabilidad de transición para la transformación geométrica. En este enfoque, en el enésimo paso de construcción, una hormiga se mueve en el camino desde el nodo (l,m) hacia su nodo vecino (i,j) y regresa a (l,m) de acuerdo a la función de probabilidad definida como

$$p_{(i,j)}^{(n)}(l,m) = \frac{\left(\tau^{(n)}(i,j)\right)^{\alpha}\left(\eta^{(n)}(i,j)\right)^{\beta}}{\sum_{(i,j)\in\Omega_{(l,m)}}\left(\tau^{(n)}(i,j)\right)^{\alpha}\left(\eta^{(n)}(i,j)\right)^{\beta}} \tag{21}$$

Donde $\tau^{(n)}(i,j)$ es la cantidad de feromona en el nodo (i,j), $\Omega_{(l,m)}$ son los nodos vecinos del nodo (l,m), $\eta^{(n)}(i,j)$ representa la intensidad en el nodo (i,j), y n es el número de iteración. Los parámetros constantes α y β representan la influencia de la matriz de feromona y de la matriz de visibilidad, respectivamente.

La dirección de movimiento de una hormiga se selecciona de una de sus coordenadas vecinas que tiene la probabilidad más alta basada en (21). Luego de decidir la dirección, la

distancia de avance es calculada de acuerdo la búsqueda de alimento en (20) utilizando

$$\begin{cases} D_x^{(n)}(l,m) = \left| r\left(I_{\text{diff}}^{(n)}\right)\left(T_x^{(n)}(l,m)\right)\right| \\ D_y^{(n)}(l,m) = \left| r\left(I_{\text{diff}}^{(n)}\right)\left(T_y^{(n)}(l,m)\right)\right| \end{cases} \tag{22}$$

Donde $T_x^{(n)}(l,m)$ es el gradiente de la imagen de plantilla registrada en la dirección x a la enésima iteración, $T_y^{(n)}(l,m)$ es el gradiente de la imagen de plantilla registrada en la dirección y en la enésima iteración, y la constante r representa la escala de las distancias de avance $D_x^{(n)}(l,m)$ y $D_y^{(n)}(l,m)$ en el eje x y y, respectivamente.

Actualización

El problema restante es la actualización de la feromona. Esto se realiza luego de cada movimiento de todas las hormigas utilizando

$$\tau^{(n+1)} = e^{-\rho} \times \tau^{(n)} + \tau_{init} \tag{23}$$

Donde la constante ρ tiene valores entre 0 y 1, y es el coeficiente de decaimiento determinado al comienzo del algoritmo.

Resultados del algoritmo

Para evaluar de forma cuantitativa el algoritmo propuesto, se utilizan la suma de diferencias cuadradas (SSD) y el coeficiente de correlación (CC) entre la imagen de plantilla deformada y la imagen de referencia. Estas ecuaciones se muestran a continuación, respectivamente

$$SSD = \sum\sum \frac{\|A(x,y) - B(x,y)\|^2}{N} \tag{24}$$

$$CC = \frac{\sum\sum(A(x,y) - \bar{A}(x,y))\sum\sum(B(x,y) - \bar{B}(x,y))}{\sqrt{\sum\sum(A(x,y) - \bar{A}(x,y))^2 \sum\sum(B(x,y) - \bar{B}(x,y))^2}} \tag{25}$$

En la Figura 4 se muestran un resultado de la modificación del *Ant Colony Optimization* para registro de imágenes.

Figura 4: Ejemplo de registro de imágenes con ACO. Primera columna: imagen plantilla, segunda columna: imagen de referencia, tercera columna: imagen registrada, cuarta columna: mapa de cuadrícula de deformación [3].

6.4. Coeficiente de correlación

El coeficiente de correlación es una medida específica que cuantifica la intensidad de relación lineal entre dos variables en un análisis de correlación [15]. Comúnmente se simboliza con una r.

Para el caso de dos variables, la fórmula compara la distancia de cada dato puntual respecto a la media de la variable y utiliza esta comparación para indicar hasta qué punto la relación entre las variables se ajusta a una línea imaginaria trazada entre los datos [15].

El valor del coeficiente de correlación puede variar entre -1 y 1. Cuando este tiene un valor de 0, la relación lineal entre las variables es mala. En el caso de tener un valor de 1, la relación es buena.

El coeficiente de correlación puede encontrarse con la siguiente ecuación.

$$r = \frac{\sum \left[(x_i - \bar{x}) \left(y_i - \bar{y} \right) \right]}{\sqrt{\Sigma \left(x_i - \bar{x} \right)^2 * \Sigma \left(y_i - \bar{y} \right)^2}} \tag{26}$$

Donde x_i son los datos de la primer variable, $\bar{x}$ es la media la primer variable, y_i los datos de la segunda variable y $\bar{y}$ la media de la segunda variable.

6.5. Suma de diferencias cuadradas

La suma de diferencias cuadradas es un criterio que se utiliza para encontrar correspondencias [16]. En imágenes, la suma de diferencias cuadradas es una medida de coincidencia basada en las diferencias de intensidad píxel por píxel entre las dos imágenes [17]. Para obtener su valor se calcula la suma de cuadrados para el producto de la resta de píxeles entre dos imágenes [18].

6.6. Modelado de robots móviles con ruedas

Los robots móviles con ruedas pueden describirse y modelarse sólo con su cinemática. En este tipo de robot los motores se encuentran en las ruedas. La distancia recorrida s por cada rueda se encuentra mediante

$$s = r\varphi \tag{27}$$

donde r es el radio de la rueda y φ es el desplazamiento angular de la misma.

En el caso de robots móviles, se tiene un set de restricciones no integrables, es decir que si se integra el camino no se puede obtener información de dónde empezó. Estas restricciones no integrables también se conocen como restricciones no holonómicas, y se presentan como restricciones de velocidad de la configuración. Se quiere la cinemática del robot, pero se quieren establecer restricciones en la velocidad, por lo tanto se utiliza cinemática diferencial para los robots móviles [19].

Se tiene la condición de no deslizamiento, en la cual los robots no pueden desplazarse hacia los lados. Estas restricciones de movimiento son fáciles de ver en el marco de referencia del robot, donde la velocidad se representa como

$${}^{B}\dot{\xi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \\ 0 \\ \omega \end{bmatrix} \tag{28}$$

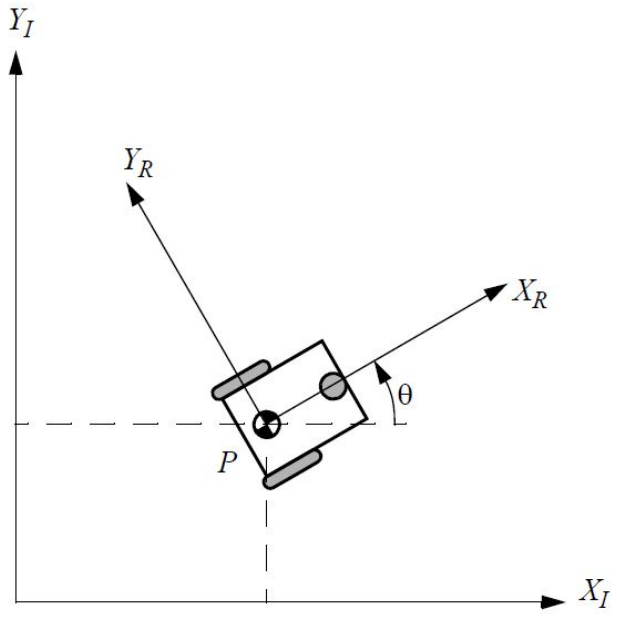


Figura 5: Marcos de referencia inercial y del robot [20].

La velocidad en el marco de referencia del robot se relaciona con la velocidad en el marco de referencia inercial mediante el jacobiano $J(\theta)$ [20]. De este modo, la cinemática diferencial del robot se obtiene mediante

$$ {}^{I}\dot{\xi} = J(\theta)\,{}^{B}\dot{\xi} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 0 \\ \omega \end{bmatrix} \tag{29} $$

Expandiendo la expresión para la cinemática diferencial se obtiene el siguiente modelo

$$ \begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\operatorname{sen}\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{30} $$

el cual se conoce como el modelo uniciclo para la cinemática de robots móviles.

6.6.1. Modelo uniciclo

Un uniciclo es un vehículo con una sola rueda orientable [21]. Este modelo es dinámico no lineal, y por medio de este se plantea una metodología de control en donde se hace control al uniciclo y se mapea a un sistema más complejo. La idea es agarrar las entradas de control (v, ω) abstractas y mapearlas a velocidades específicas a los actuadores del robot [19]. El modelo se expresa como

$$ \begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\operatorname{sen}\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{31} $$

6.6.2. Modelo robot diferencial (*differential drive*)

Se enfoca en robots de dos ruedas como el que se muestra en la Figura 6. Este tipo de robot se utilizará en esta tesis. Lo que se quiere es planificar trayectorias y controlar a los robots, por lo tanto es necesario utilizar el mapeo inverso entre el modelo uniciclo y el modelo de robot diferencial.

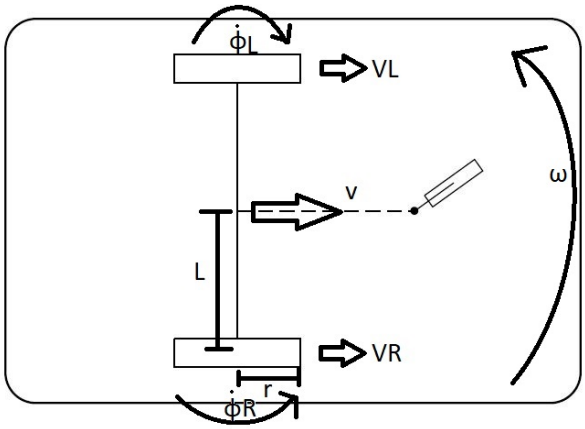


Figura 6: Robot diferencial con sus velocidades [21].

El mapeo inverso que relaciona las velocidades del uniciclo con las velocidades del robot diferencial es el siguiente

$$\dot{\varphi}_R = \frac{2v + 2\omega l}{2r} \quad \dot{\varphi}_L = \frac{2v - 2\omega l}{2r} \tag{32}$$

6.6.3. Difeomorfismo

Controlar al uniciclo es más fácil que controlar al robot diferencial. Por esta razón se busca controlar al uniciclo y mapear el control al modelo del robot diferencial. La idea es controlar al uniciclo según los requerimientos dados, y luego mapear las velocidades del uniciclo a las velocidades del robot móvil deseado [22].

El difeomorfismo se utiliza para mapear las velocidades lineales controladas u_1 y u_2 a las velocidades del uniciclo. Las ecuaciones modificadas son las siguientes

$$\begin{aligned} v &= u_1 \cos\varphi + u_2 \sin\varphi \\ w &= \frac{-u_1 \sin\varphi}{l} + \frac{u_2 \cos\varphi}{l} \end{aligned} \tag{33}$$

6.7. Controladores de posición y velocidad de robots diferenciales

6.7.1. LQR

El regulador cuadrático lineal (LQR por sus siglas en inglés), es un controlador óptimo que asegura la estabilidad del sistema en un lazo cerrado a través de ganancias de realimentación [23]. Utiliza la menor cantidad de control para llegar a la respuesta deseada. Debido

a que el sistema que se va a controlar (Robot diferencial) no es controlable se debe utilizar el difeomorfismo

$$u^* = -k\left(x - x_{ss}\right) + u_{ss} \tag{34}$$

6.7.2. LQI

El LQR es sensible a perturbaciones, por lo tanto se le agrega una parte integral para compensar el error en estado estable [24]. Este controlador es el LQI, que como el anterior se debe utilizar el difeomorfismo

$$u^* = -K \begin{bmatrix} x \\ \sigma \end{bmatrix} = -Kx - K_I\sigma \tag{35}$$

6.7.3. Controlador de pose

Este controlador toma en cuenta la pose final del robot. Este controlador se explica en [7]. Las ecuaciones que definen al controlador son las siguientes

$$\begin{aligned} \alpha &= -\theta + \theta_g \\ \beta &= -\theta - \alpha \\ v &= k_p\rho \\ w &= k_\alpha\alpha + k_\beta\beta \end{aligned} \tag{36}$$

donde las constantes siguen las reglas dadas en [7], y ρ es el error de posición, que se calcula como

$$\rho = \sqrt{\left(x_g - x\right)^2 + \left(y_g - y\right)^2} \tag{37}$$

6.7.4. Controlador de pose de Lyapunov

Este controlador se basa en el criterio de estabilidad de Lyapunov para que el sistema sea asintóticamente estable [6]. Los cálculos de α y ρ se realizan de la misma manera que con el controlador de pose. Las ecuaciones que definen al controlador son las siguientes

$$\begin{aligned} v &= k_\rho \cdot \rho \cdot \cos(\alpha) \\ w &= k_\rho \cdot \sin(\alpha) \cdot \cos(\alpha) + k_\alpha \cdot \alpha \end{aligned} \tag{38}$$

En [7] se detalla más sobre este controlador.

CAPÍTULO 7

Algoritmo de planificación de trayectorias y evasión de obstáculos

En este capítulo se presenta la metodología empleada para el algoritmo de planificación de trayectorias y evasión de obstáculos. Asimismo, se menciona el procedimiento realizado para obtener los parámetros óptimos, los escenarios en los que se validó el algoritmo, y la validación del mismo. Este algoritmo toma a las hormigas como masas puntuales sin restricción física ni actuadores, las hormigas se mueven únicamente en líneas rectas y cambian de dirección con ángulos de 45° y 90° respecto al movimiento anterior. Para estas simulaciones se utilizó el lenguaje MATLAB versión 2020a.

7.1. Metodología

Se tomó como base el algoritmo de la fase anterior [6] y se modificó de acuerdo al algoritmo que se plantea en [4]. El grafo que es la base del ACO, se tradujo a un mapa formado por casillas de 1×1. Por simplicidad se creó un borde en el mapa que es interpretado como varios obstáculos dentro del algoritmo modificado. En la Figura 7 se muestra un ejemplo del mapa sin obstáculos con los bordes dibujados. Cada punto en el centro de las cuadrículas representa un nodo en el grafo, y las líneas representan a las aristas. El mapa completo es representado dentro del algoritmo como una matriz de $(l+2) \times (l+2)$.

Los obstáculos se definen en un vector columna, en el cual se escribe el número del grafo en el que se quiere un obstáculo. Los obstáculos están representados como 1 dentro de la matriz del mapa, y se dibujan utilizando la función `rectangle` de Matlab.

Dentro del proceso de encontrar el nodo de destino se obtiene la posición actual de la hormiga para el tiempo t, y se calcula la matriz que representa al submapa de la hormiga en esta posición. Este submapa contiene la información que está en las ocho casillas del alrededor, así como lo que se observa en la Figura 3. Al tener la información del alrededor

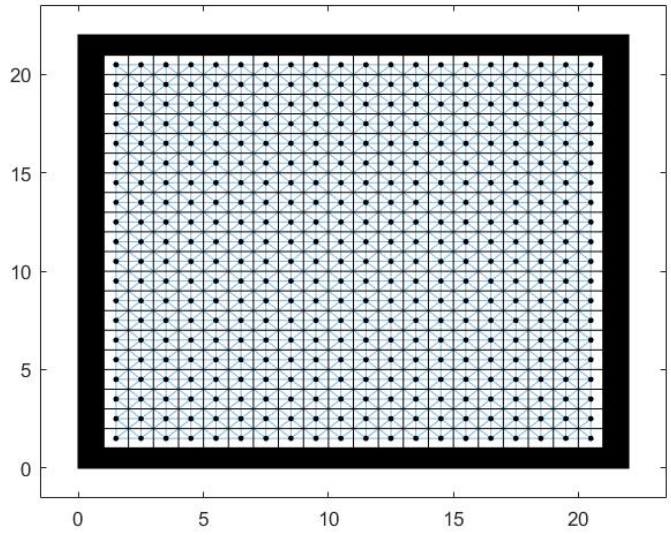


Figura 7: Mapa con bordes.

de la casilla actual, se procede a identificar si existe algún obstáculo.

Se crearon dos matrices de 3×3 para identificar si existe algún obstáculo en alguna diagonal, o lado de la casilla actual. Una matriz representa obstáculos en diagonal y la otra en las direcciones restantes. Utilizando las condiciones (9) y (10) se identifica si existe algún obstáculo en diagonal o hacia los lados de la casilla actual. Si se detecta algún obstáculo, se modifica el valor de la casilla correspondiente en la matriz que representa la dirección en la que se detectó el obstáculo por un 1. Si no detecta obstáculos, el valor en las matrices se mantiene en 0.

Con la matriz que representa los obstáculos de los lados, se identifica si alguna casilla es igual a 1. Si esto se cumple, se procede a identificar la dirección exacta del obstáculo. Utilizando el submapa se verifica si el obstáculo está a la derecha, izquierda, arriba o abajo. al identificar esto, se obtiene el número del nodo que está bloqueado y se agrega al conjunto de nodos bloqueados por la hormiga en el tiempo t.

Luego, se identifican los pasillos estrechos utilizando las condiciones (11)-(14).Al identificar si hay un pasillo estrecho al rededor, se bloquean los nodos que forman al pasillo y al que está atrás de este.

Utilizando la matriz que representa los obstáculos en diagonal, se busca si algún elemento de esta es 1. Si esto se cumple, se procede a identificar la dirección exacta de los obstáculos en las diagonales. De nuevo, utilizando el submapa se verifican las diagonales con obstáculos.

Utilizando la ecuación (15) se calcula la matriz de adyacencia para cada nodo que se visita por primera vez. Con D se encuentra la distancia más corta entre la posición actual y la posición objetivo S_{jo}, y se procede a calcular λ_{jo} utilizando la ecuación (17). La función de probabilidad utilizada para escoger el siguiente nodo es la ecuación (16).

Este algoritmo no se pudo paralelizar debido a que la matriz D es calculada con ciclos

`for` anidados, y la función `parfor` tomaba esto como procesos dependientes [25]. Para que se pueda paralelizar un algoritmo, los procesos deben ser independientes.

La actualización de feromona se hizo por partes. Al inicio del algoritmo se calcula con la ecuación (5) y calculando $\Delta\tau(i,j)$ como se plantea en la ecuación (6). Si converge el 40 % del porcentaje de convergencia especificado, y si el costo o largo de la trayectoria actual de la hormiga es el mínimo encontrado por el algoritmo, la concentración de feromona se calcula mediante la ecuación (18) tomando $\psi = Q$.

Las modificaciones mencionadas anteriormente se hicieron sobre el algoritmo desarrollado en la fase anterior en [6].

7.2. Elección de parámetros

En [4] se menciona que los parámetros utilizados para la experimentación son los siguientes:

Parámetro	**Valor**
ρ	0.43
α	1
β	7
γ	3
Q	100
Hormigas	30

Cuadro 1: Parámetros utilizados en [4].

En la fase anterior [6] se realizó un barrido de parámetros y se obtuvieron los que se muestran a continuación:

Parámetro	**Valor**
ρ	0.6
α	1
β	1
Q	2.1
Hormigas	60

Cuadro 2: Parámetros encontrados en la fase anterior [6].

Debido a que se implementó un nuevo algoritmo, se consideró necesario encontrar los mejores parámetros para este. Por lo tanto, se realizó un barrido de los mismos. Los parámetros que se tomaron como los mejores fueron los que brindaron un balance entre el menor tiempo de ejecución y el menor costo. Se utilizó un mapa en el que se demostrara que se esquivaban los obstáculos, y sin ser muy complejo. Este mapa se muestra en la siguiente figura:

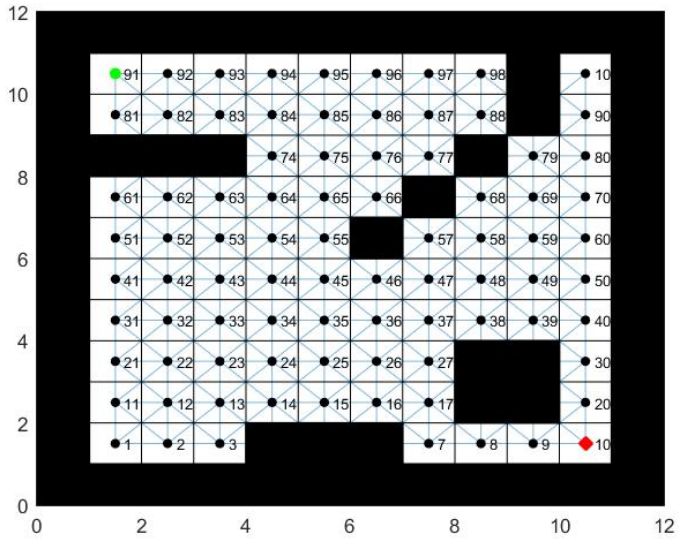


Figura 8: Mapa utilizado para barrer los parámetros.

Durante el barrido las hormigas debían ir del nodo 91 al 10. El rango en el que se realizaron los barridos se muestra en el Cuadro 3. Se realizaron 15 barridos por cada parámetro para poder determinar de forma estadística, los mejores parámetros.

Parámetro	**Rango**	**Paso**
ρ	(0.1,0.9)	0.1
α	(1,2.4)	0.2
β	(0.5,2.25)	0.25
γ	(0.2,4)	0.2
Q	(5,100)	5
Hormigas	(5,100)	5

Cuadro 3: Rango en el que se realizó el barrido de parámetros.

Los barridos se realizaron en el siguiente orden: $\rho, \alpha, \beta, \gamma$, Q, número de hormigas. Los resultados de cada barrido se usaron en el barrido del siguiente parámetro. El costo fue la característica que se tomó como la más importante durante los barridos; es decir, con base al costo se determinó cual era el mejor parámetro para el algoritmo.

7.2.1. Rho

En el Cuadro 4 se muestran los mejores resultados de las 15 corridas de ρ. Con estos valores se obtuvo el camino más corto entre los nodos de inicio y final. Utilizando la moda de los valores de ρ obtenidos, se obtuvo que los valores que más se repiten son $\rho = 0.8$ y $\rho = 0.9$. Estos resultados se tomaron para hacer el barrido de los demás parámetros, y así encontrar un set de parámetros que presente un buen rendimiento para el algoritmo.

Barrido	**t(s)**	**Costo**	**Iteraciones**	ρ
1	68.68	9.66	5	0.7
2	71.10	9.66	5	0.7
3	59.08	9.66	5	0.9
4	73.98	9.66	5	0.9
5	70.26	9.66	5	0.8
6	61.55	9.66	6	0.6
7	77.80	9.66	6	0.7
8	79.59	9.66	6	0.7
9	71.52	9.66	5	0.8
10	72.89	9.66	5	0.8
11	60.97	9.66	5	0.9
12	74.07	9.66	6	0.8
13	62.46	9.66	4	0.8
14	67.83	9.66	5	0.9
15	72.37	9.66	5	0.9

Cuadro 4: Resultados del barrido de ρ.

En la Figura 9 se muestra la gráfica de la frecuencia con la que se encontró el costo mínimo para cada valor de ρ. Se observa que para $\rho = 0.8$ en 12 de 15 barridos se se llegó al costo mínimo, y para $\rho = 0.9$ en 8 de 15 barridos se llegó al costo mínimo. En la Figura 10 se observa que al disminuir ρ el tiempo de ejecución aumenta. Por esto, se armaron los conjuntos de parámetros con valores para ρ de 0.8 y 0.9.

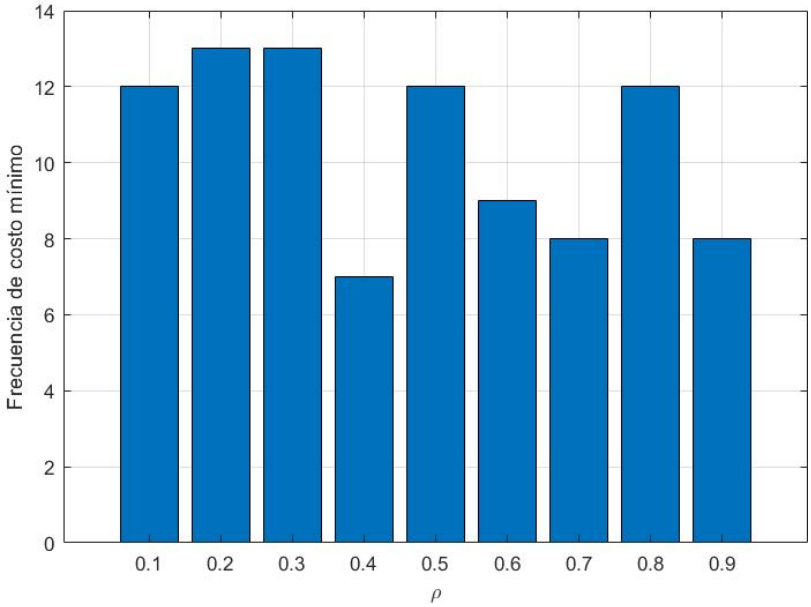


Figura 9: Frecuencia de costo mínimo para ρ.

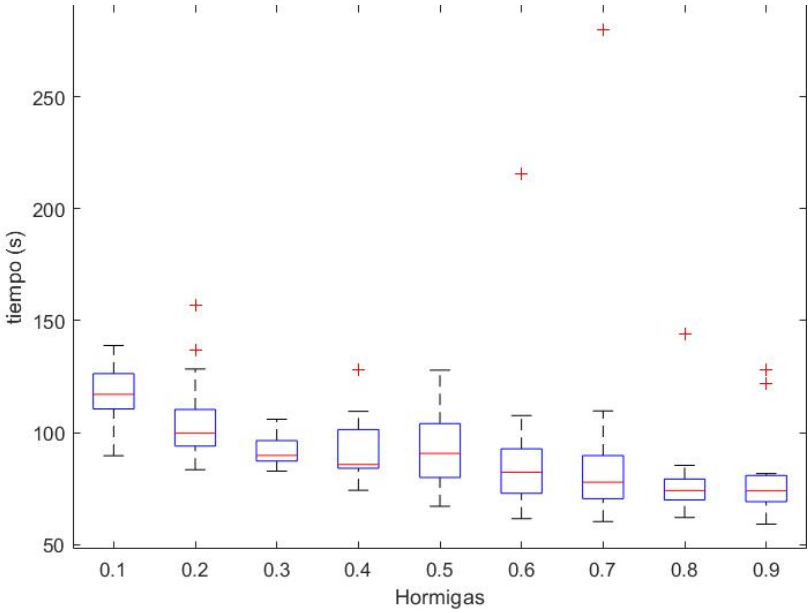


Figura 10: Tiempo para cada valor de ρ.

7.2.2. Alpha

El primer barrido de α se realizó para $\rho = 0.8$. Se esperaba que el resultado fuera cercano a 1 ya que en los Cuadros 1 y 2 se tiene que $\alpha = 1$.En el Cuadro 5 se muestran los mejores resultados de los 15 barridos de α para $\rho = 0.8$. Con la moda se obtuvo que el valor que más se repitió fue $\alpha = 2.2$.

Barrido	**t(s)**	**Costo**	**Iteraciones**	α
1	71.76	9.66	6	2.2
2	67.31	9.66	5	2
3	55.25	9.66	5	1.6
4	55.20	9.66	4	2.4
5	77.29	9.66	6	1.4
6	67.02	9.66	6	1.8
7	63.72	9.66	6	2.2
8	62.93	9.66	5	2.2
9	81.34	9.66	5	2.4
10	63.37	9.66	5	2.2
11	58.90	9.66	5	2.2
12	72.82	9.66	5	2.2
13	64.32	9.66	6	1.4
14	72.81	9.66	5	2.2
15	62.86	9.66	5	1.8

Cuadro 5: Resultados del barrido de α para $\rho = 0.8$.

En la Figura 11 se observa que para $\alpha = 1$ se obtuvo el costo mínimo para todos los

barridos. Para $\alpha = 2.2$ se obtuvo 10 de 15 barridos con costo mínimo. En la Figura 12 se observa que $\alpha = 1$ el tiempo de ejecución aumenta respecto a $\alpha = 2.2$. Se prefirió aumentar el tiempo de ejecución con el fin de obtener trayectorias óptimas. Por lo tanto para el conjunto de parámetros de $\rho = 0.8$ se tomo $\alpha = 1$.

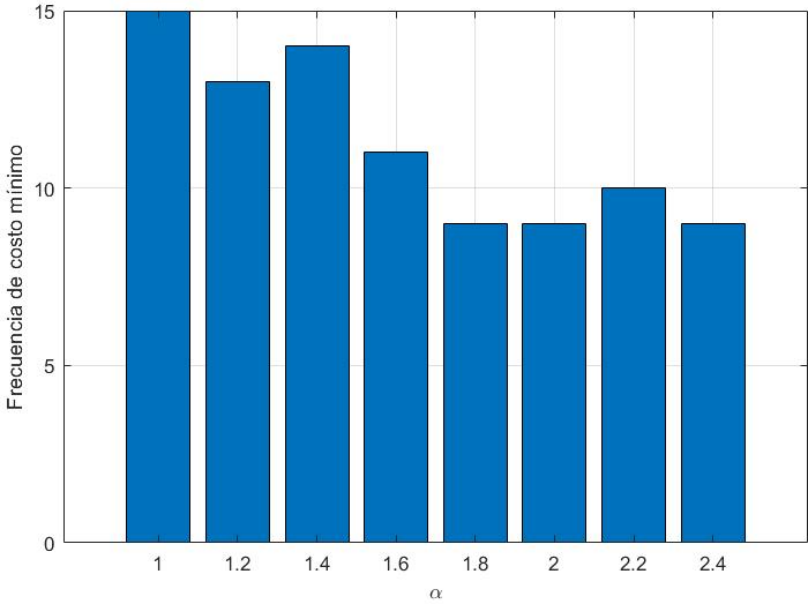


Figura 11: Frecuencia de costo mínimo de α para $\rho = 0.8$.

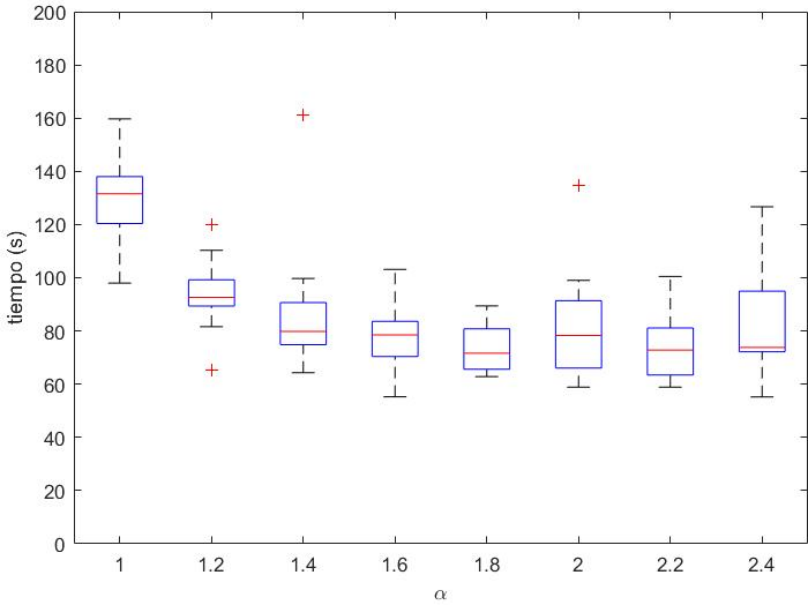


Figura 12: Tiempo para cada valor de α para $\rho = 0.8$.

El segundo barrido se realizó para $\rho = 0.9$. En el Cuadro 6 se muestran los mejores resultados de los 15 barridos de α. Se observa que el valor que más se repite para los barridos es $\alpha = 2$.

Barrido	**t(s)**	**Costo**	**Iteraciones**	α
1	60.57	9.66	5	1.8
2	60.63	9.66	5	2.2
3	69.94	9.66	5	2
4	69.51	9.66	6	2
5	65.45	9.66	5	2.4
6	65.37	9.66	5	1.8
7	69.11	9.66	6	1.4
8	52.71	9.66	4	2.2
9	62.99	9.66	4	2.4
10	54.74	9.66	4	2.2
11	62.22	9.66	5	1.6
12	72.60	9.66	6	1.8
13	58.41	9.66	5	2
14	65.47	9.66	5	2
15	66.30	9.66	6	1.6

Cuadro 6: Resultados del barrido de α para $\rho = 0.9$.

En las Figuras 13 y 14 se observa el mismo comportamiento presentado en el barrido de $\rho = 0.8$. En este caso también se tomó α como 1 aunque esto aumente un poco el tiempo de ejecución del algoritmo.

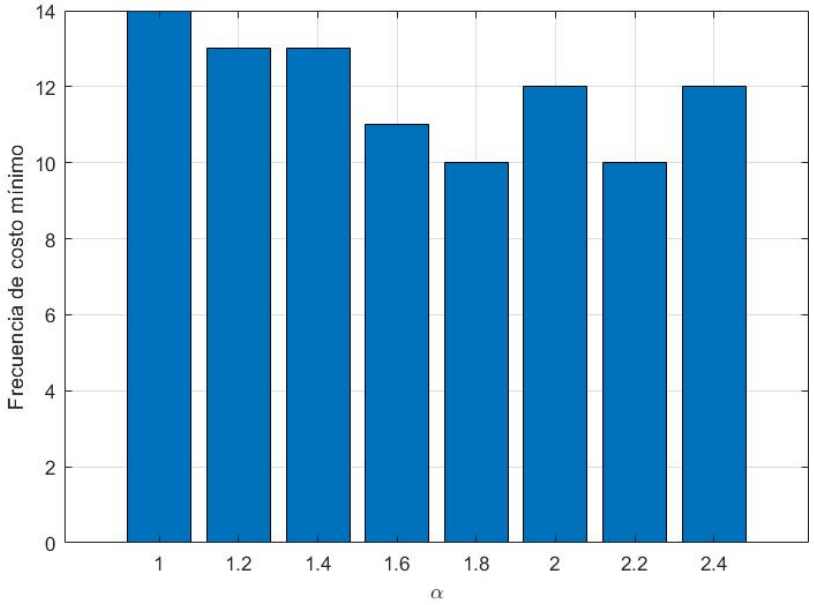


Figura 13: Frecuencia de costo mínimo de α para $\rho = 0.9$.

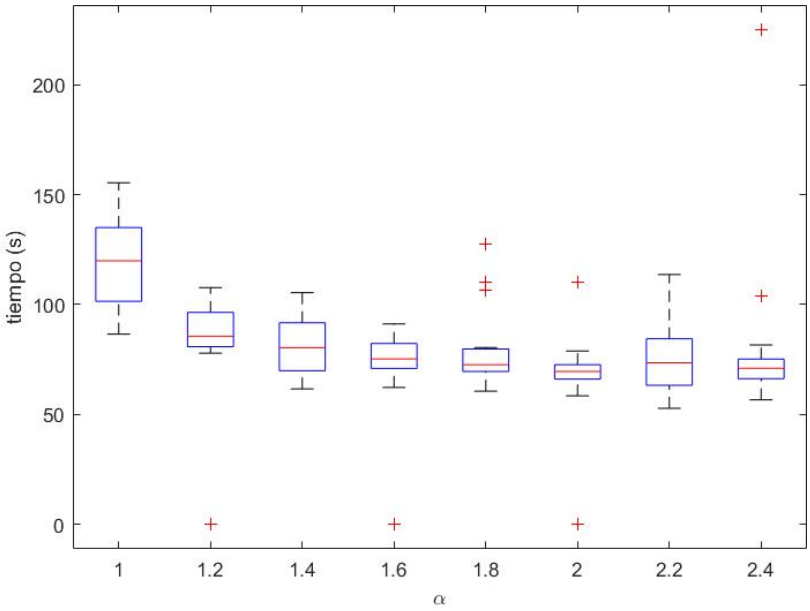


Figura 14: Tiempo para cada valor de α para $\rho = 0.9$.

7.2.3. Beta

Del Cuadro 1 se sabe que β tiene un valor de 7 y del Cuadro 2 se sabe que el valor es de 1. Debido a que hay mucha diferencia entre estos valores, se decidió hacer un pre-barrido con pasos más grandes y así saber más o menos el rango al cual hacer el barrido. Los resultados del barrido previo se observan en el Cuadro 7. Este pre-barrido se realizó con $\rho = 0.8$. En los resultados se observa que por el costo y tiempo el barrido se debía realizar para valores de β menores a 2.5.

β	**t(s)**	**Costo**	**Iteraciones**
1	186.71	9.66	5
1.5	163.48	9.66	4
2	260.16	9.66	5
2.5	4691.84	No hubo convergencia	150
3	8492.98	No hubo convergencia	150
3.5	303.03	14.12	6
4	7475.98	18.00	138
4.5	8337.48	No hubo convergencia	150
5	3412.58	18.00	59
5.5	4568.57	14.12	72
6	2291.09	18.00	39
6.5	4231.05	15.41	99
7	212.57	16.71	4

Cuadro 7: Resultados del pre-barrido de β.

Los mejores 15 resultados del barrido barrido para $\rho = 0.8$ se muestran en el Cuadro 8.

Se observa que el valor más frecuente de β con el que se obtienen los mejores resultados es de 0.5.

Barrido	**t(s)**	**Costo**	**Iteraciones**	β
1	63.11	9.66	5	0.5
2	82.04	9.66	5	0.75
3	74.10	9.66	6	0.5
4	56.49	9.66	5	0.5
5	68.23	9.66	7	1
6	54.36	9.66	5	1.25
7	73.39	9.66	5	0.5
8	84.42	9.66	5	1.25
9	52.63	10.24	4	1.5
10	85.38	9.66	6	1
11	63.71	9.66	5	0.5
12	71.16	9.66	5	0.5
13	86.88	9.66	7	0.75
14	67.32	9.66	5	0.5
15	63.89	9.66	5	1.25

Cuadro 8: Resultados del barrido de β para $\rho = 0.8$.

En la Figura 15 se observa que la máxima frecuencia de convergencia para el costo mínimo se obtiene con $\beta = 0.5$. En la Figura 16 se observa que con este valor de β el tiempo de ejecución es bajo respecto a los otros valores de este parámetro. Por lo tanto, para $\rho = 0.8$ el valor de β corresponde a 0.5.

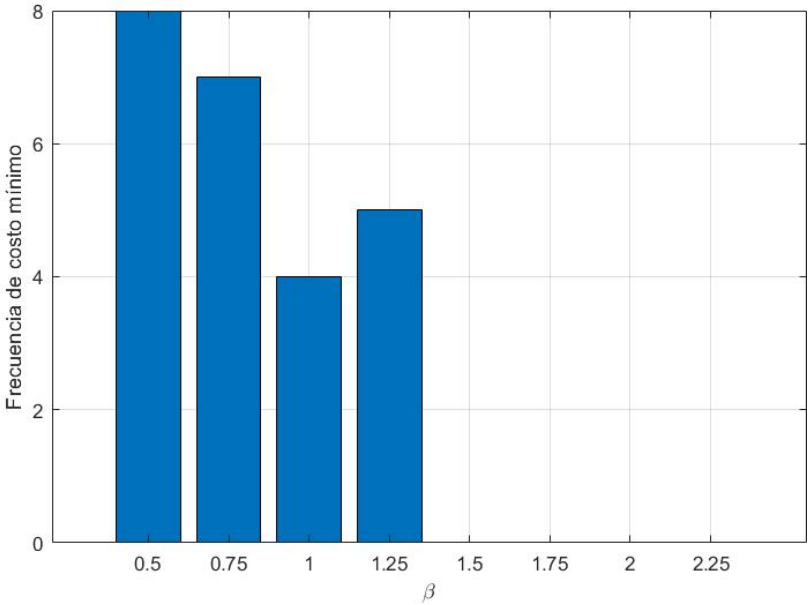


Figura 15: Frecuencia de costo mínimo de β para $\rho = 0.8$.

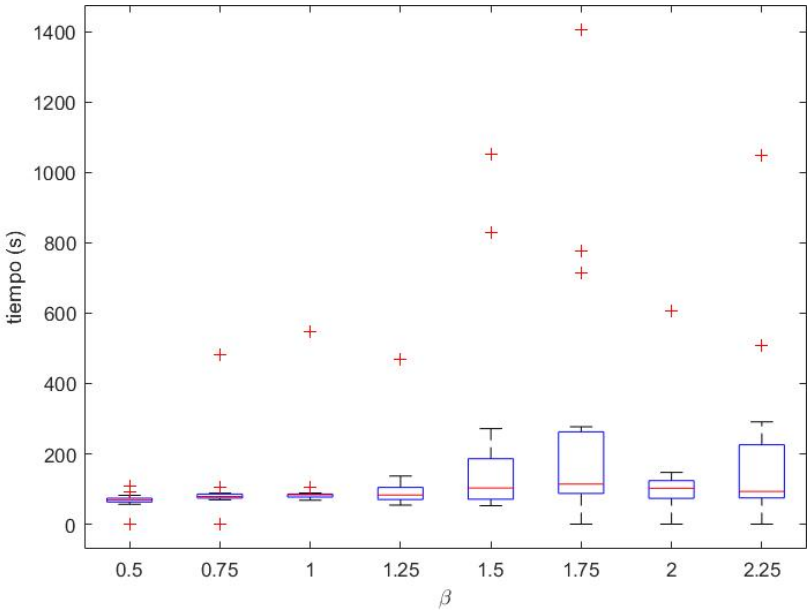


Figura 16: Tiempo para cada valor de β para $\rho = 0.8$.

En el Cuadro 9 se muestran los resultados de los barridos de β con $\rho = 0.9$. El valor que obtuvo mayor frecuencia fue el de $\beta = 0.5$.

Barrido	**t(s)**	**Costo**	**Iteraciones**	β
1	58.71	9.66	4	0.5
2	72.08	9.66	5	0.5
3	91.30	9.66	7	0.5
4	72.50	9.66	6	0.5
5	82.18	9.66	7	1
6	65.74	9.66	5	0.75
7	67.42	9.66	5	0.75
8	71.09	9.66	5	0.5
9	57.77	9.66	4	0.75
10	69.47	9.66	5	0.5
11	66.24	9.66	5	0.5
12	63.25	10.24	5	1.5
13	52.44	10.24	4	0.5
14	61.38	10.24	5	0.5
15	69.34	9.66	6	0.75

Cuadro 9: Resultados del barrido de β para $\rho = 0.9$.

En la Figura 17 se observa que con $\beta = 0.5$ se obtiene la mayor frecuencia de costo mínimo. En la Figura 18 se observa que el tiempo de ejecución con este valor es bajo respecto a los demás valores. Por lo tanto, para $\rho = 0.9$ se agrega $\beta = 0.5$.

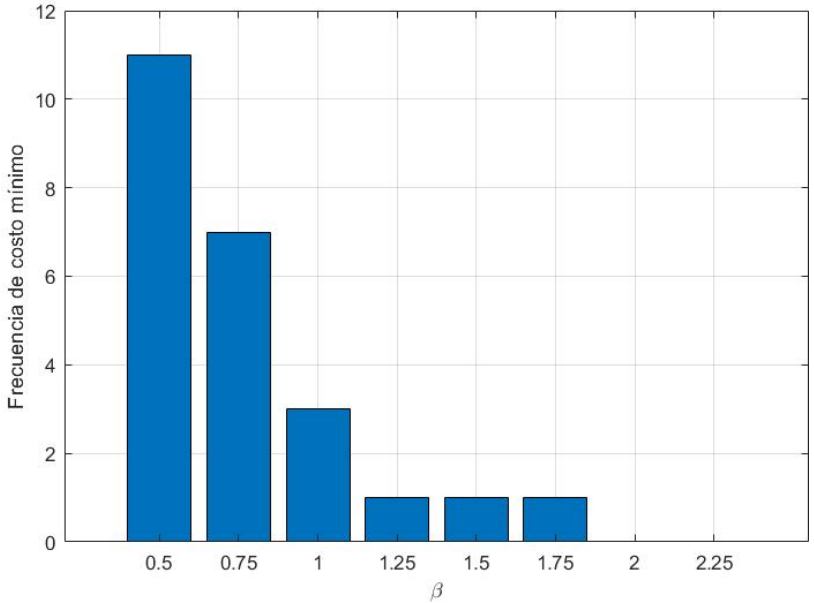


Figura 17: Frecuencia de costo mínimo de β para $\rho = 0.9$.

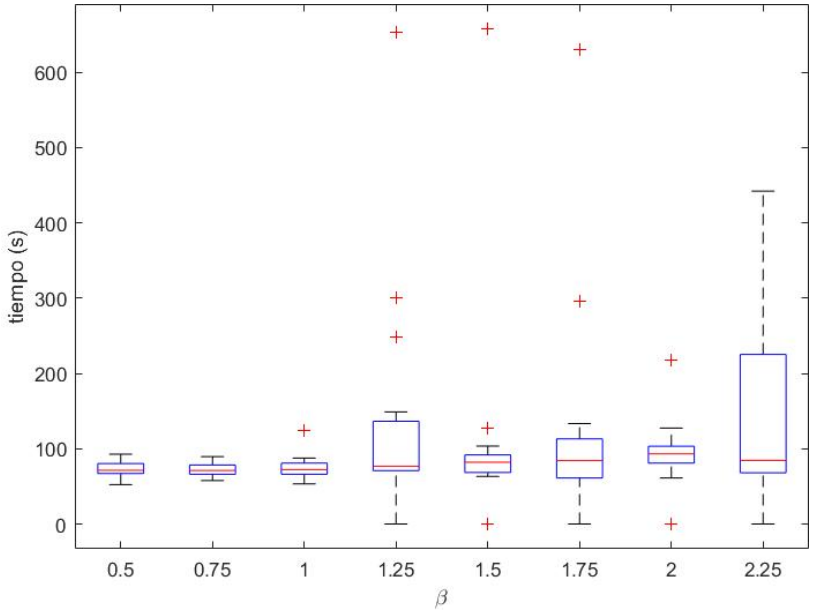


Figura 18: Tiempo para cada valor de β para $\rho = 0.9$.

7.2.4. Gamma

Se realizó un pre-barrido para este parámetro y así establecer el rango para realizar los barridos. Los resultados del pre-barrido se muestran en el Cuadro 10, donde se observa que los resultados son aceptables, por lo que el barrido de γ se realizó entre 0.2 y 4.

γ	**t(s)**	**Costo**	**Iteraciones**
0.5	233.72	9.66	6
1	195.73	10.24	5
1.5	198.27	9.66	5
2	193.59	9.66	5
2.5	219.59	10.24	6
3	221.44	9.66	5
3.5	193.34	10.24	4
4	229.85	9.66	6

Cuadro 10: Resultados del pre-barrido de γ.

En el Cuadro 11 se muestran los resultados de las mejores corridas del barrido de γ para $\rho = 0.8$. El valor con mayor frecuencia es $\gamma = 3.4$.

Barrido	**t(s)**	**Costo**	**Iteraciones**	γ
1	63.64	9.66	4	1.8
2	66.20	9.66	4	1.8
3	68.19	9.66	5	4
4	61.81	9.66	5	2.6
5	59.59	9.66	5	3.8
6	65.54	9.66	5	0.2
7	58.27	9.66	5	3.4
8	60.35	9.66	5	4
9	59.90	9.66	5	3.4
10	60.70	9.66	6	3.4
11	56.78	9.66	5	3.4
12	53.58	9.66	4	2.4
13	59.28	9.66	5	3
14	63.25	9.66	6	0.8
15	62.89	9.66	5	2.4

Cuadro 11: Resultados del barrido de γ para $\rho = 0.8$.

En la Figura 19 se observa que para el valor de $\gamma = 3.4$ la frecuencia de costo mínimo es de 11. En la Figura 20 se observa que el tiempo para $\gamma = 3.4$ se mantiene dentro de un buen rango. Por lo tanto, el valor de γ se mantiene como 3.4 para el conjunto de parámetros de $\rho = 0.8$.

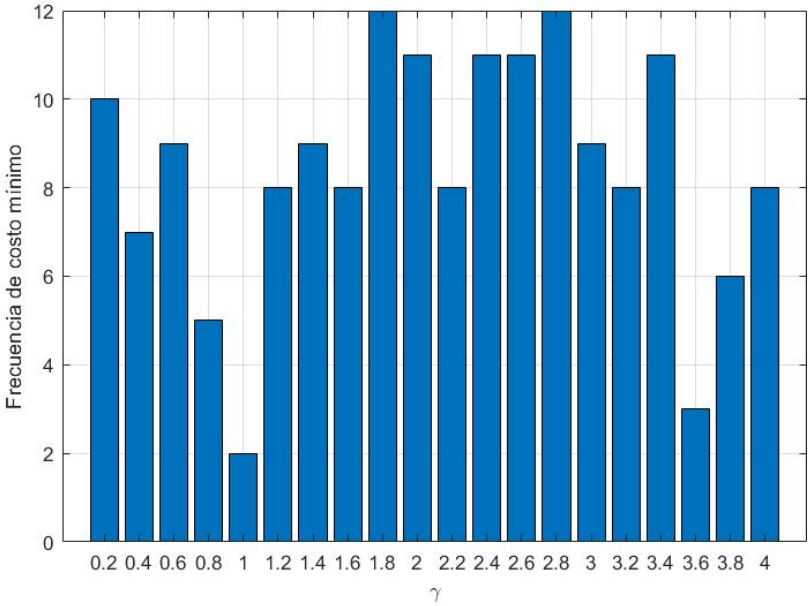


Figura 19: Frecuencia de costo mínimo de γ para $\rho = 0.8$.

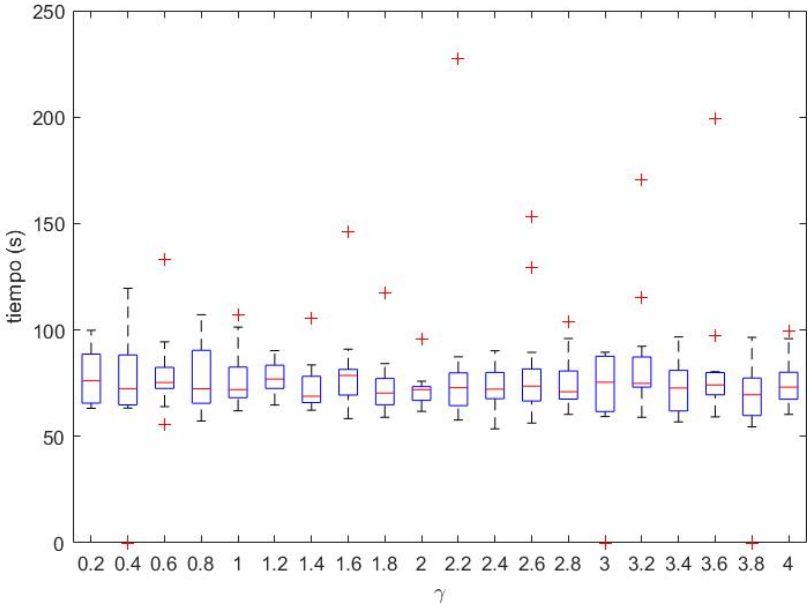


Figura 20: Tiempo para cada valor de γ para $\rho = 0.8$.

En el Cuadro 12 se muestran los resultados de barrer γ con $\rho = 0.9$. Con la moda se obtiene que el valor de γ debe ser de 2.2 para este conjunto de parámetros.

Barrido	**t(s)**	**Costo**	**Iteraciones**	γ
1	57.18	9.66	5	0.8
2	62.07	9.66	5	2.2
3	57.59	9.66	4	1.8
4	54.53	9.66	5	2.2
5	50.33	9.66	4	3.2
6	51.57	9.66	4	3.6
7	62.05	9.66	4	2.2
8	49.82	9.66	5	1.6
9	64.27	9.66	5	3
10	54.77	9.66	5	1.6
11	53.90	9.66	5	0.8
12	60.92	9.66	5	0.4
13	48.73	9.66	4	2.4
14	57.83	9.66	5	2.8
15	59.78	9.66	5	1.4

Cuadro 12: Resultados del barrido de γ para $\rho = 0.9$.

En la Figura 21 se observa que para el valor de $\gamma = 2.2$ la frecuencia de costo mínimo es de 12, que fue el valor máximo obtenido para frecuencia. En la Figura 22 se muestra que para $\gamma = 2.2$ el tiempo se ejecución se encuentra en un rango aceptable. Para el conjunto de parámetros de $\rho = 0.9$ el valor de $\gamma = 2.2$.

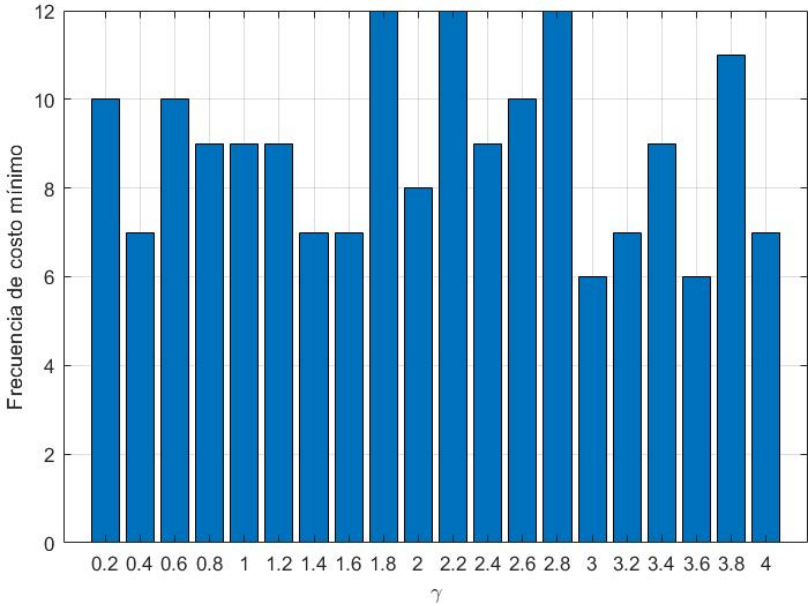


Figura 21: Frecuencia de costo mínimo de γ para $\rho = 0.9$.

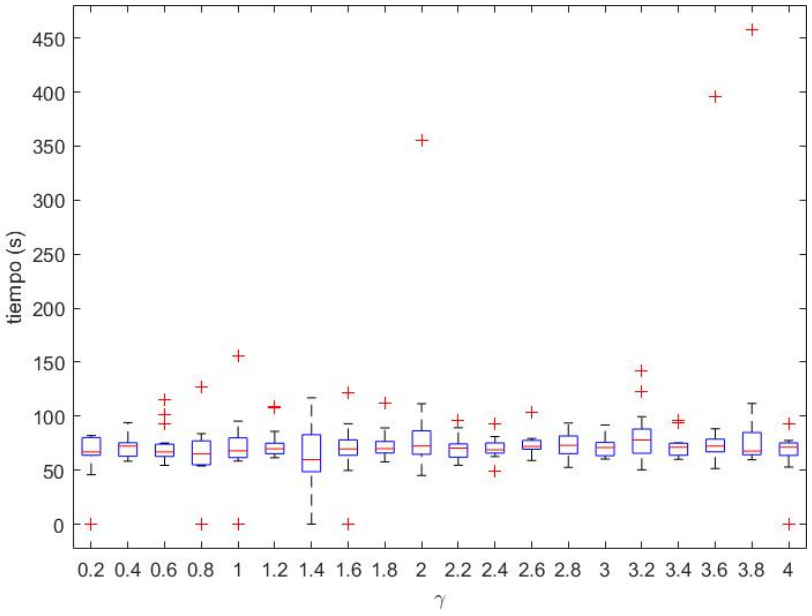


Figura 22: Tiempo para cada valor de γ para $\rho = 0.9$.

7.2.5. Q

En el Cuadro 1 se muestra que $Q = 100$, mientras que en el Cuadro 2 se muestra que $Q = 2.1$. Debido a que estos valores no son cercanos, el barrido de este parámetro se realizó desde 5 hasta 100.

En el Cuadro 13 se muestran los resultados de los barridos de Q para $\rho = 0.8$. El valor que obtuvo mayor frecuencia fue $Q = 65$.

Barrido	**t(s)**	**Costo**	**Iteraciones**	**Q**
1	52.80	9.66	4	65
2	54.65	9.66	4	80
3	56.01	9.66	5	65
4	60.49	9.66	5	65
5	57.18	9.66	4	65
6	60.99	9.66	5	100
7	65.63	9.66	5	95
8	59.54	9.66	5	35
9	56.25	9.66	5	15
10	57.73	9.66	5	70
11	56.99	9.66	5	70
12	50.64	9.66	4	60
13	60.38	9.66	4	25
14	49.48	9.66	4	80
15	64.08	9.66	4	40

Cuadro 13: Resultados del barrido de Q para $\rho = 0.8$.

En la Figura 23 se muestran las frecuencias con las que se encontró el costo mínimo para cada valor de Q con $\rho = 0.8$. La frecuencia de $Q = 60$ es mayor que la de $Q = 65$. Por lo tanto, se tomó $Q = 65$ para este set de parámetros, aunque en la Figura 26 se observe que el tiempo aumenta.

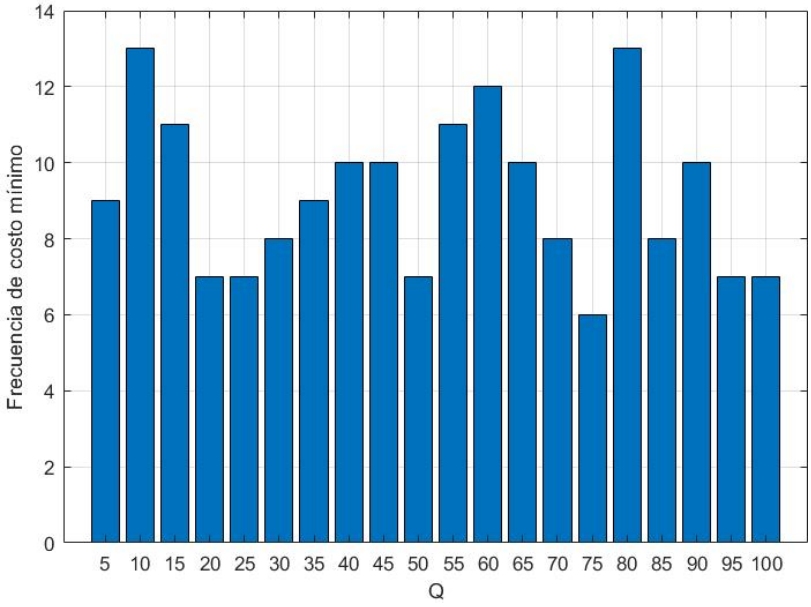


Figura 23: Frecuencia de costo mínimo de Q para $\rho = 0.8$.

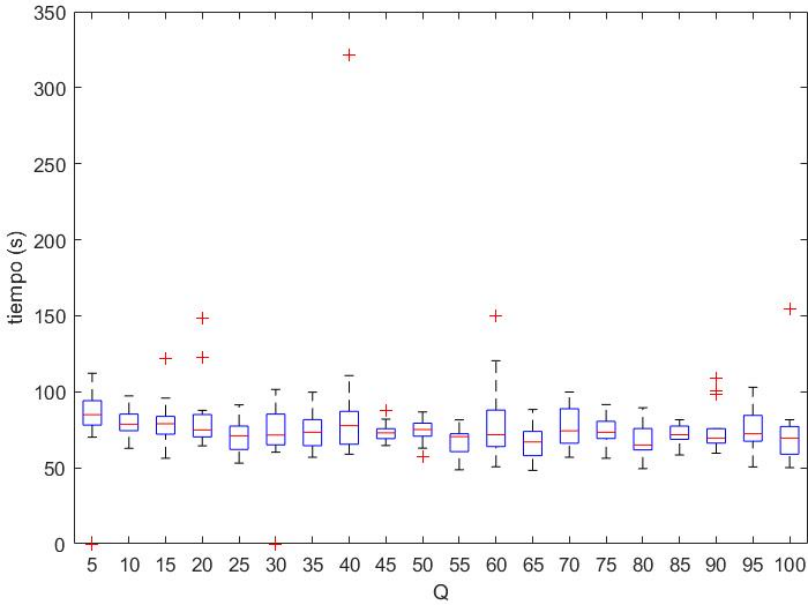


Figura 24: Tiempo para cada valor de Q para $\rho = 0.8$.

En el Cuadro 14 se muestran los 15 mejores resultados de los barridos. Se obtuvo que para este conjunto de parámetros $Q = 40$, ya que este valor fue el que tuvo mayor frecuencia en los resultados.

Barrido	t(s)	Costo	Iteraciones	Q
1	59.48	9.66	4	40
2	57.33	9.66	5	10
3	51.64	9.66	4	80
4	58.42	9.66	5	40
5	57.11	9.66	4	30
6	60.86	9.66	5	60
7	53.68	9.66	4	90
8	49.31	9.66	4	90
9	55.34	9.66	4	55
10	59.00	9.66	5	10
11	58.94	9.66	5	25
12	58.52	9.66	5	40
13	50.86	9.66	4	80
14	64.43	9.66	5	70
15	56.71	9.66	5	40

Cuadro 14: Resultados del barrido de Q para $\rho = 0.9$.

En la Figura 23 se observa que para $Q = 30$ se obtiene una frecuencia de costo mínimo mayor que con $Q = 45$. Además, en la Figura 24 se observa que con $Q = 30$ el tiempo de ejecución es menor. Por lo tanto, para $\rho = 0.9$ el valor de $Q = 30$.

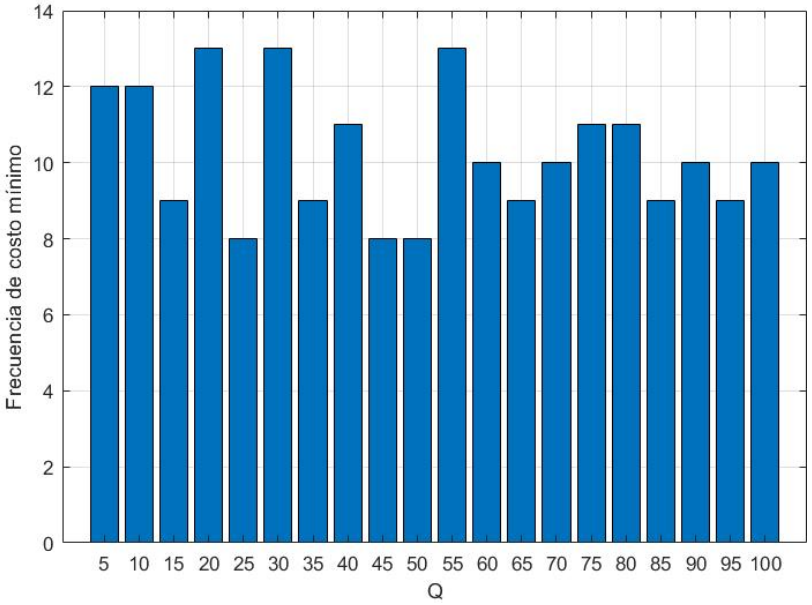


Figura 25: Frecuencia de costo mínimo de Q para $\rho = 0.9$.

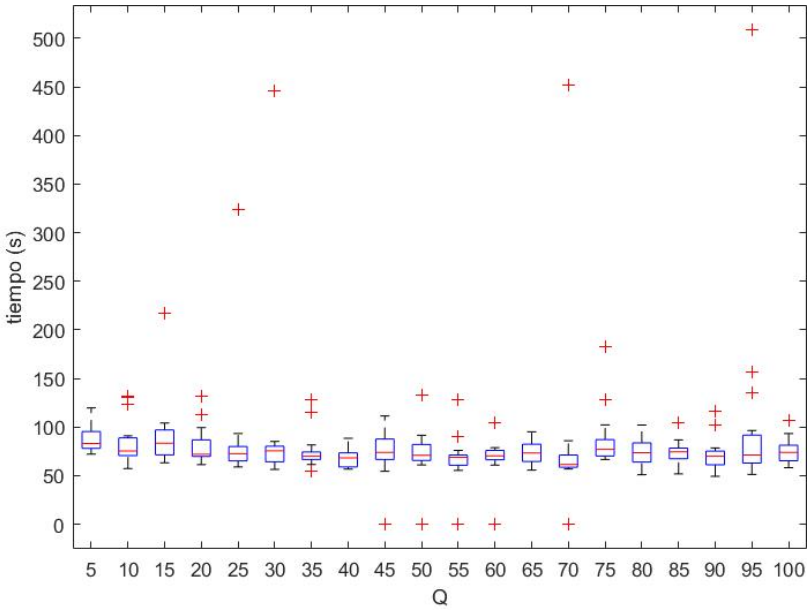


Figura 26: Tiempo para cada valor de Q para $\rho = 0.9$.

7.2.6. Cantidad de hormigas

En el Cuadro 1 se muestra que se experimentó con 30 hormigas, mientras que en el Cuadro 2 se observa que se utilizaron 60 hormigas para las pruebas. Como estos valores no son muy cercanos, el barrido se realizó desde 5 hasta 100 hormigas.

En el Cuadro 15 se muestran los resultados de los 15 barridos de la cantidad de hormigas para $\rho = 0.8$. El valor que tuvo mayor frecuencia fue 25.

Barrido	**t(s)**	**Costo**	**Iteraciones**	**Hormigas**
1	55.18	9.66	5	20
2	22.19	9.66	6	10
3	72.24	9.66	5	30
4	37.80	9.66	5	15
5	23.22	9.66	5	10
6	57.72	9.66	5	20
7	29.09	9.66	5	15
8	67.64	9.66	6	25
9	36.49	9.66	5	15
10	43.83	9.66	5	25
11	70.02	9.66	5	25
12	66.70	9.66	5	25
13	75.16	9.66	6	30
14	56.90	9.66	6	20
15	28.01	9.66	7	10

Cuadro 15: Resultados del barrido de hormigas para $\rho = 0.8$.

En la Figura 27 se observa que con 60 hormigas la frecuencia de convergencia es de 15. En la Figura 28 se observa que para 60 hormigas el tiempo de ejecución es mayor que para 25 hormigas. Sin embargo, como lo que se quiere es optimizar trayectorias, se escogieron 60 hormigas para $\rho = 0.8$.

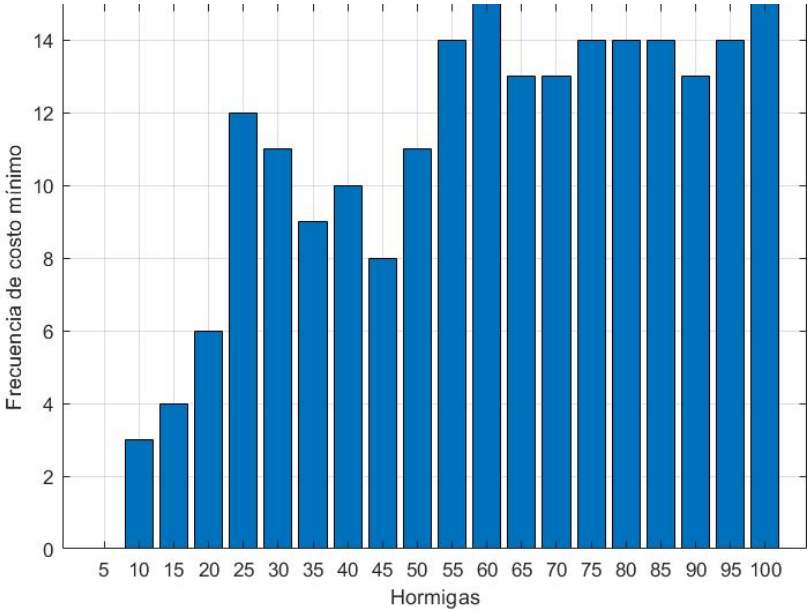


Figura 27: Frecuencia de costo mínimo de hormigas para $\rho = 0.8$.

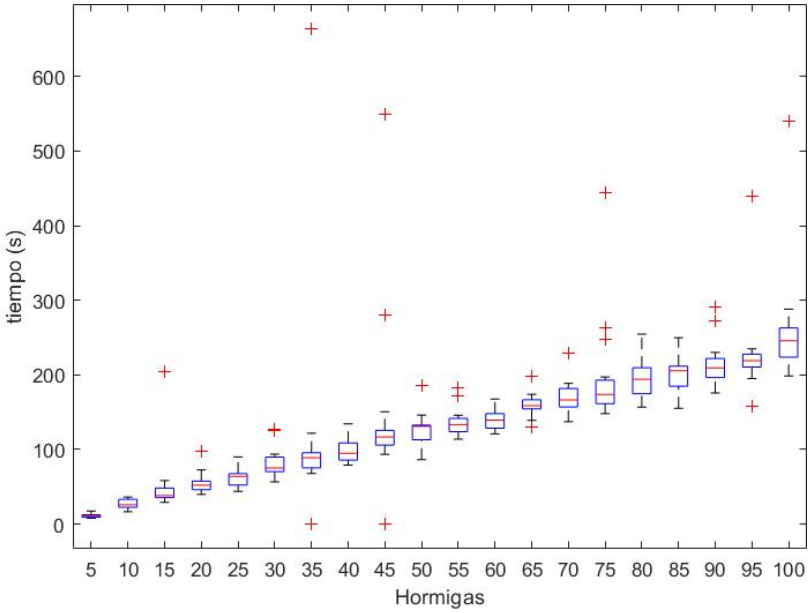


Figura 28: Tiempo para cada valor de hormigas para $\rho = 0.8$.

En el Cuadro 16 se muestran los resultados de los 15 barridos para la cantidad de hormigas, utilizando $\rho = 0.9$. El valor de este parámetro que obtuvo mayor frecuencia fue 20.

Barrido	t(s)	Costo	Iteraciones	Hormigas
1	71.35	9.66	5	30
2	42.56	9.66	5	20
3	25.18	9.66	6	10
4	53.25	9.66	5	25
5	13.71	9.66	6	5
6	50.31	9.66	4	20
7	37.56	9.66	4	20
8	43.23	9.66	5	20
9	11.83	9.66	5	5
10	92.06	9.66	10	30
11	39.71	9.66	5	20
12	44.26	9.66	5	15
13	46.13	9.66	6	20
14	22.58	9.66	5	10
15	62.76	9.66	6	25

Cuadro 16: Resultados del barrido de hormigas para $\rho = 0.9$.

En la Figura 30 se observa que para 20 hormigas se obtiene 8 veces de 15 el costo mínimo. Se tomaron 45 hormigas para completar este conjunto de parámetros ya que con 45 hormigas se obtienen 13 de 15 barridos el costo mínimo y no se incrementa mucho el tiempo de ejecución.

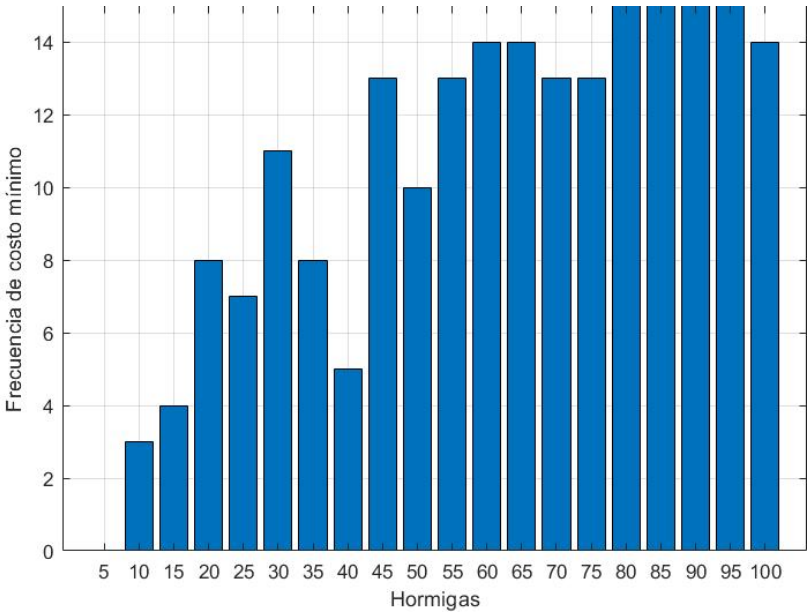


Figura 29: Frecuencia de costo mínimo de hormigas para $\rho = 0.9$.

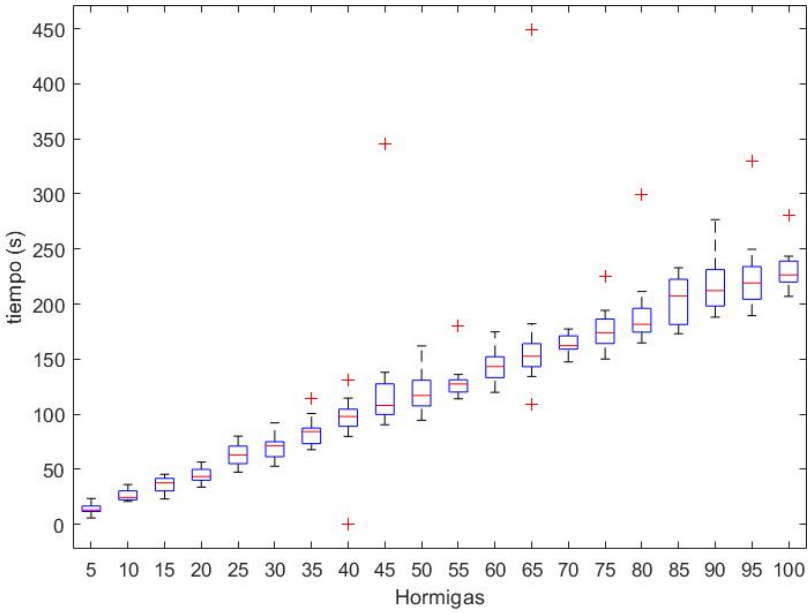


Figura 30: Tiempo para cada valor de hormigas para $\rho = 0.9$.

7.2.7. Resultados del barrido de parámetros

En el Cuadro 17 se muestra el conjunto de parámetros obtenidos para $\rho = 0.8$. A estos valores se les referirá como el primer conjunto de parámetros.

Parámetro	**Valor**
ρ	0.8
α	1
β	0.5
γ	3.4
Q	60
$Hormigas$	60

Cuadro 17: Primer conjunto de parámetros.

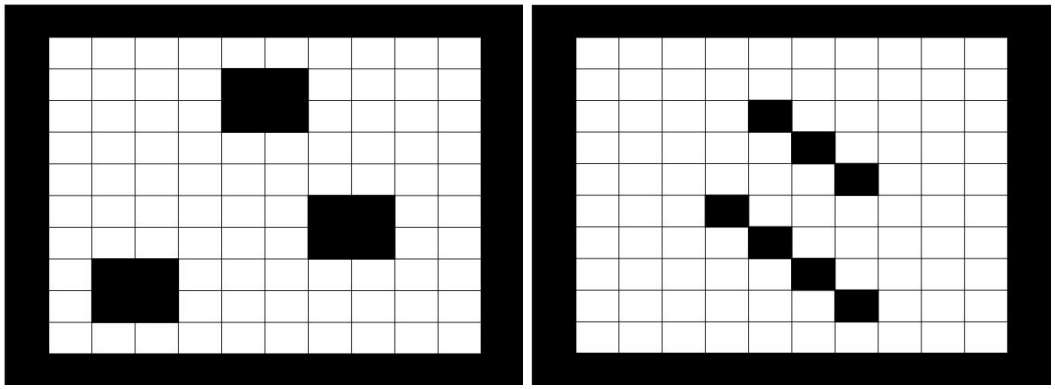
En el Cuadro 18 se muestran los parámetros encontrados para $\rho = 0.9$. A este conjunto se le referirá como el segundo conjunto de parámetros.

Parámetro	**Valor**
ρ	0.9
α	1
β	0.5
γ	2.2
Q	30
$Hormigas$	45

Cuadro 18: Segundo conjunto de parámetros.

7.3. Escenarios

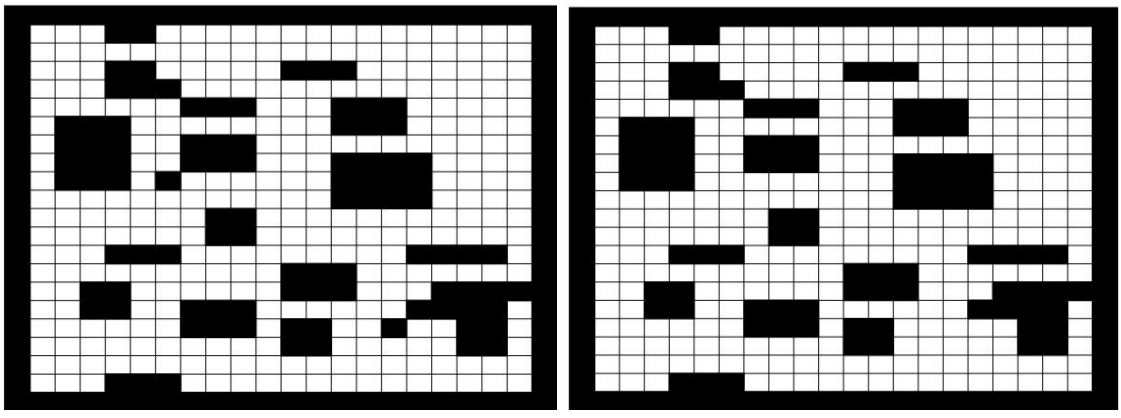
Para validar el algoritmo se utilizaron los mapas que se presentan en la Figura 31. Estos mapas son simples en cuanto a obstáculos y son de 10×10 cuadros. Se hace referencia a estos mapas en otros algoritmos que se mencionan más adelante.



(a) Mapa A. (b) Mapa B.

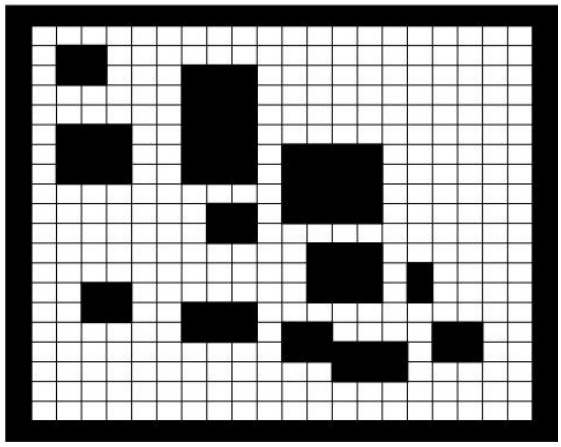
Figura 31: Escenarios para la validación del algoritmo.

Se replicaron los mapas utilizados en [4] para la validación. Estos escenarios se muestran en la Figura 32, y se hace referencia a estos en otros algoritmos que se mencionan más adelante. Estos mapas son de 20×20 cuadros.



(a) Mapa C.

(b) Mapa D.



(c) Mapa E.

Figura 32: Escenarios para la validación del algoritmo.

7.4. Validación del algoritmo de Planificación y Evasión de Obstáculos

A continuación se muestran los resultados del algoritmo implementado. El porcentaje mínimo de convergencia para la validación fue de 95 %.

Se validó el funcionamiento del algoritmo utilizando las constates de la fase anterior [6], que se muestran en el Cuadro 2. Para este conjunto de parámetros se utilizó $\gamma = 3$, ya que es el valor que se utilizó en [4] y este parámetro no fue utilizado en el algoritmo de la fase anterior. Asimismo, se utilizaron las constantes encontradas en el barrido de parámetros, que se muestran en los Cuadros 17 y 18.

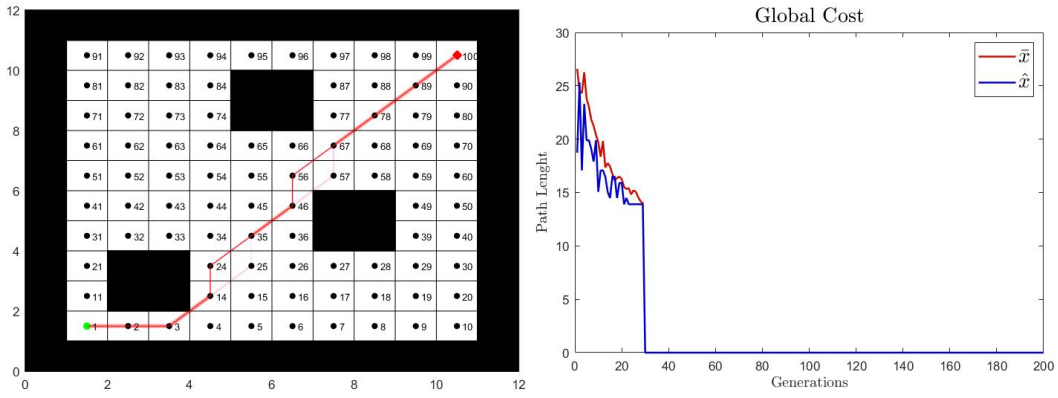
En los mapas cuadriculados generados, el punto verde representa al nodo de inicio y el punto rojo al nodo de destino. Las pruebas de los mapas A y B se realizaron en una computadora portátil ASUS X556UQ [26]. Las pruebas para los mapas C, D y E se realizaron en la computadora de alto rendimiento de la universidad [27].

7.4.1. Mapa A

En las Figuras 33, 34 y 35 se muestran los resultados del algoritmo utilizando las constantes de la fase anterior. Se observa que las trayectorias son óptimas entre los puntos. En la Figura 33a se observa la trayectoria encontrada para la primera prueba, esta es representada como la línea sólida. En la misma figura se observa la cantidad de feromona depositada en cada arista del grafo, esta es representada como las líneas opacas. Si la arista tiene mayor concentración de feromona, se vuelve más gruesa y su color más fuerte. En la Figura 33b se observa la media $\bar{x}$ y la moda $\hat{x}$ de los caminos encontrados. Se observa en la gráfica de la moda que la trayectoria final tiene el costo mínimo encontrado por el algoritmo.

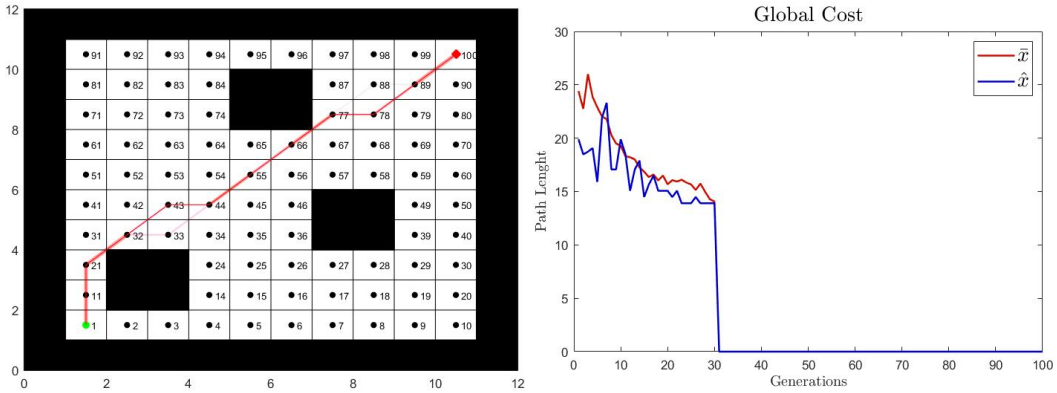
En las Figuras 36, 37 y 38 se muestran los resultados de las pruebas del algoritmo utilizando los parámetros presentados en [4]. Se observa que las trayectorias encontradas no son óptimas. En las Figuras 39, 40 y 41 se muestran los resultados obtenidos para el mapa A con las constantes encontradas en el primer barrido de parámetros. Se observa que para este conjunto de parámetros las trayectorias también son óptimas. Además, las gráficas de la moda muestran que el costo de la trayectoria fue el mínimo encontrado por el algoritmo.

En las Figuras 42, 43 y 44 se muestran los resultados de las pruebas del algoritmo utilizando el segundo conjunto de parámetros. Las trayectorias encontradas por el algoritmo fueron óptimas, por lo tanto, el costo fue el mínimo.



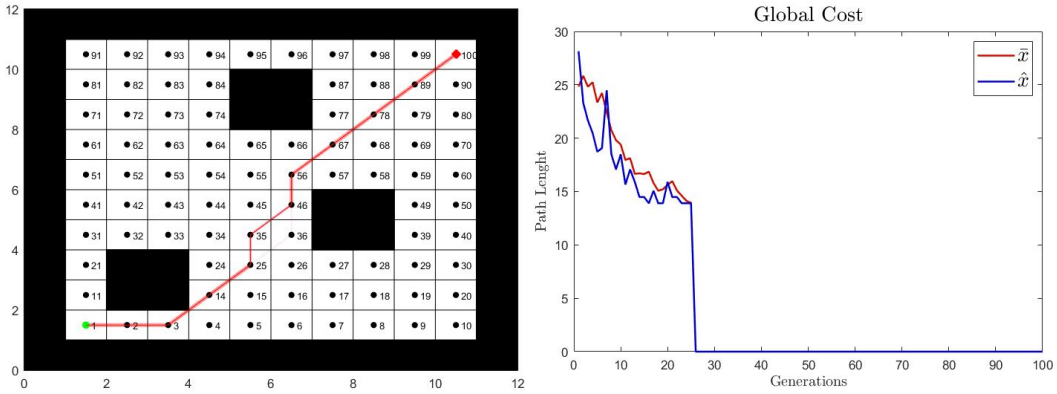
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 33: Primera validación del algoritmo con el mapa A con parámetros de la fase anterior.



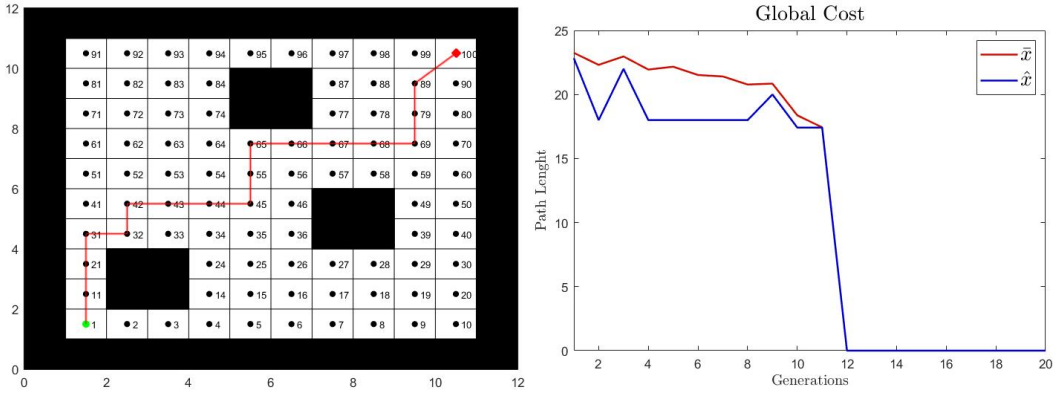
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 34: Segunda validación del algoritmo con el mapa A con parámetros de la fase anterior.



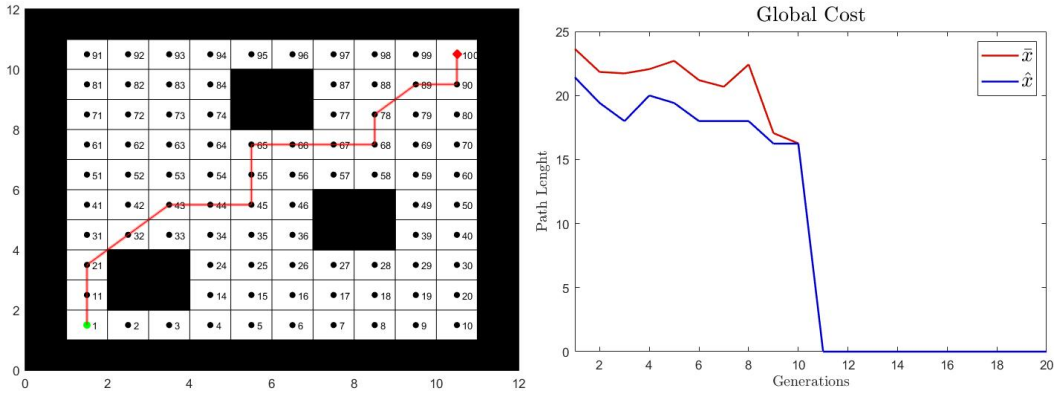
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 35: Tercera validación del algoritmo con el mapa A con parámetros de la fase anterior.



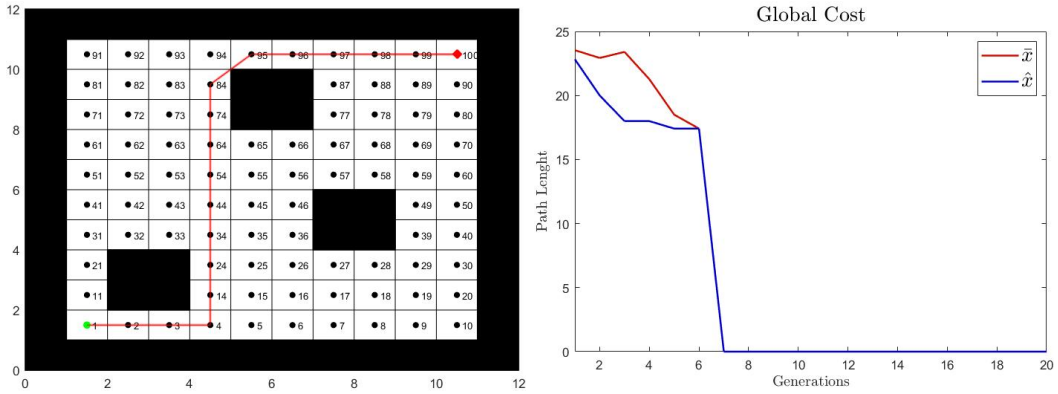
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 36: Primera validación del algoritmo con el mapa A con los parámetros de [4].



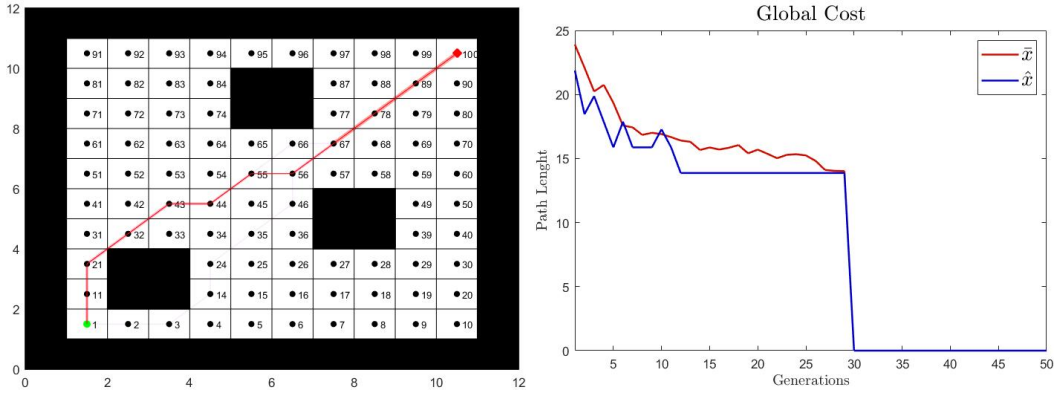
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 37: Segunda validación del algoritmo con el mapa A con los parámetros de [4].



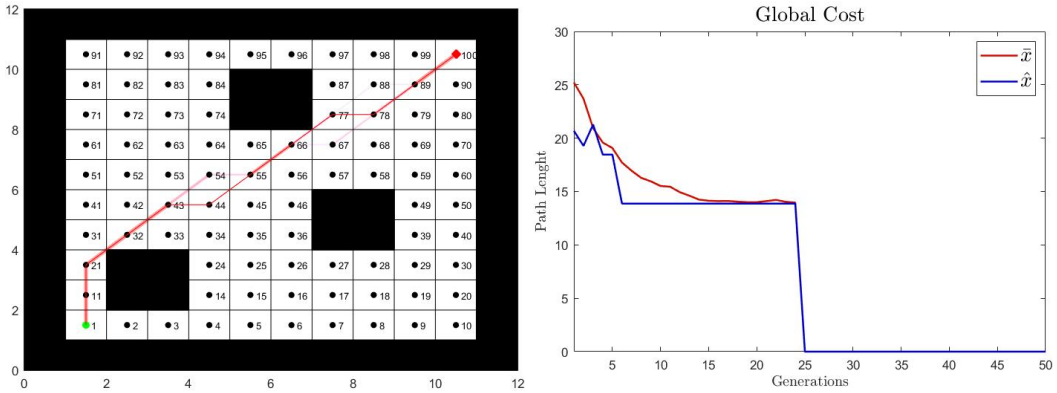
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 38: Tercera validación del algoritmo con el mapa A con los parámetros de [4].



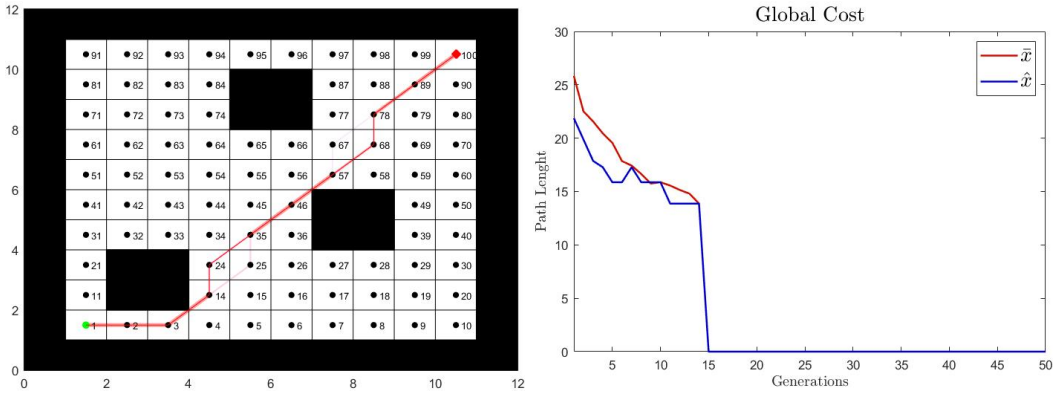
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 39: Primera validación del algoritmo con el mapa A con el primer conjunto de parámetros.



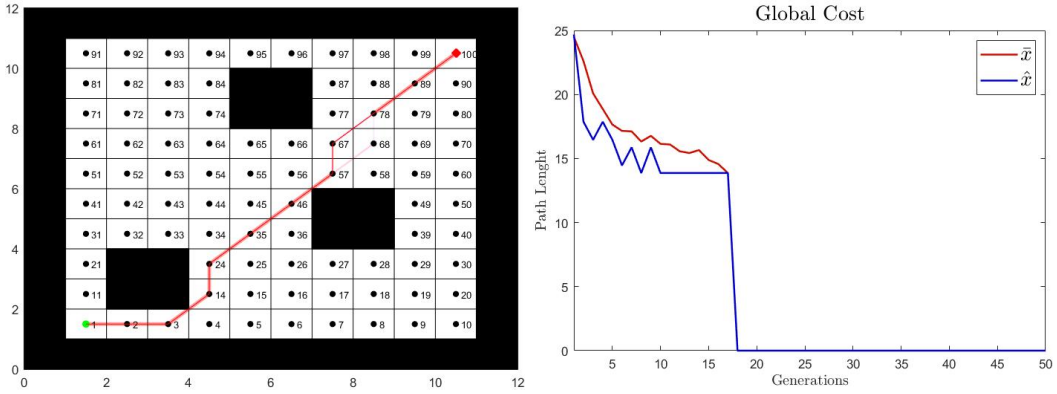
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 40: Segunda validación del algoritmo con el mapa A con el primer conjunto de parámetros.



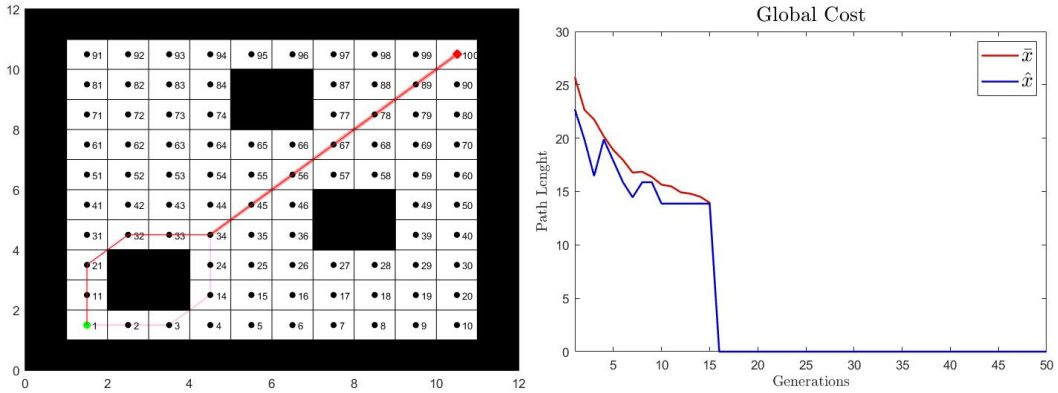
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 41: Tercera validación del algoritmo con el mapa A con el primer conjunto de parámetros.



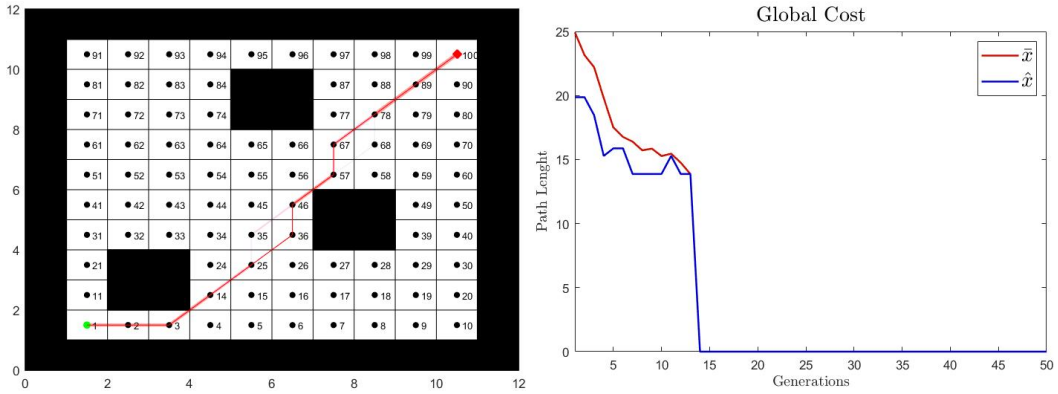
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 42: Primera validación del algoritmo con el mapa A con el segundo conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 43: Segunda validación del algoritmo con el mapa A con el segundo conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 44: Tercera validación del algoritmo con el mapa A con el segundo conjunto de parámetros.

En el Cuadro 19 se muestran los resultados de todas las pruebas realizadas para el mapa A. El único conjunto de parámetros que no devolvió una ruta óptima fue el que se menciona en [4]. En el Cuadro 20 se muestran los promedios de iteraciones, tiempo y costo por cada conjunto de parámetros. El único conjunto de parámetros con el que no se obtuvo el costo mínimo fue el que se utiliza en [4], y esto está relacionado con que la trayectoria encontrada no fue la óptima. Sin embargo, con este conjunto de parámetros se obtuvo el menor promedio de iteraciones y el menor tiempo promedio.

Corrida	Parámetros	Iteraciones	Tiempo (min)	Costo
1	Fase anterior	29	22.57	13.9
2	Fase anterior	30	19.09	13.9
3	Fase anterior	25	20.53	13.9
1	[4]	11	15.44	17.4
2	[4]	10	16.11	16.2
3	[4]	6	8.68	17.4
1	Primer conjunto	29	35.54	13.9
2	Primer conjunto	24	32.75	13.9
3	Primer conjunto	14	15.21	13.9
1	Segundo conjunto	17	17.46	13.9
2	Segundo conjunto	15	13.75	13.9
3	Segundo conjunto	13	10.49	13.9

Cuadro 19: Resultados del mapa A.

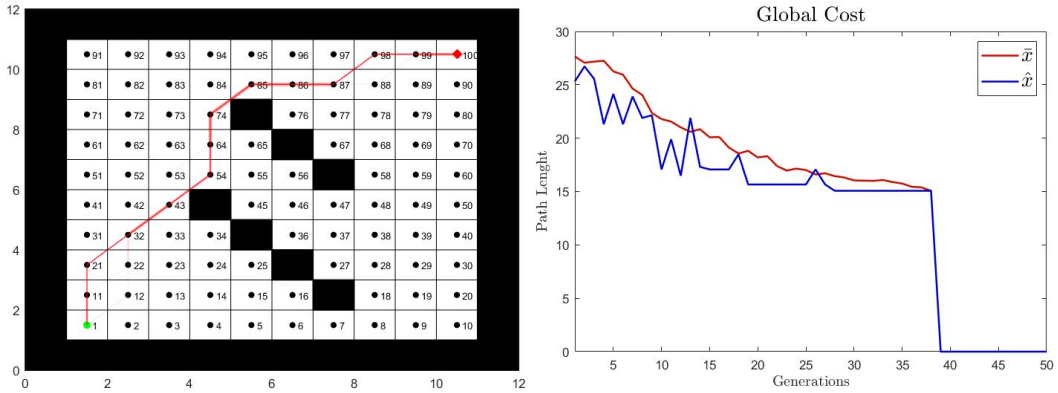
Parámetros	Promedio de iteraciones	Tiempo promedio (min)	Costo promedio
Fase anterior	28	20.73	13.9
[4]	9	13.41	17.02
Primer conjunto	22.33	27.83	13.9
Segundo conjunto	15.00	13.90	13.9

Cuadro 20: Promedio de los resultados del mapa A.

7.4.2. Mapa B

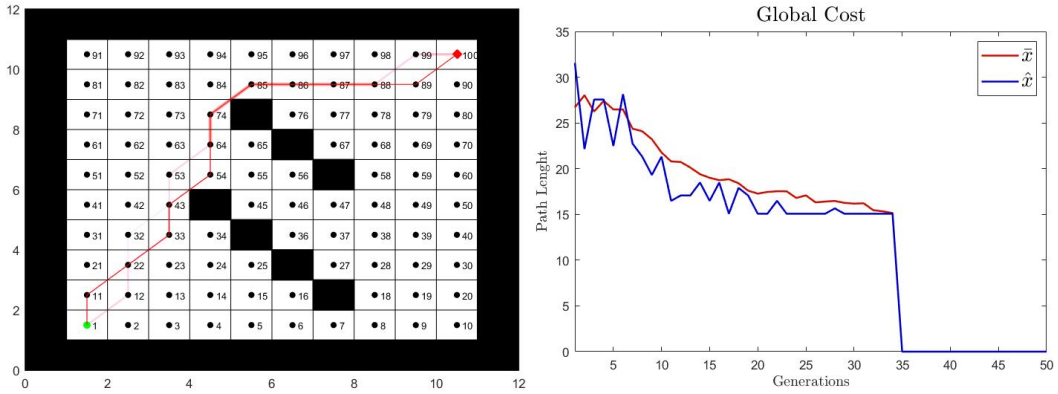
En las Figuras 45, 46 y 47 se muestran los resultados generados al utilizar los parámetros de la fase anterior. Se observa que las trayectorias encontradas son óptimas y que por lo tanto, el costo de esta es el mínimo. En las Figuras 48, 49 y 50 se muestran los resultados obtenidos al utilizar los parámetros de [4]. Se observa que las trayectorias encontradas no son óptimas. En las Figuras 51, 52, 53 y 54, 55, 56 se muestran los resultados de utilizar los parámetros del primer conjunto y del segundo conjunto, respectivamente. Las trayectorias fueron óptimas y el costo de estas fue el mínimo.

En estos resultados se observa que el algoritmo siempre cumple con esquivar los obstáculos del mapa. Además, el mapa está compuesto de pasillos estrechos donde es físicamente imposible que pase un robot. Se puede observar que las trayectorias no pasan por ningún pasillo estrecho.



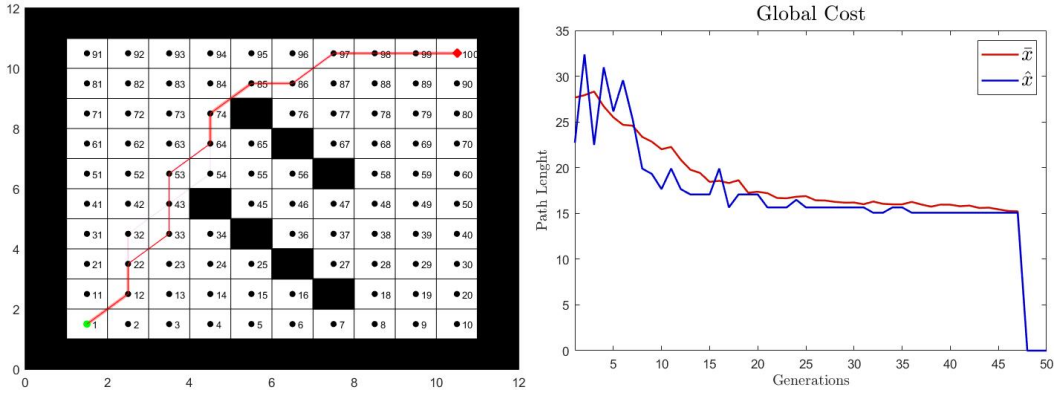
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 45: Primera validación del algoritmo con el mapa B con parámetros de la fase anterior.



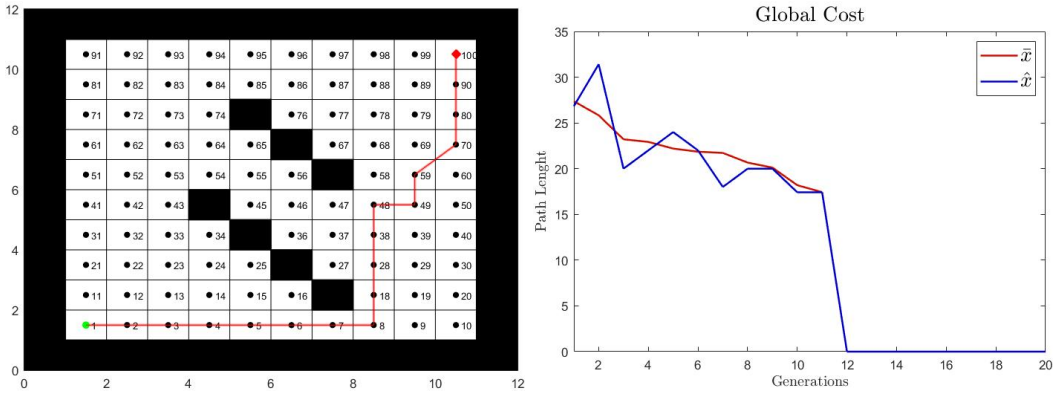
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 46: Segunda validación del algoritmo con el mapa B con parámetros de la fase anterior.



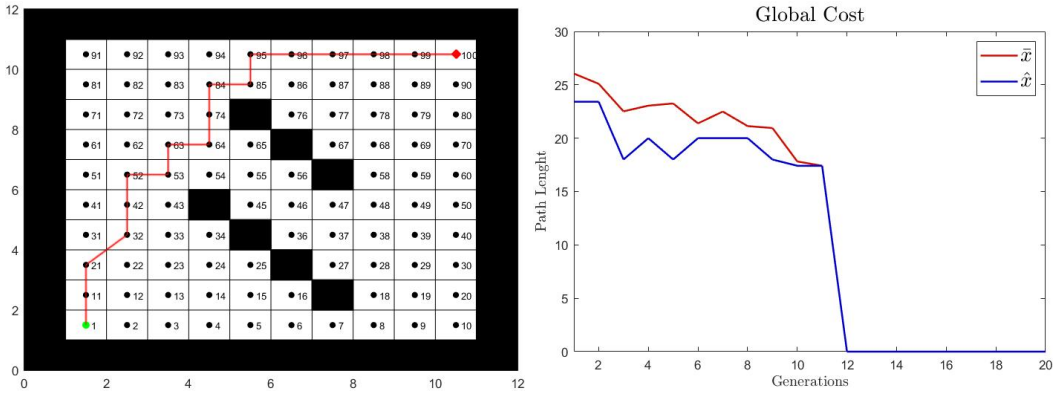
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 47: Tercera validación del algoritmo con el mapa B con parámetros de la fase anterior.



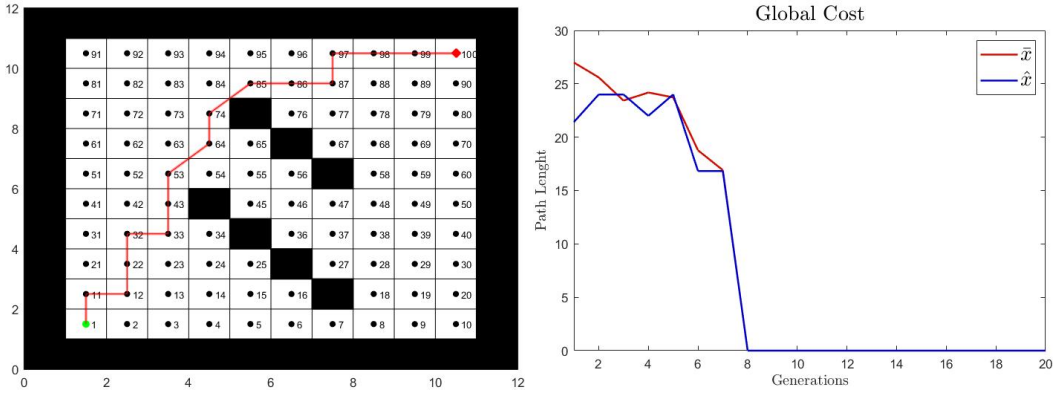
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 48: Primera validación del algoritmo con el mapa B con los parámetros de [4].



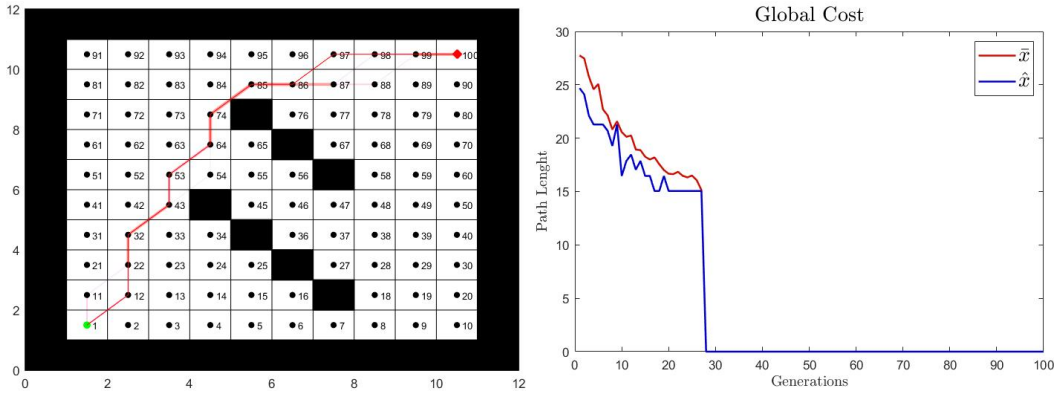
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 49: Segunda validación del algoritmo con el mapa B con los parámetros de [4].



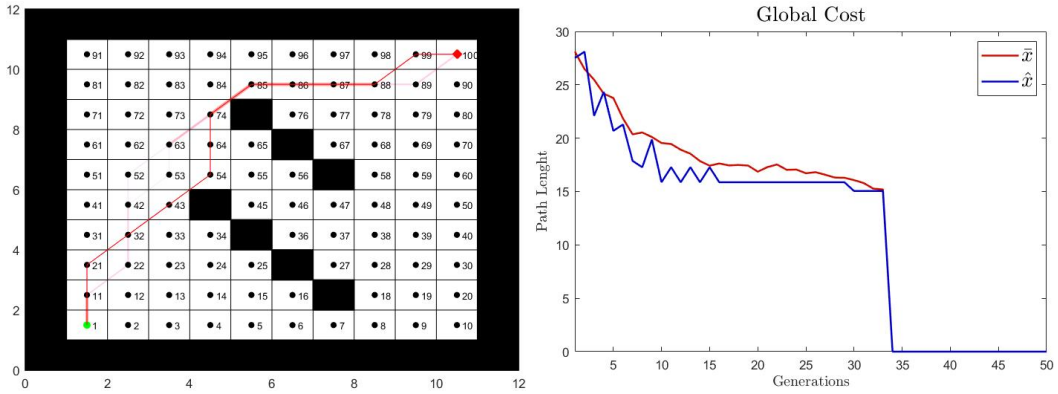
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 50: Tercera validación del algoritmo con el mapa B con los parámetros de [4].



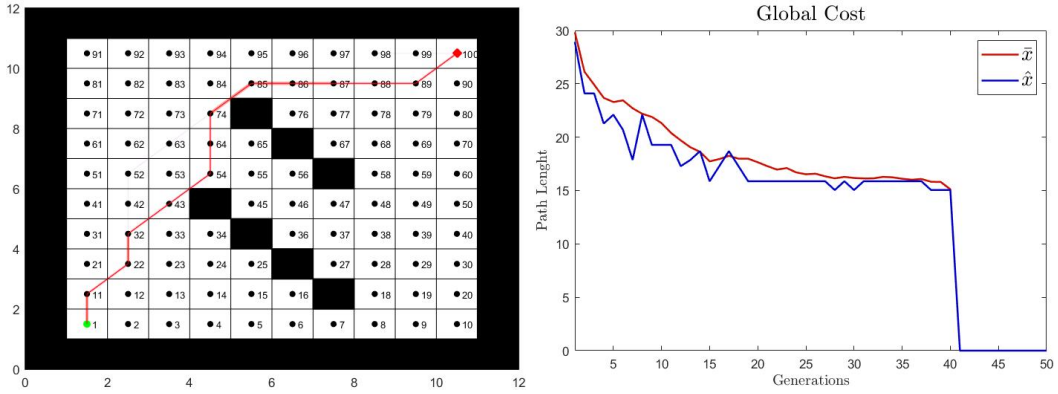
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 51: Primera validación del algoritmo con el mapa B con el primer conjunto de parámetros.



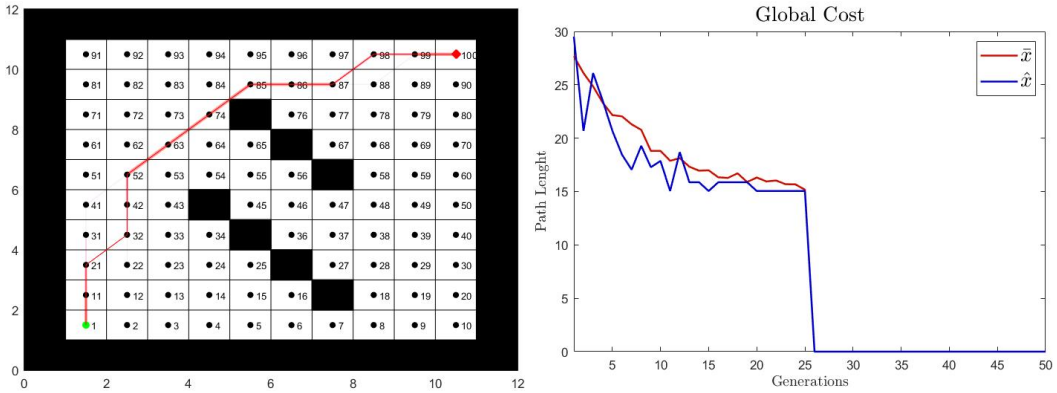
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 52: Segunda validación del algoritmo con el mapa B con el primer conjunto de parámetros.



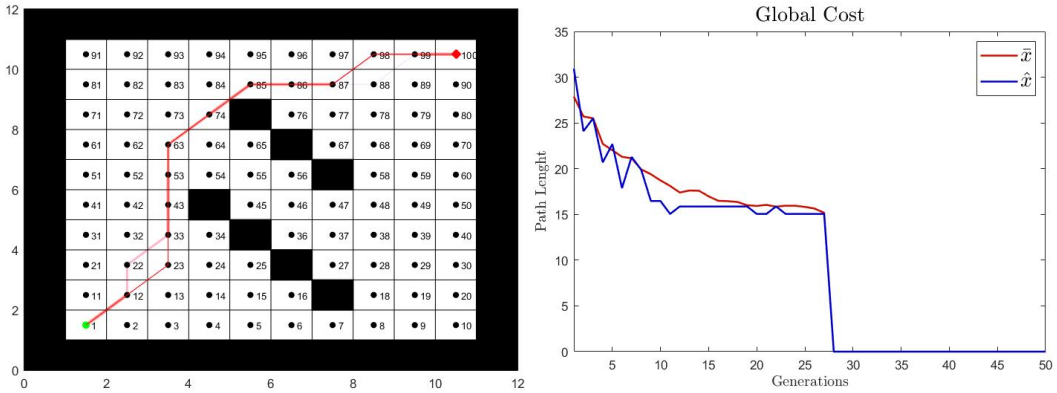
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 53: Tercera validación del algoritmo con el mapa B con el primer conjunto de parámetros.



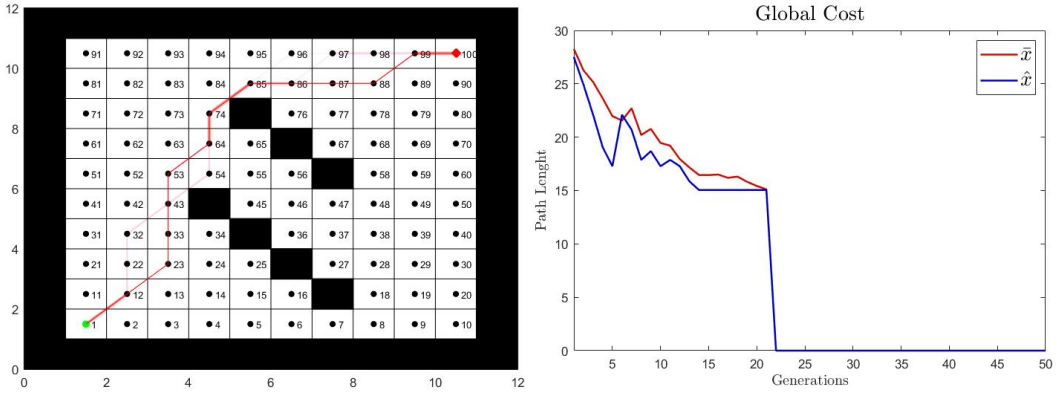
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 54: Primera validación del algoritmo con el mapa B con el segundo conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 55: Segunda validación del algoritmo con el mapa B con el segundo conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 56: Tercera validación del algoritmo con el mapa B con el segundo conjunto de parámetros.

En el Cuadro 21 se muestran los resultados de todas las pruebas realizadas con el mapa B. Se observa que únicamente con los parámetros de [4] no se encontraron trayectorias

óptimas. En el Cuadro 22 se muestran los promedios de las iteraciones, tiempo y costo de los resultados antes mencionados. Se observa que con las constantes de [4] se obtuvo la menor cantidad de iteraciones en promedio y el menor tiempo promedio. Sin embargo, las trayectorias no fueron óptimas.

Corrida	**Parámetros**	**Iteraciones**	**Tiempo (min)**	**Costo**
1	Fase anterior	38	45	15.05
2	Fase anterior	34	42.87	15.05
3	Fase anterior	47	62.69	15.05
1	[4]	11	13.81	17.41
2	[4]	11	14	17.41
3	[4]	7	10.03	16.83
1	Primer conjunto	27	29.75	15.05
2	Primer conjunto	33	33.32	15.05
3	Primer conjunto	40	38.08	15.05
1	Segundo conjunto	25	19.43	15.05
2	Segundo conjunto	27	20.98	15.05
3	Segundo conjunto	21	17.92	15.05

Cuadro 21: Resultados del mapa B.

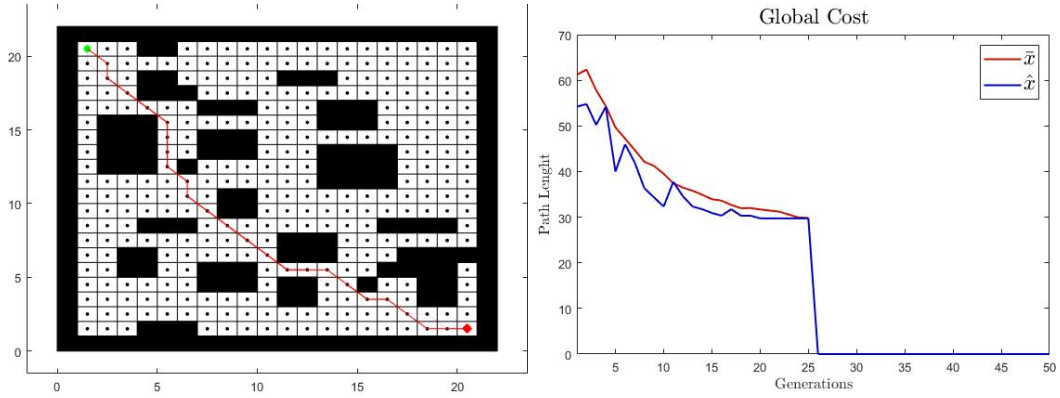
Parámetros	**Promedio de iteraciones**	**Tiempo promedio (min)**	**Costo promedio**
Fase anterior	39.67	50.19	15.05
[4]	9.67	12.61	17.22
Primer conjunto	33.33	33.72	15.05
Segundo conjunto	24.33	19.44	15.05

Cuadro 22: Promedio de los resultados del mapa B.

7.4.3. Mapa C

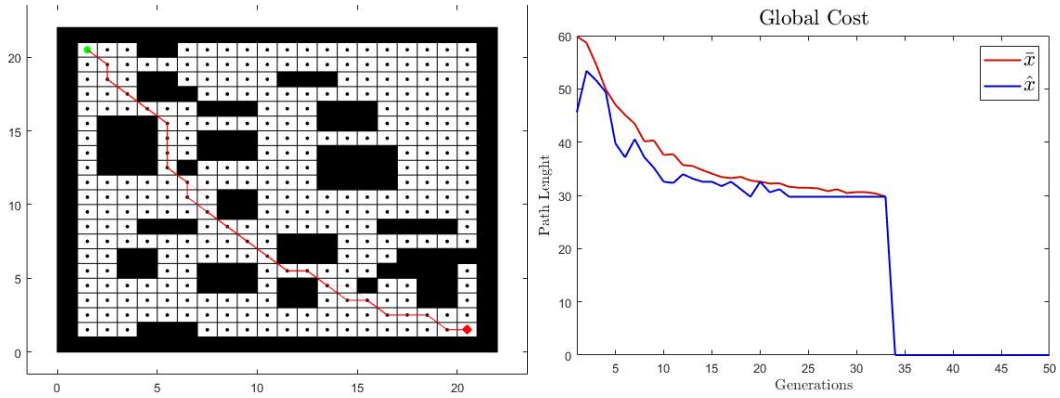
En este mapa se utilizaron los parámetros del primer y segundo conjunto. No se utilizaron los parámetros que se presentan en [4] ya que al utilizar estos con los mapas A y B no se obtuvieron trayectorias óptimas. No se utilizaron los parámetros de la fase anterior ya que el tiempo promedio de ejecución del algoritmo fue mayor en los mapas A y B, respecto al primer y segundo conjunto de parámetros.

En las Figuras 57, 58 y 59 se muestran los resultados obtenidos con el primer conjunto de parámetros para el mapa C. En las Figuras 60, 61 y 62 se muestran los resultados obtenidos con el segundo conjunto de parámetros. Se observa que casi todas las trayectorias obtenidas son óptimas. Asimismo, se observa que la trayectoria evade los obstáculos del mapa y que no pasa por medio de pasillos estrechos.



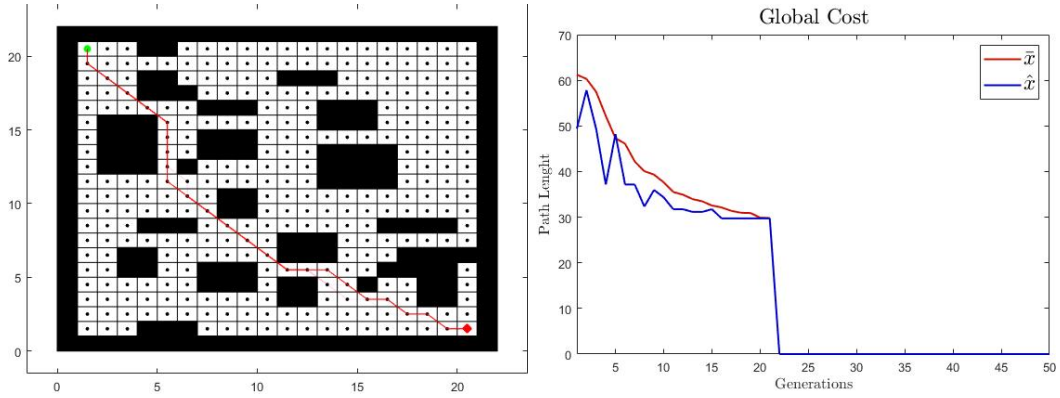
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 57: Primera validación del algoritmo con el mapa C con el primer conjunto de parámetros.



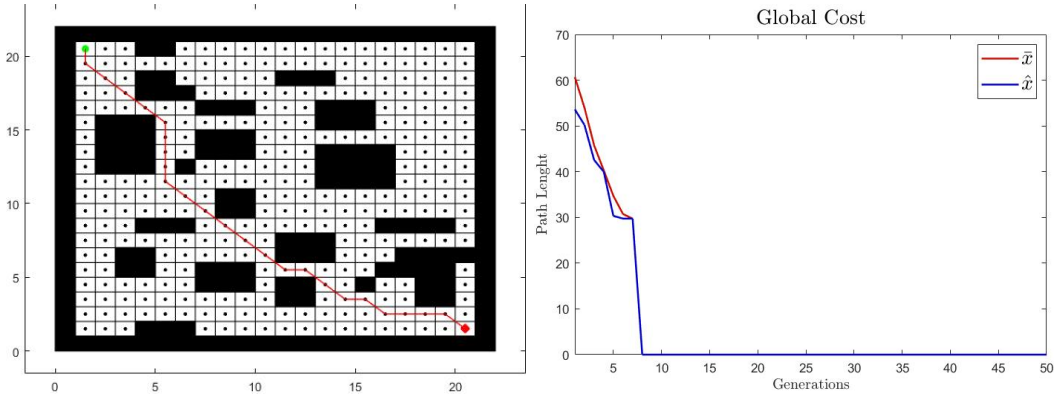
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 58: Segunda validación del algoritmo con el mapa C con el primer conjunto de parámetros.



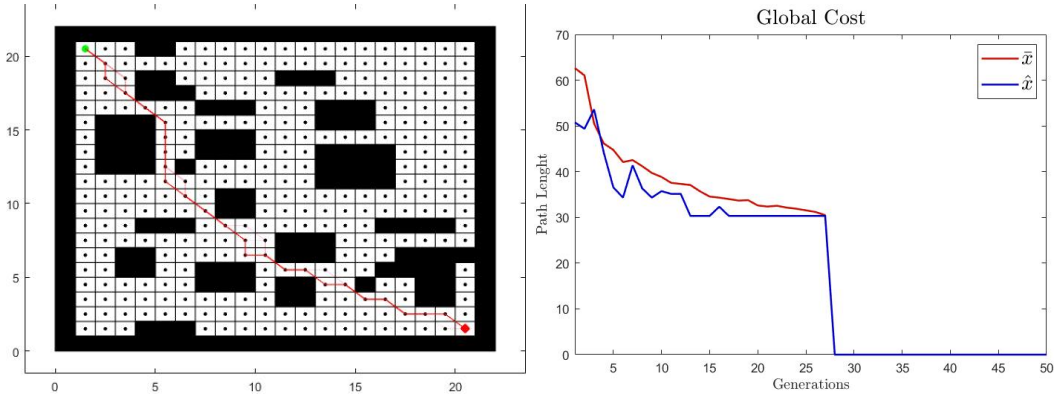
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 59: Tercera validación del algoritmo con el mapa C con el primer conjunto de parámetros.



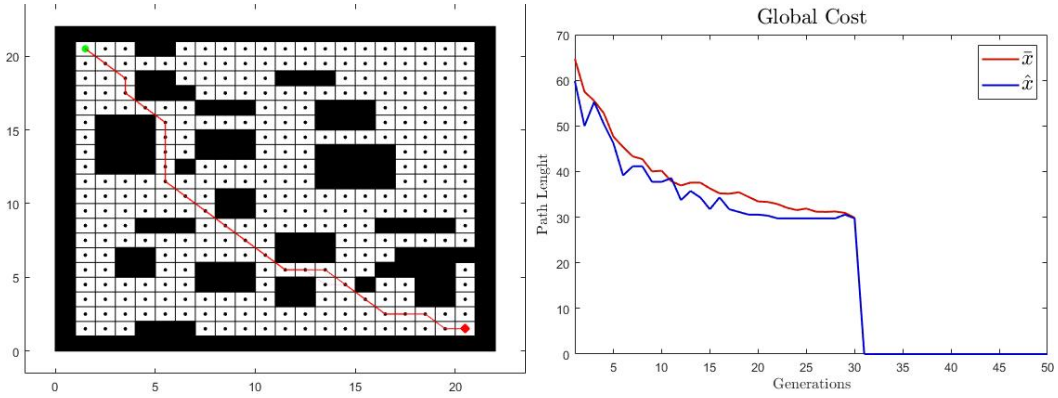
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 60: Primera validación del algoritmo con el mapa C con el segundo conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 61: Segunda validación del algoritmo con el mapa C con el segundo conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 62: Tercera validación del algoritmo con el mapa C con el segundo conjunto de parámetros.

En el Cuadro 23 se muestran los resultados obtenidos para el mapa C. Se observa que en una iteración del segundo conjunto de parámetros no se logró una trayectoria óptima ya

que el costo no fue el mínimo. En el Cuadro 24 En promedio, se obtienen tiempos menores y menos iteraciones con el segundo conjunto de parámetros. Con el primer conjunto de parámetros se tienen más aciertos de encontrar trayectorias óptimas.

Corrida	Parámetros	Iteraciones	Tiempo (min)	Costo
1	Primer conjunto	25	61.73	29.74
2	Primer conjunto	33	68.81	29.74
3	Primer conjunto	21	52.33	29.74
1	Segundo conjunto	7	23.43	29.74
2	Segundo conjunto	27	44.64	30.33
3	Segundo conjunto	30	52.68	29.74

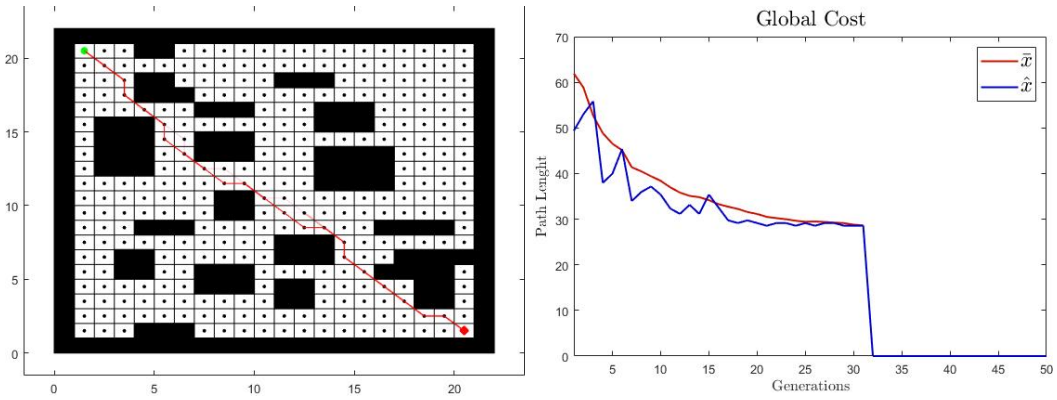
Cuadro 23: Resultados del mapa C.

Parámetros	Promedio de iteraciones	Tiempo promedio (min)	Costo promedio
Primer conjunto	26.33	60.96	29.74
Segundo conjunto	21.33	40.25	29.94

Cuadro 24: Promedio de los resultados del mapa C.

7.4.4. Mapa D

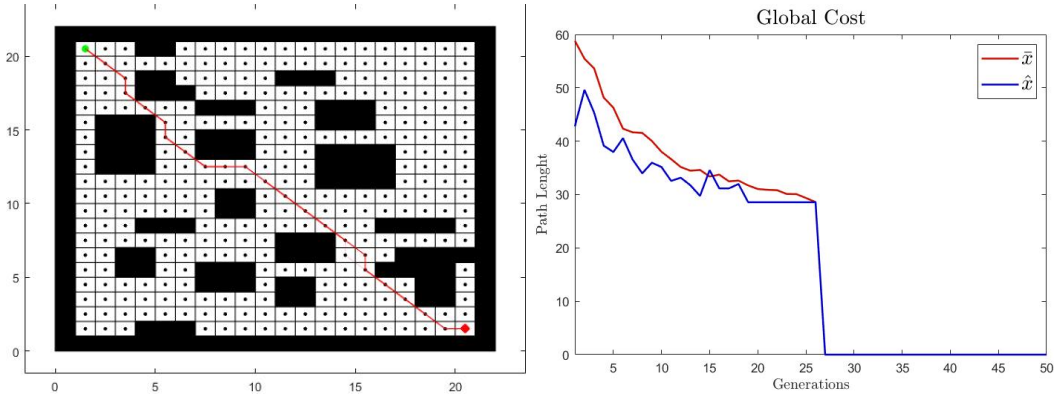
En las Figuras 63, 64 y 65 se muestran los resultados obtenidos para el mapa D. Se observa que las trayectorias encontradas son óptimas ya que se mantienen cerca de la diagonal del mapa. Los obstáculos se esquivan de forma satisfactoria y en las gráficas de media y moda se observa que la moda llega a su valor mínimo, lo cual asegura que la trayectoria es la mínima encontrada por el algoritmo. En el Cuadro 25 se observan los resultados tabulados del mapa D. Se observa que el costo fue el mismo para todas las iteraciones. En el Cuadro 26 se observan los promedios de los resultados.



(a) Trayectoria y animación de feromona.

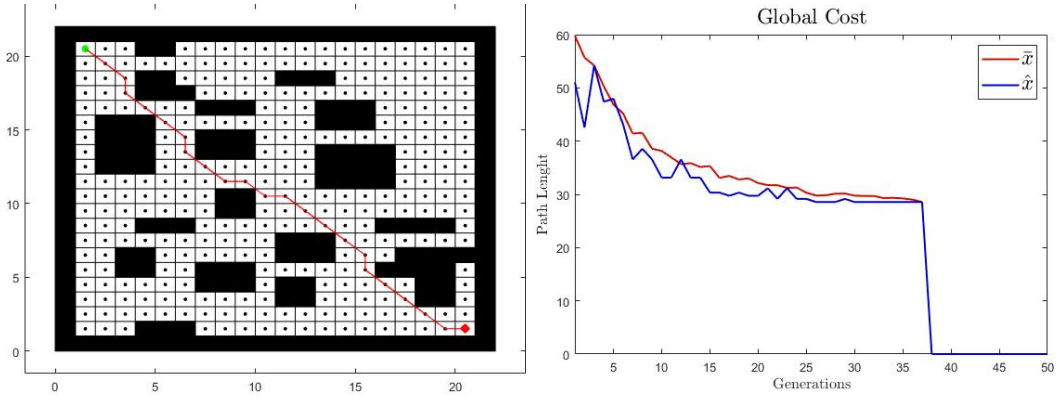
(b) Media y moda de los caminos.

Figura 63: Primera validación del algoritmo con el mapa D con el primer conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 64: Segunda validación del algoritmo con el mapa D con el primer conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 65: Tercera validación del algoritmo con el mapa C con el primer conjunto de parámetros.

Corrida	Parámetros	Iteraciones	Tiempo (min)	Costo
1	Primer conjunto	31	60.36	28.56
2	Primer conjunto	26	51.66	28.56
3	Primer conjunto	37	67.9	28.56

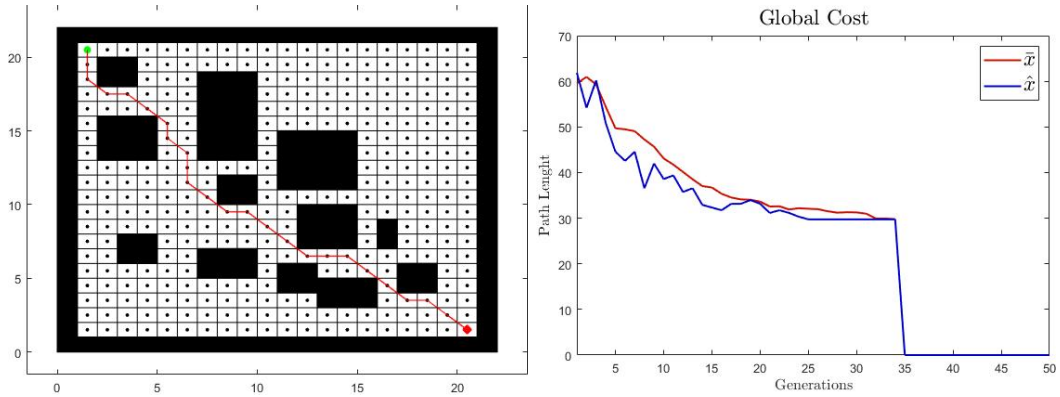
Cuadro 25: Resultados del mapa D.

Parámetros	Promedio de iteraciones	Tiempo promedio (min)	Costo promedio
Primer conjunto	31.33	59.97	28.56

Cuadro 26: Promedio de los resultados del mapa D.

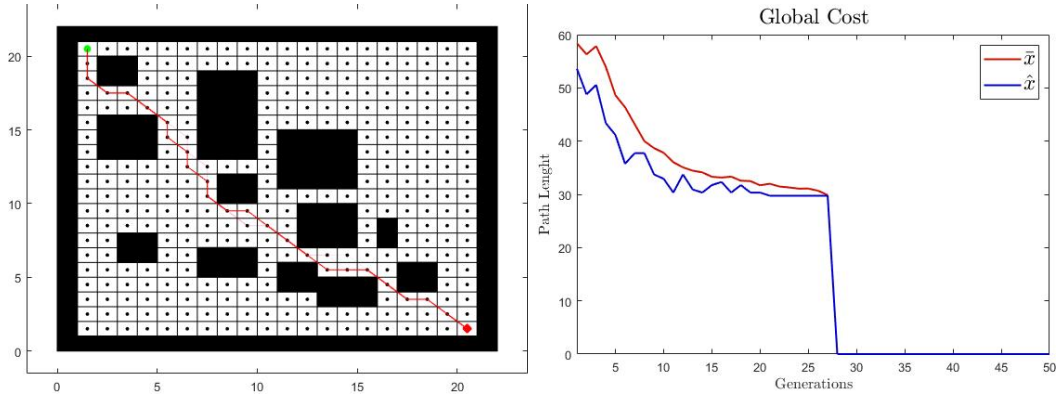
7.4.5. Mapa E

En las Figuras 66, 67 y 68 se muestran los resultados obtenidos para el mapa E. Se observa que la moda llega a su valor mínimo. Por lo tanto, las trayectorias encontradas son óptimas. El algoritmo logró esquivar obstáculos de forma satisfactoria para este mapa. En el Cuadro 25 se observan los resultados obtenidos para el mapa E. Se observa que el costo es el mismo para todas las trayectorias. En el Cuadro 26 se observa el promedio de los resultados presentados en el Cuadro antes mencionado.



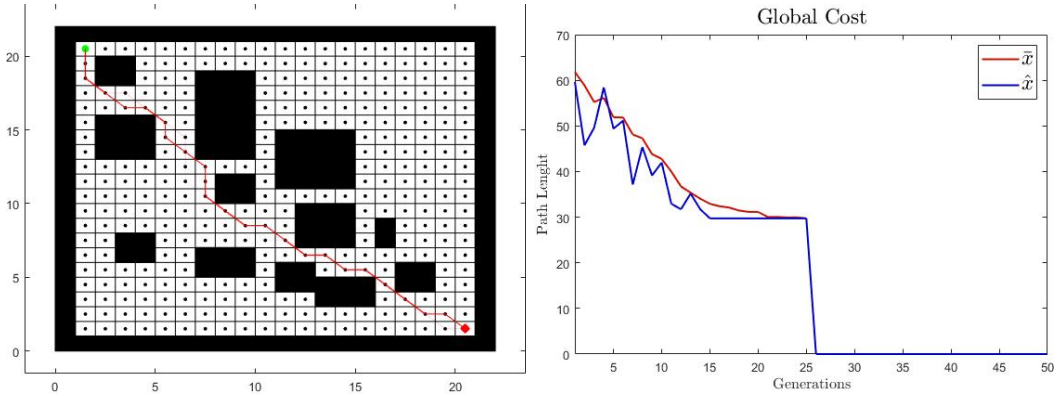
(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 66: Primera validación del algoritmo con el mapa E con el primer conjunto de parámetros.



(a) Trayectoria y animación de feromona. (b) Media y moda de los caminos.

Figura 67: Segunda validación del algoritmo con el mapa E con el primer conjunto de parámetros.



(a) Trayectoria y animación de feromona.

(b) Media y moda de los caminos.

Figura 68: Tercera validación del algoritmo con el mapa E con el primer conjunto de parámetros.

Corrida	Parámetros	Iteraciones	Tiempo (min)	Costo
1	Primer conjunto	34	73.28	29.74
2	Primer conjunto	27	56.21	29.74
3	Primer conjunto	25	55.79	29.74

Cuadro 27: Resultados del mapa E.

Parámetros	Promedio de iteraciones	Tiempo promedio (min)	Costo promedio
Primer conjunto	28.67	61.76	29.74

Cuadro 28: Promedio de los resultados del mapa E.

CAPÍTULO 8

Validación de trayectorias en Webots

En las pruebas realizadas con Matlab no se consideraron las restricciones de los actuadores, ni algún modelo de robot en específico. Para considerar las restricciones de velocidad de los motores y garantizar que el robot hace lo que se quiere, es necesario implementar controladores. En este capítulo se presenta el proceso de validación de las trayectorias obtenidas por el algoritmo de planificación de trayectorias y evasión de obstáculos. Esto abarca la metodología empleada, los controladores utilizados y los resultados obtenidos. La validación se realizó en la plataforma de Webots versión 2021a debido a la familiarización con dicha plataforma.

8.1. Metodología

Se colocó una mesa de 2 $\times$ 2 metros al igual que en la fase anterior [6]. Esta mesa se colocó en el origen del mundo, por lo tanto el marco de referencia de la mesa es el mismo que el marco de referencia inercial. La cuadrícula del piso de la mesa se hizo de 0.2 $\times$ 0.2 metros, de tal modo que se obtiene una cuadrícula de 10 $\times$ 10 cuadros. Los obstáculos se colocaron como objetos tipo *WoodenBox* que tienen el tamaño de un cuadro.

Para la validación se utilizó un mapa más sencillo que los utilizados anteriormente. Este mapa se muestra en la Figura 69 junto con la trayectoria generada por el algoritmo. La trayectoria va del nodo 1 al nodo 100. Los parámetros utilizados en el algoritmo fueron los del segundo conjunto de parámetros.

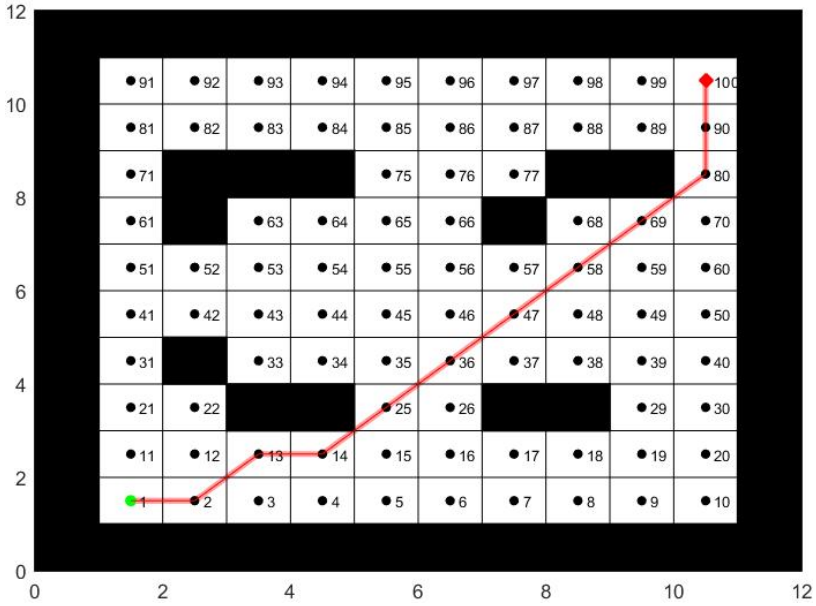


Figura 69: Mapa utilizado para la validación en Webots.

Se utilizaron controladores punto a punto, donde las coordenadas del siguiente nodo en la trayectoria son el siguiente punto a llegar. Estos puntos son obtenidos con el algoritmo de planificación de trayectorias y evasión de obstáculos. El punto objetivo se cambia al siguiente hasta que el robot llega a una posición muy cercana al punto objetivo actual.

El ACO no puede implementarse directamente en Webots, ya que en este algoritmo las hormigas regresan en cada iteración al nodo inicial. Por esta razón, se utilizó únicamente un robot diferencial para replicar la trayectoria. El robot que se utilizó fue el e-puck ya que este se utilizó en las fases anteriores a este trabajo.

En la Figura 70 se muestra el mapa de la validación replicado en Webots. En la parte inferior derecha se muestra el sistema de coordenadas utilizado en la mesa.

El ajuste de las constantes de los controladores se hizo a prueba y error. Para tener una idea de dónde empezar, se tomaron las constantes de la fase anterior [6]. Para la implementación de los controladores se tomó como referencia el código del controlador trabajado en la fase anterior.

8.2. Controladores

En la fase anterior [6] se encontró que los contorladores LQI, controlador de pose y controlador de pose de Lyapunov eran los que mejor funcionaban con el ACO. Por esta razón, se tomaron estos controladores para la validación de las trayectorias que genera el algoritmo de planificación de trayectorias y evasión de obstáculos.

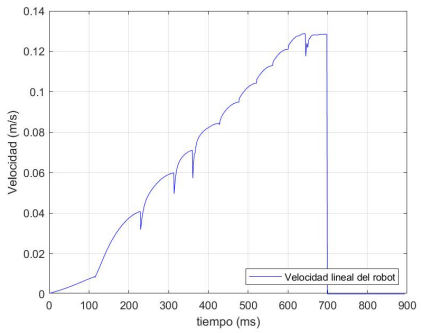
Figura 70: Mapa en Webots.

8.2.1. LQI

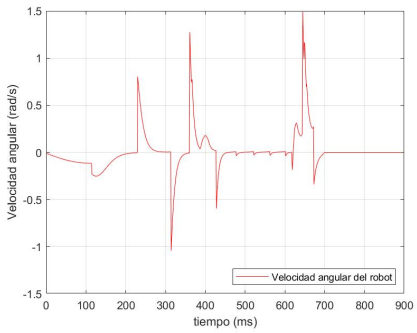
Los parámetros del controlador se muestran en el Cuadro 29. En la Figura 71 se muestran los resultados obtenidos con este controlador. La velocidad lineal y angular presenta varios picos y cambios bruscos. La velocidad de los motores también presenta varios picos y cambios bruscos. Las discontinuidades en estas gráficas ocurren debido a que los controladores implementados son de punto a punto. Entonces, al llegar al siguiente punto el robot disminuye un poco la velocidad y al cambiar de punto aumenta la velocidad. En la Figura 71d se observa que la trayectoria obtenida es bastante suave. Este controlador funciona para replicar las trayectorias obtenidas con el algoritmo de planificación de trayectorias y evasión de obstáculos. El robot logró llegar al nodo deseado a pesar de los obstáculos.

Parámetro	Tamaño de Matriz	Valor de Matriz
Q	2×2	$\boldsymbol{I}_{2\times2}$
R	2×2	$2000\boldsymbol{I}_{2\times2}$
QQ	4×4	$\boldsymbol{I}_{4\times4}$, QQ(3,3) = 5, QQ(4,4) = 0.0002
K_{lqi}	2×4	$\boldsymbol{0}_{4\times4}$, K_{lqi}(1,1) = 0.317, K_{lqi}(2,2) = 0.0337, K_{lqi}(1,3)=0.05, K_{lqi}(2,4)=0.0003

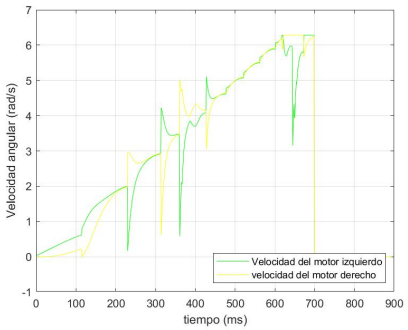
Cuadro 29: Parámetros utilizados para el controlador LQI.



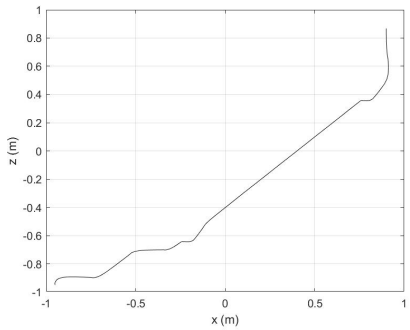
(a) Velocidad lineal.



(b) Velocidad angular.



(c) Velocidad en los motores.



(d) Trayectoria generada.

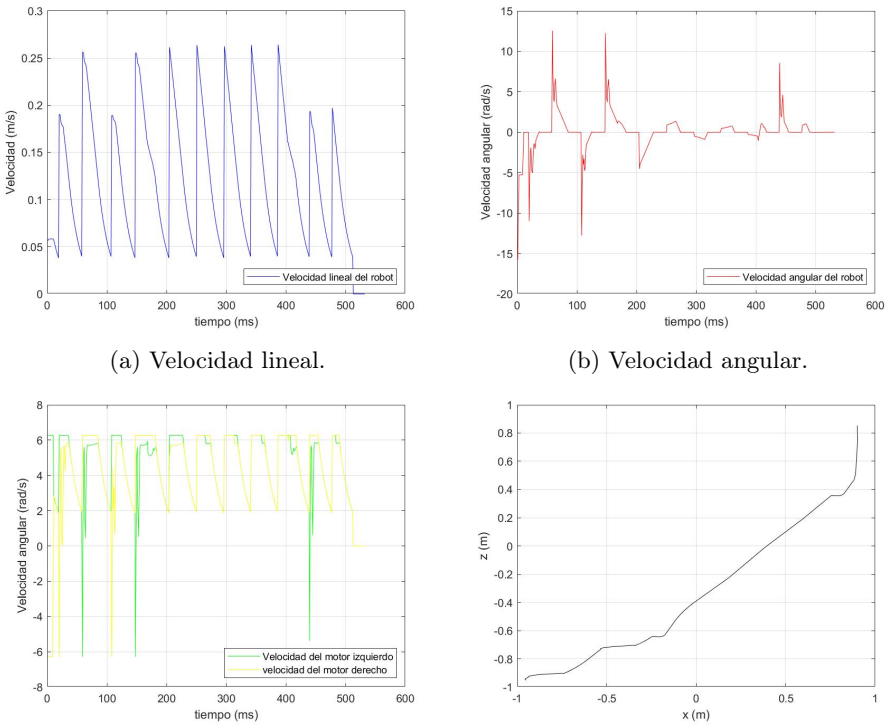
Figura 71: Resultados del controlador LQI.

8.2.2. Controlador de pose

En el Cuadro 30 se muestra los parámetros utilizados para el controlador de pose. Estos se utilizaron con las ecuaciones (36)-(37). Los resultados obtenidos con este controlador se muestran en la Figura 72. Las velocidades angular y lineal presentan bastantes picos. Las velocidades de los motores presentan cambios bruscos. En la Figura 72d se observa que la trayectoria seguida por el robot fue suave. El controlador cumplió con llevar al robot en la trayectoria de forma satisfactoria. En la escala de tiempo de las gráficas se observa que este controlador es más rápido que el controlador anterior.

Parámetro	**Valor**
K_ρ	0.8
K_α	20
K_β	-0.05

Cuadro 30: Parámetros utilizados para el controlador de pose.



(a) Velocidad lineal.

(b) Velocidad angular.

(c) Velocidad en los motores.

(d) Trayectoria generada.

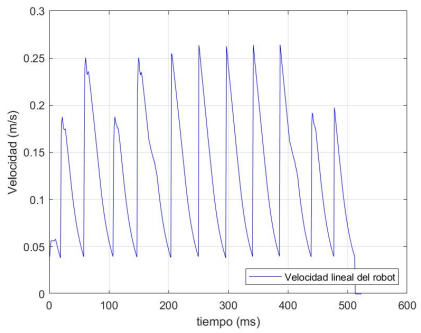
Figura 72: Resultados del controlador de pose.

8.2.3. Controlador de pose de Lyapunov

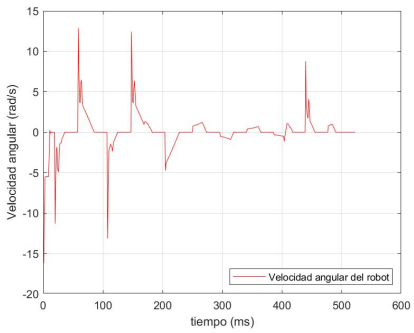
En el Cuadro 31 se muestran los parámetros utilizados para el controlador. Estos valores se utilizaron con las ecuaciones (38) para la implementación. El comportamiento en las velocidades es bastante similar al comportamiento del controlador anterior. La trayectoria generada es bastante parecida a la obtenida con el controlador antes mencionado. En este caso, el controlador también cumplió con seguir la trayectoria dada y resultó ser más rápido que el LQI.

Parámetro	**Valor**
K_ρ	0.8
K_α	20

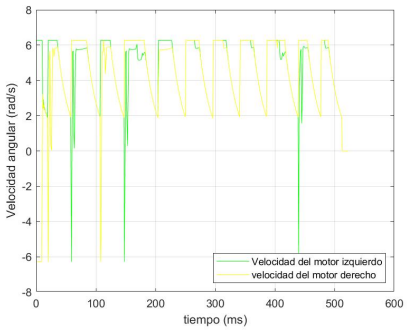
Cuadro 31: Parámetros utilizados para el controlador de pose de Lyapunov.



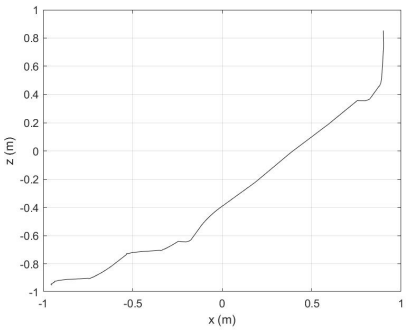
(a) Velocidad lineal.



(b) Velocidad angular.



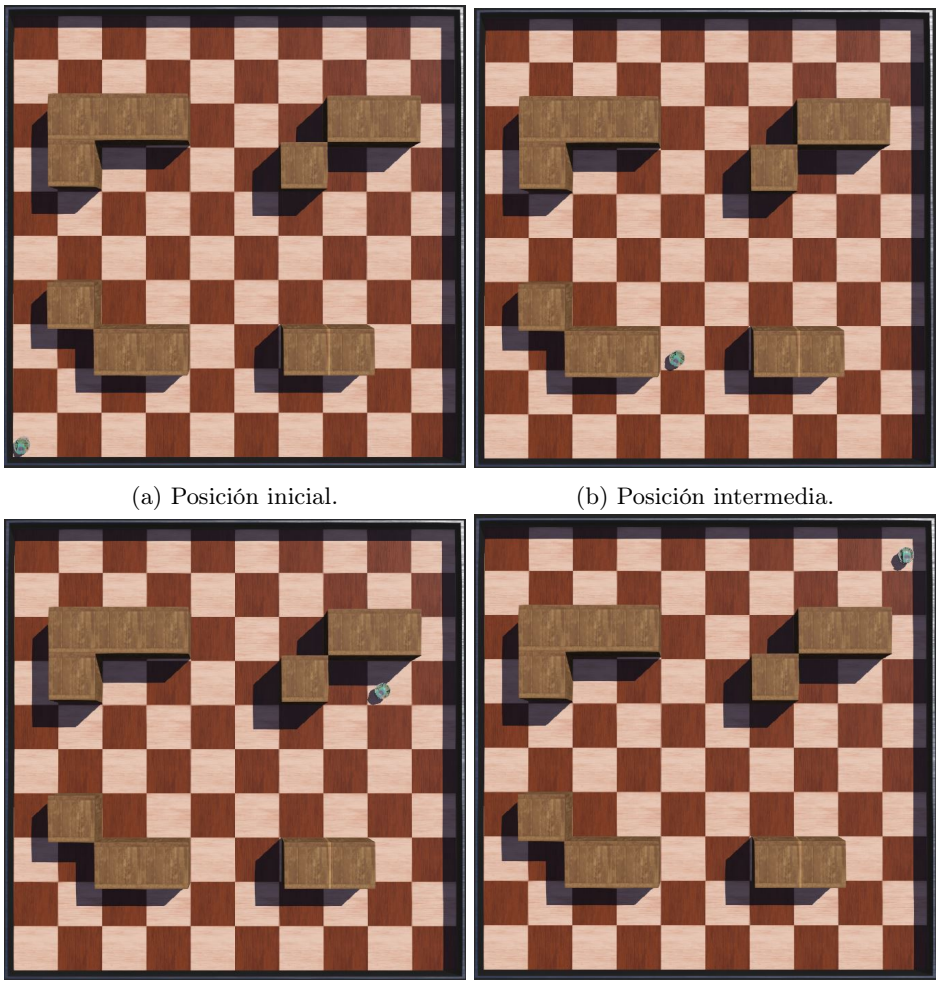
(c) Velocidad en los motores.



(d) Trayectoria generada.

Figura 73: Resultados del controlador de pose de Lyapunov.

En la Figura 74 se muestra la secuencia de movimientos realizada por el robot en Webots. Durante este procedimiento se utilizó el controlador de pose de Lyapunov. Se observa que las posiciones del e-puck de las Figuras están dentro de la trayectoria mostrada en la Figura 69.



(a) Posición inicial.

(b) Posición intermedia.

(c) Posición intermedia.

(d) Posición final.

Figura 74: Secuencia de movimientos del e-puck en Webots.

CAPÍTULO 9

Algoritmo de exploración de terrenos

En este capítulo se muestra la metodología utilizada para la implementación del algoritmo de exploración de terrenos. Asimismo, se muestran los resultados obtenidos para dicho algoritmo. El objetivo de este algoritmo es encontrar el camino entre dos nodos, considerando obstáculos, y realizar un mapa de lo que encontraron las hormigas explorando. Esta implementación se realizó en Matlab.

9.1. Metodología

Se tomó como base el algoritmo de planificación y evasión de obstáculos. Se eliminó la parte que involucra la matriz de adyacencia D que se calcula con la ecuación (15), esto se hizo para bajar el tiempo de ejecución del programa y que el algoritmo se pudiera paralelizar.

El algoritmo de planificación y evasión de obstáculos es capaz de encontrar la dirección en la que está un obstáculo respecto a la posición actual de cada hormiga. Se aprovechó esta detección para que cada hormiga guarde en su estructura de datos, en qué nodo encontró un obstáculo. Luego, se obtienen todos los obstáculos que detectó cada hormiga, se eliminan las repeticiones y se crea un nuevo mapa, que es el mapa explorado.

Se utilizó la ecuación (3) como función de probabilidad. Esta función se utilizó en la fase anterior [6]. Por esta razón, este algoritmo no utiliza el parámetro γ.

Las pruebas se realizaron en la computadora de alto rendimiento de la universidad. De este modo se aprovechó que el algoritmo se paralelizó y así poder reducir el tiempo de ejecución.

9.2. Validación del algoritmo de exploración de terrenos

La validación del algoritmo se realizó en los tres mapas antes mencionados. Se utilizaron los parámetros del primer conjunto. Se realizaron corridas desde diferentes puntos de inicio y final, y así identificar las diferencias con el mapa devuelto con el algoritmo. En los mapas cuadriculados, el punto verde representa al nodo de inicio y el punto rojo al nodo de destino.

En la Figura 75 se muestra la numeración de los nodos de un mapa de 20×20 cuadros. Se evitó colocar la numeración de los nodos en los resultados para evitar saturar las imágenes.

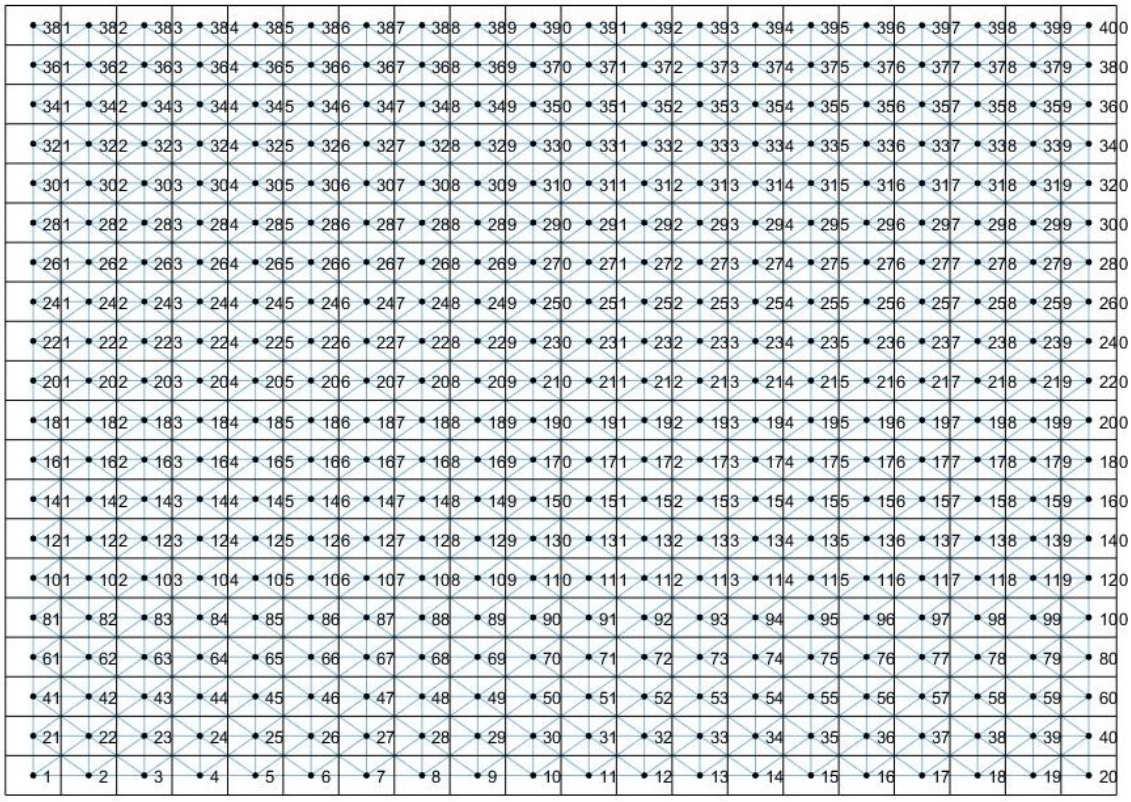
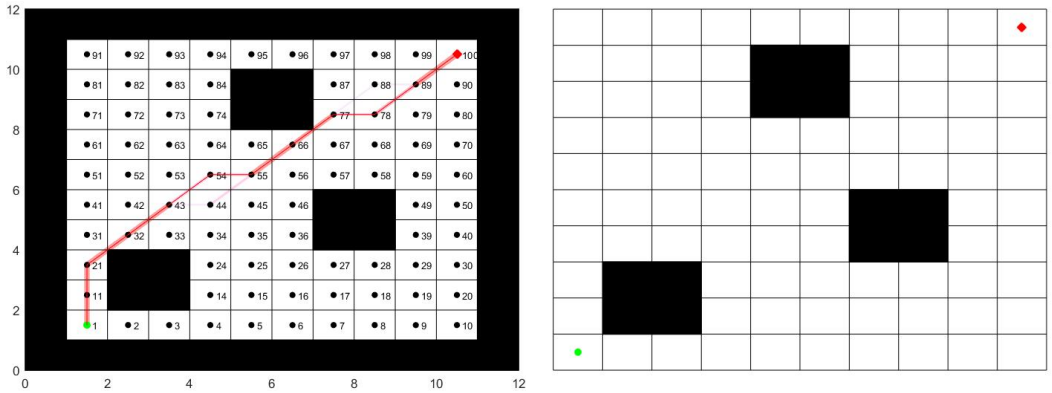


Figura 75: Mapa de 20×20 cuadros con numeración de nodos.

9.2.1. Mapa A

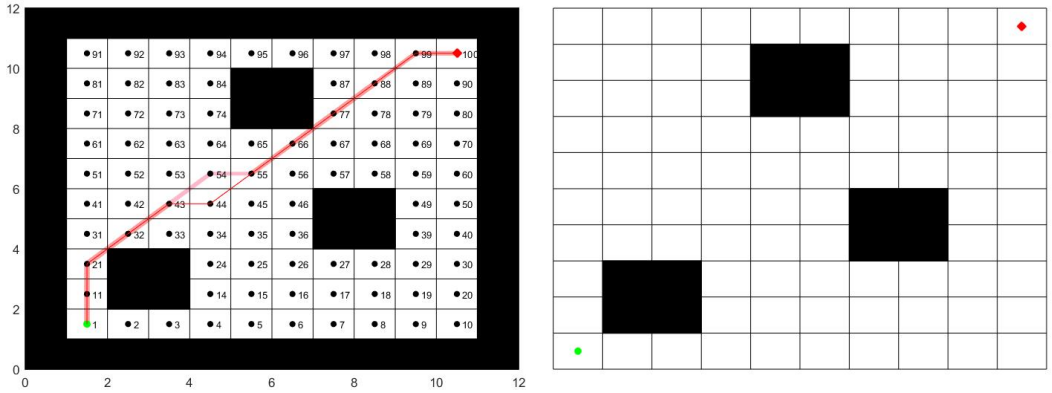
En las Figuras 76, 77 y 78 se muestran los resultados de explorar el mapa A desde el nodo 1 al 100. En la Figura 76a se observa la trayectoria obtenida para el mapa. En la Figura 76b se observa el mapa que se genera con la información obtenida de la exploración. En los resultados se observa que para los tres casos se replicó el mapa completamente.

En el Cuadro 32 se muestran los resultados obtenidos para el mapa A. El costo de la trayectoria fue el mismo para los tres casos. En el Cuadro 33 se muestra el promedio de los resultados obtenidos.



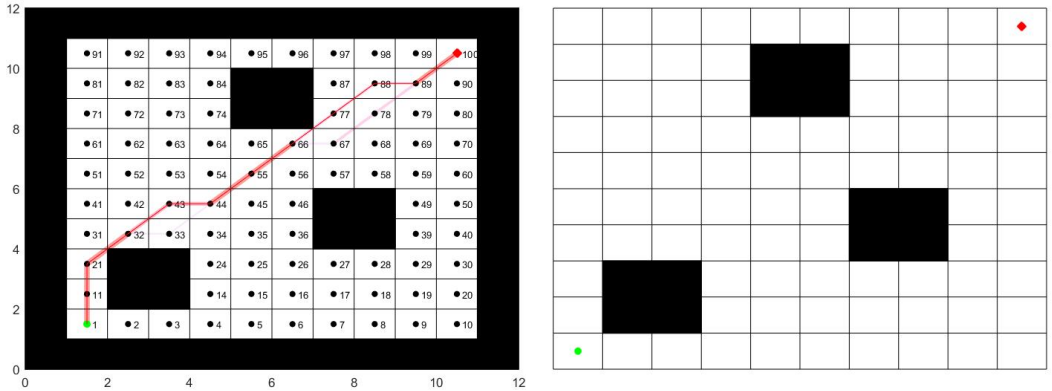
(a) Mapa A con trayectoria del nodo 1 al 100. (b) Mapa A generado.

Figura 76: Exploración del mapa A con trayectoria del nodo 1 al 100.



(a) Mapa A con trayectoria del nodo 1 al 100. (b) Mapa A generado.

Figura 77: Exploración del mapa A con trayectoria del nodo 1 al 100.



(a) Mapa A con trayectoria del nodo 1 al 100. (b) Mapa A generado.

Figura 78: Exploración del mapa A con trayectoria del nodo 1 al 100.

Corrida	Iteraciones	Tiempo (min)	Costo
1	25	14.98	13.9
2	16	12	13.9
3	33	17.22	13.9

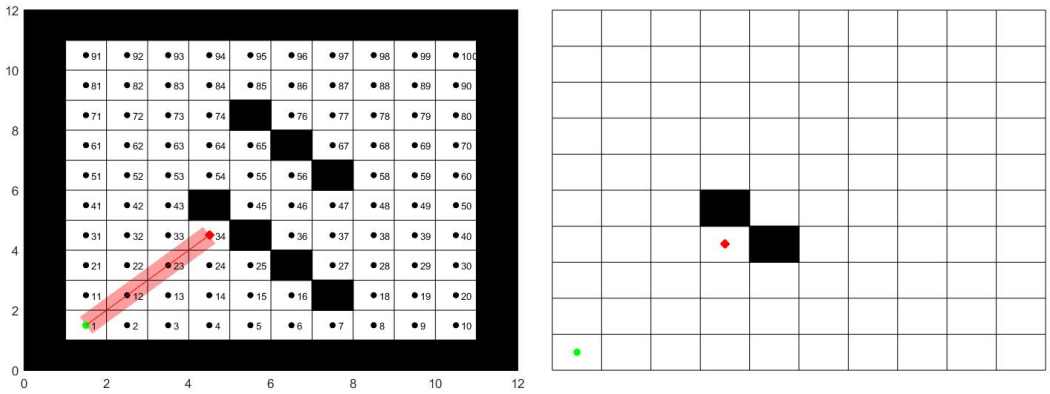
Cuadro 32: Resultados del mapa A.

Promedio de iteraciones	Tiempo promedio (min)	Costo promedio
24.67	14.73	13.9

Cuadro 33: Promedio de los resultados del mapa A.

9.2.2. Mapa B

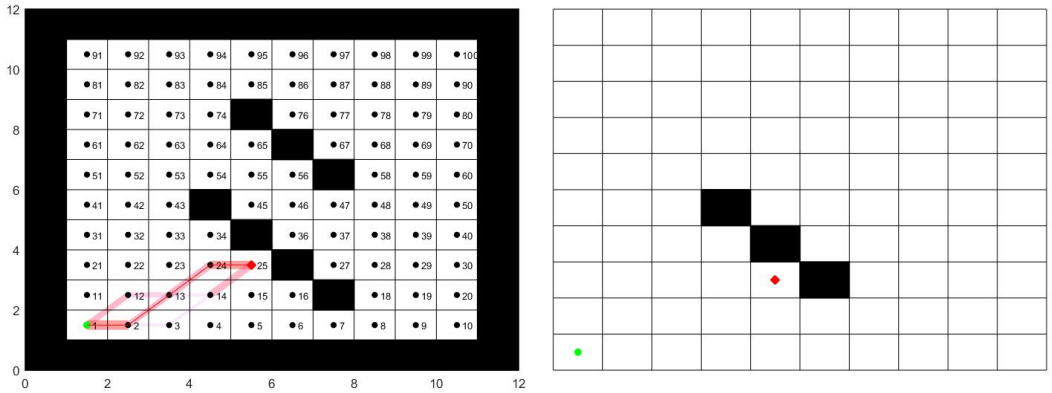
En la Figura 79 se observan los resultados de explorar el mapa B desde el nodo 1 al 34. En este caso los únicos obstáculos que detectó el algoritmo fueron los que están en la vecindad al nodo 34. Esto quiere decir que las hormigas encontraron rápido le nodo de destino y ya no siguieron explorando.



(a) Mapa B con trayectoria del nodo 1 al 34. (b) Mapa B generado.

Figura 79: Exploración del mapa B con trayectoria del nodo 1 al 34.

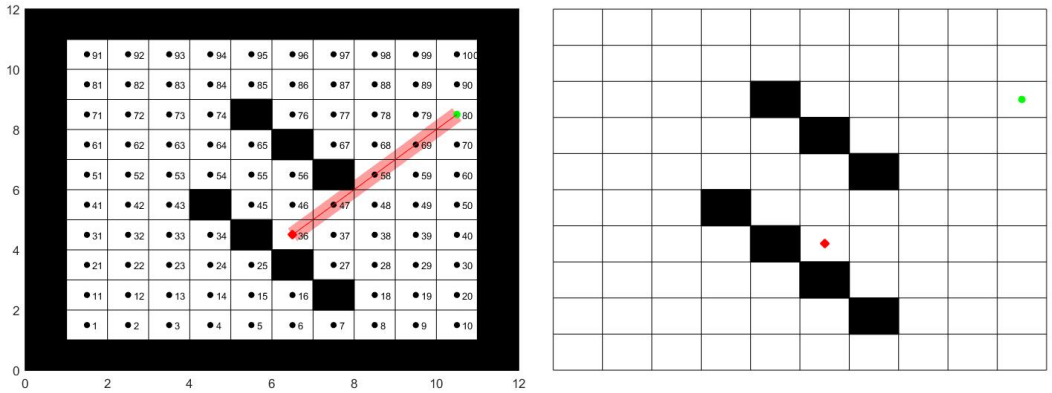
Utilizando el mapa B y colocando como nodo de destino al nodo 25 y como nodo de inicio al nodo 1, se obtuvieron los resultados que se muestran en la Figura 80. En este caso el mapa generado no presenta muchos obstáculos, lo que indica que no fue necesario explorar mucho para llegar al destino.



(a) Mapa B con trayectoria del nodo 1 al 25. (b) Mapa B generado.

Figura 80: Exploración del mapa B con trayectoria del nodo 1 al 25.

En la Figura 81 se observa el resultado de la exploración desde el nodo 80 hasta el 36. Se observa que el mapa generado es igual al original. En este caso se exploró más el mapa, por lo que se detectaron todos los obstáculos.



(a) Mapa B con trayectoria del nodo 80 al 36. (b) Mapa B generado.

Figura 81: Exploración del mapa B con trayectoria del nodo 80 al 36.

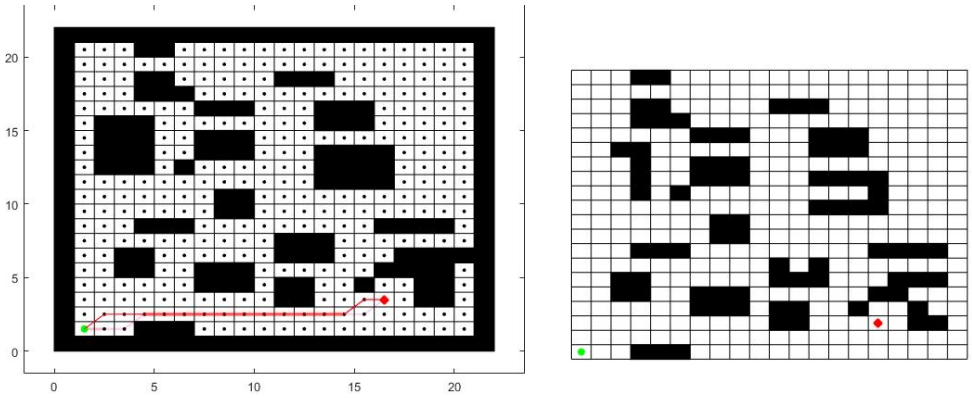
En el Cuadro 34 se muestran los resultados obtenidos de las pruebas anteriores. El tiempo de ejecución fue bastante bajo para los tres casos.

Corrida	**Iteraciones**	**Tiempo (min)**	**Costo**
1	5	0.78	4.23
2	5	0.98	4.82
3	6	1.26	5.64

Cuadro 34: Resultados del mapa B.

9.2.3. Mapa C

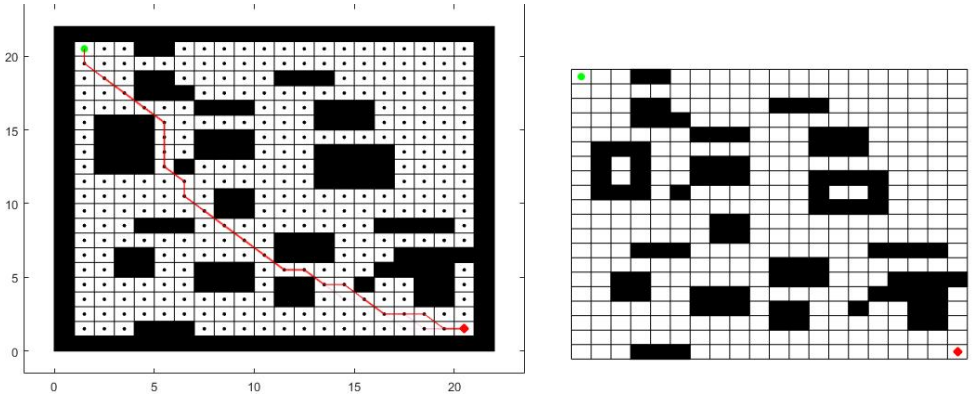
Estableciendo el nodo 1 como inicio y el nodo 56 como final se obtuvieron los resultados de la Figura 82. La diferencia en el mapa original y el mapa explorado se debe a que como el mapa es grande, y el nodo de destino está cerca del nodo de inicio, las hormigas no exploran todo el mapa, sino que solo pasan por los puntos que las pueden llevar al nodo de destino. Cuando los obstáculos son muy grandes el algoritmo deja huecos en el centro de estos. Esto se debe a que como las hormigas no pueden ver qué hay dentro de un obstáculo, nunca almacenan en su estructura de datos a los nodos que quedan en el centro de un obstáculo grande.



(a) Mapa C con trayectoria del nodo 1 al 56. (b) Mapa C generado.

Figura 82: Exploración del mapa C con trayectoria del nodo 1 al 56.

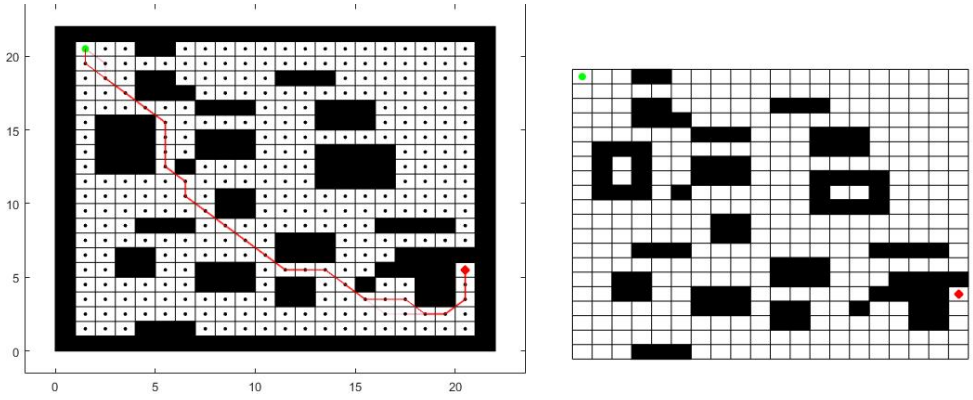
Con una trayectoria del nodo 381 al 20 se obtuvieron los resultados de la Figura 83. Se observa que el mapa de la Figura 83a y el de la Figura 83b son muy parecidos, y son más similares entre sí que los resultados de la Figura 82. La mejora en la similitud se debe a que los nodos de inicio y destino están en esquinas opuestas, entonces las hormigas deben atravesar todo el mapa para llegar a la meta.



(a) Mapa C con trayectoria del nodo 381 al 20. (b) Mapa C generado.

Figura 83: Exploración del mapa C con trayectoria del nodo 381 al 20.

Al tomar como nodo de inicio al 381 y al 100 como nodo de destino se obtuvieron los resultados de la Figura 84. El mapa generado y el original son casi idénticos, con la única diferencia que los obstáculos grandes no pueden ser rellenados.



(a) Mapa C con trayectoria del nodo 1 al 100.

(b) Mapa C generado.

Figura 84: Exploración del mapa C con trayectoria del nodo 1 al 100.

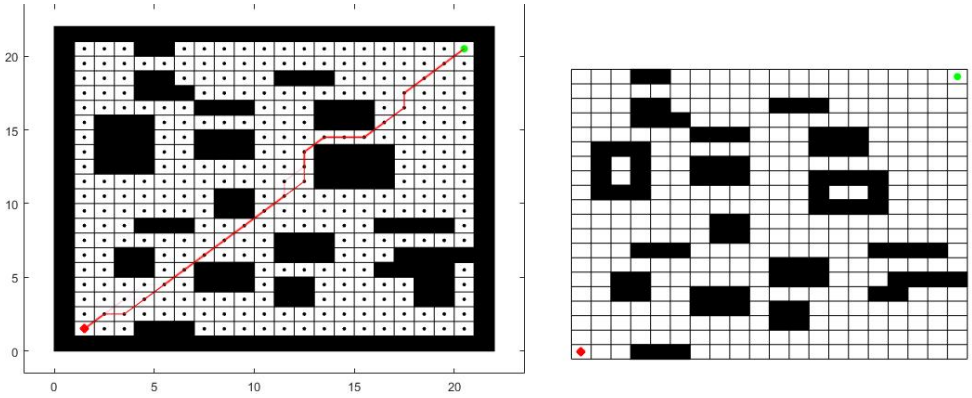
En el Cuadro 35 se muestran los resultados de las pruebas anteriores. Se puede notar que el tiempo requerido para este algoritmo es mucho menor que el requerido para el algoritmo de planificación de trayectorias y evasión de obstáculos. La corrida 2 tiene las mismas condiciones que las corridas con el mapa C con el algoritmo anterior, los resultados obtenidos fueron bastante similares en cuanto a costo e iteraciones. Tomando en cuenta que el tiempo ahora fue mucho menor y que la trayectoria fue bastante similar a las obtenidas anteriormente, se puede decir que este algoritmo presenta mejor rendimiento que el anterior.

Corrida	**Iteraciones**	**Tiempo (min)**	**Costo**
1	23	1.91	15.82
2	25	5.1	29.74
3	32	9.49	31.74

Cuadro 35: Resultados del mapa C.

9.2.4. Mapa D

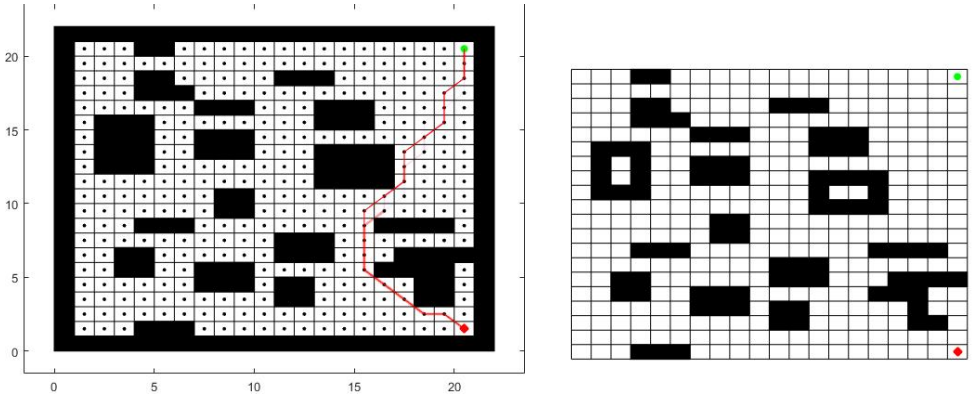
En la Figura 85 se muestran los resultados de realizar la exploración en el mapa D, tomando como nodo de inicio el 400 y como nodo de destino al 1. El mapa generado es bastante similar al original, con la diferencia que el obstáculo de la esquina inferior derecha no lo generó completo. Como en el mapa anterior, la diferencia se debe a que las hormigas no necesitan explorar el mapa completo para encontrar la trayectoria entre los nodos.



(a) Mapa D con trayectoria del nodo 400 al 1. (b) Mapa D generado.

Figura 85: Exploración del mapa D con trayectoria del nodo 400 al 1.

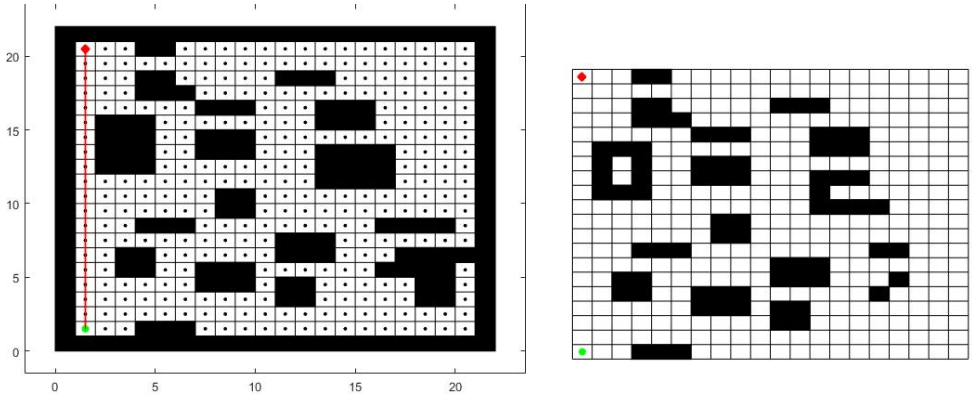
En la Figura 86 se muestran los resultados del algoritmo con el mapa D, al buscar una trayectoria desde el nodo 400 hasta al 20. En el mapa generado de la Figura 86b se observa que los obstáculos se replicaron casi igual que los del mapa original, que se muestra en la Figura 86a. Se observa que el obstáculo de la esquina inferior derecha no se completó. Esta y la siguiente prueba se realizaron con la finalidad de ver qué ocurre cuando los nodos inicial y objetivo se encuentran al costado del mapa.



(a) Mapa D con trayectoria del nodo 400 al 20. (b) Mapa D generado.

Figura 86: Exploración del mapa D con trayectoria del nodo 400 al 20.

En la Figura 87 se muestran los resultados de la trayectoria entre el nodo 381 y 1. En el mapa generado de la Figura 87b se observa que los obstáculos en lado derecho no fueron replicados completamente. Sin embargo, se obtuvo un mapa bastante similar al original.



(a) Mapa D con trayectoria del nodo 381 al 1. (b) Mapa D generado.

Figura 87: Exploración del mapa D con trayectoria del nodo 381 al 1.

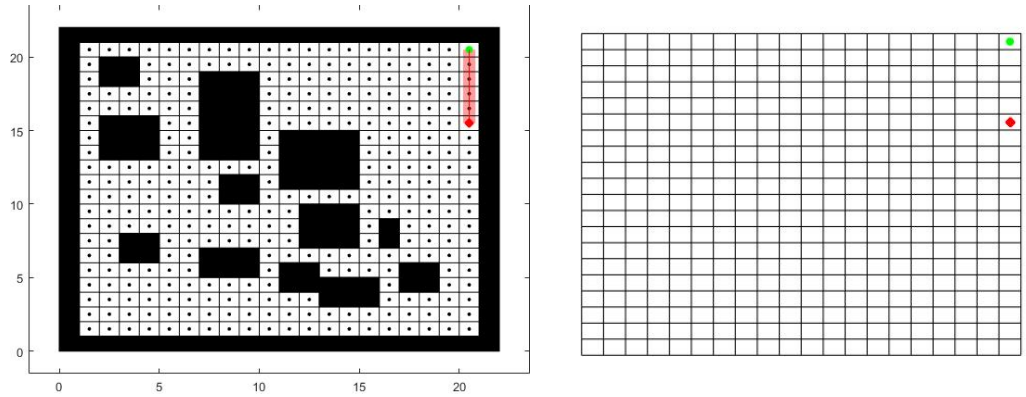
En el Cuadro 36 se muestran los resultados de las pruebas mencionadas anteriormente. El tiempo de ejecución es bastante bajo en comparación al tiempo de ejecución del algoritmo de planificación de trayectorias y evasión de obstáculos para este mapa.

Corrida	**Iteraciones**	**Tiempo (min)**	**Costo**
1	29	4.91	28.56
2	39	4.78	23.69
3	58	5.79	19

Cuadro 36: Resultados del mapa D.

9.2.5. Mapa E

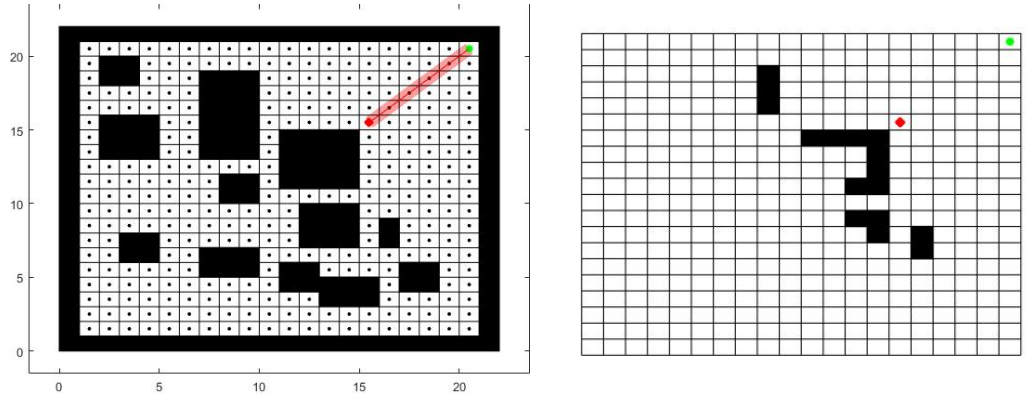
Las pruebas que se hicieron con este mapa fueron para ver qué pasaba si los nodos de inicio y destino se encontraban cerca. Para la primera prueba se tomó como inicio al nodo 400 y como final al nodo 300. En la Figura 88 se observa que el mapa generado no presenta obstáculos. En esto también influye que en el área que se colocaron los nodos de inicio y final casi no hay obstáculos alrededor.



(a) Mapa E con trayectoria del nodo 400 al 300. (b) Mapa E generado.

Figura 88: Exploración del mapa E con trayectoria del nodo 400 al 300.

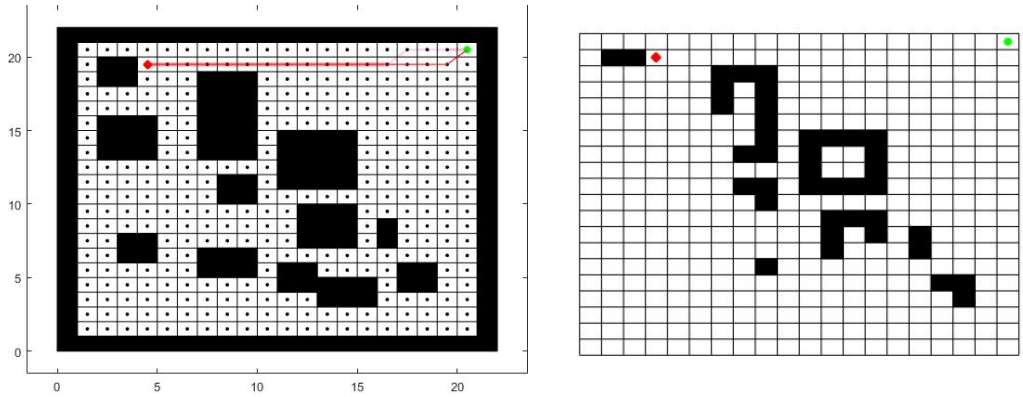
En la segunda prueba se tomó como inicio al nodo 400 y como final al nodo 295. En la Figura 89 se observa que el mapa explorado tiene más obstáculos que el de la Figura 88. La parte inferior izquierda del mapa generado está casi vacía, por lo que se puede notar que las hormigas no exploraron el mapa más de lo necesario.



(a) Mapa E con trayectoria del nodo 400 al 295. (b) Mapa E generado.

Figura 89: Exploración del mapa E con trayectoria del nodo 400 al 295.

La tercera prueba se realizó tomando el nodo 400 como inicio y al nodo 364 como final. En este caso los nodos están un poco más alejados que en los dos casos anteriores. Se observa en la Figura 90 que el mapa generado y el original no son muy parecidos y únicamente se replican los obstáculos que están cerca de los nodos de inicio y final.



(a) Mapa E con trayectoria del nodo 400 al 364.

(b) Mapa E generado.

Figura 90: Exploración del mapa E con trayectoria del nodo 400 al 364.

En el Cuadro 37 se muestran los resultados de las pruebas anteriores. Se observa que el tiempo de ejecución es muy pequeño cuando no hay obstáculos de por medio en la trayectoria.

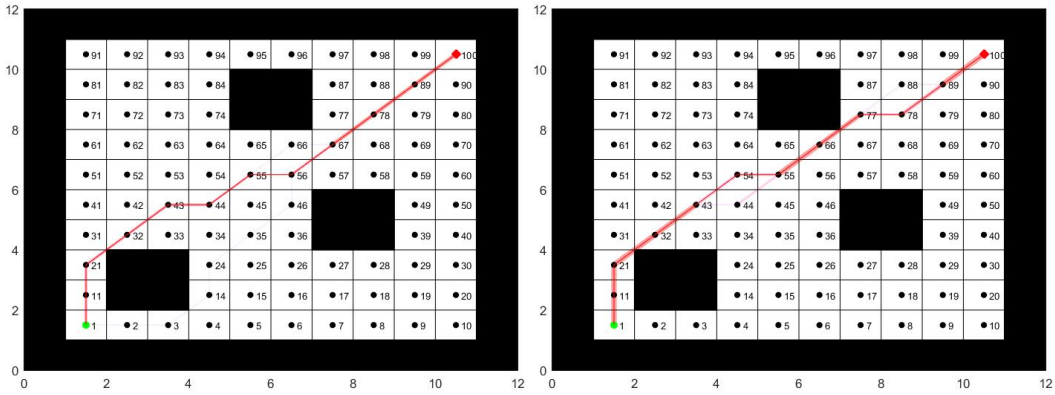
Corrida	**Iteraciones**	**Tiempo (min)**	**Costo**
1	7	0.27	5
2	8	0.35	7.05
3	24	2.04	16.41

Cuadro 37: Resultados del mapa E.

9.3. Comparación de los algoritmos

En la Figura 91a se muestra la trayectoria de la primera corrida del algoritmo de planificación de trayectorias y evasión de obstáculos. En la Figura 91b se muestra la trayectoria de la primera corrida del algoritmo de exploración de terrenos, con trayectoria desde el nodo 1 hasta el 100.

En el Cuadro 38 se observa el promedio de los resultados obtenidos para el mapa A, con el algoritmo de planificación de trayectorias y evasión de obstáculos. Asimismo, se muestran los resultados de la corrida del algoritmo de exploración de terrenos antes mencionada. El tiempo fue menor para el segundo algoritmo. Sin embargo, la cantidad de iteraciones fue menor para el primer algoritmo. Dado que el algoritmo de exploración de terrenos está paralelizado, se puede notar la ventaja que esto tiene en cuanto a tiempo de ejecución. El algoritmo paralelizado es casi 2 veces más rápido de ejecutar.



(a) Trayectoria del primer algoritmo. (b) Trayectoria del segundo algoritmo.

Figura 91: Comparación de trayectorias obtenidas por los algoritmos.

Algoritmo	**Iteraciones**	**Tiempo (min)**	**Costo**
Planificación de trayectorias y evasión de obstáculos	22.33	27.83	13.9
Exploración de terrenos	24.67	14.73	13.9

Cuadro 38: Comparación entre algoritmos.

CAPÍTULO 10

Algoritmo de registro de imágenes médicas

En este capítulo se muestran los resultados obtenidos de utilizar las funciones de registro de imágenes de Matlab. Asimismo, se explica la metodología empleada para la implementación del algoritmo de registro de imágenes médicas, basado en el *Ant Colony Optimization*. Asimismo, se mencionan los resultados obtenidos de esta implementación. Este algoritmo fue desarrollado en Matlab versión 2020a. Durante el desarrollo del algoritmo se trabajó con imágenes utilizadas comúnmente en procesamiento de imágenes. Luego, se realizaron pruebas con imágenes médicas.

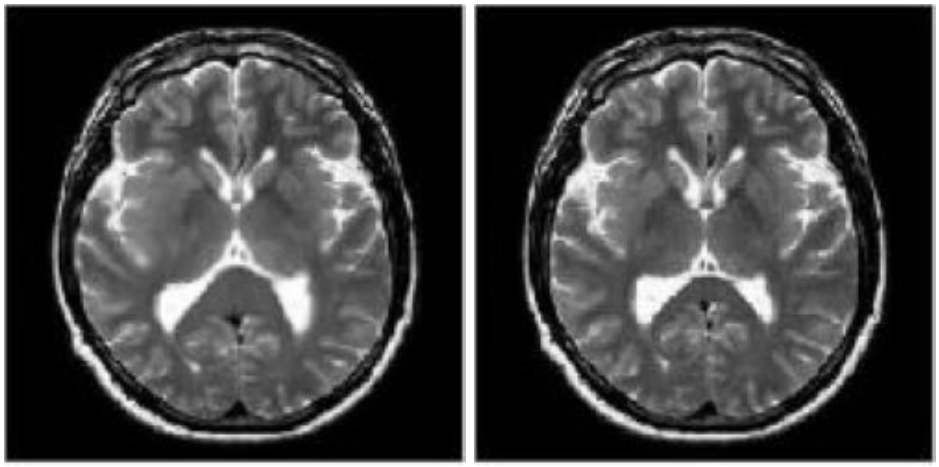
10.1. Registro de imágenes con Matlab

Se realizaron pruebas de registro de imágenes con funciones de Matlab [28] [29]. Esto para comparar con los resultados obtenidos con el algoritmo. En Matlab es necesario indicar el tipo de transformación para el registro, en todos los casos se utilizó la opción de transformación afín. Los resultados se muestran a continuación.

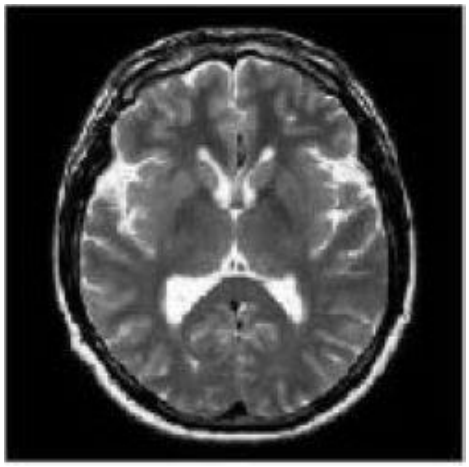
En las Figuras 92 a 96 se muestran las imágenes registradas. En el Cuadro 39 se observa que para los cerebros los resultados fueron muy buenos ya que el coeficiente de correlación fue de 1. Para el caso de la imagen de la rodilla y Lena los resultados no fueron tan buenos, esto se puede observar con el valor del coeficiente correlación final. En el caso de la imagen con forma de óvalo a forma de "C" los resultados no fueron buenos, el coeficiente de correlación disminuyó y la suma de diferencias cuadradas se mantuvo en valores muy grandes.

Prueba	*SSD*		*CC*	
	Inicial	*Final*	*Inicial*	*Final*
Figura 92	484.5881	0	0.9894	1
Figura 93	1.0117×10^3	0	0.9502	1
Figura 94	1.8483×10^3	813.2215	0.8877	0.9385
Figura 95	577.2524	103.3937	0.9378	0.9878
Figura 96	1.5270×10^4	7.3654×10^3	0.5632	0.2757

Cuadro 39: Análisis cuantitativo del registro de imágenes con funciones de Matlab.

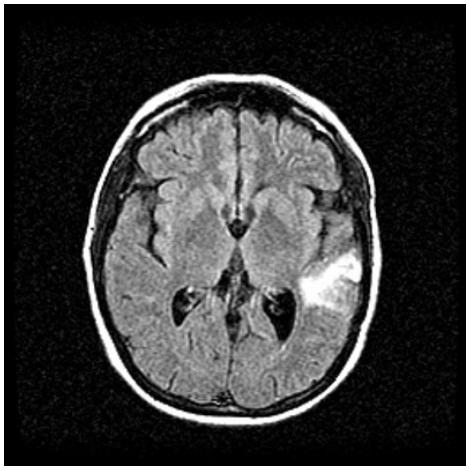


(a) Imagen plantilla. (b) Imagen de referencia.

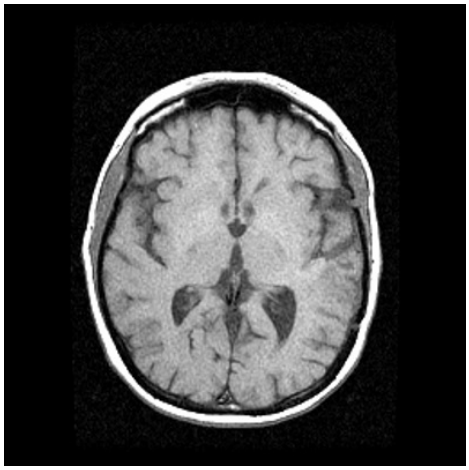


(c) Imagen registrada.

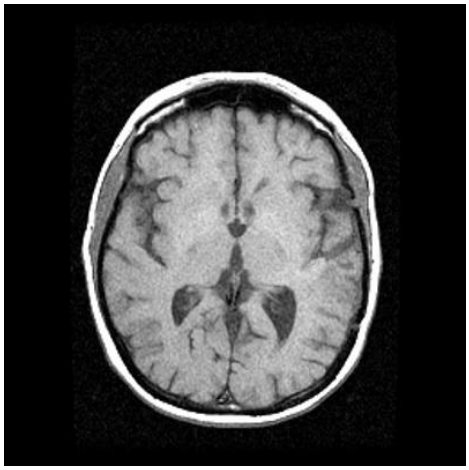
Figura 92: Registro de la imagen de un cerebro con funciones de Matlab.



(a) Imagen plantilla.

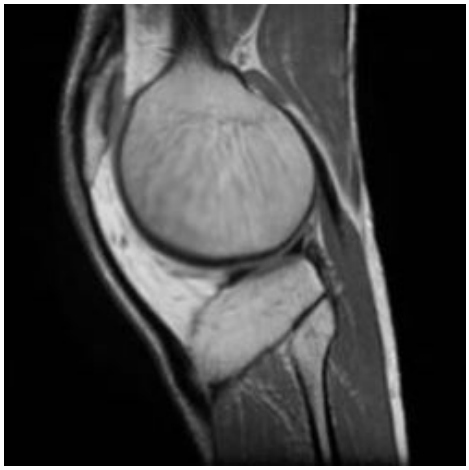


(b) Imagen de referencia.



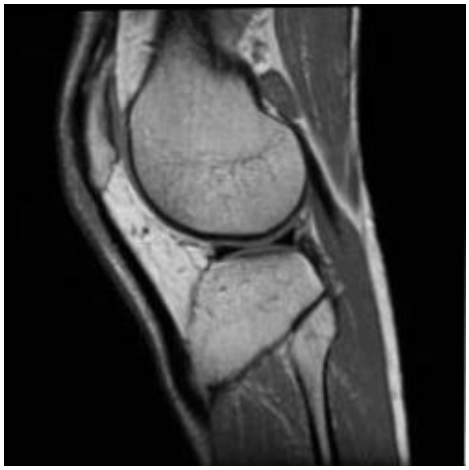
(c) Imagen registrada.

Figura 93: Registro de la imagen de un cerebro con funciones de Matlab.



(a) Imagen plantilla.

(b) Imagen de referencia.



(c) Imagen registrada.

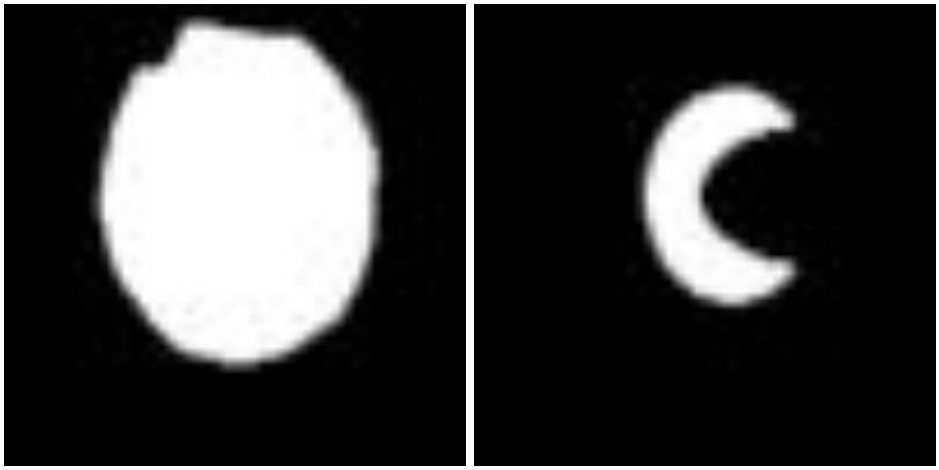
Figura 94: Registro de la imagen de una rodilla con funciones de Matlab.

(a) Imagen plantilla.

(b) Imagen de referencia.

(c) Imagen registrada.

Figura 95: Registro de la imagen de Lena.



(a) Imagen plantilla. (b) Imagen de referencia.

(c) Imagen registrada.

Figura 96: Registro de forma de óvalo a forma de “C” .

10.2. Metodología

Se tomó como base el algoritmo de la fase anterior [6] y se realizaron algunas modificaciones tomando como referencia al algoritmo que se presenta en [3]. En la Figura 97 se muestra el diagrama de flujo del algoritmo implementado. Este algoritmo trabaja con imágenes en blanco y negro, y las imágenes deben ser cuadradas. El algoritmo consta de 4 partes: proceso de desface, proceso de inicialización, proceso de búsqueda y proceso de actualización.

En esta implementación, el orden en que se mueven las hormigas es en forma de onda cuadrada. Por ejemplo, para una imagen de 10×10 píxeles después de que se mueve la hormiga 10, se mueve la hormiga 20. Luego, después de moverse la hormiga 11, se mueve la hormiga 21. Esto se hizo para evitar estancamientos en el movimiento del alimento. Un ejemplo del orden en que las hormigas se mueven se muestra en la Figura 98. En la siguiente iteración el orden se invierte.

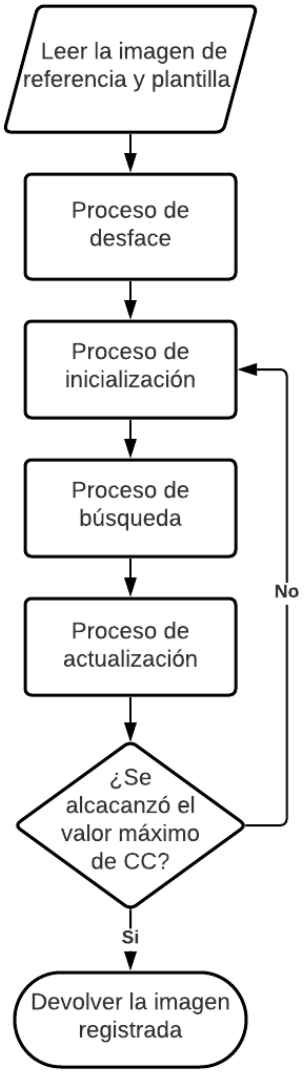


Figura 97: Diagrama de flujo del algoritmo de registro de imágenes.

10.2.1. Proceso de desface

La imagen de referencia y la plantilla pueden tener intensidades totales de imagen diferentes. La primera transformación que realiza el algoritmo es igualar la intensidad total de la imagen plantilla a la de la referencia. Para esto, se calcula el desface de intensidad total que existe entre las imágenes. Luego, el desface se reparte en partes iguales entre todos los píxeles de la imagen. En caso de haber un residuo en el valor del desface que no se pueda repartir, este se asigna aleatoriamente a un píxel de la imagen. Si el desface es positivo se suma, si es negativo se resta a la imagen plantilla para formar la primera versión de la imagen registrada.

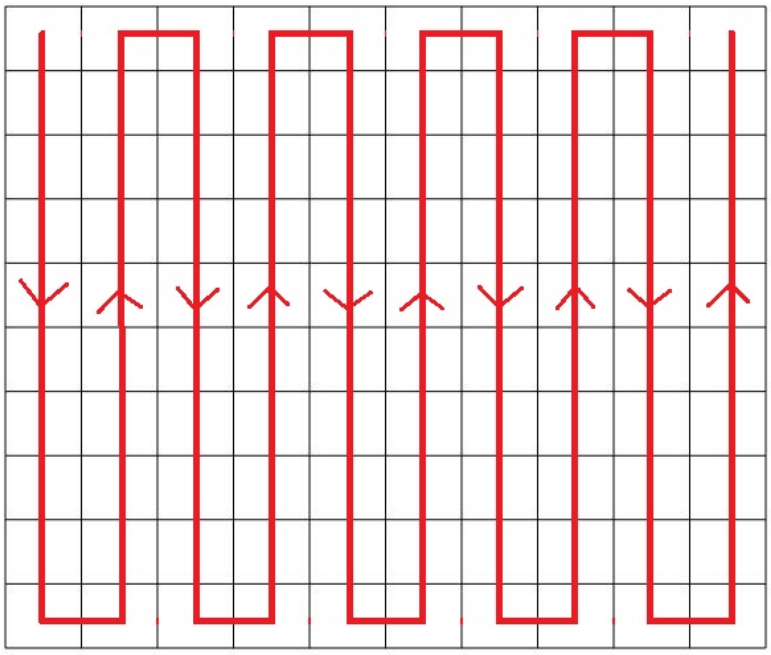


Figura 98: Orden de movimiento de las hormigas.

10.2.2. Proceso de inicialización

La imagen ingresada es de dimensión $N \times N$. Por lo tanto, se asignan $N \times N$ hormigas de manera uniforme a cada píxel de la imagen. Cada píxel representa un nodo que tiene inicialmente un valor τ_0 de feromona. La comida está representada por la diferencia de intensidades entre la imagen de referencia y la imagen plantilla [3]. La diferencia de intensidades se calcula usando la ecuación (20).

En los nodos donde $I_{diff}\,(l,m) < 0$, las hormigas extraen alimento para llevarlo a los nodos donde $I_{diff}\,(l,m) > 0$.

10.2.3. Proceso de búsqueda

El siguiente paso es escoger hacia qué nodo se mueve cada hormiga. En este enfoque, en la enésima iteración, una hormiga se mueve desde el nodo (l,m) hasta el nodo vecino (i,j) y regresa al nodo (l,m) [3] con el alimento que encontró. El movimiento de las hormigas se calcula mediante la ecuación (21).

10.2.4. Proceso de actualización

En este proceso se forma la matriz de alimento, cuyo valor correspondiente al nodo (l,m) es igual $I_{\text{diff}}\ \ (i,j)$. Si el valor de la matriz de alimento en el nodo $(l,m) < 0$, se mueve alimento del nodo (i,j) al nodo (l,m). Este último paso corresponde a la actualización de

la imagen registrada. Para obtener la matriz que contiene las transformaciones que llevan de la imagen plantilla a la registrada, se suma o resta el valor de la comida que se pone o quita en los nodos, según corresponda. Luego, se calcula de nuevo $I_{\text{diff}}^{(n)}$ y con los valores obtenidos, se actualiza la intensidad (η) de los nodos (l, m) e (i, j).

Por último, se actualiza la feromona presente en cada nodo. Esta se calcula mediante la siguiente ecuación

$$\tau^{(n+1)} = e^{-\rho} \times \tau^{(n)} + \tau_0 \times \left| I_{\text{diff}}^{(n)} \right| \times H\left(-I_{\text{diff}}^{(n)}\right) \tag{39}$$

Donde la constante ρ tiene valores entre 0 y 1, y representa el coeficiente de decaimiento de feromona. La ecuación (39) es una modificación de la ecuación (23). El término $\left|I_{\text{diff}}^{(n)}\right|$ se agregó para que las hormigas prefieran ir a los nodos donde la diferencia de intensidad no es cero. El término $H\left(-I_{\text{diff}}^{(n)}\right)$ corresponde a la función $Heaviside$ evaluada en $-I_{\text{diff}}^{(n)}$. Este último término se agregó para que las hormigas prefieran ir a los nodos con diferencia de intensidad negativa, es decir, donde hay comida.

10.2.5. Métricas de desempeño

Para evaluar el desempeño se experimentó primero con imágenes que son utilizadas comúnmente en procesamiento de imágenes y luego se hicieron pruebas con imágenes médicas. Para analizar cuantitativamente el algoritmo, se calculó la suma de diferencias cuadradas (SSD), con la ecuación (24), y el coeficiente de correlación (CC) [15] entre la imagen de referencia y la registrada, con la siguiente ecuación

$$CC = \frac{\sum_x \sum_y [(A(x,y) - \bar{A})(B(x,y) - \bar{B})]}{\sqrt{\sum_x \sum_y (A(x,y) - \bar{A})^2 * \sum_x \sum_y (B(x,y) - \bar{B})^2}} \tag{40}$$

Donde $A\,(x, y)$ y $B\,(x, y)$ representan la intensidad de las imágenes A y B en en (x, y), representa la norma euclidiana, N representa el total de pixeles, $\bar{A}\,(x, y)$ y $\bar{B}\,(x, y)$ representa el promedio de intensidades en las imágenes A y B, respectivamente.

10.3. Resultados

10.3.1. Registro de imágenes

Para validar el funcionamiento del algoritmo se hicieron pruebas con imágenes. A continuación se muestran los resultados obtenidos. En las figuras, la primera Figura corresponde a la imagen plantilla, la segunda figura a la imagen de referencia, la tercera figura corresponde a la imagen registrada y la cuarta figura corresponde a la diferencia de intensidad inicial entre imagen de referencia y la imagen plantilla.

En la Figura 99 se muestra el registro de la imagen de Lena [30]. La imagen plantilla se obtuvo como producto de distorsionar la imagen de referencia con una herramienta de distorsión [31]. Se observa que la imagen registrada es bastante parecida a la de referencia. Lo que indica que se lograron hacer las transformaciones necesarias a la imagen plantilla para llegar a un resultado muy parecido a la referencia. En el Cuadro 40 se observa que el coeficiente de correlación final para esta imagen es de 1 y que la suma de diferencias cuadradas es bastante cercana a 0. Esto indica que se registró la imagen de Lena de forma satisfactoria.

En la Figura 100 se observa el registro de una forma de óvalo a una forma de "C", cuyas imágenes plantilla y referencia se obtuvieron de [3]. En la imagen de la tercera columna se observa que el resultado es bastante similar a la imagen de referencia. Para este caso se tuvo un coeficiente de correlación final de 1 y la suma de diferencias cuadradas final fue de 0. Por lo tanto, se registró la imagen de forma satisfactoria. Los resultados obtenidos con el algoritmo fueron mejores que los obtenidos con las funciones de Matlab, en este caso.

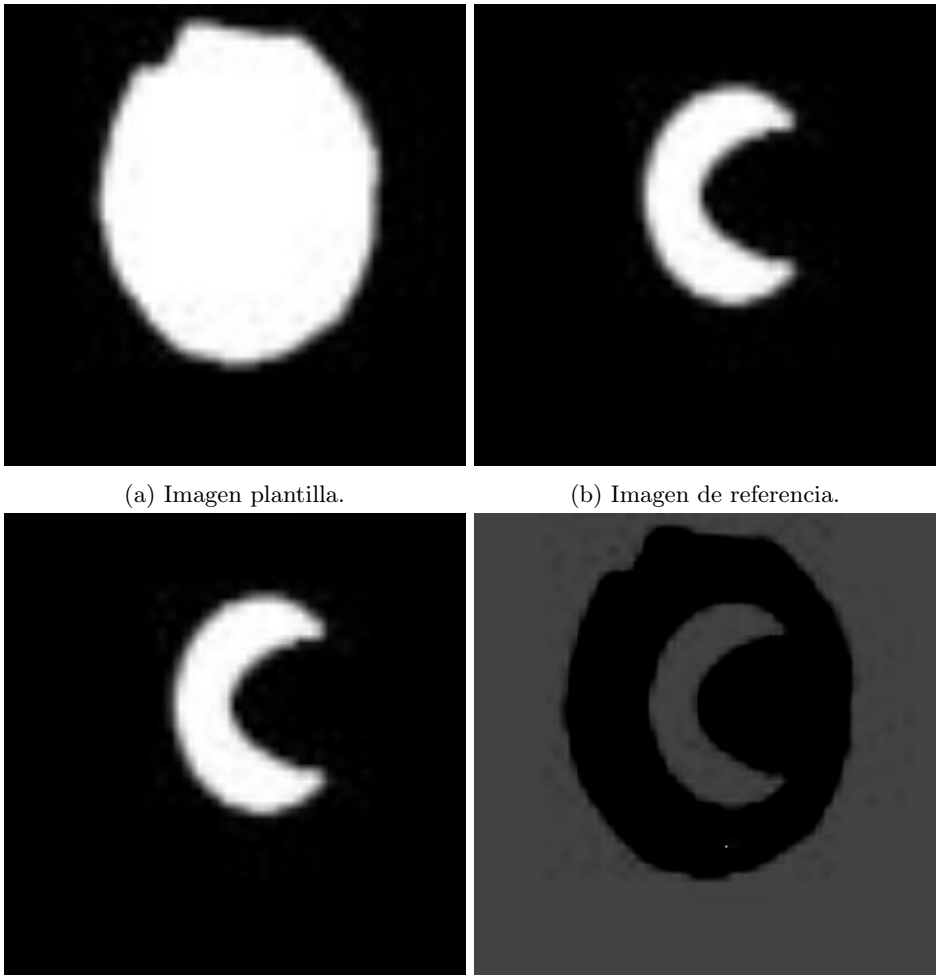
Prueba	*SSD*		*CC*	
	Inicial	*Final*	*Inicial*	*Final*
Figura 99	577.2524	0.0032	0.9378	1
Figura 100	1.5270×10^4	0	0.5632	1

Cuadro 40: Análisis cuantitativo de las pruebas del algoritmo con imágenes.

(a) Imagen plantilla. (b) Imagen de referencia.

(c) Imagen registrada. (d) Diferencia de intensidad inicial.

Figura 99: Registro de la imagen de Lena.



(a) Imagen plantilla. (b) Imagen de referencia.

(c) Imagen registrada. (d) Diferencia de intensidad inicial.

Figura 100: Registro de forma de óvalo a forma de “C” .

10.3.2. Registro de imágenes médicas

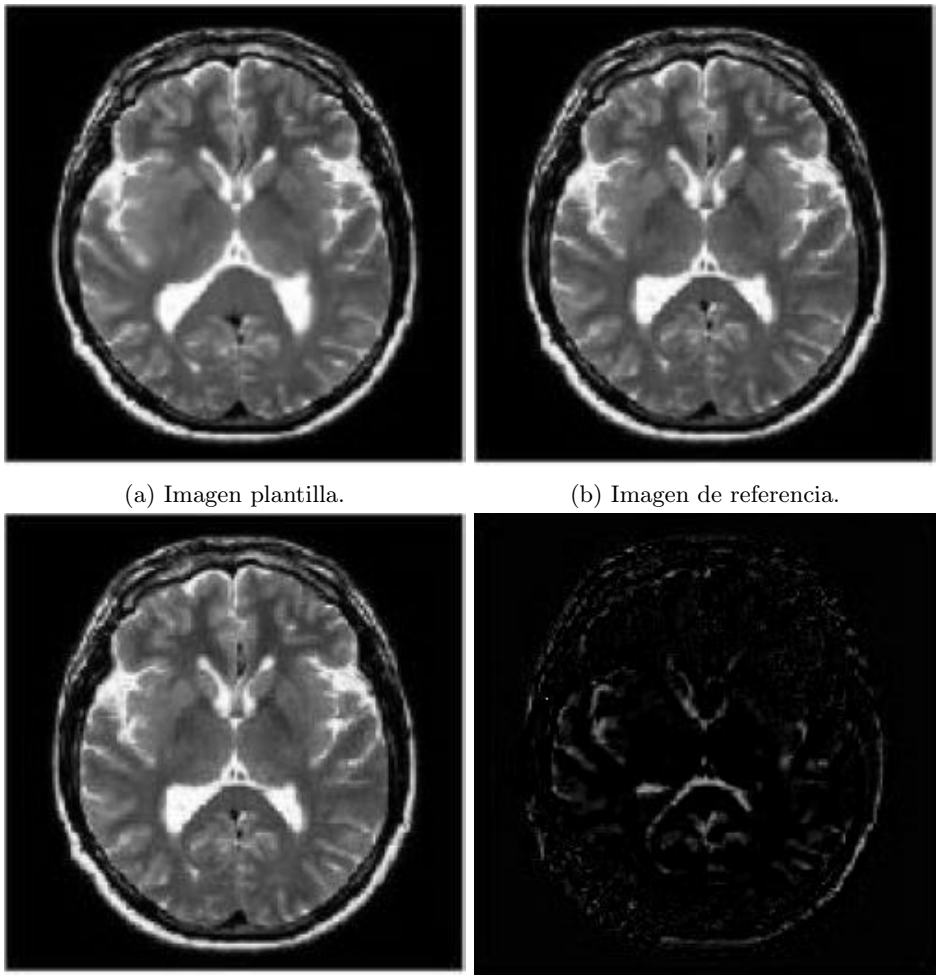
Se utilizaron imágenes médicas para validar el algoritmo. A continuación se muestran los resultados obtenidos de realizar pruebas con imágenes médicas. En las figuras, la primera Figura corresponde a la imagen plantilla, la segunda Figura a la imagen de referencia, la tercera Figura corresponde a la imagen registrada y la cuarta Figura corresponde a la diferencia de intensidad inicial entre imagen de referencia y la imagen plantilla.

En la Figura 101 se muestra el resultado del registro de un cerebro, las imágenes de plantilla y referencia se obtuvieron de [32]. Se observa que se logró transformar la imagen plantilla para que fuera lo más parecida posible a la referencia. En la Cuadro 41 se observa que el coeficiente de correlación para este caso fue de 1 y la suma de diferencias cuadradas fue de 0.

En la Figura 102 se observa el registro de la imagen una resonancia magnética de un cerebro. Las imágenes plantilla y referencia se obtuvieron de [33]. Se observa que la imagen

registrada es bastante similar a la imagen de referencia. El coeficiente de correlación obtenido fue de 1 y la suma de diferencias cuadradas de 0.008, lo cual es un valor cercano a 0. De este modo, se obtuvo la transformación necesaria para registrar la imagen de forma satisfactoria.

En la Figura 103 se muestra el resultado de registrar la imagen de una rodilla. Las imágenes necesarias para el registro se obtuvieron de [3]. Se observa que la imagen registrada es bastante parecida a la referencia, lo cual indica que logró aplicar las transformaciones necesarias a la imagen plantilla para hacer el registro. El coeficiente de correlación para este caso fue de 1 y la suma de diferencias cuadradas tuvo un valor final de 0.



(a) Imagen plantilla.

(b) Imagen de referencia.

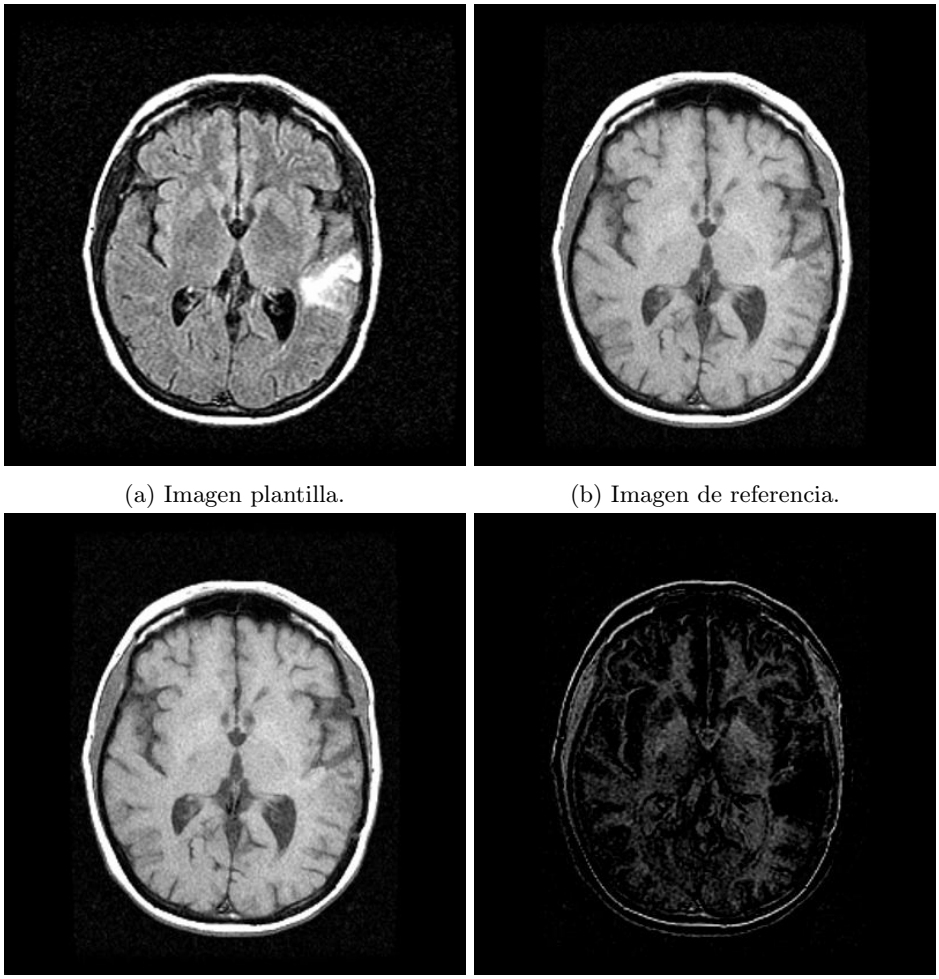
(c) Imagen registrada.

(d) Diferencia de intensidad inicial.

Figura 101: Registro de la imagen de un cerebro.

Prueba	*SSD*		*CC*	
	Inicial	*Final*	*Inicial*	*Final*
Figura 101	484.5881	0	0.9894	1
Figura 102	1.0117×10^3	0	0.9502	1
Figura 103	1.8483×10^3	0	0.8877	1

Cuadro 41: Análisis cuantitativo de las pruebas del algoritmo con imágenes médicas.



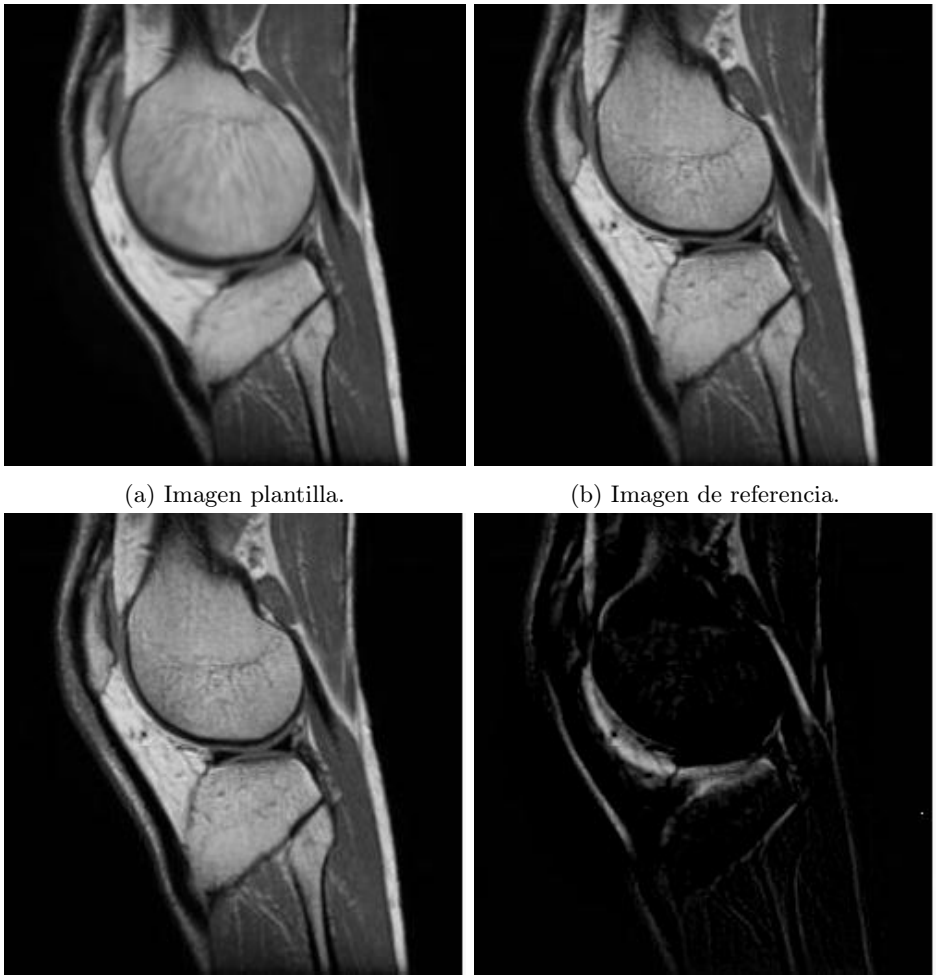
(a) Imagen plantilla.

(b) Imagen de referencia.

(c) Imagen registrada.

(d) Diferencia de intensidad inicial.

Figura 102: Registro de la imagen de un cerebro.



(a) Imagen plantilla.

(b) Imagen de referencia.

(c) Imagen registrada.

(d) Diferencia de intensidad inicial.

Figura 103: Registro de la imagen de una rodilla.

CAPÍTULO 11

Conclusiones

- Es posible modificar el algoritmo *Ant Colony Optimization* para que en la trayectoria encontrada evada obstáculos y pasillos estrechos, donde físicamente no podría pasar un robot.
- Se logró replicar mapas explorados con obstáculos al modificar el ACO. La exactitud de la réplica de los mapas explorados depende de que tanto atraviesa la trayectoria entre los nodos al mapa.
- Al utilizar algoritmos paralelizados se disminuye considerablemente el tiempo de ejecución.
- El algoritmo de exploración de terrenos presenta mejor rendimiento, en cuanto a tiempo, que el algoritmo de planificación de trayectorias y evasión de obstáculos, a pesar que las trayectorias encontradas son bastante similares.
- Se logró replicar en Webots, mediante simulaciones que involucran restricciones físicas, las trayectorias obtenidas con el algoritmo de planificación de trayectorias y evasión de obstáculos.
- Es posible modificar el algoritmo *Ant Colony Optimization* para realizar registro de imágenes médicas.
- El algoritmo de registro de imágenes médicas logró realizar las transformaciones necesarias a las imágenes plantilla, para obtener imágenes bastante similares a las imágenes de referencia. De este modo, se registraron las imágenes de forma satisfactoria.

CAPÍTULO 12

Recomendaciones

- Implementar el algoritmo de planificación de trayectorias y evasión de obstáculos en un lenguaje/plataforma que lo haga más eficiente en cuanto a tiempo de ejecución.
- Realizar el barrido de parámetros con paso más fino y con más iteraciones para el algoritmo de planificación y evasión de obstáculos. De esta manera se pueden generar resultados más óptimos.
- Implementar otros algoritmos para aplicaciones biomédicas y/o procesamiento de imágenes utilizando como base algoritmos de inteligencia de enjambre. Por ejemplo, detección de bordes en imágenes médicas.
- Validar de forma física las trayectorias generadas por los algoritmos de planificación de trayectorias y evasión de obstáculos. Utilizar sensores en la implementación para evitar totalmente la colisión con obstáculos.
- En el algoritmo de exploración de terrenos colocar a las hormigas de forma aleatoria en el mapa o colocar múltiples nodos de destino. De esta forma, explorar terrenos completos.

CAPÍTULO 13

Bibliografía

[1] *Robotarium | Institute for Robotics and Intelligent Machines*. dirección: `http://www.robotics.gatech.edu/robotarium` (visitado 29-03-2021).

[2] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, en. Springer Science & Business Media, nov. de 2011, ISBN: 978-3-642-20143-1.

[3] T. X. Lin y H. H. Chang, «Medical Image Registration Based on an Improved Ant Colony Optimization Algorithm,» en, vol. 5, n.º 1, pág. 6, 2016.

[4] H.-J. Wang, Y. Fu, Z.-Q. Zhao e Y.-J. Yue, «An Improved Ant Colony Algorithm of Robot Path Planning for Obstacle Avoidance,» *Journal of Robotics*, vol. 2019, K. Watanabe, ed., pág. 6 097 591, jun. de 2019, Publisher: Hindawi, ISSN: 1687-9600. DOI: `10.1155/2019/6097591`.

[5] E.-H. Guerrout, R. Mahiou y S. Ait-Aoudia, «Hidden Markov Random Fields and Particle Swarm Combination for Brain Image Segmentation,» en, vol. 15, n.º 3, pág. 7, 2018.

[6] G. I. Colmenares, «Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica,» Tesis de licenciatura, Universidad del Valle de Guatemala, 2020.

[7] A. S. A. Nadalini, «Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO),» Tesis de licenciatura, Universidad del Valle de Guatemala, 2019.

[8] B. S. G. d. Almeida y V. C. Leite, «Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems,» en, *Swarm Intelligence - Recent Advances, New Perspectives and Applications*, dic. de 2019, Publisher: IntechOpen. DOI: `10.5772/intechopen.89633`.

[9] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd. Wiley Publishing, 2007, ISBN: 0470035617.

[10] M. N. Ab Wahab, S. Nefti-Meziani y A. Atyabi, «A Comprehensive Review of Swarm Optimization Algorithms,» *PLOS ONE*, vol. 10, n.º 5, págs. 1-36, mayo de 2015. DOI: `10.1371/journal.pone.0122827`.

[11] C. A. R. Algarín, «Optimización por colonia de hormigas:» ES, *Ingeniería Solidaria*, vol. 6, n.º 10-11, págs. 83-89, 2010, Number: 10-11, ISSN: 2357-6014.

[12] *Image Registration - an overview | ScienceDirect Topics*. dirección: https://www.sciencedirect.com/topics/neuroscience/image-registration (visitado 03-05-2021).

[13] D. L. G. Hill, P. G. Batchelor, M. Holden y D. J. Hawkes, «Medical image registration,» *Physics in Medicine and Biology*, vol. 46, n.º 3, R1-R45, 2001. DOI: 10.1088/0031-9155/46/3/201.

[14] W. Peng, R. Tong, G. Qian y J. Dong, «A constrained ant colony algorithm for image registration,» vol. 4115, págs. 1-11, 2006. DOI: 10.1007/11816102_1.

[15] S. I. Inc., *Coeficiente de correlación*. dirección: https://www.jmp.com/es_cl/statistics-knowledge-portal/what-is-correlation/correlation-coefficient.html (visitado 16-11-2021).

[16] *Squared Difference - an overview | ScienceDirect Topics*. dirección: https://www.sciencedirect.com/topics/engineering/squared-difference (visitado 16-11-2021).

[17] S. Ourselin, R. Stefanescu y X. Pennec, «Robust Registration of Multi-modal Images: Towards Real-Time Clinical Applications,» en, vol. 2489, G. Goos, J. Hartmanis, J. van Leeuwen, T. Dohi y R. Kikinis, eds., págs. 140-147, 2002. DOI: 10.1007/3-540-45787-9_18.

[18] L. Di Stefano y S. Mattoccia, «Fast template matching using bounded partial correlation,» *Mach. Vis. Appl.*, vol. 13, págs. 213-221, feb. de 2003. DOI: 10.1007/s00138-002-0070-5.

[19] Miguel Zea, *MT3005 Lección 12 - modelado de robots móviles*, mar. de 2020. dirección: https://www.youtube.com/watch?v=5AqXC8ivZ8U (visitado 15-09-2021).

[20] R. Siegwart, I. R. Nourbakhsh y D. Scaramuzza, *Introduction to autonomous mobile robots*, en, 2nd ed. Cambridge, Mass: MIT Press, 2011, ISBN: 978-0-262-01535-6.

[21] B. Siciliano, L. Sciavicco, L. Villani y G. Oriolo, *Robotics*, en, M. J. Grimble y M. A. Johnson, eds., ép. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2009, ISBN: 978-1-84628-641-4. DOI: 10.1007/978-1-84628-642-1.

[22] Mezeaa, *MT3005 Lección 13 - control de robots móviles*, 2020. dirección: https://www.youtube.com/watch?v=bCCIEPiNPgM (visitado 17-09-2021).

[23] C. D. G. Uribe, *Comparación de un controlador LQR vs un controlador PID implementados en un helicoptero de dos grados de libertad Pivotado*. es, 2016. dirección: https://repository.udistrital.edu.co/bitstream/handle/11349/4020/GonzalezUribeChristianDavid2016.pdf?sequence=1&isAllowed=y (visitado 17-09-2021).

[24] O. F. Kececioglu, A. Gani, H. Açıkgöz y M. Sekkeli, *Dynamic Performance Comparison of LQR and LQI Controllers on Buck Converter*, nov. de 2018.

[25] *Using parfor Loops: Getting Up and Running*, en. dirección: https://blogs.mathworks.com/loren/2009/10/02/using-parfor-loops-getting-up-and-running/ (visitado 17-09-2021).

[26] *Asus X556UQ, 15.6, Core i7, 8GB, 1TB, W10*. dirección: https://intercompras.com/p/laptop-asus-x556uq-intel-core-i7-7500u-8gb-1tb-gtx-2gb-windows-home-124960 (visitado 16-10-2021).

[27] *Especificaciones de productos*, es. dirección: https://ark.intel.com/content/www/es/es/ark/products/120491/intel-xeon-gold-6152-processor-30-25m-cache-2-10-ghz.html (visitado 15-09-2021).

[28] *Configuraciones para el registro basado en la intensidad - MATLAB imregconfig - MathWorks España*. dirección: https://es.mathworks.com/help/images/ref/imregconfig.html (visitado 06-12-2021).

[29] *Registro de imágenes basado en la intensidad - MATLAB imregister - MathWorks España*. dirección: https://es.mathworks.com/help/images/ref/imregister.html (visitado 06-12-2021).

[30] E. E. País. (22 de oct. de 2019). «Lena Söderberg: el póster central que ayudó a crear el jpg,» Verne. Publisher: Ediciones El País, dirección: https://verne.elpais.com/verne/2019/10/21/articulo/1571661515_258318.html (visitado 04-12-2021).

[31] *Processing result - IMG online*. dirección: https://www.imgonline.com.ua/eng/picture-distortion-result.php (visitado 30-08-2021).

[32] L. Tsai y H.-H. Chang, «An incompressible fluid flow model with mutual information for MR image registration,» en *Image Processing: Machine Vision Applications VI*, P. R. Bingham y E. Y. Lam, eds., International Society for Optics y Photonics, vol. 8661, SPIE, 2013, págs. 254-261. DOI: 10.1117/12.2005441. dirección: https://doi.org/10.1117/12.2005441.

[33] F. P. Margall, *Resonancia Magnética cerebral como herramienta de investigación - Biotech Spain*. dirección: http://biotech-spain.com/es/articles/resonancia-magntica-cerebral-como-herramienta-de-investigaci-n/ (visitado 24-11-2021).

CAPÍTULO 14

Anexos

14.1. Repositorio en Github

Para obtener información sobre el código desarrollado puede ingresar al siguiente link: `https://github.com/larivera-UVG/Inteligencia-Computacional-y-Robotica-Swarm-2021/tree/main/Daniela%20Baldiz%C3%B3n`