INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

LABORATORIO DE RECONOCIMIENTO DE PATRONES

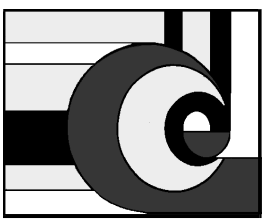
ESTUDIO COMPARATIVO DE DIFERENTES TÉCNICAS BIO – INSPIRADAS PARA ENCONTRAR EL CAMINO MÁS CORTO DE UN ROBOT MÓVIL DENTRO DE UN LABERINTO

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A LA

ING. BEATRIZ AURORA GARRO LICÓN



DIRECTOR DE TESIS: **DR. JUAN HUMBERTO SOSSA AZUELA**

MÉXICO, D. F. JUNIO 2007

A mis Padres

Martha Patricia
Rodolfo
A quienes dedico esta parte de mis sueños

A mis Abuelos

Manuel
Hermila
Rodolfo
María de la Paz

De quienes aprendí cosas maravillosas de la vida, siempre estarán en mi recuerdo y mi corazón

AGRADECIMIENTOS

El Instituto Politécnico Nacional, mi segunda Patria, ha sido mi orgullo y emblema desde hace 10 años. Más que una Institución, es el corazón de mi vida en la que me formé con ética en el ámbito cultural, deportivo y académico-científico. El IPN ha sido testigo y parte de innumerables escenarios durante mi estancia en el mismo, los cuales serán inolvidables. En sus cimientos, mis fracasos se volvieron el coraje para emprender un nuevo camino hacia el éxito, formándome como un ser íntegro. Por ello, en lo que reste de mi existencia seguiré honrando con placer y honor los colores guinda y blanco, así como su escudo y su noble mascota; cantando con amor y decoro su himno, y siempre llevaré muy en alto las siglas IPN, ya que soy politécnico por convicción y no por circunstancia.

Agradezco también al Centro de Investigación en Computación por brindarme la oportunidad de desenvolverme como investigadora, contribuyendo con su apoyo económico la SIP y el CONACYT con el proyecto cuyo registro es el 46805.

A los profesores, investigadores y amigos que dejaron en mí durante estos años enseñanzas, sus consejos y su apoyo incondicional en mi formación académica y como ser humano, muchas gracias.

Agradezco al Dr. Juan Humberto Sossa, el que me haya abierto las puertas para seguir adelante con ésta investigación. Gracias a su apoyo, confianza, enseñanzas e interés no perdí el entusiasmo de seguir adelante ya que me mostró que es un gran investigador y un gran ser humano. Al Dr. Marco Antonio Moreno Armendáriz y a los investigadores que integraron la comisión evaluadora, por sus consejos e interés mostrados en la elaboración de esta investigación .

Finalmente quiero agradecer a los pilares de mi vida: a Dios por darme vida y permitirme disfrutar de cada instante. A mi Familia: mis tíos y tías por preocuparse por mí y darme palabras de aliento. A mis padres, los cuales son una bendición de Dios, gracias por su apoyo, por sus consejos, por compartir mis tristezas y alegrías, por todo su amor, por sus preocupaciones al querer que sea mejor día con día; gracias por darme sus vidas en cada paso, en cada pensamiento, en cada acción. Sin ustedes yo no sería lo que soy ahora. Gracias por enseñarme a seguir mis convicciones; a ustedes papá y mamá, les debo la vida y el que sea un ser humano feliz e íntegro, los amo. Gracias a mis hermanas Alejandra y Mariana, quienes me han brindado su apoyo, sus consejos, su cariño y por existir en mi vida, saben que las amo. Y a Roberto Antonio quien me ha dado la oportunidad de crecer académicamente y como persona; te agradezco la dedicación que siempre me brindas, el apoyo, tus consejos, el amor que me has demostrado día con día, por creer en mí; gracias por enseñarme a no rendirme nunca y motivarme a realizar mis sueños. Gracias por ser parte de mi vida, te amo.

Beatriz Aurora Garro Licón

Me encanta Dios. Es un viejo magnífico que no se toma en serio. A él le gusta jugar y juega, y a veces se le pasa la mano y nos rompe una pierna o nos aplasta definitivamente. Pero esto sucede porque es un poco cegatón y bastante torpe con las manos. Nos ha enviado a algunos tipos excepcionales como Buda, o Cristo, o Mahoma, o mi tía Chofi, para que nos digan que nos portemos bien. Pero esto a él no le preocupa mucho: nos conoce. Sabe que el pez grande se traga al chico, que la lagartija grande se traga a la pequeña, que el hombre se traga al hombre. Y por eso inventó la muerte: para que la vida - no tú ni yo - la vida, sea para siempre. Ahora los científicos salen con su teoría del Big Bang... Pero ¿que importa si el universo se expande interminablemente o se contrae? Esto es asunto sólo para agencias de viajes.

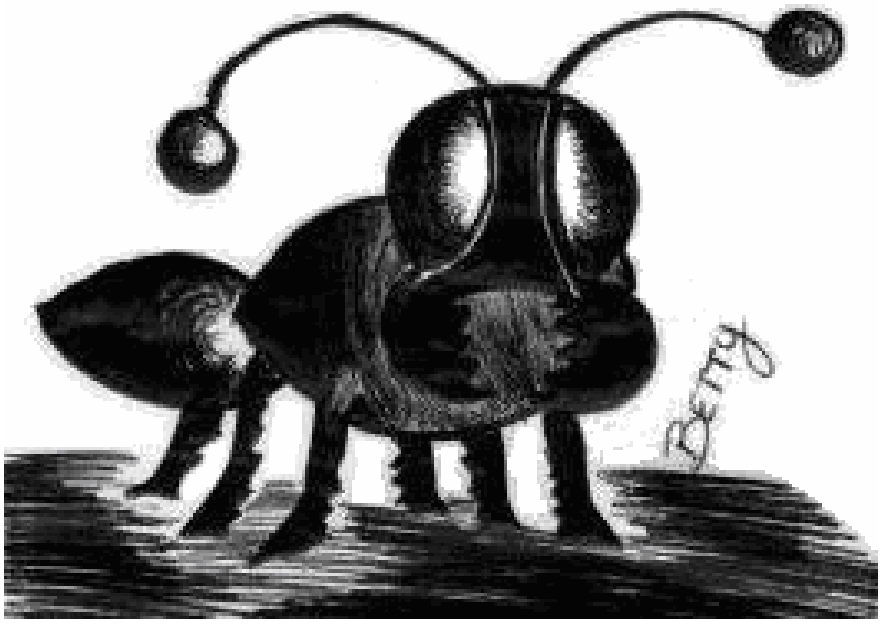
A mi me encanta Dios. Ha puesto orden en las galaxias y distribuye bien el tránsito en el camino de las hormigas. Y es tan juguetón y travieso que el otro día descubrí que ha hecho frente al ataque de los antibióticos con ¡bacterias mutantes! Viejo sabio o niño explorador, cuando deja de jugar con sus soldaditos de plomo de carne y hueso, hace campos de flores o pinta el cielo de manera increíble. Mueve una mano y hace el mar, y mueve la otra y hace el bosque. Y cuando pasa por encima de nosotros, quedan las nubes, pedazos de su aliento. Dicen que a veces se enfurece y hace terremotos, y manda tormentas, caudales de fuego, vientos desatados, aguas alevosas, castigos y desastres. Pero esto es mentira. Es la tierra que cambia- y se agita y crece- cuando Dios se aleja. Dios siempre está de buen humor. Por eso es el preferido de mis padres, el escogido de mis hijos, el más cercano de mis hermanos, la mujer mas amada, el perrito y la pulga, la piedra mas antigua, el pétalo mas tierno, el aroma más dulce, la noche insondable, el borboteo de luz, el manantial que soy. A mi me gusta, a mi me encanta Dios. Que Dios bendiga a Dios.

Jaime Sabines
1926 - 1999

“La vida artificial busca simular o crear nuevas formas de vida no basadas en carbono si no en estructuras de información”

“Se pretende estudiar la vida cómo podría ser y no como es en realidad”

Chris Langton
Padre de la Vida Artificial



GLOSARIO

Alelo. Cada uno de los valores que puede adquirir un gen.

Algoritmo de Dijkstra. También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un nodo origen al resto de nodos en un grafo dirigido y con pesos en cada arista.

Algoritmo de Hormigas. Algoritmo utilizado para la optimización de problemas, El cual se basa en el comportamiento colectivo de las hormigas en la búsqueda de alimentos para su subsistencia.

Algoritmos bio-inspirados. Modelan de forma aproximada un fenómeno existente de la naturaleza.

Algoritmos genéticos. Sistema adaptativo, el cual se basa en la evolución y selección natural.

Aptitud. Valor que indica que tan bueno es un individuo con respecto a otro.

Cromosoma. Estructura de datos que contiene una cadena de parámetros o genes, emulando a una molécula de ADN.

Cruza. Operador genético que forma un nuevo cromosoma a partir de la combinación de partes de los cromosomas de sus padres.

Entorno. Espacio de trabajo donde las hormigas llevan a cabo una búsqueda.

Extinción. Desaparición de uno o más individuos que conforman una población.

Feromona. Sustancia química producida por las glándulas sexuales de algunos animales para atraer o comunicarse con individuos de su misma especie.

Gen. O gene, es una sección de un cromosoma que codifica un solo parámetro.

Heurísticas. Las heurísticas son reglas no formales que reducen el espacio de búsqueda de problemas complejos, donde el número de posibles soluciones es muy grande. Mejoran el desempeño promedio de la solución de un problema, aunque no mejore necesariamente el desempeño en el peor caso.

Individuo. Un solo miembro de una población de soluciones para un problema.

Mutación. Operador genético que forma un nuevo cromosoma debido a alteraciones de los alelos de un cromosoma.

Optimización. La optimización se encarga de resolver problemas del tipo $\max(\min) f(x) | x \in A \subseteq \mathbb{R}^n$ donde $x = (x_1, ..., x_n)$ es un vector y representa variables de decisión, f es la llamada función objetivo y representa o mide la calidad de las decisiones (números) y A es el conjunto de decisiones factibles o restricciones del problema. Dichas restricciones a veces se pueden representar como solución de un sistema de igualdades y desigualdades.

Planificación de rutas. En robótica, es una técnica que permite encontrar un camino específico que nos lleve a un punto destino partiendo de un nodo origen.

Regla de evaporación. Regla que permite simular la disminución de la cantidad de feromona en una arista dada.

Regla de transición. Regla que permite elegir un posible nodo a visitar desde un punto de origen.

Selección. Operador genético que muestra el proceso de obtener individuos de una población en base a su aptitud para poder reproducirse.

Swarm intelligence. Es un comportamiento que tienen los individuos aislados de su sociedad, en la que demuestran un comportamiento no inteligente con capacidades limitadas, pero que en conjunto pueden realizar tareas muy complejas.

ESTUDIO COMPARATIVO DE DIFERENTES TÉCNICAS BIO – INSPIRADAS PARA ENCONTRAR EL CAMINO MÁS CORTO DE UN ROBOT MÓVIL DENTRO DE UN LABERINTO

RESUMEN

La Planificación de Rutas o (*path planning*) es una tarea difícil en robótica, así como el de construir un robot y controlarlo. La planificación de rutas tiene por objetivo determinar un camino específico que nos lleve a la meta o destino.

Esta investigación resuelve el problema de planificación de rutas y la optimización de ésta.

Dado un entorno, sobre el cual un robot móvil debe determinar una ruta para llegar a un punto destino, se encuentra el camino más corto que dicho robot pueda seguir. Ésto se realiza utilizando técnicas Bio-inspiradas, tal como Optimización por Colonia de Hormigas (ACO) y los algoritmos Genéticos (AG).

ACO está basado en Swarm Intelligence, donde el complejo comportamiento colectivo emerge de agentes. Estos agentes poseen simples capacidades, que al ser comparadas con su comportamiento colectivo son mucho más complejas. Un ejemplo de esto son las colonias de hormigas cuando buscan alimento para su supervivencia. Los AG´s se basan en principios evolutivos y neo-darwinianos al utilizar operadores basados en la selección, cruza y mutación.

En esta investigación se propone una modificación al algoritmo de optimización por colonia de hormigas. Este algoritmo se aplica a entornos para obtener el camino más corto.

Una vez que se obtiene el grafo de un entorno dado, se aplicará la modificación del algoritmo propuesto de Optimización por Colonia de Hormigas. Debido a que en esta modificación existen varios parámetros que controlan el comportamiento del algoritmo, se optimizarán los parámetros de dicho algoritmo con ayuda de un algoritmo genético.

Finalmente para probar la metodología propuesta y medir su eficiencia, se presenta un estudio comparativo entre las técnicas utilizando entornos reales.

A COMPARATIVE STUDY OF DIFFERENT BIO-INSPIRED TECHNIQUES APPLIED TO FIND THE BEST ROUTE THAT A MOBILE ROBOT COULD FOLLOWS IN A LABYRINTH

ABSTRACT

Path planning it is a difficult task in robotics, as well as construct and control a robot. The main propose of path planning is find a specific route in order to reach the target destination.

This research tries to solve the path planning problem and its optimization.

Given an environment, where a mobile robot must determine a route in order to reach a target destination, we found the shortest path that this robot can follow. This goal is reach using bio-inspired techniques, as Ant Colony Optimization (ACO) and the Genetics Algorithms (GA).

ACO is based in Swarm Intelligence, where the complex collective behavior emerges from agents. These agents have simple capacities, but when they are compared against its collective behavior, they are much more complex. An example of this is an ant's colony when they are looking for food for its survival. The GA´s are based on evolutionary principles and neo-darwinian principles using operators based on selection, a crossover and mutation.

In this research we propose a modification to ant colony optimization algorithm. This algorithm is applied to different environments for obtain the shortest path.

Once the graph of given environment is obtained, the modification of the proposed ACO algorithm is applied. Due to in this modification have several parameters that determine the behavior of proposed algorithm, these parameters were optimized using a genetic algorithm.

Finally, in order to test the accuracy of the proposed method, we perform a comparative study between the proposed techniques using real environments.

ÍNDICE

CAPÍTULO 6

ÍNDICE DE ALGORITMOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ESTUDIO COMPARATIVO DE DIFERENTES TÉCNICAS BIO-INSPIRADAS PARA ENCONTRAR EL CAMINO MÁS CORTO DE UN ROBOT MÓVIL DENTRO DE UN LABERINTO

RESUMEN

La Planificación de Rutas o (*path planning*) es una tarea difícil en robótica, así como el de construir un robot y controlarlo. La planificación de rutas tiene por objetivo determinar un camino específico que nos lleve a la meta o destino.

Esta investigación resuelve el problema de planificación de rutas y la optimización de ésta.

Dado un entorno, sobre el cual un robot móvil debe determinar una ruta para llegar a un punto destino, se encuentra el camino más corto que dicho robot pueda seguir. Ésto se realiza utilizando técnicas Bio-inspiradas, tal como Optimización por Colonia de Hormigas (ACO) y los algoritmos Genéticos (AG).

ACO está basado en Swarm Intelligence, donde el complejo comportamiento colectivo emerge de agentes. Estos agentes poseen simples capacidades, que al ser comparadas con su comportamiento colectivo son mucho más complejas. Un ejemplo de esto son las colonias de hormigas cuando buscan alimento para su supervivencia. Los AG´s se basan en principios evolutivos y neo-darwinianos al utilizar operadores basados en la selección, cruza y mutación.

En esta investigación se propone una modificación al algoritmo de optimización por colonia de hormigas. Este algoritmo se aplica a entornos para obtener el camino más corto.

Una vez que se obtiene el grafo de un entorno dado, se aplicará la modificación del algoritmo propuesto de Optimización por Colonia de Hormigas. Debido a que en esta modificación existen varios parámetros que controlan el comportamiento del algoritmo, se optimizarán los parámetros de dicho algoritmo con ayuda de un algoritmo genético.

Finalmente para probar la metodología propuesta y medir su eficiencia, se presenta un estudio comparativo entre las técnicas utilizando entornos reales.

CAPÍTULO 1

INTRODUCCIÓN

La robótica es un campo muy amplio de la Inteligencia Artificial, la cual incluye diferentes disciplinas que se incorporan a ella (ciencias de la computación, ingeniería eléctrica, entre otras). Una rama que se deriva de la robótica es la que estudia los robots móviles. Estos robots móviles se dividen en varios tipos, los cuales ayudan en tareas en las que se necesita precisión, calidad y gran fuerza, amen de otras, en las que el ser humano pondría en riesgo su vida o simplemente no las podría realizar. Algunos ejemplos de estas tareas son: la exploración de lugares de difícil o peligroso acceso, la transportación de algún objeto delicado o gran peso o el guiado de artefactos o del mismo robot a través de un entorno, etc. Para realizar tareas del mundo real, los robots móviles dependen de un sistema de control que consta de 3 etapas, ver Figura 1.1.

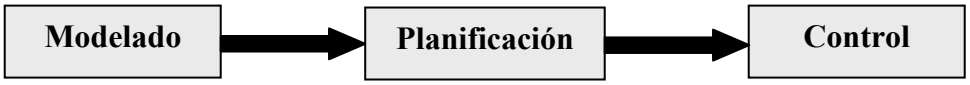


Figura 1. 1 Sistema de control de un robot móvil.

En el primer módulo se procesan los datos recibidos de los sensores, dichos datos son usados para construir un modelo del mundo real. El segundo módulo (el cual será de nuestro interés) usa el modelado del mundo para ejecutar un plan de acción y el tercer módulo realiza las acciones indicadas controlando los motores del robot.

La planificación de la ruta (en inglés *path planning*), requiere la determinación de la posición actual del robot y la posición de la meta o punto destino, ambas en el mismo marco de referencia [1]. Consiste en determinar la ruta que dicho robot puede seguir desde un punto de inicio hasta un punto final, dentro de un entorno determinado, tratando de moverse con un costo mínimo. Esta tarea se puede llevar a cabo por medio de varios

algoritmos clásicos como lo son planificación de rutas basados en mapas de rejilla, planificación de rutas con mapas topológicos, evadir colisiones [2], entre otros. Sin embargo, existen otros tipos de algoritmos que podrían ayudar a planificar la ruta con un costo mínimo. Estos algoritmos son los llamados algoritmos Bio-inspirados, los cuales serán el centro de atención de esta investigación.

Hasta el momento las ciencias computacionales, no han dejado de ser eso, algo sólido, rígido, donde la ciencia y la tecnología no han aportado un cambio radical y brusco en ella. Por años el ser humano ha tratado de hacer que las computadoras se asemejen a él para resolver tareas fáciles y complejas, sin embargo, éstas trabajan de una manera contraria a la propia naturaleza humana.

Las computadoras no son fáciles de influenciar por su entorno, es decir, no realizan una acción con respecto a un cambio en el mismo, para ello se necesita ser metódico y preciso; con ello podemos concluir que las computadoras son incapaces de adaptarse. La adaptación se da en los sistemas cuya habilidad de cambiar interiormente por ellos mismos está acorde con los cambios dados en su entorno, pero con la condición que continúe su funcionamiento. Los ejemplos estereotipados de sistemas adaptativos no son creaciones humanas, si no de la naturaleza. En ésta podemos encontrar a los diferentes microorganismos como bacterias, virus, microbios e insectos como las hormigas, las abejas y termitas, etc. Hay muchos ejemplos de sistemas inspirados biológicamente conocidos como Sistemas Bio-inspirados. Estos Sistemas Bio-inspirados adaptativos son complejos, ya que la mayoría de las pequeñas partes que constituyen esos sistemas son complicados, debido a la manera de construirlos o comprenderlos.

Peter Coveney y Roger Highfield escribieron en *Fronteras de la Complejidad*: “Dentro de la ciencia, la complejidad es una palabra para una nueva manera de pensar acerca del comportamiento colectivo de muchas básicas, pero interesantes unidades, como los átomos, las moléculas, neuronas o bits en una computadora. Para ser más precisos nuestra definición es que la complejidad es el estudio del comportamiento de colecciones macroscópicas cuyas unidades están dotadas con el potencial de evolucionar en el tiempo. Sus interacciones conducen a un coherente fenómeno colectivo llamado propiedades emergentes que solo pueden ser descritas en niveles mas altos que las unidades. En este sentido, el total es más que la suma de sus componentes…” [3].

Así pues nos introducimos a un mundo donde técnicas computacionales que resuelven problemas de manera abstracta, evolucionan al basarse en aproximaciones de acciones biológicas. Las acciones biológicas se realizan paso a paso y tienen la característica (en algunos individuos) de llevarse a cabo de manera colectiva, resolviendo problemas tan complejos que ni la ciencia misma podría comprender y la tecnología no podría reproducir.

En los últimos años, se ha intentado copiar funciones biológicas y proponer soluciones basadas en ellas para resolver problemas complejos del mundo real. Como consecuencia se han hecho investigaciones acerca del comportamiento de grupos como las colonias de hormigas en la búsqueda de alimento para su supervivencia. Se han estudiado también las abejas, la forma en que se comunican entre ellas y su significado la cual

emplean para llamar a más de su especie, o las termitas con sus inmensos nidos que llegarían a rebasar la altura de un ser humano. En cada uno de estos grupos de insectos, los integrantes de manera aislada no sobrevivirían, pero en grupo pueden realizar sorprendentes y a veces inexplicables tareas. Las hormigas por ejemplo, son insectos diminutos, ciegos que pueden tener una organización fascinante. Cada uno de sus miembros tiene una jerarquía y una responsabilidad; estos insectos controlan la temperatura dentro del nido sin importar las condiciones climáticas en el exterior en el nido y entre otras actividades encuentran la ruta más corta que siguen desde su nido hacia la fuente de alimentación. Esta última tarea se ha simulado en las Ciencias Computacionales con propósitos de optimización; por ello, el campo de las Heurísticas ha abierto una rama más, dando lugar a los algoritmos Bio-inspirados (los cuales se basan en modelos biológicos).

Sin embargo, esto no quiere decir que en dichos algoritmos bio-inspirados se dé exactamente el proceso biológico, físico y químico hecho por la naturaleza (ya que la precisión de la naturaleza nos deja en desventaja), sólo se trata de hacer una simulación de esos procesos, la cual nos llevará a obtener un comportamiento colectivo emergente. Los algoritmos bio-inspirados, en especial el basado en el comportamiento de las hormigas en la búsqueda de su alimento se puede aplicar al problema de optimización de rutas, es decir, el clásico problema de planificación de rutas en robótica.

1.1 Problema a resolver

Dado un entorno, sobre el cual un robot móvil debe determinar una ruta para llegar a un punto destino, encontrar el camino más corto que dicho robot pueda seguir. El entorno debe de tener la característica de contener la posición de inicio y la posición a la que se quiere llegar (destino).

1.2 Objetivo

Proponer e implementar técnicas Bio-inspiradas basadas en algoritmos clásicos para resolver el problema de planificación de rutas en un entorno dado.

1.3 Propuesta

Se propone una metodología para resolver el problema de planificación de rutas en un entorno dado. Una vez que se obtiene el grafo de un entorno dado (utilizando cualquier técnica) se realizarán los siguientes 3 pasos:

1. Se aplicará la modificación del algoritmo propuesto de Optimización por Colonia de Hormigas sobre el grafo.
2. Se optimizarán los parámetros del algoritmo propuesto con ayuda de un algoritmo genético.

3. Se aplicará la metodología en entornos reales.

1.4 Aportaciones

La siguiente investigación aporta contenidos importantes para la robótica, resolviendo el problema de planificación de rutas, dichas aportaciones son las siguientes:

- Se desarrolla una modificación del clásico algoritmo de Optimización por Colonia de Hormigas (ACO por sus siglas en inglés). Esto con el fin de resolver el problema de planificación de rutas para un robot móvil.

- Se desarrolla un algoritmo híbrido que mezcla Algoritmos genéticos (AG) y (ACO), con el fin de optimizar los diferentes parámetros que se encuentran en el algoritmo propuesto.

- Se realiza una comparativa entre las técnicas Bio-inspiradas propuestas y un algoritmo clásico de optimización de rutas (Algoritmo de Dikjstra).

1.5 Contenido de la tesis

La siguiente tesis se encuentra dividida como sigue:

Capítulo 2: contiene la descripción y clasificación de algunos trabajos publicados en revistas que tratan de dar solución a la planificación de rutas.

Capítulo 3: se dan los conceptos básicos de los que parte esta investigación, para una mayor comprensión y entendimiento del problema, así como de sus soluciones y resultados.

Capítulo 4: se enfoca a cada uno de los 3 algoritmos clásicos basados en el comportamiento de la colonia de hormigas, explicando la teoría en la que se basan y se expone el pseudo-código de cada uno.

Capítulo 5: explica la metodología propuesta, el desarrollo de un algoritmo basado en la optimización por colonia de hormigas y una combinación de éste con los algoritmos genéticos; los cuales ayudarán para resolver nuestro problema particular.

Capítulo 6: se muestran los resultados experimentales y sus interpretaciones, comparando los algoritmos propuestos y un algoritmo clásico que resuelve el problema directamente.

Capítulo 7: finalmente se dan las conclusiones acerca de esta tesis y los trabajos a futuro que se derivan de ésta.

CAPÍTULO 2

ESTADO DEL ARTE

Los animales han sido una inspiración fascinante para científicos; su adaptabilidad, su flexibilidad y su gran variedad de comportamientos, los han hecho un "benchmark" para probar la eficiencia de dispositivos robóticas.

La planificación de rutas, es una de las tareas más importantes en robótica, por ello una serie de investigadores dedicados a esta rama de la inteligencia artificial han dedicado su tiempo en desarrollar algoritmos que nos permitan manipular robots móviles para realizar una tarea específica, encontrando el camino que nos lleve al objetivo. Sin embargo el encontrar caminos que sean los más cortos u óptimos son indispensables para algunos problemas, por ello la propuesta de utilizar algoritmos que optimicen la planificación de rutas ha contribuido al avance de la evolución artificial y la robótica autónoma.

Los algoritmos Bio-inspirados en Colonias de Hormigas, así como los algoritmos genéticos son un ejemplo para optimizar la planificación de rutas. En éste capítulo se describirán algunos trabajos donde se utilizan dichos algoritmos para optimizar rutas, así como otras aplicaciones. A través del tiempo han surgido múltiples trabajos que involucran los siguientes temas.

2.1 Robots móviles

La robótica, al principio de su nacimiento solo podía ofrecer para el hombre una satisfacción como la creación de una muñeca mecánica que cantaba o la aparición de Jacques de Vaucanson quien desarrolló una figura de tamaño natural de un pastor que tocaba el tabor y la flauta y tenía un repertorio de doce canciones y un pato *con aparato digestivo*, que es considerado su pieza maestra. El pato tenía más de 400 partes móviles, y podía batir sus alas, beber agua, digerir grano, y defecar.

Ahora la robótica es una necesidad para las tareas difíciles de realizar para el ser humano, como el de entrar a los cráteres de volcanes en actividad, dar mantenimiento a naves espaciales, ir a planetas lejanos, cargar equipos pesados y ensamblarlos, etc., sin embargo, actividades tan sencillas para el hombre como correr, andar o coger un objeto sin romperlo, no ha dado resultados satisfactorios.

Los primeros robots y algunos dedicados a tareas específicas, contienen procesos que nunca cambian ante el entorno, con lo cual sólo responden ante una misma situación, a diferencia de los robots autónomos, que realizan tareas bajo entornos dinámicos.

Algunos trabajos como en [4], la aplicación de ejecutores de movimiento primitivo como la evitación de obstáculos, "goal following", "wall following", "docking" y la trayectoria para la navegación del robot móvil son integrados con un planificador de movimiento, asistentes y un "behavior arbitrator", basados en una arquitectura descentralizada del control y una memoria compartida.

En [5] los autores proponen un controlador general para cualquier vehículo que sea conducido, cambiando vectores del objeto destino. Muestran que la velocidad restringida del vehículo puede ser modelada con un elemento de saturación no lineal.

La manipulación de un objeto mediante un robot móvil usando palillos como herramientas [6] trabaja de manera cooperativa con ayuda de múltiples robots móviles utilizando una técnica basada en el control de la fuerza. Estos robots se guían por medio de posición-control. La técnica que se propone es la manipulación sin utilizar la información de sensores, sólo se calculan las condiciones en las cuales se encuentra el objeto y se manipulan los robots conforme a los análisis obtenidos.

El desarrollo de un sistema óptico para múltiples robots móviles en un entorno [7], opera con un modelo del entorno, se comunica con los robots móviles e indica sus posiciones destino, por medio de una luz láser proyectada hacia la tierra.

En [8] Nakamura y Sekiguchi, desarrollan un método para dar a conocer la naturaleza caótica de un robot móvil con un controlador de movimiento caótico. El movimiento es caracterizado por la transitividad topológica y la dependencia sensible en las condiciones iniciales. Para el movimiento que se realiza durante la exploración, el robot caótico nunca necesita el mapa de trabajo ni planea un movimiento global, sólo requiere la normal local del límite del espacio de trabajo cuando está en el.

Trevai, *et. al.* en [9] presentan un método para exploración cooperativa a través de robots móviles múltiples en un área de trabajo restringida. Estos robots generan y comparten la trayectoria de la longitud mínima para cada robot móvil, en la cual hay varios puntos de observación para explorar el área de trabajo.

2.2 Planificación de rutas

Robots equipados con una sola rueda han sido utilizados para llevar a cabo investigaciones sobre conducta, navegación, y planificación de rutas. La planificación de rutas es una tarea difícil en robótica, por la complejidad del entorno dado. Algunos trabajos son los siguientes:

Los autores en [10] muestran un trabajo en donde se exploran los problemas 3D de planificación de rutas, queriendo dar soluciones al tomar en cuenta una planificación global y dinámica.

En 1995 se propuso en [11] un marco computacional para la planificación de rutas dinámicas. Esto se hizo mediante una función armónica, la cual no tiene un mínimo local. La solución usa un acercamiento digital sobre un conjunto de computadoras en red usando un paquete de software.

En [12] se describe un algoritmo evolutivo para la solución multi-dimensional de problemas de planificación de rutas para aplicaciones robóticas. El algoritmo utiliza una multi-resolución de la representación del camino para codificar los caminos candidatos. Dicho algoritmo es flexible, puede manejar problemas multidimensionales de la planeación de rutas, utiliza un dominio específico del conocimiento para tomar decisiones y puede acomodar diferentes criterios de optimización.

En [13] los autores proponen la idea del cómo aplicar un Algoritmo Genético para encontrar camino globalmente sub-óptimo para un grupo de robots que trabajan en realizar ciertas tareas. Los modelos utilizados trabajan para los dos tipos del problema del Múltiple Agente Viajero.

Los autores en [14] proponen el usar Algoritmos Genéticos para la planeación de rutas de robots móviles en entornos no estructurados. El algoritmo genético usa una simple y única representación del camino, usa coordenadas naturales para representar el entorno y mapas de rejillas que forman los nodos de los caminos. El algoritmo genético propuesto también incorpora conocimiento del dominio en sus tres operadores genéticos, problema especifico para la planeación de la ruta del robot.

La planeación de la trayectoria para el tradicional robot pulidor utiliza el método de "teach-play-back", es decir, después de ajustar los errores, el robot pulidor repite el mismo programa fijo en una trayectoria predeterminada; en [15] los autores describen esquemas de la planeación de la trayectoria que proponen en un algoritmo genético llamado IGA. Este algoritmo tiene más ventajas que el algoritmo tradicional.

En [16] Daniel Ashlock, *et. al*., hacen un estudio donde presentan un sistema de computación evolutiva que genera problemas de planeación de rutas substancialmente diversos y de tamaño específico.

2.3 Algoritmos bio-inspirados

Algunos trabajos realizados para el desarrollo de este tema, desde su comienzo y su evolución a través del tiempo se muestran a continuación:

En [17], realizan uno de los primeros trabajos con colonia de hormigas. Presentan un algoritmo distribuido para solución de problemas de optimización combinatorial. Se inspiraron por la observación real de las colonias de hormigas, aplicando el algoritmo a el problema del agente viajero de manera simétrica y asimétrica.

Una mejora al Algoritmo de hormigas se da en [18] cuando Marco Dórigo *et. al.* Proponen un nuevo acercamiento en optimización combinatorial estocástica un modelo con una retroalimentación positiva, computación distribuida y el uso de una heurística constructiva. El uso de dicha retroalimentación positiva provoca el descubrir rápidamente buenas soluciones, el cómputo distribuido evita la convergencia prematura. La heurística encuentra soluciones aceptables en los primeros tiempos del proceso de búsqueda. El método se aplica al problema del agente viajero comparando con la búsqueda Tabú y el Recocido Simulado.

En [19] los autores hacen una comparativa entre los algoritmos: Recocido Simulado, “elastic net”, programación evolutiva y algoritmo genético; todos aplicados al problema del agente viajero.

En el año de 1997 el Sistema de Colonia de Hormigas fue propuesto en [20] por Marco Dórigo y Gambardella. En él proponen un algoritmo distribuido en donde un conjunto de agentes llamadas hormigas, cooperan para buscar soluciones al problema del agente viajero. Las hormigas usan una forma de comunicación mediante la feromona que depositan en aristas de un grafo, mientras construyen soluciones; este algoritmo es comparado con una variante llamada ACS-3-opt.

Una implementación diferente se refleja en [21], donde los autores describen cómo al utilizar agentes biológicamente inspirados se resuelve el control y mantenimiento de problemas de telecomunicaciones. La colección de agentes exhibe una forma de control distribuido con la comunicación a través del entorno. La aplicación de las hormigas como agentes se aplica al problema de ruteo en una red de comunicaciones.

Hozefa y Erick Bonabeau en [22] usan un algoritmo genético para la búsqueda de los mejores parámetros a utilizar en el Sistema de Colonia de Hormigas aplicado al problema del agente viajero.

Pierre Delisle *et. al.* en [23] presentan una implementación paralela de una colonia de hormigas para solución de un manejador de problemas de un centro de aluminio. Implementan una memoria paralela compartida de una colonia de hormigas al utilizar OpenMP.

El uso de hormigas elitistas aparece en [24] donde Tony White utiliza la información local para reforzar las mejores soluciones, de tal manera que realza la capacidad del algoritmo de hormigas para el multiproceso de implementación actual de redes. En el modelo propuesto, las hormigas son dotadas con una memoria de su mejor recorrido actualizado.

Los autores en [25] utilizan un Sistema de hormigas basado en un método de optimización. El método se aplica al instalar un modelo que aprende para la máscara "Tuned", la cual es usada para la clasificación de textura. La aplicación de la simulación del comportamiento de hormigas, genera una máscara óptima para dicha clasificación y presenta un modelo de aprendizaje híbrido, basado en los algoritmos de hormigas, algoritmos genéticos y un algoritmo simplex.

Un artículo, también interesante, consiste en mejorar un algoritmo de hormigas para problemas de clasificación en minería de datos Ant-Miner en [26] propuesto por BoLiu, *et.al.*

En [27] los autores aplican el algoritmo de Sistema de Colonia de Hormigas, al problema del grafo coloreado. El algoritmo conforma el max-min de la estructura del sistema de hormigas y explota una búsqueda heurística local para mejorar su eficiencia.

Dandan Zhang *et. al.* en [28] describen un método adaptativo de asignamiento de tarea para un equipo distribuido de robots móviles, con funcionalidades iniciales idénticas en un entorno desconocido. Los robots adquieren la llamada "auto-organización".

El algoritmo de sistema de hormigas es una simplificación de la optimización por colonia de hormigas, y puede ser usado para resolver una gran variedad de problemas de optimización discreta. En [29] los autores muestran como la función objetivo basada en modelos de clusterización tal como hard y fuzzy c-means, pueden ser optimizados usando extensiones particulares de la simplificación de la optimización por colonia de hormigas.

Los "constraint satisfaction problems (CSPs)" buscan una asignación de valores a las variables que satisfacen un conjunto de restricciones entre esas variables. Este problema general tiene muchas aplicaciones en la vida real, tales como el planear, el de reservar recursos para la computadora, reconocimiento de patrones o visión por computadora. En [30] los autores modelan una técnica genérica que resuelve cualquier problema CSP, como la búsqueda del mejor conjunto de vértices de un grafo. También se estudia la influencia de los valores de los parámetros en el problema del SAT sobre el proceso de resolución.

En [31] Rafael Parpinelli, *et. al.* proponen un algoritmo para minería de datos llamado Ant-Miner, que tiene como propósito extraer reglas de clasificación desde los datos y se compara con un algoritmo clásico llamado CN2.

José Aguilar, *el. al.* en [32] presentan un nuevo algoritmo distribuido llamado el General Sistema de Hormigas, para resolver problemas de optimización combinatorial. Realizan un mapeo del espacio de solución del problema combinatorial en el espacio donde

las hormigas caminarán. Se probó el en problema del agente viajero y el del “Graph Partitioning”.

Los autores en [33] presentan una nueva versión del algoritmo Ant-Miner, el cual es llamado Unorderer Rule Set Ant-Miner. El algoritmo tiene la ventaja de obtener más reglas modulares, es decir reglas que pueden ser interpretadas independientemente de otras reglas, no como las reglas en una lista ordenada donde la interpretación requiere conocimiento previo.

En [34] los autores desarrollan un algoritmo para encontrar el menor costo en las restricciones de grado en un “spanning tree” de un grafo dado. Las hormigas buscan las aristas candidatas con las cuales se puede construir el árbol.

En [35] presentan un algoritmo para resolver un número generalizado de problemas de grafos coloreados. Usan el método estándar del procesamiento de la entrada el mismo algoritmo y puede ser usado para resolver los problemas del multi-coloreado y “Bandwidth Multicoloring”.

2.4 Resumen

En este capítulo se describió una serie de trabajos que dejan ver el cómo han evolucionado los tres temas principales que hacen posible esta investigación: la robótica, la planificación de rutas y los algoritmos bio-inspirados.

En cada uno de estos temas nos encontramos con algunos artículos que han podido ayudar al avance científico y tecnológico del ser humano. Este capítulo presentó una pequeña reseña histórica para poder adentrarnos al mundo de la inteligencia y la vida artificial.

CAPÍTULO 3

CONCEPTOS BÁSICOS

La robótica trata de cubrir todos los temas necesarios para programar un robot con inteligencia artificial en aplicaciones de sensado y planificación de rutas para dar solución a un problema.

A través del tiempo, el campo de la robótica se ha transformado, al pasar de máquinas rústicas hasta la construcción de humanoides, que permiten resolver tareas sencillas o complicadas.

Algunos de estos robots, se han construido basándose en el comportamiento de seres vivos dándole un toque sutil del raciocinio. No sólo la parte física y de comportamiento se ha basado en la vida, si no también los algoritmos que controlan dichos robots, los cuales nos dan una mayor posibilidad de encontrar soluciones a problemas difíciles de resolver para el ser humano.

A continuación se introducen los conceptos básicos de la siguiente investigación.

3.1 Robótica

La robótica es la disciplina cuyo objetivo es el diseño y puesta en operación de entes artificiales a nuestra imagen y semejanza o, por lo menos con capacidades que les permitan cierta autonomía [36].

En el campo de la robótica se ha trabajado con procedimientos de ingeniería como la creación y construcción de robots. Estos procedimientos siguen los principios tradicionales como la descomposición del problema en partes; por ejemplo el diseño manual, comprobación de cada componente, verificación e integración (entre otras), con ayuda de metodologías de otros campos. Sin embargo en la última década ha aparecido una

nueva visión que trabaja con características de la evolución así como la selección (ambas artificiales) para realizar las mismos objetivos.

La robótica evolutiva, utiliza la evolución artificial en el desarrollo de los robots autónomos. La evolución artificial simula en la computadora la evolución natural, y al igual que en el mundo natural, se pretende que en sucesivas generaciones vayan apareciendo individuos mejor adaptados a la supervivencia en su hábitat. La aplicación de los principios de la evolución artificial, lleva a la creación de individuos que consiguen sobrevivir en un entorno, realizando determinadas tareas.

La vida artificial simula o crea nuevas formas de vida no basadas en carbono, sino en estructuras de información. Como ya decía Chris Langton, considerado padre de la Vida Artificial: "se pretende estudiar la vida como podría ser y no sólo como es".

El término *robot* fue usado por primera vez por el escritor checo Karen Capek en su obra de teatro R.U.R. (Rossum´s Universal Robots), escrita en 1921. Karen elige la palabra robot como derivación de la palabra eslovaca *robota*, que significa trabajo o servidumbre y de la que se obtiene *robotnik* o *servidor.*

También el famoso divulgador científico estadounidense de origen ruso Isaac Asimov, contribuyó al establecimiento del término robot. En su relato de ciencia-ficción Runaround (1942), se utilizó por primera vez la palabra "robótica", enunciando las tres Leyes de la Robótica, las cuales son:

- Un robot no puede dañar a un ser humano o, por omisión de acciones, permitir que sufra algún daño.
- Un robot debe obedecer las órdenes de los seres humanos, excepto cuando tales órdenes entren en conflicto con la primera ley.
- Un robot debe proteger su propia existencia, salvo cuando dicha protección no entre en contradicción con las dos anteriores.

Y la cuarta ley conocida como Ley Cero, publicada en Robots e Imperio (1985), es una síntesis de las anteriores:

- Un robot no debe perjudicar a la humanidad o, por omisión de acciones, permitir que la humanidad sufra algún daño.

Deacuerdo con la Asociación Japonesa de robots industriales (JIRA por sus siglas en ingles), los robots se dividen en las siguientes clases [37]:

1. *Dispositivo de manejo manual:* Es un dispositivo que tiene varios grados de libertad, el cual es dirigido por un operador.
2. *Robots de secuencia fija:* Es un dispositivo manejador, el cual realiza etapas sucesivas de una tarea predeterminada, utilizando un método que es difícil de modificar o que no puede ser modificado.

3. *Robots de secuencia variable:* Es del mismo tipo de dispositivo que la clase anterior, pero los estados pueden cambiar fácilmente.
4. *Robots a control remoto:* El operador (ser humano) realiza la tarea manualmente, conduciendo o controlando el robot. Esta información es recordada cuando es necesario, y el robot puede realizar la tarea de manera automática.
5. *Robots de control numérico:* El operador humano hace trabajar al robot proveyéndole de un programa, es decir enseñándolo manualmente.
6. *Robots inteligentes:* Son aquellos que cuentan con la característica de entender su entorno, y la habilidad de completar exitosamente una tarea, sin importar que el entorno esté cambiando.

La Asociación de la Industria Robótica Americana define un robot al utilizar 3 clases: "Un robot es un manipulador reprogramable y multifuncional diseñado para mover material, partes, herramientas, o dispositivos especializados mediante movimientos variables" [36].

Muchos de los robots utilizados en la industria son manipuladores o robots ensambladores que operan en un espacio de trabajo limitado y no pueden trasladarse totalmente. Sin embargo existe otra clase de robots que si pueden cambiar su posicionamiento, éstos reciben el nombre de robots móviles.

Los robots móviles son aquellos que cambian su localización (coordenadas en el espacio) mediante su locomoción; un ejemplo de ello es el vehículo guiado automatizado (AGV por sus siglas en inglés). Este trabaja en entornos modificados previamente a la realización de su tarea, guiándose por marcas en su entorno o por la colocación de obstáculos en el mismo, realizando su tarea a través de los caminos o rutas fijas.

Debido a las características de los vehículos guiados automatizados, éstos son inflexibles y costosos. Ésto implicaría el no realizar no realizar la tarea deseada, a causa de cambios imprevistos en el entorno (como el quitar o poner algún obstáculo, marca o señal).

Debido a las limitaciones de los vehículos guiados automatizados, surgen los llamados robots móviles autónomos. Estos robots son capaces de reaccionar ante situaciones no consideradas en la programación de su control sin ninguna supervisión exterior. A pesar de que su control se defina por un programa, el robot debe realizar en todo momento los movimientos necesarios para "sobrevivir" en su entorno [73] y cumplir las tareas encomendadas, sin que el robot autónomo sea totalmente preprogramado [36].

El proceso de realizar una tarea de un robot móvil, sigue una serie de pasos que se desarrollan en los módulos que conforman la Figura 3.1.

En dicha Figura se muestran las tareas principales que realiza un robot para cumplir su objetivo exitosamente. El proceso se divide en tres módulos principales que son:

- *Modelado del entorno:* El robot recibe los datos (señal procesada) a través de sensores o de algún dispositivo de captación del entorno real (percepción). Una vez obtenida la señal, ésta se transforma en lo que será nuestro modelo del mundo real.

- *Planificación:* Este módulo utiliza el modelo del mundo, así como la percepción adquirida para decidir la realización de una acción. Éste plan se basa en la naturaleza del mundo del propio robot, donde el modelo es una generalización y abstracción del mundo real.
- *Control:* Ya que las acciones se han generado, entonces se ejecutan por medio del control de los motores del robot.

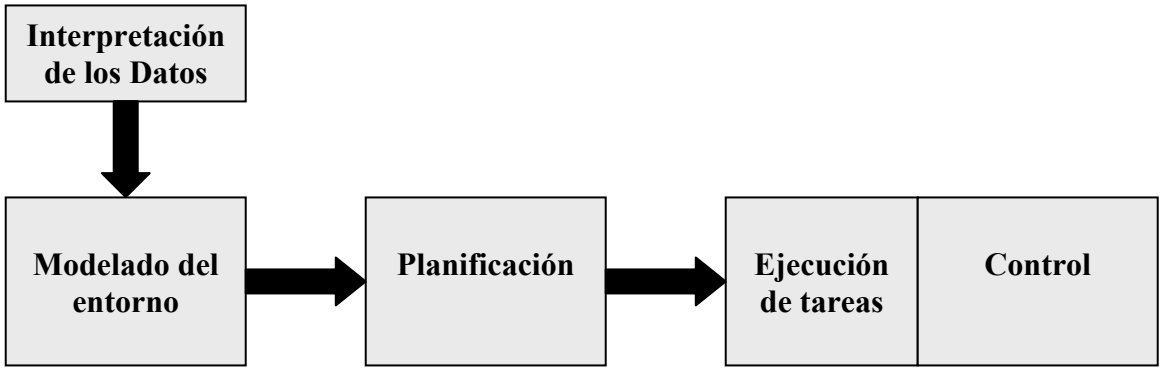


Figura 3. 1 Proceso que sigue un robot móvil para realizar una tarea.

3.2 Planificación de rutas

La Planificación de Rutas o (*path planning*) es una tarea difícil en robótica, así como el de construir un robot y controlarlo; sin embargo, en esta investigación sólo nos enfocaremos al problema de planificación de rutas, y a la optimización de ésta (ver Figura 3.2). Éstas actividades se encuentran dentro de la etapa de Planificación del robot.

Figura 3. 2 Etapa de la Planificación.

La planificación de rutas tiene por objetivo determinar un camino específico que nos lleve a la meta o destino. Usualmente descompone el camino en sub-puntos o sub-metas, los cuales constituyen los puntos del camino. Éstos tienen una posición fija en el entorno o espacio de búsqueda del robot, posición representada por las coordenadas (x, y) [38].

3.3 Optimización

La optimización se encarga de resolver problemas del tipo $\max(\min) f(x) \big| x \in A \subseteq \mathbb{R}^n$ donde $x = (x_1, ..., x_n)$ es un vector y representa variables de decisión, f es la llamada función objetivo y representa o mide la calidad de las decisiones (números) y A es el conjunto de decisiones factibles o restricciones del problema. Dichas restricciones a veces se pueden representar como solución de un sistema de igualdades y desigualdades.

Según el problema, será la resolución que se plantee utilizando algún tipo de optimización como por ejemplo:

- Optimización Clásica: Si la restricción no existe o es una restricción de igualdad con menor o igual número de variables que la función objetivo, entonces la solución se encuentra aplicando cálculo diferencial a la función dada.

- Optimización no Clásica: Si la restricción contiene mayor cantidad de variables que la función objetivo, o la restricción contiene restricciones de desigualad, existen métodos en los que se pueden encontrar o los máximos o los mínimos. Si las restricciones son lineales, entonces se aplican algoritmos de álgebra lineal como el método simplex o el dual.

- Optimización Combinatorial: Éstos resuelven problemas que no se pueden resolver con los dos tipos de optimización anteriores, ya que el espacio de soluciones es demasiado grande. Ésta optimización tiende a reducir el espacio y explorarlo eficientemente.

3.4 Heurísticas

El significado de la palabra “heurística” se deriva del griego *heuriskein* cuyo significado es “descubrir” o “encontrar”, pero a través del tiempo ha cambiado su definición.

Newell dice que una heurística es: “Un proceso que puede resolver un cierto problema, pero que no ofrece una garantía de lograrlo” [39] .

Una definición más formal es la de Reeves: “una heurística es un técnica que busca soluciones buenas (casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. En alguno de los casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible” [40].

Sin embargo, actualmente el término heurística se refiere a cualquier técnica que mejore el desempeño en promedio de la solución de un problema, aunque no mejore necesariamente el desempeño en el peor caso [41] .

Las heurísticas son reglas no formales que reducen el espacio de búsqueda de problemas complejos, donde el número de posibles soluciones es muy grande y las soluciones óptimas se dan con eventos aleatorios.

La solución de algunos problemas (partiendo de una tautología), son difíciles de resolver por las razones siguientes, (entre otras) [42]:

- El número de posibles soluciones en el espacio de búsqueda es tan grande como para ordenar una búsqueda exhaustiva para la mejor respuesta.

- El problema es muy complicado, que sólo para obtener alguna respuesta de todo, se tienen que usar los modelos simplificados de los problemas.

- La función de evaluación, que describe la calidad de cualquier solución propuesta es ruidosa o variable con respecto al tiempo y quizá requiere no sólo de una solución simple sino una serie de soluciones complejas.

- Las posibles soluciones son bastante limitadas y construir una respuesta adecuada es difícil, por eso sólo se buscan soluciones óptimas.

- La persona que resuelve el problema no está preparada o crea una barrera psicológica que evita que encuentre la solución.

3.5 Tipos de Heurísticas

Los métodos heurísticos constituyen un excelente criterio para escoger un resultado efectivo. Este resultado depende del tipo de problema al que nos enfrentamos y del tipo de heurística que se use para resolverlo, en la Figura 3.3 se muestran algunos tipos de heurísticas.

3.5.1 Monte Carlo

En este caso se toma una solución valida y se provoca una pequeña variación aleatoria. Si la nueva configuración es mejor que la anterior y cumple con las restricciones, entonces se mantiene la nueva, si no, entonces se sigue con la misma configuración. Para que haya convergencia se va haciendo cada vez más pequeña la variación aleatoria. Se podría seguir iterando aunque se llegue a un mínimo.

Este método se ha considerado como uno de los mejores algoritmos de nuestro siglo [72].

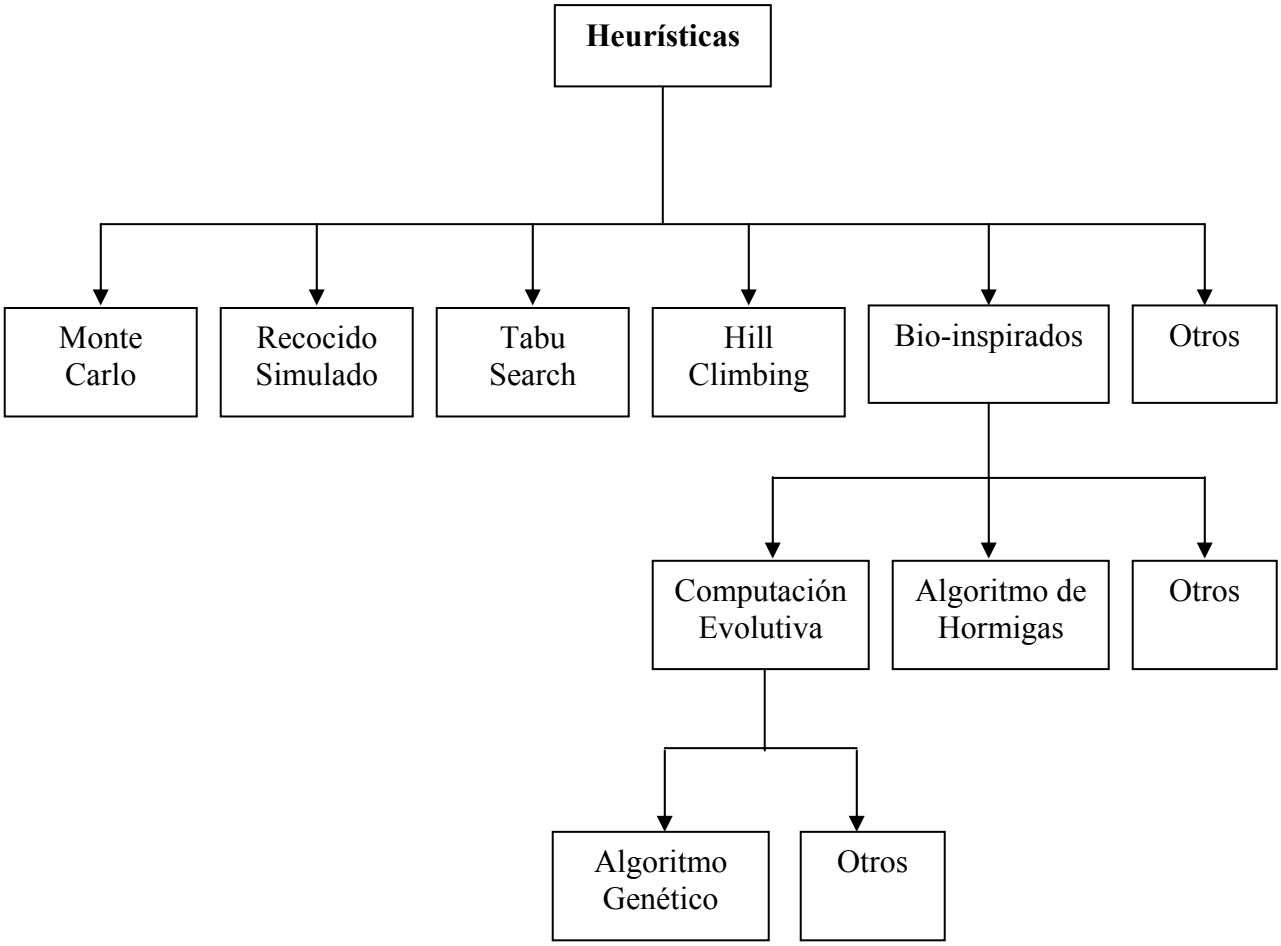


Figura 3. 3 Diferentes Métodos Heurísticos.

3.5.2 Recocido Simulado

El recocido simulado [43] o algoritmo de los Dioses (es un escalador estocástico). Se basa en el proceso de recocido, en la cual los metales se funden y se forjan a la par que se enfrían, hasta conseguir una estructura cristalina, dura y resistente.

Éste método realiza una búsqueda global que requiere de una "temperatura" inicial, una final y una función de variación de la temperatura. Consiste en un ascenso de gradiente comenzando en un punto aleatorio; dependiendo de la temperatura es como se mueve por el espacio de búsqueda. La probabilidad de escoger una solución peor que la actual, desciende con el tiempo; la solución (se aproxima a la solución óptima) se escoge de manera determinista evitando los mínimos locales.

3.5.3 Búsqueda Tabú

La búsqueda tabú [44] fue propuesta por Glover, Taillard y de Werra. El método es utilizado para resolver problemas complejos de optimización. Esta técnica, en realidad es una meta-heurística, que guiará a otras técnicas para que no se quede en óptimos locales. También ayuda a la resolución de problemas combinatorios.

La búsqueda Tabú, explora el espacio de búsqueda de todas las soluciones por medio de movimientos que se harán sólo si la solución siguiente es mejor que la anterior. Para que esta técnica no quede atrapada en un óptimo local o se generen movimientos cíclicos la búsqueda tabú utiliza una "memoria" (lista taboo) en donde se almacenan las soluciones examinadas recientemente, volviéndose prohibidas para el siguiente movimiento que se realizará. También, registra los atributos más comunes de un conjunto de soluciones para buscar en la zona que le corresponde y a largo plazo se extiende sobre regiones no exploradas.

3.5.4 Escalando la colina

Esta técnica se basa en optimización local, sigue la dirección de ascenso o descenso a partir de su posición y requiere muy poco costo computacional. Su comportamiento es irrevocable porque no permite regresarnos a otra alterativa.

La técnica de escalando la colina, evalúa la función en varios puntos, pasando de un punto a otro donde el valor de la evaluación es mayor. La búsqueda termina cuando se ha encontrado el punto con un valor máximo, por lo que no se obtendría la mejor solución.

3.5.5 Algoritmos Bio-inspirados

Son propuestas inspiradas en modelos biológicos, tales como Algoritmos de Optimización basada en Colonias de Hormigas, Algoritmos basados en Enjambres (de abejas, termitas) y Computación Evolutiva. Basan su operación en los principios de la teoría Neo – Darwiniana de la evolución natural.

3.6 Resumen

En este Capítulo se introdujeron los conceptos básicos de la rama de la Inteligencia Artificial: la robótica; así como el problema de planificación de rutas y la optimización.

Otros conceptos muy importantes que se presentaron fueron, fueron las heurísticas y los algoritmos Bio-inspirados, los cuales son el centro de esta investigación.

CAPÍTULO 4

ALGORITMOS BIO-INSPIRADOS

Los animales, nunca han dejado de fascinarnos a quienes los respetamos y admiramos; los hay de todos colores, variedades de especies y tamaños, y sobre todo cuentan con inteligencia asombrosa.

Algunos insectos sociales [45] (que trabajan en grupo, con orden y disciplina) como las hormigas, nos dejan maravillados al descubrir que son ciegas, pero que siempre encuentran su alimento y saben cuando hay que defenderse; construyen nidos desde muy pequeños en la pared y en los árboles, hasta nidos gigantes en la tierra; regulan la temperatura del nido, de modo que no les afectará el clima del exterior; construyen puentes que las ayudan a cruzar arroyos, etc. Su sociedad se encuentra organizada por jerarquías, donde existe la reina, las obreras y soldados y sobre todo, llegan a la fuente de alimentación tomando el camino más corto ¿no es maravilloso?

Gracias a estos animalitos y su comportamiento único, varios investigadores se dieron cuenta de que la forma en la que encuentran su alimento ¡puede ayudar a resolver problemas del propio ser humano!, problemas computacionales del mundo real que requieren una optimización como la que realizan las hormigas al encontrar su camino.

Por otro lado la base de la teoría de la selección natural de Charles Robert Darwin hoy en día combinada con la teoría de la genética de Gregor Mendel y el seleccionismo de August Weissman (que da origen al Neo-Darwinismo) nos permite obtener procesos estadísticos basados en dichas teorías que como resultado, nos da soluciones donde no las había.

Hoy en día, existen diferentes algoritmos que se basan en el comportamiento de las hormigas y en los procesos de supervivencia natural optimizando soluciones y acortando el tiempo de búsqueda de las mismas, gracias a la emulación de su comportamiento en una computadora.

En este capítulo se explican los algoritmos básicos de Optimización por Colonia de Hormigas y el algoritmo Genético, así como la teoría que los cimenta.

4.1 Optimización por Colonia de Hormigas

Varios investigadores han observado algunas colonias de insectos tales como las abejas, hormigas, termitas, etc. y sus comportamientos al realizar sus tareas. Estos comportamientos son sumamente complejos y en ocasiones incomprensibles, sin embargo los resultados son una prueba fiel del asombroso comportamiento colectivo de los insectos.

Las hormigas, son seres diminutos que carecen de visión, se guían solo por el olfato, al ser receptoras de señales químicas que pueden percibir de igual manera las demás de su especie. Las hormigas pueden regular la temperatura del nido sin importar el clima en el exterior, construyen sus nidos con cientos de túneles que conducen a ordenadas cavidades, tienen una jerarquía respetable en su comunidad, dan mantenimiento al nido y pueden encontrar fuentes de alimentación recorriendo el camino más corto para llegar a ella.

La increíble capacidad para llevar a acabo estas acciones, ha dado pie a que las ciencias computacionales traten de emular los difíciles procesos de la naturaleza y aplicarlos a la resolución de problemas complejos que desasosiegan al hombre.

Los grupos inteligentes de insectos o Swarm Intelligence es una propiedad de sistemas de agentes no inteligentes de capacidades individuales limitadas, pero que en conjunto con otros entes de su misma especie muestran un comportamiento inteligente y complejo [46]. Este comportamiento se basa en la Organización Propia (patrón macroscópico que se forma de múltiples iteraciones a nivel microscópico), esta organización requiere ciertas interacciones entre los insectos, las cuales son:

Interacción Directa: En las hormigas, se da mediante las señales químicas volátiles que se perciben mediante el olfato, el gusto así como el contacto con substancias llamadas feromonas (transmitidas por aire o contacto directo), las cuales se identifican por medio de las antenas (cuando intercambian líquidos o cuando existe un contacto químico, mandibular o visual.

Interacción Indirecta: Ésta se da cuando dos individuos interaccionan sin haber contacto. Se lleva a cabo cuando un individuo modifica el entorno y el otro responde al nuevo entorno tiempo después (a esta interacción indirecta se le conoce también como Stigmergy [47].

La acción de seleccionar el camino que la hormiga sigue del nido a la fuente de alimentación se basa en la Organización Propia, y es la hormiga Argentina (Iridomyrmex humilis) en la que se basan científicos para desarrollar el algoritmo de sistema de colonia de hormigas el cual, siendo una heurística y por sus características es un algoritmo de optimización combinatorial, que se ha aplicado a problemas como: el problema del agente viajero (Traveling Salesman Problem TSP) [20], el problema del asignamiento cuadrático

[49], el del grafo coloreado, la planificación del job-shop, ordenamiento secuencial y ruteo de vehículos [50], entre otros.

Las hormigas (en su mundo real), parten del nido en busca de alimento para su subsistencia. Para que las hormigas sean influenciadas en encontrar dicha fuente deben seguir rastros de feromona la cual, por medio de su intensidad de concentración guiará a la colonia al lugar donde se encuentra el sustento. Cabe destacar que con el paso del tiempo, la feromona se evapora en el medio ambiente, por lo que un constante depósito de esta sustancia en los caminos la hará más fuerte y más receptiva a las demás hormigas. Al encontrarse en un punto decisivo, ver Figura 4.1(a); las hormigas eligen al azar el camino a seguir (tomemos en cuenta que uno de los caminos es más corto que el otro), y dejan a su paso su propia feromona, ver Figura 4.1(b); las hormigas caminan a una velocidad constante y aquellas que al azar eligieron el camino más corto llegarán mas rápido a la fuente de alimentación, tomarán su bocado y regresarán al nido oliendo su propia feromona reforzando la misma, y depositando más feromona en el mismo camino. La feromona depositada en el camino más largo se evaporará, ya que la distancia a recorrer es más grande y el tiempo para que la hormiga refuerce la sustancia es menor ver Figura 4.1(c); finalmente las hormigas que les siguen, llegan al mismo punto de decisión, pero ahora ellas se moverán por el lugar donde perciban con mayor fuerza la sustancia química. Con el paso del tiempo las hormigas preferirán el camino que casualmente es el más corto Figura 4.1(d).

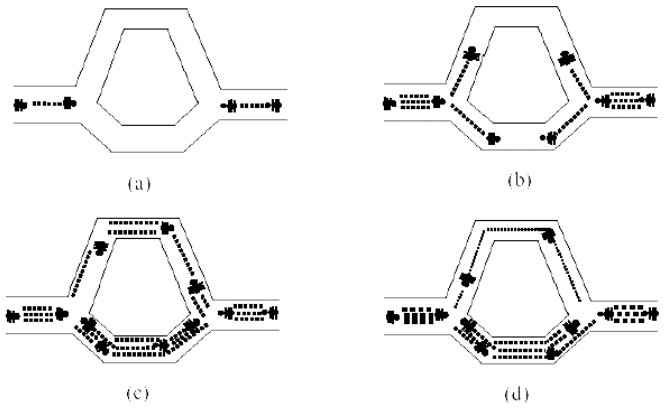


Figura 4. 1 Pasos para encontrar el camino más corto. (a) Las hormigas están en un punto decisivo, (b) se mueven al azar, (c) algunas hormigas llegan más rápido a la fuente de alimentación, dejan su feromona en el trayecto y regresan al nido reforzando dicha sustancia otras hormigas toman un camino largo y la feromona se evapora al no haber alguien más que también la deposite (d) las hormigas eligen el camino en el que perciben una mayor cantidad de feromona, el cual es al final el camino más corto.

Los Doctores Marco Dórigo, Vittorio Maniezzo y Alberto Colorni desarrollaron la metáfora del algoritmo de hormigas [18]. Este se basa en la naturaleza real de estos pequeños insectos, (ver Algoritmo 4.1), para así poder optimizar problemas

combinatoriales; de aquí surge una nueva heurística llamada Optimización por Colonia de Hormigas (ACO por sus siglas en inglés) .

Para esto, estos autores desarrollaron un puente binario en un laboratorio, donde se unen el nido y la fuente de alimentación por medio de dos puentes del mismo tamaño y sin ningún rastro de feromona. Esto permite tener la misma probabilidad de ser seleccionados en un tiempo t. Dicha probabilidad depende del número total de hormigas que usan el camino hasta ese tiempo.

Por lo tanto la probabilidad de elegir un camino está dada por la Ec. 4.1, donde Ai y Bi son el número de hormigas que han usado los puentes A y B, la probabilidad P_A (o P_B) de que la i-ésima ($i+1$) hormiga elija el camino A (ó B).

$$P_A = \frac{(k+Ai)^n}{(k+Ai)^n + (k+Bi)^n} = 1 - P_B \qquad (4.1)$$

La ecuación anterior cuantifica la manera en que una alta concentración en A da una alta probabilidad de elegir el camino A, dependido de los valores absolutos y relativos de Ai y Bi; n determina el grado de no linealidad de la función elegida: es decir, cuando n es grande, si un camino tiene ligeramente más feromona que otro, la siguiente hormiga tendrá una alta probabilidad de elegirlo; k determina la cantidad o el grado de atracción de una camino sin marcar (es decir que ninguna hormiga ha pasado sobre de él, por lo tanto no tiene huella de feromona).

Los algoritmos de colonia de hormigas hacen uso de exploraciones simultáneas de diferentes soluciones por una colección de hormigas idénticas. Las hormigas que tienen un buen rendimiento en una iteración dada, influyen en la exploración de otras hormigas en futuras iteraciones, ya que la huella de feromona es consecuencia de diferentes perspectivas en el espacio de soluciones. Cuando la hormiga de mejor rendimiento refuerza su solución se da un efecto cooperativo a través del tiempo debido a que las hormigas en su siguiente iteración utilizan la huella de feromona para guiar sus exploraciones [52].

4.1.1 Sistema de Hormigas

La Optimización por Colonia de Hormigas (ACO) fue por primera vez aplicada para resolver el problema del Agente Viajero (TSP) con el algoritmo de Sistema de Hormigas (AS Ant System por sus siglas en inglés). El TSP, consiste en visitar todas las n ciudades o puntos ($n \in N$) que se tengan en un país o entorno, pasando sólo una vez por cada uno de ellos, regresando al mismo punto de origen, obteniendo así la longitud mínima de dicho recorrido [18].

La distancia entre las ciudades i y j, para el caso del problema del Agente viajero en el espacio Euclidiano, se define por la siguiente ecuación:

$$d_{ij} = \left[\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right]^{1/2} \tag{4.2}$$

En este caso x_i y y_i son coordenadas de la ciudad i. El problema del Agente viajero se reduce a recorres un grafo **G**(N, E) donde N es el conjunto de ciudades y E es el conjunto de aristas entre las ciudades.

Tómese en cuenta que $b_i(t)$ (i,…, n) es el número de hormigas en la ciudad i en un tiempo t, y se tiene que $m = \sum_{i=1}^{n} b_i(t)$ es el número total de hormigas; para cada hormiga la transición de la ciudad i a la ciudad j en una iteración t, el algoritmo depende de [52]:

- Si la ciudad ha sido o no visitada: para cada hormiga se tiene una "memoria" o lista Taboo, en la cual se almacena el conjunto de J_i^k ciudades que han sido visitadas durante el recorrido de cada una de las hormigas cuando están en la ciudad i. La lista se puede vaciar conforme las hormigas regresan al nido o a su origen.

- El inverso de la distancia: $\eta = 1/d_{ij}$ llamada también visibilidad, representa el deseo de elegir a la ciudad j cuando se está en la ciudad i. Esta información es local y no varía mientras se busca la solución del problema.

- La cantidad de feromona virtual $\tau_{ij}(t)$ en la arista que conecta la ciudad j con la ciudad i, representa la deseabilidad aprendida al ir de una ciudad a otra. Ésta se actualiza durante la búsqueda de solución del problema, reflejando la experiencia adquirida por las hormigas.

La regla de transición o la probabilidad para la hormiga k para ir de la ciudad i a la ciudad j mientras construyen su t-ésima ruta, viene dada por:

$$p_{ij}^k(t) = \frac{\left[\tau_{ij}(t) \right]^{\alpha} \cdot \left[\eta_{ij}(t) \right]^{\beta}}{\sum_{l \in J_i^k} \left[\tau_{il}(t) \right]^{\alpha} \cdot \left[\eta_{il}(t) \right]^{\beta}} \tag{4.3}$$

Dicha regla, se cumple cuando $j \in J_i^k$, y es 0 si $j \notin J_i^k$; α y β son dos parámetros ajustables. Si $\alpha = 0$ quiere decir que la cuidad más cercana tiene una mayor probabilidad de ser seleccionada, por el contrario si $\beta = 0$, solo la amplificación de la feromona trabajará. Este resultado causará la rápida selección del tour que puede no ser una solución óptima.

Por lo tanto, el balanceo entre la longitud de la ruta y la intensidad de la huella es necesario para obtener un resultado óptimo.

Ya se vió que $\tau_{ij}(t)$ es la intensidad de feromona en una arista (i,j) en un tiempo t. Cada hormiga en un tiempo t elige la siguiente ciudad a donde debe dirigirse en un tiempo $t+1$. Una iteración consiste en realizar los m movimientos hechos por las m hormigas en el intervalo de tiempo $t+1$ (ver Algoritmo 4.1), entonces para cada n iteraciones del algoritmo (o ciclo) cada hormiga ha completado un viaje, por lo tanto la intensidad de feromona se debe actualizar. Esta actualización se lleva a cabo con la siguiente ecuación:

$$\tau_{ij}(t+n) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{4.4}$$

donde ρ es un coeficiente tal que $(1-\rho)$, $0 \le \rho < 1$ y representa la evaporización de la feromona entre el tiempo t. Y $t+n$, $\Delta\tau_{ij}^{k}$ es la cantidad por unidad de longitud de la feromona en la arista (i,j) por la k-ésima hormiga entre el tiempo t y $t+n$, viene dada por:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} Q \big/ L^{k}(t) & \text{si } (i,j) \in Taboo^{k}(\mathrm{t}) \\ 0 & \text{si } (i,j) \notin Taboo^{k}(\mathrm{t}) \end{cases} \tag{4.5}$$

donde $\Delta\tau_{ij}^{k}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$, Q es una constante y $L^{k}(t)$ es la longitud del viaje de la k-ésima hormiga; la cantidad inicial de feromona en las aristas es una constante pequeña positiva llamada τ_0.

Dórigo asume que el número m de hormigas son constantes todo el tiempo. Si se asume que son muchas, se puede reforzar la creación de sub-óptimos caminos y converger rápidamente a malas soluciones. Por otro lado si se asume un número reducido de hormigas, el efecto de cooperación no existiría ya que la evaporación de feromona ocurriría sin poder ser reforzada. Por ello, Dórigo propone que el número de hormigas sea igual al número de ciudades $m = n$; lo cual dará una buena compensación y por tanto buenos resultados [53]. Finalmente el algoritmo Básico de Sistema de Hormigas que resuelve el problema del agente viajero es representado por el Algoritmo 4.1.

Un concepto que es importante tomar en cuenta, es el de hormiga elitista (viene del concepto de algoritmos genéticos) [54]. Las hormigas elitistas son las que refuerzan las aristas que construyen el camino (tour) más corto (T^{+}) con una cantidad Q/L^{+} donde L^{+} es la longitud de T^{+}. Al introducir las hormigas elitistas surge el Algoritmo 4.2.

1. Inicializar: Fijar t=0 {t es el contador de tiempo}. Fijar NC=0 {NC es el contador de ciclos}.
   Para cada arista (i,j) fijar un valor inicial $\tau_{ij}(t)=c$ para el rastro de intensidad y $\Delta\tau_{ij}(t)=0$. Colocar m hormigas, una en cada uno de los n nodos.
2. Fijar $s=1$ {s es el índice de la lista Taboo}.
   Para k =1 hasta m, hacer: colocar la ciudad de inicio de la k-ésima hormiga en Taboo_k.
3. Repetir hasta que la lista Taboo esté llena. Hacer $s=s+1$
   Para k =1 hasta m elegir una ciudad j y moverse hasta ella con una probabilidad $p_{ij}^k(t)$.
   Mover la k-ésima hormiga a la ciudad j.
   Insertar la ciudad j en $\text{Taboo}_k[s]$.
4. Para k =1 hasta m hacer:
   Mover la k-ésima de $\text{Taboo}_k[n]$ a $\text{Taboo}_k[1]$.
   Calcular la longitud L_k de la ruta descrita por la k-ésima hormiga.
   Actualizar el camino más corto encontrado.
   Calcular la cantidad de feromona en la arista (i,j) para la k-ésima hormiga con $\Delta\tau_{ij}^k(t)$ ver la Ec. 4.5.
5. Para cada arista (i,j) calcular la Ec. 4.4.
   t = t + n ; NC = NC+1; para cada arista $(i,j):\Delta\tau_{ij}(t)=0$.
6. Si (NC < NCmax) y (se encuentra un camino)
   Limpiar todas las listas Taboo e ir al paso 2.
   Imprimir el camino más corto.
   Terminar ejecución.

Algoritmo 4. 1 Sistema de Hormigas.

Para cada iteración, las hormigas elitistas e se juntan con las otras que encontraron el camino T^+ dando un extra reforzamiento dado por $e\cdot Q/L^+$. Esto con el fin de que la feromona en T^+ se refuerce, dirigiendo la búsqueda de las demás hormigas con una probabilidad de tener cada una en su tour aristas del mejor camino, lo que hará que se converja al camino más corto.

En este algoritmo la cantidad de feromona que se deposita en cada una de las aristas, es la contribución de cada una de las *m* hormigas en la n-ésima arista; y viene dada por:

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{4.6}$$

$$\Delta\tau_{ij}^{e}(t)=\begin{cases}Q/L^{+} & \text{si } (i,j)\in T^{+}\\ 0 & \text{de otra forma}\end{cases} \tag{4.8}$$

1. Inicializar: Fijar t=0 {t es el contador de tiempo}. Fijar NC=0 {NC es el contador de ciclos}.
   Para cada arista (i,j) fijar un valor inicial $\tau_{ij}(t)=c$ para el rastro de intensidad y $\Delta\tau_{ij}(t)=0$. Colocar m hormigas, una en cada uno de los n nodos.
2. Fijar $s=1$ {s es el índice de la lista Taboo}.
   Para k =1 hasta m, hacer: colocar la ciudad de inicio de la k-ésima hormiga en Taboo_k.
3. Repetir hasta que la lista Taboo esté llena. Hacer $s=s+1$
   Para k =1 hasta m elegir una ciudad j y moverse hasta ella con una probabilidad $p_{ij}^{k}(t)$.
   Mover la k-ésima hormiga a la ciudad j,
   Insertar la ciudad j en $\text{Taboo}_k[s]$.
4. Para k =1 hasta m hacer:
   Mover la k-ésima de $\text{Taboo}_k[n]$ a $\text{Taboo}_k[1]$.
   Calcular la longitud L_k de la ruta descrito por la k-ésima hormiga.
   Actualizar el camino más corto encontrado.
   Calcular la cantidad de feromona en la arista (i,j) para la k-ésima hormiga con $\Delta\tau_{ij}^{k}(t)$ ver la Ec. 4.7.
5. Para cada arista (i,j) calcular con
   $\tau_{ij}(t)\leftarrow(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t)+e\cdot\Delta\tau_{ij}^{e}(t)$. (ver Ecs. 4.5, 4.6 y 4.8).
   t = t + n ; NC = NC+1; para cada arista $(i,j):\Delta\tau_{ij}(t)=0$.
6. Si (NC < NCmax) y (se encuentra un camino)
   Limpiar todas las listas Taboo e ir al paso 2.
   Imprimir el camino más corto.

Algoritmo 4. 2 Sistema de Hormigas al utilizar hormigas elitistas.

4.1.2 Sistema de Colonia de Hormigas

El algoritmo del Sistema de Colonia de Hormigas (ACS por sus siglas en inglés; ver Algoritmo 4.3) es una mejora del algoritmo de Sistema de Hormigas (AS); las mejoras son las siguientes [52]:

- Una regla de transición diferente: La regla de transición es modificada para permitir la exploración. Una hormiga k en una ciudad i elige la ciudad j, moviéndose con la siguiente regla:

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left\{ \left[\tau_{iu}(t) \right] \cdot \left[\eta_{iu} \right]^{\beta} \right\} & \text{si } q \le q_0 \\ J & \text{si } q > q_0 \end{cases} \tag{4.6}$$

  donde q es una variable aleatoria uniformemente distribuida sobre [0,1], q_0 es un parámetro ajustable en el rango $(0 \le q_0 \le 1)$ y $J \in J_i^k$ es una ciudad seleccionada en base a la regla de probabilidad:

$$p_{iJ}^k(t) = \frac{\left[\tau_{iJ}(t) \right] \cdot \left[\eta_{iJ} \right]^{\beta}}{\sum_{l \in J_i^k} \left[\tau_{il}(t) \right] \cdot \left[\eta_{il} \right]^{\beta}} \tag{4.7}$$

  La regla de transición del Sistema de Colonia de Hormigas es idéntica al del Sistema de Hormigas cuando $q > q_0$, y es diferente cuando $q \le q_0$. Esto quiere decir que $q \le q_0$ corresponde a una explotación del conocimiento disponible del problema, *i.e.* es un conocimiento heurístico acerca de las distancias entre las ciudades y el conocimiento aprendido memorizado en forma de feromona, mientras que $q > q_0$ favorece más exploración.

- Una nueva regla de actualización de feromona: En el sistema de hormigas todas las hormigas depositan feromona después de completar sus rutas, pero en el Sistema por Colonia de Hormigas sólo la hormiga que genera el mejor camino puede actualizar las concentraciones de feromona en las aristas. Otra diferencia es que en el Sistema de Hormigas, la regla de actualización de feromona se aplica a todas las aristas. Mientras que en el Sistema por Colonia de Hormigas la regla de actualización global de feromona se aplica sólo a las aristas que pertenecen al mejor camino.

  La regla de actualización es la Siguiente:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t) \tag{4.8}$$

donde (i, j) son las aristas que pertenecen a T^+ (el mejor camino), ρ es un parámetro que representa la evaporación de la feromona y:

$$\Delta\tau_{ij}(t) = 1/L^+ \tag{4.9}$$

donde L^+ es la longitud de T^+. Dicha regla permite reforzar solamente el mejor tour por medio de una actualización global.

1. Inicializar: Fijar t=0 {t es el contador de tiempo}. Fijar NC=0 {NC es el contador de ciclos}.
   Para cada arista (i, j) fijar un valor inicial $\tau_{ij}(t) = c$ para el rastro de intensidad y $\Delta\tau_{ij}(t) = 0$. Colocar m hormigas, una en cada uno de los n nodos. *cl* es el número de ciudades en la lista candidata.
2. Para k =1 hasta m :
   Si existe al menos una $j \in$ a la lista candidata
   Elegir una ciudad $j \in J_i^k$ con la ecuación 4.6, con una probabilidad $p_{ij}^k(t)$.
   Elegir la ciudad más cercana $j \in J_i^k$.
   Aplicar la regla de actualización local Ec. 4.10.
3. Para k =1 hasta m
   Calcular la longitud $L^k(t)$ del tour $T^k(t)$ hecho por la k-ésima hormiga.
   Actualizar L^+ y T^+ .
4. Para cada arista $(i, j) \in T^+$
   Actualizar la feromona usando la Ec. 4.8.
5. Para cada arista (i, j)
   $\tau_{ij}(t+1) = \tau_{ij}(t)$.
   Imprimir el camino más corto T^+ así como su longitud L^+.
   Terminar ejecución.

Algoritmo 4. 3 Sistema de Colonia de Hormigas.

- Usa una actualización local de feromona a favor de la exploración: Mientras se lleva a cabo un tour, la k-ésima hormiga está en la ciudad i y selecciona la ciudad $j \in J_i^k$, la concentración de la feromona en (i, j) es actualizada mediante la siguiente regla:

$$\tau_{ij}(t) \leftarrow (1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\tau_0 \qquad (4.10)$$

El valor τ_0 tiene el valor inicial de cantidad de feromona.

- Usa una lista candidata para restringir la elección de las ciudades a visitar: La lista candidata no es más que una estructura de datos, donde se almacenan las ciudades que no han sido visitadas. Estas ciudades se encuentran ordenadas desde la ciudad más cercana a la que tiene una mayor distancia con respecto a la ciudad actual. La lista se actualiza constantemente.

Primeramente se examinan todas las ciudades de la lista, pero si todas las ciudades de la lista ya fueron visitadas, entonces se toman otras ciudades como alternativa, es decir, para el problema del agente viajero, la hormiga elige la siguiente ciudad a visitar en la lista, toma la más cercana utilizando las Ecs. 4.6 y 4.7; por el contrario, si todas las ciudades ya fueron visitadas, entonces se elige alguna de ellas, tomando en cuenta la distancia.

4.2 Algoritmo Genético (AG)

A través del tiempo el hombre se ha preguntado el origen de su especie. Hombres ilustres fueron apareciendo con teorías que poco a poco fueron evolucionando nuestra manera de pensar, sin embargo la verdad aún se esconde.

La evolución mediante la selección natural, es el concepto central de la obra de Charles Robert Darwin ("El origen de las especies" publicada en 1859 [55]), teoría acerca de la adaptación, la complejidad y la diversidad entre las criaturas vivientes de la Tierra.

Esta teoría en combinación con las teorías de la selección de August Weismann y la genética de Gregor Mendel crean el paradigma del Neo-Darwinismo, que establece que la vida en nuestro Planeta Tierra, se da a través de procesos estadísticos que actúan sobre y dentro de las poblaciones y especies [56]. Tales procesos son la reproducción, la mutación, la competencia y la selección.

La reproducción es una propiedad de todas las formas de vida en nuestro Planeta, sin ésta, entonces la existencia de la vida misma, no sería. Si un sistema se reproduce a si mismo continuamente y se encuentra en equilibrio, se verá afectado por una mutación [57]. El contar con una cantidad finita de espacio para albergar la vida en la Tierra garantiza la existencia de la competencia y la selección se vuelve la consecuencia natural del exceso de organismos que han llenado el espacio de recursos disponibles.

La evolución es, por lo tanto, el resultado de esos procesos estocásticos (probabilísticos) fundamentales que interactúan entre si en las poblaciones, generación tras generación [58].

La computación evolutiva se inspira en esos procesos biológicos simulándolos en una computadora. Dichos factores o procesos consisten en codificar el espacio de solución del problema, operaciones modificadoras, una técnica de selección, así como una función de aptitud. La aptitud de un individuo se define como la probabilidad de que éste viva para reproducirse o como una función del número de descendientes que éste tiene.

Existen tres paradigmas principales en la computación evolutiva llamadas Programación Evolutiva (propuesta por Lawrence J. Fogel) [59,60], Estrategias Evolutivas (propuesto por Ingo Rechenberg) [61] y los Algoritmos Genéticos (desarrollados por John H. Holland en 1960) [62,63].

Hollan advirtió que un estudio de la adaptación debería de reconocer que [64, 63]:

- La adaptación ocurre en un ambiente,
- La adaptación es un proceso poblacional,
- Los comportamientos individuales pueden representarse a través de programas,
- Pueden generarse nuevos comportamientos mediante variaciones aleatorias de los programas, y
- Las salidas de dos programas normalmente están relacionadas si sus estructuras están relacionadas.

Esto condujo a un Sistema Adaptativo General denominado Algoritmo Genético (AG); la Figura 4.2 muestra el proceso de un algoritmo genético básico.

El algoritmo genético enfatiza la importancia de la cruza sexual (operador principal) sobre el de la mutación (operador secundario), al usar selección probabilística. La representación tradicional es la binaria tal como se muestra en la Figura 4.2 [58].

4.2.1 Operadores Genéticos

Para adentrarnos a este tema, primeramente considérese los siguientes conceptos que se pueden asociar a la Figura 4.3, al tomar en cuenta que genéticamente el ser humano contiene cadenas de información y que las computadoras también contienen su semejante.

La cadena binaria recibe el nombre de “cromosoma”, a cada posición de la cadena se le denomina “gene” y el valor que toma dentro de esa posición se le denomina “alelo”; los siguientes puntos deben de tomarse en cuenta para poder aplicar el algoritmo genético:

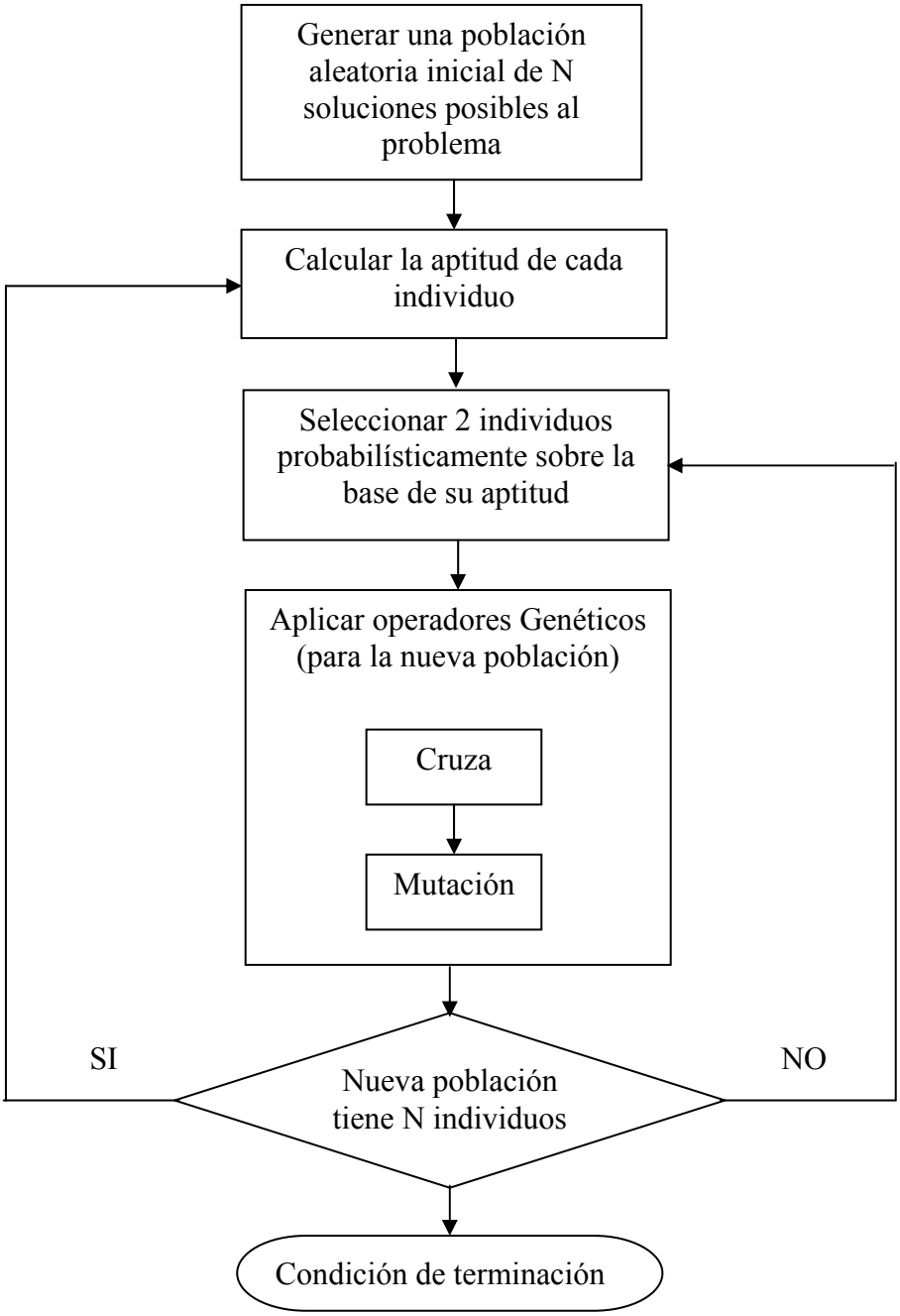


Figura 4. 2 Diagrama del proceso del Algoritmo Genético Básico.

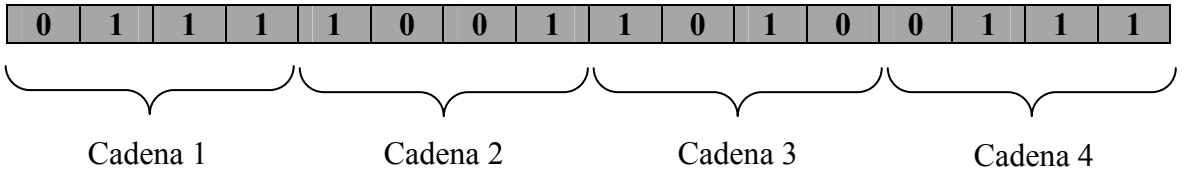


Figura 4. 3 Codificación del Algoritmo Genético.

1. Tener una representación de las soluciones potenciales del problema.
2. Contar con una forma de crear una población inicial de posibles soluciones (un proceso aleatorio).
3. Contar con una función de evaluación que juegue el papel de ambiente, clasificando las soluciones en términos de su "aptitud".
4. Contar con operadores genéticos que alteran la composición de los hijos que se producirán para las siguientes generaciones.
5. Contar con valores para los diferentes parámetros que utiliza el algoritmo genético (tamaño de la población, probabilidad y cruza, probabilidad de mutación número máximo de generaciones).

Nota: Se ha demostrado [66] que el Algoritmo Genético requiere de elitismo (retener intacto al mejor individuo de cada generación) para poder converger al óptimo.

4.2.2 Técnicas de Selección

Un algoritmo genético al igual que la naturaleza y el hombre, debe seleccionar a los individuos más calificados, mejor adaptados al medio, para que tengan mayor oportunidad de reproducción. De ésta manera se incrementa la probabilidad de tener individuos "buenos" en el futuro. La selección, explota el conocimiento que se ha obtenido hasta el momento, procurando elegir lo mejor que se haya encontrado, elevando así el nivel de adaptación de toda la población, por lo que la selección es el proceso mediante el cual algunos individuos en una población son seleccionados para reproducirse, típicamente en base a su aptitud.

Un aspecto a tomar en cuenta es que si sólo se hace la selección forzando que sea más probable elegir al mejor individuo de la población, pero sin asegurarlo, es posible que este individuo se pierda y no forme parte de la siguiente generación. Para evitar esta situación se fuerza a la selección de los mejores *n* individuos de la generación para pasar intactos en la siguiente. A esta estrategia se le denomina elitismo y puede ser generalizada especificando que permanezcan en la población los *n* mejores individuos de las pasadas *k* generaciones [67].

Existen muchas técnicas para la selección como lo son [58]:

- Selección proporcional
- Selección mediante torneo
- Selección de estado uniforme
- Brecha generacional, entre otras.

4.2.3 Técnicas de Cruza

En los sistemas biológicos, la cruza se da en los cromosomas, es decir el código genético de los padres de un individuo, los cuales se alinean, fraccionándose en ciertas partes y posteriormente intercambian fragmentos entre si, heredando sus características a los hijos. Si estas características les permitieron a los ancestros una alta aptitud de supervivencia, entonces el nuevo individuo también podrá sobrevivir.

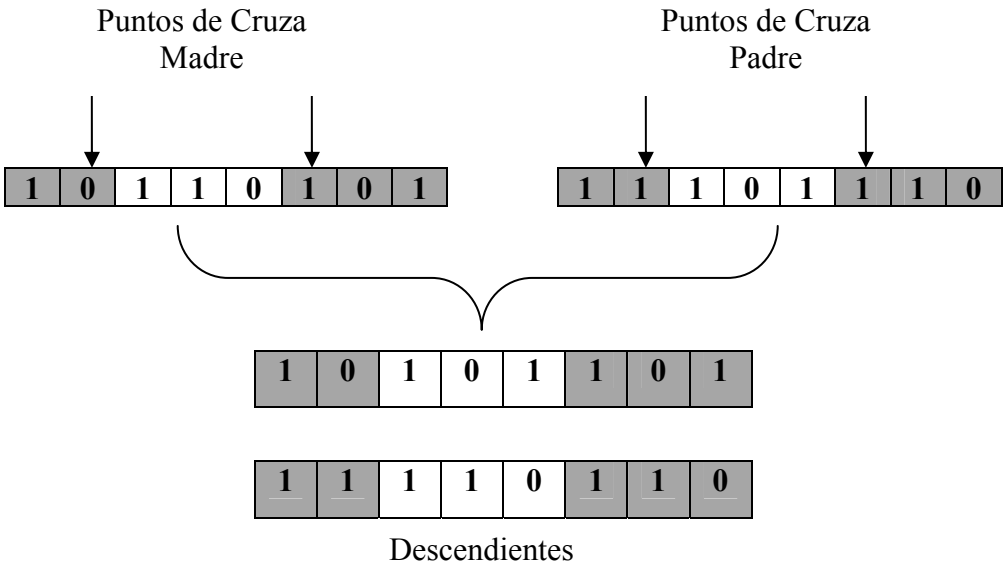


Figura 4. 4 Cruza de dos puntos.

En algoritmos genéticos, reproducirse significa que dado dos individuos seleccionados en función de su grado de adaptación, pasarán a formar parte de la siguiente generación o al menos mezclarán sus códigos genéticos para generar hijos [67].

La cruza se simula intercambiando segmentos de cadenas lineales de longitud fija.

Algunas técnicas para la Cruza son las siguientes [58]:

- Cruza de un punto
- Cruza de dos puntos
- Cruza Uniforme
- Cruza Acentuada

La técnica utilizada en la investigación es la de uno y dos puntos: De Jong [68] fue el primero en implementar una cruza de *n* puntos como una generalización de la cruza de un punto, en ella se tiene que la cruza de dos puntos (*n*=2, ver la Figura 4.4) es mejor que la de un punto, por lo cual es mas utilizada junto con la cruza uniforme [69].

La Figura 4.4 muestra cómo se lleva a cavo la cruza de dos puntos. Por medio de dicha cruza, se obtendrán dos descendientes, los cuales heredarán los valores de sus padres. Primero se obtienen los puntos de cruza o los puntos a partir de donde se heredará la información. El primer descendiente heredará los primeros dos y los tres últimos genes del cromosoma de la madre, y los valores restantes serán ocupados por los genes centrales del padre. El segundo descendiente heredará los genes del cromosoma del padre donde se realizó el corte (los primeros dos y los últimos tres genes), y los genes restantes serán tomados de la parte central del cromosoma de la madre.

4.2.4 Técnicas de Mutación

En el ámbito biológico la mutación es una alteración accidental en el código genético de los seres vivos.

En los algoritmos genéticos para emular la mutación, el código de ciertos individuos se altera a propósito; se seleccionan aleatoriamente y se modifica el valor del alelo.

El objetivo es generar nuevos individuos, que exploren regiones del dominio del problema que probablemente no se han visitado aún. Esta exploración se realiza en forma aleatoria, con el objetivo de buscar nuevas soluciones posibles que quizá, superen las encontradas hasta el momento [67].

Algunas técnicas de la mutación son las siguientes [58]:

- Mutación por Inserción
- Mutación por Desplazamiento
- Mutación por Intercambio Recíproco
- Mutación Heurística

4.2 Resumen

En éste capítulo se describió la teoría de los algoritmos Bio-inspirados, el comportamiento de una colonia de hormigas y los algoritmos que surgen a partir de la inspiración de las mismas. Los algoritmos basados en colonias de hormigas se explicaron detalladamente, así como su base matemática.

También se describió una heurística y su algoritmo básico apoyado en la evolución y selección natural, la cuál recibe el nombre de algoritmo genético. En el siguiente capítulo se describe la metodología propuesta para resolver la planificación de rutas mediante las técnicas de algoritmos Bio-inspirados.

CAPÍTULO 5

METODOLOGÍA PROPUESTA

El mundo en el que vivimos no se encuentra en un estado favorable para cada situación que se da en ella, por ello algunos problemas se resuelven en condiciones iniciales ideales para poder darles una solución e interpretación; sin embargo lo ideal no existe.

Al tratar de resolver los hechos tan complejos que nos rodean día a día, con ayuda de las Ciencias Computacionales, hoy se buscan sistemas, técnicas que resuelvan familias de problemas. Esto tampoco es posible ya que no hay sistemas o técnicas generales para dar soluciones. Hasta ahora, se tienen diferentes técnicas que puedan resolver diferentes problemas con ciertas características en común y condiciones semejantes, pero con sus propias limitantes.

En este capítulo se describirá la metodología propuesta para resolver el problema de planificación de rutas.

5.1 Surgimiento de la propuesta

A través de los tiempos el hombre ha deseado ser reemplazado en algunas de sus tareas, que resultan pesadas, difíciles de realizar por el grado de precisión, tareas complejas y tardadas, que podrían en riesgo la vida de un ser humano. Con ello surgieron los robots, herramientas que pueden ejecutar varias tareas según sea su tipo; esto no significa que el robot pueda tomar sus propias decisiones, ya que se requiere de la programación de tareas específicas, movimientos que deben de ejecutar para llevar a cabo un trabajo, instrucciones que los hará útiles, y que con el paso del tiempo dichas instrucciones y robots se han vuelto más complejos según las necesidades del ser humano (ver Capítulo 3).

Los robots móviles han sido una parte importante en la robótica ya que pueden ir a donde el ser humano aún no ha pisado, un ejemplo de ello son los robots enviados a planetas lejanos, robots que necesitan moverse dentro de un entorno yendo de un punto

específico a otro ya sea por diversión, por trabajos que ayudan al ser humano moralmente o en situaciones bélicas. Por ello uno de los problemas más importantes de la robótica es la planificación de rutas, donde se tiene que dirigir al robot, evitando que se dañe, que destruya el medio donde se encuentra y que ejecute con presición y rapidez su tarea, guiándose en un entorno desconocido. Para ello en los últimos años se han desarrollado diferentes técnicas que nos ayudan a optimizar las trayectorias de un robot móvil (ver Capítulo 2).

Como se mencionó en el Capítulo 4, la Optimización por Colonia de Hormigas se desarrolló para resolver el problema del agente viajero, cuya representación se puede llevar al problema de la planificación de rutas, (recuérdese las características de dicho problema y nuestro problema a resolver descrito en el Capítulo 1).

El problema del agente viajero es un problema de optimización que al querer encontrar sobre un conjunto de *n* ciudades una longitud mínima desde un punto de inicio hasta el mismo. Esto significa que se estaría encontrando una trayectoria cerrada de longitud mínima (para mayor detalle ver Capítulo 4). Este algoritmo básico no puede dar solución a nuestro problema, ya que se necesita encontrar el camino más corto de un punto de origen a uno destino (que no sea el mismo origen) de entre otros posibles trayectos en un entorno.

Por ello se modificó el algoritmo básico que resolviera el problema de planificación de rutas (ACO), para llevarlo a otros tipos de entornos, donde las trayectorias no fuesen cerradas, así también, se mezcló el algoritmo de Optimización por Colonia de Hormigas con Algoritmos Genéticos, para un mejor desempeño al evolucionar algunos parámetros del algoritmo de hormigas.

5.2 Descripción General de la Metodología

La metodología propuesta es una modificación del algoritmo básico de Optimización por Colonia de Hormigas (ACO), con el fin de aplicarlo a la planificación de rutas y así poder encontrar la ruta más corta que pueda seguir un robot móvil de un punto de origen a un punto destino en un entorno dado.

La Figura 5.1 muestra los pasos generales de la metodología.

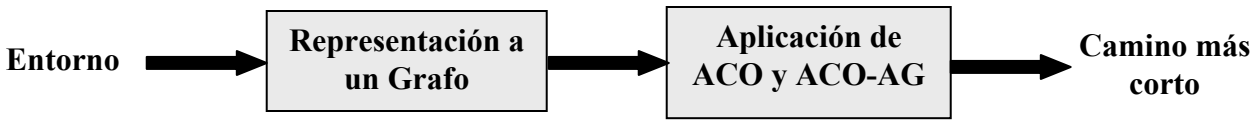


Figura 5. 1 Diagrama General de la Metodología propuesta.

Primeramente se obtiene el entorno real donde se pretende encontrar la solución dada por medio de una imagen, después se necesita obtener su símil, representado con un grafo, cuyos nodos serán los puntos a visitar o las posiciones a donde se desea ir. Las aristas representan los posibles caminos para llegar hasta ellos. Al obtener el grafo, se selecciona el nodo de inicio y el nodo final (objetivo) para que se puedan aplicar los algoritmos propuestos y encuentren la solución (el camino más corto). A continuación, se describen en más detalle los pasos de la Metodología propuesta.

5.3 Construcción del grafo a partir de un entorno

Para aplicar los algoritmos propuestos y los básicos, se requiere conocer de alguna manera el entorno por donde se desplazará el robot. Por medio de dispositivos de captación de imágenes se puede obtener el escenario o el área donde el robot se desplazará. Se le asociará un grafo que se obtendrá de operaciones básicas de morfología.

No se utilizaron métodos más robustos para encontrar el grafo correspondiente a un entorno ya que no es de vital importancia su método de obtención, sólo basta tener los grafos para poder ejecutar en ellos los algoritmos propuestos.

1. Aplicar un umbral [72] a la imagen del entorno seleccionado (Figura 5.2(a)).
2. Obtener el esqueleto morfológico de la imagen binaria.
3. Obtener el máximo grosor de la erosión de la imagen binaria.
4. Aplicar una operación de sustracción entre la imagen del paso 2 y el paso 3.
5. Aplicar una dilatación a la imagen obtenida en el paso 4 para obtener el vértice del grafo.
6. Aplicar una operación OR entre la imagen del paso 5 y el paso 6 para obtener las aristas del grafo.
7. Aplicar una operación XOR entre la imagen del paso 5 y del paso 6 para obtener las aristas del grafo.
8. Para la imagen del paso 5, aplicar etiquetado por componentes conectadas [73] y a esas componentes conectadas calcular su centro de masa. Con ello obtendremos la tabla con los vértices etiquetados ver Tabla 5.1.
9. Para la imagen del paso 7, aplicar etiquetado por componentes conectadas [73] y esas componentes deben de ser asociadas con la etiqueta de sus respectivos vértices. Esta asociación es calculada usando la mínima distancia Euclidiana del extremo de las aristas a los vértices.
10. Finalmente, obtenemos las tablas de las aristas en términos de las etiquetas de sus vértices, ver tabla 5.2.

Algoritmo 5. 1 Algoritmo para obtener un grafo representativo a partir de un entorno.

El algoritmo propuesto (Algoritmo 5.1) [70] se basa en operaciones morfológicas [71], las cuales no serán explicadas a detalle ya que éste no es el propósito de la investigación, sin embargo se presentan los resultados en imágenes, de los pasos que se siguieron para la obtención de grafos.

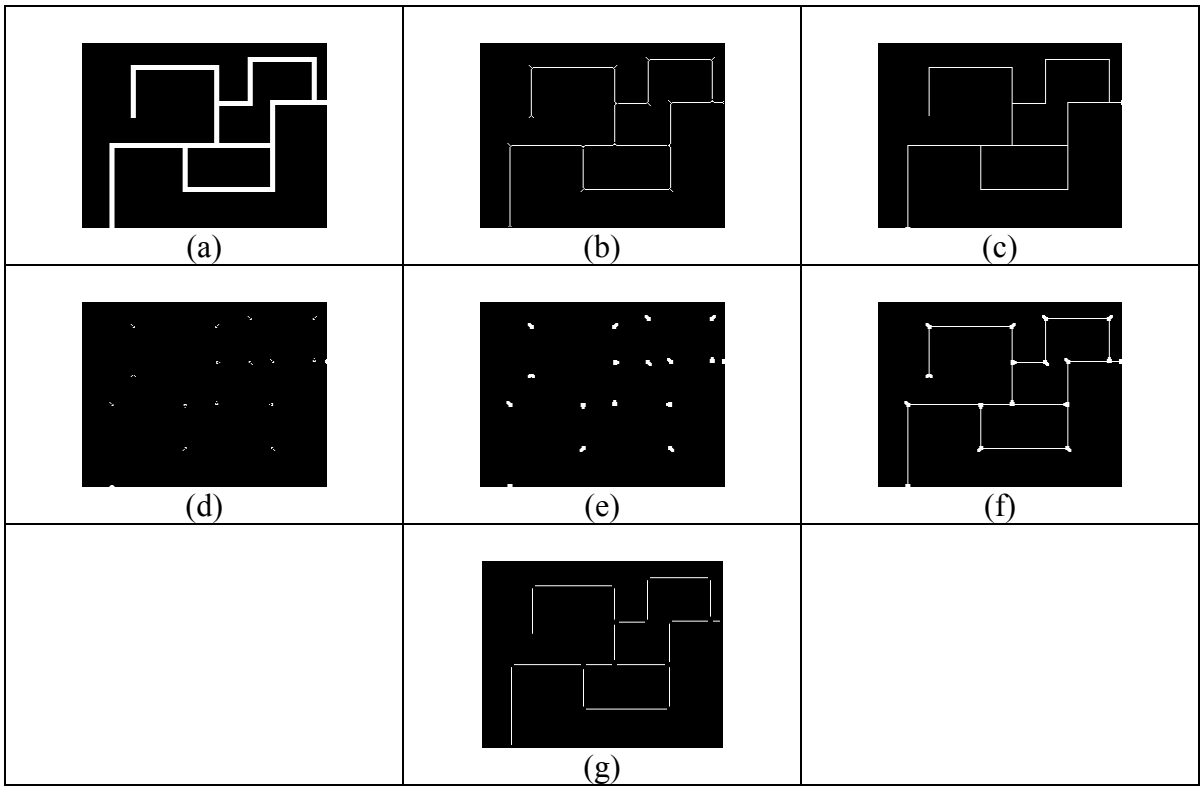


Figura 5. 2 Obtención de un grafo. (a) Imagen binaria. (b) Esqueleto morfológico de la imagen a. (c) Grosor máximo de erosión para la imagen a. (d) Operación de sustracción entre las imágenes b y c. (e) Vértices del grafo obtenidos de la dilatación de la imagen d. (f) Operación OR entre las imágenes c y e. (g) Aristas obtenidas del grafo al aplicar la operación XOR entre las imágenes e y f.

Primeramente se puede observar en la Figura 5.2(a) un laberinto, el cual representa el entorno a recorrer. Esta imagen debe ser binaria para poder obtener de ella el esqueleto morfológico [72] (Figura 5.2 (b)), es decir reducir completamente la región de interés sin perder información.

A continuación se realiza una erosión [72] sin perder información de la imagen original Figura 5.2(c). Después se lleva a cabo una sustracción entre las imágenes b y c, obteniendo así los píxeles representativos de los nodos Figura 5.2(d). Debido a que no se distinguen dichos píxeles se dilatarán [72]. Esto nos ayudará a visualizarlos (Figura 5.2(e)) con mayor nitidez. Estos píxeles son los vértices de nuestro grafo.

Posteriormente se realiza una operación OR entre las imágenes (c) y (e), ver Figura 5.2 (e). Después se aplica una operación XOR entre las imágenes (e) y (f), obteniendo así las aristas que conectan los nodos.

Finalmente se obtiene un grafo con vértices $V = \{C_1, C_2, ..., C_n\}$ y con aristas $A = \{(i, j) : i, j \in V\}$.

En la Tabla 5.1, se puede apreciar el número de vértices y la posición en (x, y) que le corresponde a cada uno. En la Tabla 5.2 se aprecia el conjunto de vértices que están unidos por una cierta arista.

Vértice	Posición (x,y)	Vértice	Posición (x,y)
0	(164,15)	9	(50,73)
1	(228,15)	10	(132,100)
2	(49,23)	11	(28,101)
3	(132,23)	12	(185,102)
4	(228,57)	13	(101,103)
5	(186,58)	14	(100,146)
6	(239,59)	15	(187,146)
7	(133,60)	16	(29,183)
8	(165,60)		

Tabla 5. 1 Vértices de la Figura 5.2(e).

Arista	(V,V)	Arista	(V,V)
0	(0,1)	9	(5,12)
1	(0,8)	10	(7,10)
2	(1,4)	11	(11,13)
3	(2,3)	12	(13,10)
4	(2,9)	13	(10,12)
5	(3,7)	14	(11,16)
6	(5,6)	15	(12,15)
7	(3,6)	16	(13,14)
8	(7,8)	17	(14,15)

Tabla 5. 2 Aristas de la Figura 5.2(g).

Una vez que se tienen los grafos representativos de los entornos, se ejecuta uno a uno, los algoritmos con sus diferentes configuraciones sobre dichos entornos.

Una nota importante es que en ésta investigación no se enfoca a la implementación de la mejor técnica para obtener un grafo del entorno dado. El método que se propone es sencillo y con él se obtiene una buena representación del entorno, para después aplicar los algoritmos propuestos. Si se desea obtener el grafo de manera óptima, una técnica morfológica podría ser la clave para resolverlo, tal como la Transformada de Distancia [74].

5.4 Modificación al Sistema de Colonia de Hormigas

Para resolver el problema de planificación de rutas (Capitulo 1) se hicieron modificaciones al algoritmo original ya que éste, como se sabe, permite resolver el problema del agente viajero, el cuál posee características diferentes a las esenciales de nuestro problema. El algoritmo propuesto se muestra en el Algoritmo 5.2. Las modificaciones mencionadas anteriormente se encuentran subrayadas en el mismo.

Dichas modificaciones son:

1. Que todas las hormigas comienzan su exploración desde el mismo nodo: en el algoritmo básico de hormigas que resuelve el problema del agente viajero (capítulo 4), teníamos que por cada nodo había una hormiga y éstas recorrían el grafo encontrando un tour cada una, regresando a sus respectivos nodos de origen. En el caso del algoritmo propuesto, todas las hormigas creadas son colocadas en un mismo nodo. Este nodo es designado como el nodo de origen. De esta manera las hormigas cumplirán con el criterio de moverse de un punto de origen a uno destino para encontrar un camino óptimo.
2. Se crean hormigas con una frecuencia f. Esto se hace para asegurar que la colonia de hormigas no tendrá un tamaño variable.
3. La actualización de la longitud se evalúa con un costo de longitud máximo de la ruta. Si la longitud actual supera la longitud máxima (significa que alguna ciudad fué visitada más de una vez), entonces la hormiga tiene que morir.
4. Se utiliza una regla de transición diferente a la propuesta por Dórigo (Ec. 4.3), la cual será explicada en el siguiente sub-tema.

La cantidad de feromona dada por τ_{ij}^{k}, puede tomar tanto valores positivos como negativos, donde su dominio son los números enteros $\mathbb{Z}$. A continuación, se explica brevemente, cómo la cantidad de feromona trabaja en el dominio establecido. Se sabe que entre mayor cantidad de feromona se tenga, ésta tardará más en evaporarse. En el caso de tener una cantidad de feromona positiva que se evapora hasta llegar a cero se podría decir que "no existe feromona" en ese caso, (ya que el cero representa ausencia de algo). Sin embargo para ésta investigación no es así, ya que no interesa la ausencia total de feromona, si no la cantidad de la misma y aquella que sea la mayor. Si tenemos una cantidad pequeña de feromona positiva, se evaporará rápidamente, entrando en el rango de los valores negativos. Aun así, dentro del rango $\mathbb{Z}^{-}$, existen números mayores que >.

Por otro lado, si se toman solamente el rango de los números naturales $\mathbb{N}$ y el cero para evaluar la cantidad de feromona, no se tendría un criterio de elección, ya que al evaporarse la cantidad de feromona en las aristas, en algún momento la intensidad de feromona tomaría el valor de cero y en este caso no habría una cantidad menor a esta referencia.

5.4.1 Regla de Transición

La regla de transición utilizada en el algoritmo del Sistema de Colonia de Hormigas propuesto por Dorigo, posee varias desventajas:

1. La regla tiene un factor que se determina a partir de la distancia que hay entre un nodo i y un nodo j, y se necesita conocer la distancia de las aristas que aún no hay sido visitadas (Ec. 4.3).

2. Las hormigas en el mundo real no saben de las distancias que hay en los caminos y mucho menos de los no han visitado previamente, estás sólo se guían por la feromona y por otros factores naturales; por lo que dicha regla se aleja del proceso biológico que siguen las hormigas y que quiere ser emulado en una computadora.

Con el propósito de reducir el efecto de estas desventajas, en esta investigación se propone una nueva regla de transición (Ec. 5.1), la cual permitirá explorar los caminos sin utilizar la distancia de las aristas; sólo se guiará por la llamada feromona y su instinto natural. La regla propuesta es la siguiente:

$$P_{ij}^{k}(t) = \alpha \cdot sen\left(\beta \cdot \tau_{ij}^{k}(t) + \gamma\right) \tag{5.1}$$

donde τ_{ij}^{k} es la intensidad de feromona en una arista, α, β y γ son parámetros $\in \mathbb{R}$ en el dominio [-5,5] loa cuales hacen que se tengan en la colonia, diferentes individuos u hormigas. Para entender mejor la importancia de la función de transición (Ec. 5.1), descompóngasele en sus partes más simples (los siguientes dos casos y la función propuesta Ec. 5.1). Primeramente recuérdese que los valores de la arista que maximice esta regla, determinará el camino a seguir.

Caso 1: Supóngase que la función de transición a utilizar es la dada en la Ec. 5.2.

$$P_{ij}^{k}(t) = \tau_{ij}^{k} \tag{5.2}$$

Como ya se mencionó, dicha función significa la intensidad de feromona en una arista. Para elegir un camino sobre la base de la Ec. 5.2, se tomará en cuenta la arista con la mayor cantidad de feromona, esto es, donde la función de transición adquiera un valor máximo. Las aristas con valores más negativos, serán ignoradas para la elección, ocasionando una exploración insuficiente y con ello la posibilidad de encontrar el camino

óptimo, y las aristas con valores que son siempre positivos y con valores grandes serán elegidos siempre, probablemente tomando soluciones sub-óptimas o no encontrar alguna solución.

Caso 2: Supóngase que la función de transición a evaluar viene dada por la Ec. 5.3.

$$P_{ij}^{k}(t)=\left(\beta\cdot\tau_{ij}^{k}(t)+\gamma\right) \tag{5.3}$$

Para que una hormiga elija un camino sobre la base de la Ec. 5.3, se tomará en cuenta la intensidad de feromona en las aristas, dada por τ_{ij}^{k} la cuál será operada por los gametos β y γ, de la hormiga k (ver sección 5.5). La arista que maximice la función de transición será el camino a tomar. Si los parámetros β y γ toman signos negativos, el valor de la función de transición P_{ij}^{k} será un factor para no elegir el camino correspondiente (i,j), ya que entre más negativo sean los parámetros, el resultado de la función de transición tenderá a ser negativo (ver Figura 5.3). Lo mismo pasaría si sólo uno de los parámetros tuviera signo negativo y fuese muy grande. Por otro lado si ambos parámetros tuviesen signo positivo la posibilidad de elegir el correspondiente camino sería muy alta, ya que el valor de la función de transición tendería a ser un número grande y positivo. Nuevamente la desventaja sería que no habría una buena exploración del entorno, debido a la función de transición elegida.

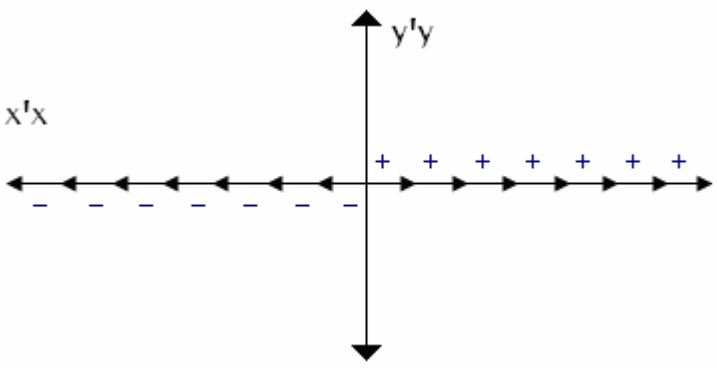


Figura 5. 3 Tendencia del valor de la función de transición (Ec. 5.3).

Caso 3: La función de transición propuesta en esta investigación ver Ec. 5.1, utiliza la función seno (Figura 5.4), la cuál nos permite tener una mayor posibilidad para escoger una arista que nos pueda llevar de la ciudad i a la ciudad j, a pesar del signo que acompaña a los operandos; es decir se puede explorar ampliamente, lo que evitará caer en una solución sub-óptima. La función seno tiene un dominio en los números reales, con un rango que va de [-1,1], es periódica en 2π y simétrica con respecto al origen; además de que es una función impar y se intersecta con el origen en el $y'y$. En el eje de las ordenadas, cuando x toma los valores $\pi/2, 3\pi/2, 5\pi/2, 7\pi/2, ..., etc$, el valor de la función

seno es: $1,-1,1,-1,1,...,etc$, por otro lado la función seno se intersecta en el $x'x$ cuando toma los valores $\pi,2\pi,3\pi,4\pi,...,etc$.

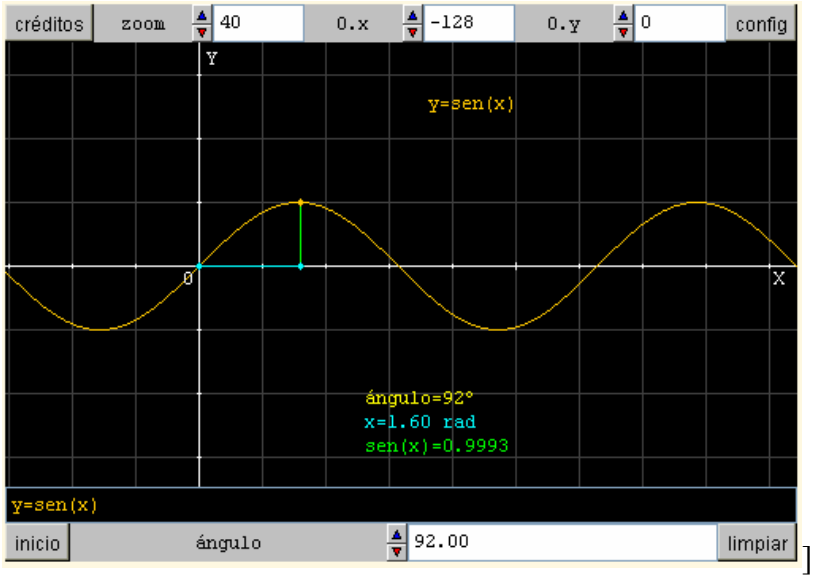


Figura 5. 4 Función seno.

Dadas las características anteriores analicemos la función de transición dada en la Ec. 5.1 y la Figura 5.4.Ya que cada hormiga consta de un cromosoma con tres gametos (parámetros) el cuál determina su comportamiento (ver siguiente sección 5.5), si el argumento del seno es operado con valores negativos, la función seno podrá obtener un valor negativo o positivo. Por el contrario si el argumento del seno es operado con valores positivos, la función seno podría adoptar cualquiera de los dos signos (ver la Figura 5.5). Esto se debe a que la función seno es muy variable, al ser periódica y simétrica como ya hemos mencionado.

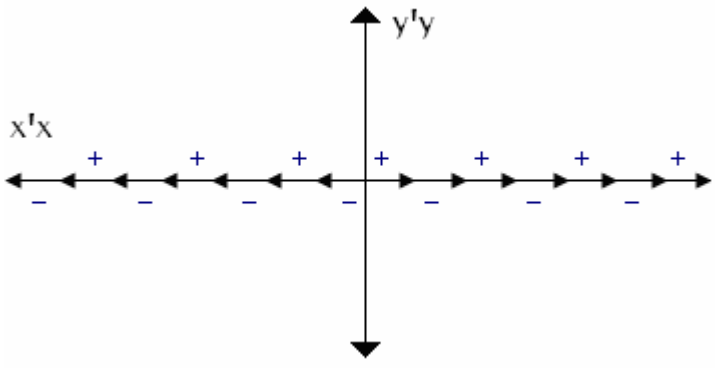


Figura 5. 5 Tendencia del valor de la función de transición propuesta (Ec. 5.1).

La función seno en la Ec. 5.1, es operada finalmente por el parámetro α, que al tener un signo aleatorio, hace que el resultado de la función seno tenga todavía más posibilidades de cambiar. Esto permite que el resultado de la función de transición (Ec. 5.1) sea más flexible al elegir los posibles caminos, sin importar la intensidad de feromona depositada en ellos. Gracias a los tres parámetros pertenecientes a cada una de las hormigas y la función seno se explorarán todos los posibles caminos que componen el entorno, haciendo que la posibilidad de encontrar la solución óptima sea cada vez más grande y exitosa.

1. Inicializar: Fijar $t := 0$ {t es el contador de tiempo}.
   Fijar $NC := 0$ {NC es el contador de ciclos}
   Para cada arista (i, j) fijar un valor inicial $\tau_{ij}(t) = c$ para la intensidad de feromona y $\Delta\tau_{ij}(t) = 0$.
   Colocar las m hormigas en el nodo de inicio.
   Crear nuevas hormigas con una frecuencia f.
2. Fijar $s := 1$ {s es el índice de la lista Taboo}
   Para $k := 1$ a m hacer:
   Colocar el nodo de inicio de la k-ésima hormiga en $taboo_k(s)$.
3. Repetir hasta que el nodo destino se halla encontrado:
   Fijar $s := s + 1$
   Para $k := 1$ a m hacer
   Elegir la ciudad j a visitar con una regla de transición $p_{ij}^k(t)$ dada por la Ec. 5.1.
   Mover la k-ésima hormiga a la ciudad o nodo j.
   Actualizar la longitud de de la ruta con: $cl^k = cl^k + d_{ij}$
   Si ($cl > cl_{\max}$) matar la k-ésima hormiga,
   Else inserta la ciudad j en $taboo_k(s)$.
4. Para $k := 1$ a m hacer:
   Mover la k-ésima hormiga de $taboo_k(n)$ hasta $taboo_k(1)$.
   Calcular la longitud de L^k del tour hecho por la k-ésima hormiga.
   Actualizar el camino más corto encontrado.
   Para cada arista (i, j) aplicar la Ec. 4.6.
5. Para cada arista (i, j) aplicar la Ec. 4.4.
   Fijar $t := t + n$ y $NC := NC + 1$.
   Para cada arista (i, j) hacer $\Delta\tau_{ij}(t) = 0$
6. Si (NC < NCMAX) y (se encuentra un camino) Limpiar todas las listas Taboo.
   Si no ir al paso 2.

Algoritmo 5. 2 Modificación del Sistema de Colonia de Hormigas.

Como ya se mencionó, la función seno es periódica, en todo el dominio real. Debido a ésta característica, se puede tener en la regla de transición valores positivos como negativos. Si se cambia la función seno por un coseno, de igual manera se tendría esa variación de signos, ya que también es periódica y cuenta con el mismo rango que la función seno.

Por el contrario, utilizando una función definida por intervalos y no periódica, como lo es la función signo o la función de escalón unitario entre otras, se caerá en los primeros dos casos de la regla de transición propuesta en esta investigación.

Ya que se requiere una exploración amplia del entorno, la regla de transición debe arrojar valores positivos y negativos. Esto se obtiene al utilizar una función periódica; en este caso se utilizó la función seno.

5.4.2 Algoritmo Propuesto (ACO)

El Algoritmo 5.2 es la implementación el algoritmo modificado de ACO, como ya se mencionó anteriormente; las modificaciones se encuentran subrayadas y explicadas en el sub-tema anterior.

Las cuatro características principales del algoritmo propuesto son esenciales para que el algoritmo de Sistema por Colonia de Hormigas pueda ser aplicado a la optimización de planificación de rutas para un robot móvil. Como ya vimos, el algoritmo original fue planeado para planificar rutas cerradas.

5.5 ACO evolucionado con Algoritmos Genéticos

Como ya vimos, la regla de transición Ec. 4.3 depende de los parámetros α y β. Estos parámetros ayudan a seleccionar el tour óptimo. Si estos parámetros no se ajustan bien, se puede converger a una solución sub-óptima o no obtener solución alguna; ya que como ya vimos se puede tomar en cuenta sólo la cantidad de feromona en una arista y olvidar la distancia que tiene ésta y viceversa. Esto puede llevar a que el algoritmo se incline por una solución incorrecta.

Para evitar el problema de "adivinar" los valores que le corresponden a α y β se decidió utilizar otra heurística que ayude a optimizar dichas variables. Debido a que es muy incierto el ajustar manualmente los parámetros, se decidió evolucionar, el algoritmo de Optimización por Colonia de Hormigas, al aplicar un algoritmo genético que permita obtener dichos parámetros, los cuales serán los más adecuados para encontrar un resultado.

Y ¿cómo nos ayuda un algoritmo genético en el algoritmo de ACO?

Al evolucionar ACO se estarán generando nuevas hormigas que sustituirán a las que no ayuden en la búsqueda de la solución; las hormigas que se generen tienen que ser mejores que las que se tenían. Esto se hace mediante los operadores de la selección natural. Por lo tanto se realiza una selección de los mejores individuos, se lleva a cavo una cruza entre ellos y se muten sus cadenas genéticas para encontrar nuevas soluciones.

Las mejores hormigas son aquellas que tienen un valor muy alto de aptitud en comparación a las demás y se dividen en dos tipos:

- Las exploradoras: son aquellas que encuentran el nodo destino y colectan una gran cantidad de alimento, mientras que
- Las trabajadoras: son aquellas que buscan caminos alternativos para llegar al nodo destino, no acarrean comida durante la búsqueda y no pasan más de una vez por un nodo.

Las hormigas que estén perdidas durante la búsqueda de la solución, (las que han pasado más de cinco por un nodo), o se encuentren atrapadas en un circuito en el grafo, serán eliminadas (extintas) y reemplazadas por las nuevas descendientes, las cuales serán mejores que sus padres, garantizando una convergencia a la solución.

5.5.1 Implementación de Operadores genéticos

La selección natural es implementada como sigue:

1. Se seleccionan cuatro miembros de la población (los mejores dos exploradores y los mejores dos trabajadores).
2. Se aplica la cruza entre parejas.
3. Se aplica la mutación con una probabilidad igual al porcentaje de mutación.
4. Se realiza una extinción de los individuos que no benefician a la colonia o población.

5.5.1.1 Selección

La selección consiste en tomar a individuos (los más aptos) de una población para reproducirse. Estos heredarán sus características a sus hijos o descendientes, creando así una población más fuerte.

Para saber el grado de qué tan aptas son las hormigas trabajadoras se debe de saber por medio de un contador que tanta comida han recolectado. Al ser alta dicha cantidad serán las mejores hormigas trabajadoras y para las exploradoras el qué tanto han pasado por la misma ruta significaría una aptitud baja (lo cual indica que son ineficientes; para este caso el pasar más de cinco veces por un mismo nodo).

5.5.1.2 Cruza

Se sabe que al reproducirse dos individuos se generaran descendientes. En este caso se generarán dos: uno de la cruza de los mejores exploradores y otro descendiente de los mejores trabajadores. Al combinarse los cromosomas de los padres se obtendrá un nuevo cromosoma, el cual representa al nuevo individuo. El cromosoma de cada hormiga viene representado por la Figura 5.6.

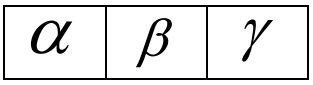


Figura 5. 6 Cromosoma.

Para entender mejor en que consiste la cruza véase la Figura 5.7. De esta Figura se puede ver que se obtienen 4 cromosomas pertenecientes a los 2 individuos más aptos de los exploradores y dos cromosomas pertenecientes a los mejores trabajadores. De cada pareja se obtiene un hijo. Estos hijos heredan las características o genes de sus respectivos padres, ayudando así a tener una colonia con los individuos más aptos para sobrevivir.

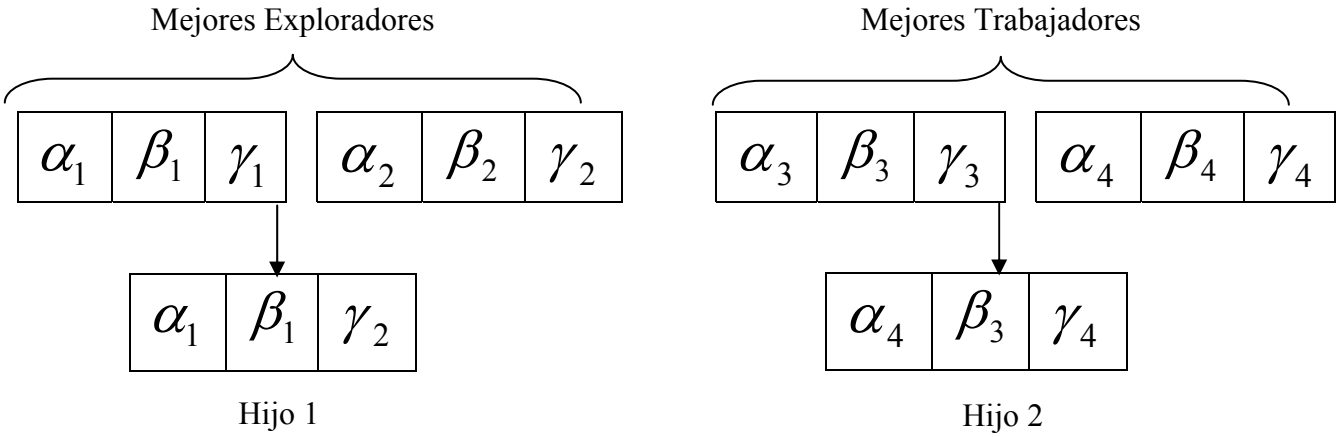


Figura 5. 7 Cruza entre los exploradores y trabajadores más aptos.

Para la implementación del Algoritmo Genético en el algoritmo de Sistema por Colonia de Hormigas, la cruza se genera aleatoriamente, es decir el hijo recibe algún alelo de uno de sus padres.

La Figura 5.8 muestra como el Hijo hereda los genes de sus padres al realizarse la cruza. El hijo cuenta con un cromosoma del mismo tamaño que el cromosoma de los padres. Los cromosomas están compuestos de un gen α, de un gen β y un gen γ.

Supóngase que se tiene una máscara del tamaño de un cromosoma , la cual contiene una cadena binaria. En este cromosoma el valor de 0 corresponde a los genes de la madre y el valor 1 corresponde a los genes del padre.

Como primer paso, los valores α, β y γ de la máscara, son generados aleatoriamente. Esta máscara ayuda a crear el cromosoma correspondiente al hijo. Si el valor del gen α de la máscara es un 1, entonces se copia en el gen α del hijo, el valor del gen α del padre. Si el valor del gen α de la máscara es un 0 se copiará el gen de la madre. Esta regla será aplicada también a los genes β y γ.

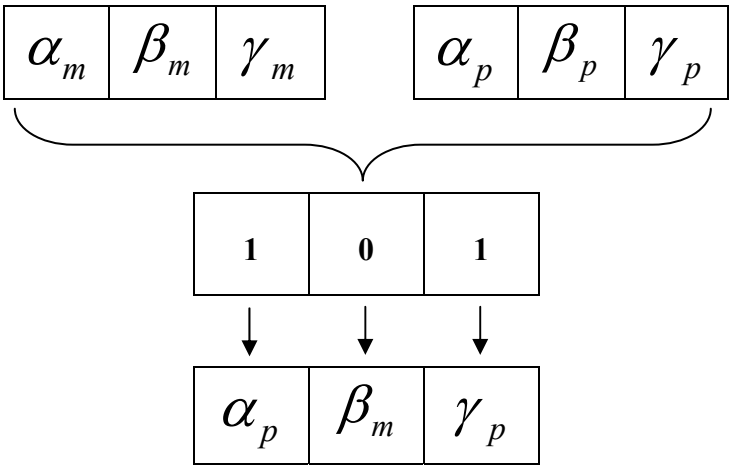


Figura 5. 8 Codificación del nuevo descendiente.

5.5.1.3 Mutación

Para mantener la diversidad de los individuos en la población y tener mayor posibilidad de realizar búsquedas más extensas, con el objetivo de encontrar una solución óptima, se realiza una mutación aleatoria en el cromosoma del hijo.

La mutación se lleva a cabo, seleccionando aleatoriamente uno de los tres genes del cromosoma del hijo. El gen que sea seleccionado tendrá que cambiar el alelo, el cual fue heredado por sus padres. Dicho valor será reemplazado por un número aleatorio.

Esta operación se realiza con una probabilidad pequeña, es decir que no todos los hijos tienen que mutar alguno de sus genes. Para seleccionar el hijo que sufrirá una mutación se generan números aleatorios en el intervalo $[1,20] \in \mathbb{N}$. Si el número es el 5, entonces se realiza la mutación

Si el número es diferente de 5, entonces los nuevos individuos se suman a la población, sin haber cambiado los alelos de sus cromosomas.

5.5.1.4 Extinción

Después de haber hecho la cruza o la mutación (si es que la hubo) se requiere matar o eliminar a aquellas hormigas que no nos han beneficiado con su búsqueda.

Para ello se cuenta con una condición que informa de las hormigas que se hayan perdido más de 5 veces, las cuales deberán de ser eliminadas de la colonia.

5.5.2 Algoritmo Propuesto (ACO-AG)

Finalmente el algoritmo 5.3 es la implementación de ACO-AG la cual realiza una emulación de la selección natural y la evolución en una población o colonia dentro de un entorno. La implementación que se encuentra subrayada se hace con las características de la sección 5.5.1.

Inicializar:

1. Colocar la ciudad de origen de la k-ésima hormiga en $taboo_k(s)$.
2. Repetir hasta que el nodo destino se encuentre:
   Si (NC%p= =0)
   Se hace una Selección Natural.
   Para $k := 1$ a m hacer:
   Elegir la ciudad j a moverse, con una probabilidad $p_{ij}^k(t)$ dada por la Ec. 5.1.
   Mover la k-ésima hormiga a la ciudad j.
   Actualizar la longitud de la ruta $cl^k = cl^k + d_{ij}$.
   Si ($cl > cl_{\max}$) Mata la k-ésima hormiga,
   Si no Inserta la ciudad j en $taboo_k(s)$.
3. Para $k := 1$ a m hacer:
   Mover la k-ésima hormiga de $taboo_k(n)$ a $taboo_k(1)$.
   Calcular la longitud L^k del tour hecho por la k-ésima hormiga.
   Actualizar la ruta más corta encontrada.
   Para cada arista (i, j) aplicar la Ec. 4.6.
4. Para cada arista (i, j) aplicar la Ec. 4.4.
   Poner $t := t + n$ y $NC := NC + 1$.
   Para cada arista (i, j) , poner $\Delta\tau_{ij}(t) = 0$.
5. Si (NC < NCMAX) y (no se encuentra un camino) Limpiar todas las listas taboo.
   Si no, Ir al paso 2.

Algoritmo 5. 3 Combinación de 2 heurísticas: ACO-AG.

5.6 Resumen

En este capítulo se explicó a detalle la metodología propuesta. Esta se fundamenta en la obtención de un grafo a partir de un entorno dado. Los algoritmos propuestos como la modificación del ACO y su evolución por medio de AG se explicaron a detalle. Se describieron las principales diferencias y ventajas con respecto al algoritmo básico.

En el siguiente capítulo se presentan los resultados obtenidos al implementar los algoritmos propuestos (Algoritmos 5.2 y 5.3) en los diferentes grafos representativos de sus respectivos entornos.

CAPÍTULO 6

RESULTADOS EXPERIMENTALES

Para probar la Metodología propuesta y medir su eficiencia, se presenta un estudio comparativo entre las técnicas descritas en el Capítulo 5. Los primeros experimentos se hicieron tomando en cuenta los siguientes 5 laberintos (ver Figura 6.1), los cuales representan distintos entornos.

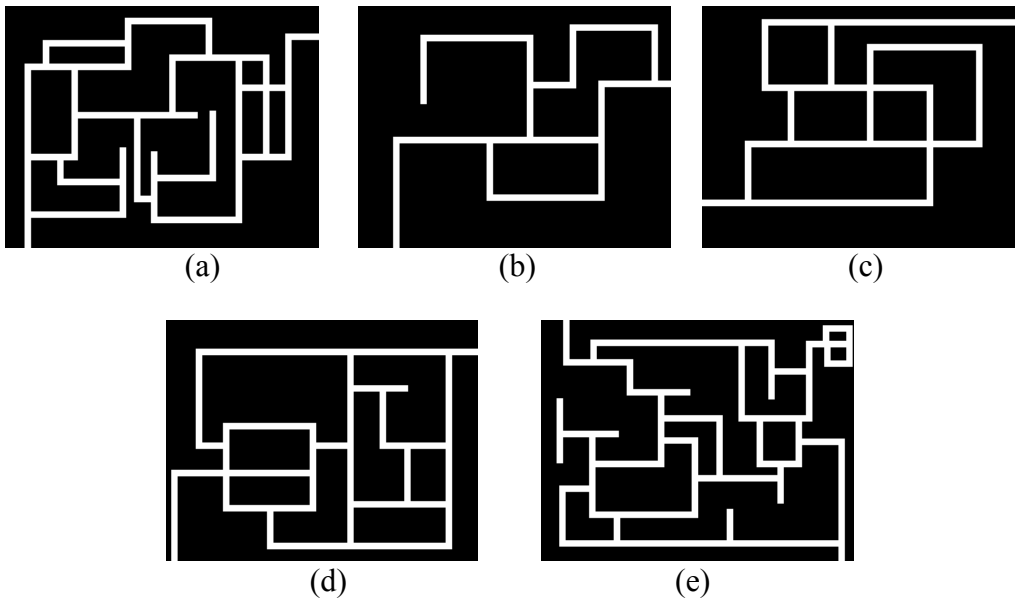


Figura 6. 1 Laberintos como entornos. (a) Laberinto 1. (b) Laberinto 2. (c) Laberinto 3. (d) Laberinto 4. (e) Laberinto 5.

A cada uno de ellos se les asignó un grafo representativo (ver Figura 6.2). A cada uno de estos grafos se aplicaron los algoritmos propuestos 5.2 y 5.3.

Para evaluar el desempeño de los algoritmos propuestos, primero se utilizó el Algoritmo clásico de Dijkstra (el cual encuentra la distancia más corta dentro de un grafo). Después se ejecutaron los algoritmos propuestos y el camino encontrado, fue comparado con el encontrado por Dijkstra. A partir de esto se genera una regla que nos permite encontrar el porcentaje de error en la solución de nuestros algoritmos en relación con la solución del Algoritmo de Dijkstra.

El algoritmo Modificación del Sistema de Colonia de Hormigas (Algoritmo 5.2) se corrió sobre cada laberinto con 7 configuraciones diferentes. Los valores de los parámetros fueron alterados con el objetivo de observar su comportamiento. Al aplicar el algoritmo de Combinación de 2 heurísticas: ACO-AG (Algoritmo 5.3) se obtuvo la mejor configuración de los parámetros que permitió encontrar el mejor camino.

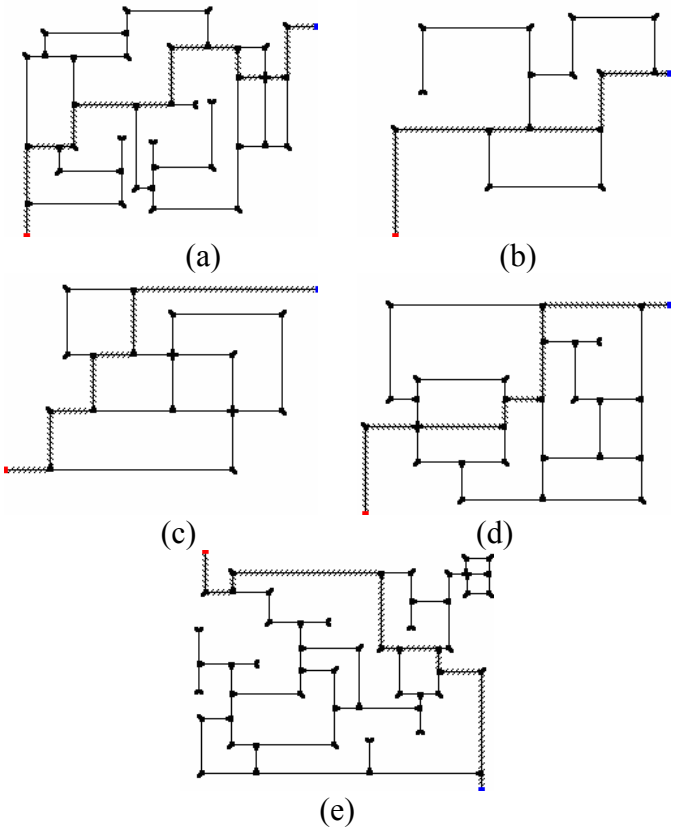


Figura 6. 2 Grafos de los laberintos y sus caminos más cortos. (a) Grafo del laberinto 1. (b) Grafo del laberinto 2. (c) Grafo del laberinto 3. (d) Grafo del laberinto 4. (e) Grafo del laberinto 5.

En éste Capítulo se presentan los diferentes experimentos realizados durante la investigación, así como su descripción y gráficas que ayudarán a entender mejor el comportamiento de la metodología propuesta y sustentar su creación.

6.1 Descripción General de la Experimentación

Esta etapa se divide en tres diferentes experimentaciones. La primera consiste en aplicar el Algoritmo propuesto 5.2 y modificar los parámetros α , β y γ para encontrar diferentes resultados, tratando de acercarse a la solución óptima. La segunda experimentación consiste en calcular de manera automática (al utilizar el Algoritmo 5.3) los valores de los parámetros α , β y γ , es decir, encontrar la mejor configuración. El último experimento consiste en aplicar las configuraciones encontradas en el segundo experimento, y aplicarlos al algoritmo 5.2, esperando obtener el camino mas corto en un tiempo menor.

A continuación se explican de manera más detallada cada uno de los experimentos mencionados anteriormente.

6.1.1 Aplicación del algoritmo modificación de ACO

El algoritmo propuesto 5.2 se corrió sobre los 5 diferentes laberintos con 7 configuraciones diferentes. En este caso se variaron los valores de los parámetros con el fin de encontrar varias soluciones.

Los experimentos realizados en ésta prueba consistieron en variar los parámetros α, β y γ al azar. Se emplearon las siguientes configuraciones: $\alpha > \gamma$, $\alpha > \beta$, $\beta > \alpha$, $\beta > \gamma$, $\gamma > \alpha$, $\gamma > \beta$ y $\alpha = \beta = \gamma$.

Las Tablas 6.1 a la 6.35, muestran el porcentaje de error del camino encontrado con respecto al camino más corto generado por el algoritmo de Dijkstra. En dichas tablas se puede observar que al obtener el camino más corto se obtuvo un porcentaje del 0.0%. En los demás casos (donde existe un porcentaje $\neq 0.0\%$) se encontró un camino alternativo que no es el óptimo, pero que es una de las posibles soluciones; el aumento del porcentaje de error, indica qué tanto se aproxima la longitud encontrada por el algoritmo propuesto respecto a la longitud encontrada por el Algoritmo de Dijkstra. Como consecuencia, entre mayor es el error, la longitud del camino encontrado se aproxima menos a la longitud del camino más corto.

La columna *M* es la cantidad de hormigas que se utilizaron en una corrida, y el número de épocas significa el número de iteraciones del algoritmo. El hecho de no haber encontrado ningún camino se representa con una X.

Para poder observar mejor el comportamiento de las tablas, cada una es acompañada por dos gráficas. La primera gráfica representa la cantidad de error al aumentar el número de épocas (de 1000 hasta 10000 épocas), la segunda representa el error encontrado al aumentar el número de hormigas (5, 10, 15, 20 y 25 hormigas). El error que se grafica es el porcentaje de error promedio.

Experimento 1.1.1

La siguiente tabla es el resultado de aplicar el algoritmo propuesto 5.2 (modificación del ACO) al laberinto 1, al utilizar la configuración 1 ($\alpha > \gamma$), donde $\alpha = 2.5$, $\beta = 1.3$ y $\gamma = 0.23$.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	0.0%	0.0%	X	X	X	X	X
10	X	X	48.23%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%
15	X	X	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%
20	26.86%	26.86%	26.86%	26.86%	26.86%	26.86%	26.86%	26.86%	26.86%	26.86%
25	14.66%	14.66%	14.66%	14.66%	14.66%	14.66%	14.66%	14.66%	14.66%	14.66%

Tabla 6. 1 Porcentajes de error en el laberinto 1, al utilizar la configuración 1.

En la Tabla 6.1, se puede observar que solamente en dos ocasiones se obtuvo el camino más corto: con 5 hormigas en 4000 y 5000 épocas; en los demás casos, los porcentajes se mantuvieron casi iguales durante todo el experimento (ya que con la mínima cantidad de hormigas y la mínima cantidad de épocas no se encontró ninguna solución).

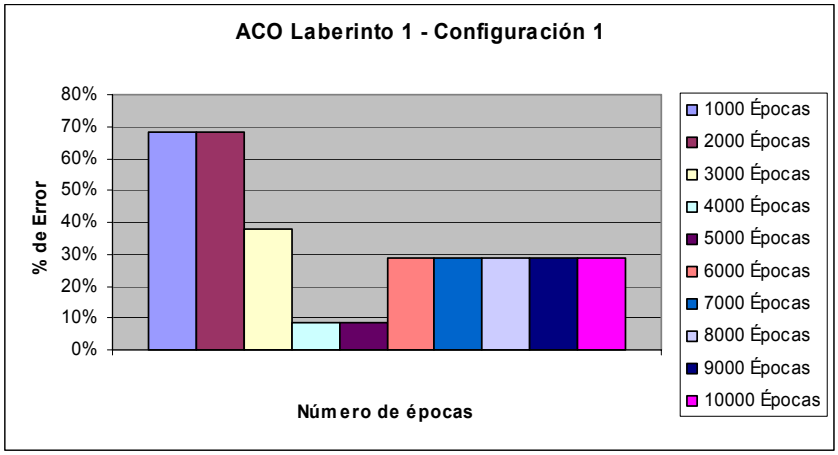


Figura 6. 3 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 1.

La Figura 6.3, muestra el porcentaje del error promedio, al aumentar el número de épocas. Como se puede ver, el error disminuye al aumentar el número de épocas, ya que las hormigas tienen mayor tiempo de hacer la búsqueda y acercarse a la solución óptima, sin

embargo existen mejores soluciones en 4000 y 5000 épocas, debido a que se está trabajando con un algoritmo basado en heurísticas y no tiene un comportamiento constante, además de que la configuración utilizada fue elegida al azar.

En la Figura 6.4, se muestra el comportamiento del algoritmo en el Laberinto 1, con la misma configuración 1, solo que ahora se presenta el porcentaje de error al aumentar el número de hormigas. En general, el comportamiento mostrado en la gráfica es que al aumentar el número de hormigas, disminuye el porcentaje de error promedio. Esto se debe a que se refuerza la cooperación entre hormigas para encontrar la solución óptima.

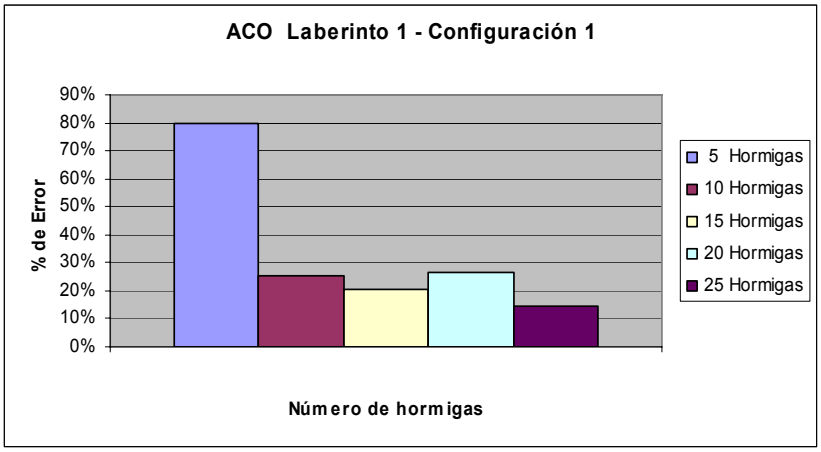


Figura 6. 4 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 1.

Experimento 1.1.2

La siguiente Tabla, es el resultado de aplicar el algoritmo 5.2 al laberinto 1, al utilizarla configuración 2 ($\alpha > \beta$), donde $\alpha = 1.8$, $\beta = 1.0$ y $\gamma = 2.2$.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%
10	X	14.14%	14.14	14.14%	14.14	14.14%	14.14%	14.14%	14.14%	14.14%
15	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
20	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
25	X	X	X	X	X	X	X	X	X	X

Tabla 6. 2 Porcentajes de error en el laberinto 1, al utilizar la configuración 2.

Los porcentajes obtenidos muestran que el comportamiento del algoritmo con esta nueva configuración, no varió, aunque en algunos casos no se encontró una solución.

La Figura 6.5 muestra que al aumentar el número de épocas el error promedio tiende a disminuir, siendo el error promedio máximo de 66% y el mínimo de 31.19%.

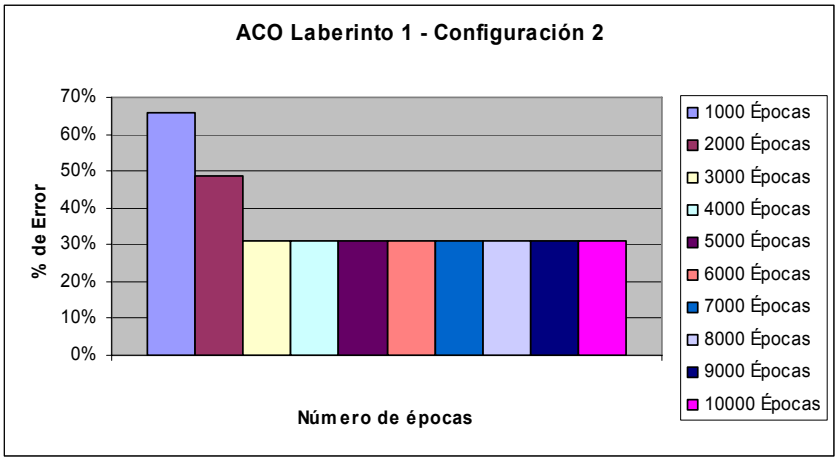


Figura 6. 5 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 2.

En la Figura 6.6 se esperaría ver que al aumentar el número de hormigas el error promedio disminuyera. En este caso pareciera que cumple con ello, pero al aumentar la cantidad de hormigas a 25, el error se dispara hasta un 100%. En la tabla se verifica este fenómeno ya que en ninguna época se encontró alguna solución. Este comportamiento se debe a la naturaleza del algoritmo ya que éste es una heurística y su comportamiento puede oscilar, pero de manera general encontrar una tendencia.

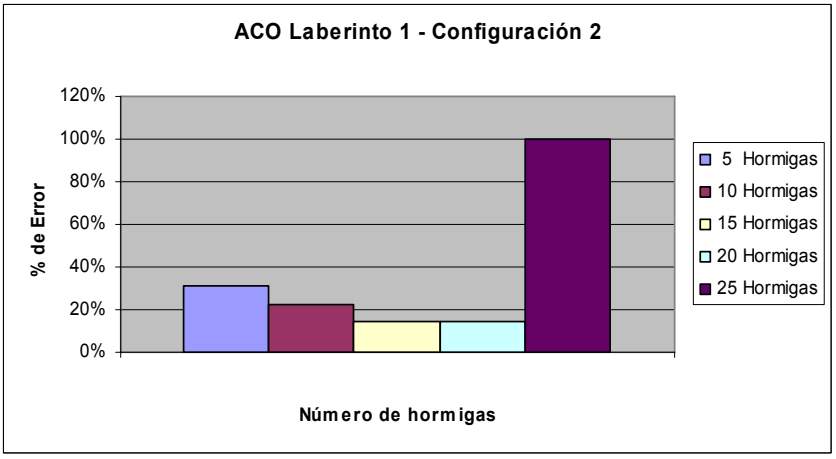


Figura 6. 6 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 2.

Experimento 1.1.3

En la Tabla 6.3, se muestra el resultado de aplicar el algoritmo 5.2 al laberinto 1, al utilizar la configuración 3 ($\beta > \alpha$), donde $\alpha = 2.4$, $\beta = 3.0$ y $\gamma = 0.5$.

Los porcentajes de error mostrados en la siguiente tabla son muy grandes, cabe señalar que la configuración no tiene que ver con dicho comportamiento, si no con los valores que toman los parámetros; en este caso los valores no nos acercaron a la solución óptima.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	100%	100%	100%	100%	100%	100%	45.72%	45.72%
10	82.57%	49.01%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
15	X	X	X	X	X	X	X	X	X	X
20	X	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
25	X	X	X	X	X	X	X	X	X	X

Tabla 6. 3 Porcentajes de error en el laberinto 1, al utilizar la configuración 3.

La Figura 6.7, muestra que al aumentar el número de épocas, el error promedio tiende a disminuir.

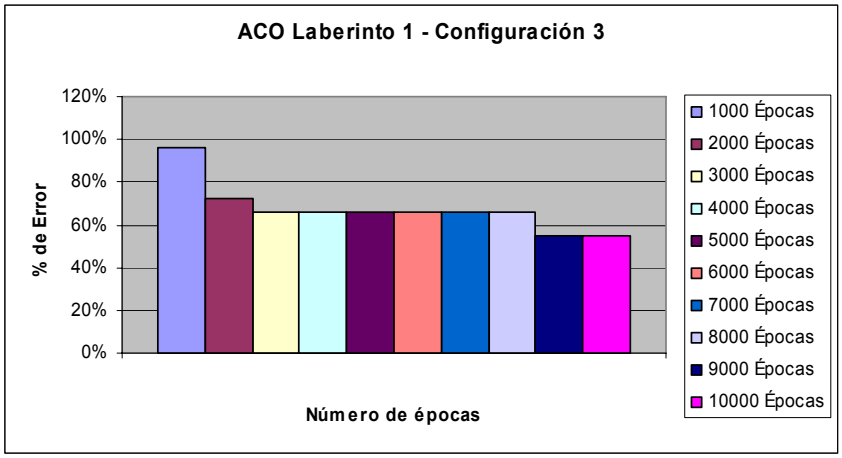


Figura 6. 7 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 3.

Por el contrario, se puede ver que en la Figura 6.8, el porcentaje de error promedio se comporta de manera variable.

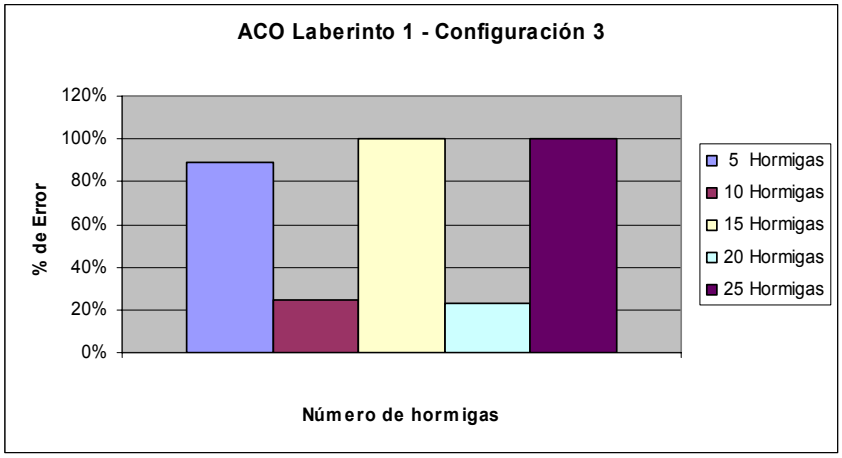


Figura 6. 8 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 3.

Experimento 1.1.4

En la Tabla 6.4, se aprecia el resultado de aplicar el algoritmo 5.2 al laberinto 1, al utilizar la configuración 4 ($\beta > \gamma$), donde $\alpha = 0.9$, $\beta = 2.81$ y $\gamma = 1.14$.

Los porcentajes de error mostrados en dicha tabla son grandes, pero se pudo obtener en su mayoría una solución alterna a la solución óptima.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%
10	X	38.89%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%
15	X	47.7%	14.14%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%
20	X	X	44.61%	44.61%	44.61%	44.61%	14.66%	14.66	14.66	14.66%
25	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%

Tabla 6. 4 Porcentajes de error en el laberinto 1, al utilizar la configuración 4.

La Figura 6.9 muestra de manera eficiente que entre mayor es el número de épocas, menor es el porcentaje de error promedio.

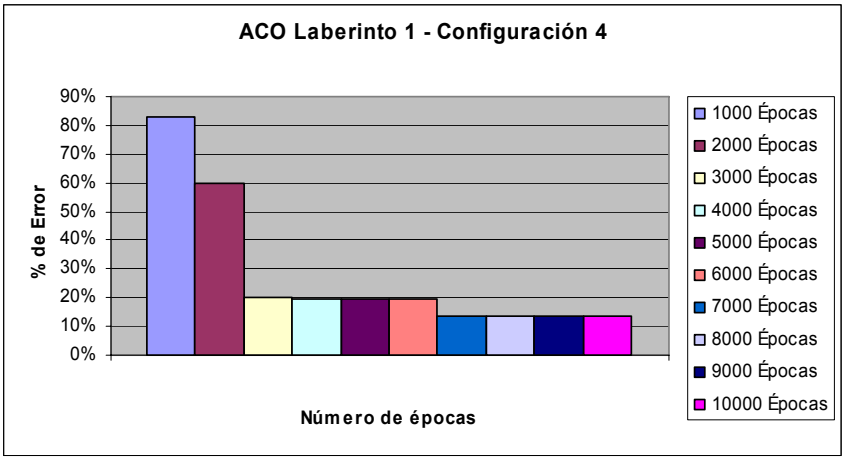


Figura 6. 9 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 4.

De manera general, se puede ver que en la Figura 6.10, al aumentar la cantidad de hormigas, disminuye el error promedio, aunque con 20 hormigas se dispare en un 44%.

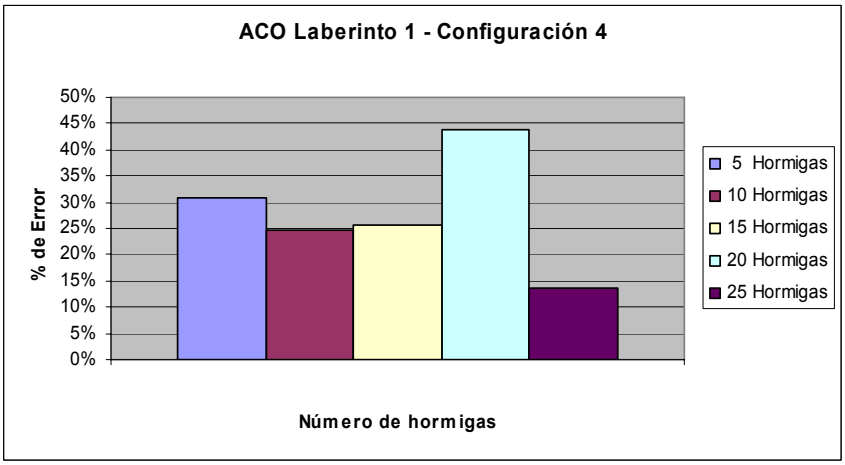


Figura 6. 10 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 4.

Experimento 1.1.5

En la Tabla 6.5, se muestra el resultado de aplicar el algoritmo 5.2 al laberinto 1, al utilizar la configuración 5 ($\gamma > \alpha$), donde $\alpha = 0.13$, $\beta = 1.66$ y $\gamma = 0.97$.

Esta tabla muestra el pésimo comportamiento del algoritmo con los valores de la configuración dada, como se pude observar en ningún momento se pudo obtener una solución.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	X	X	X	X	X	X	X
10	X	X	X	X	X	X	X	X	X	X
15	X	X	X	X	X	X	X	X	X	X
20	X	X	X	X	X	X	X	X	X	X
25	X	X	X	X	X	X	X	X	X	X

Tabla 6. 5 Porcentajes de error en el laberinto 1, al utilizar la configuración 5.

La Figura 6.11, muestra que desde las 1000 hasta las 10000 épocas, hubo un porcentaje de error promedio del 100%.

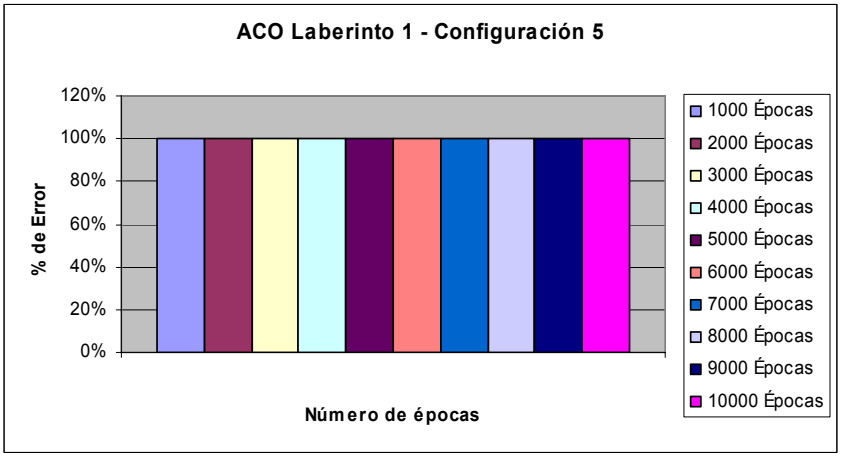


Figura 6. 11 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 5.

Al igual que la Figura 6.11, la Figura 6.12 muestra que en todos los casos se obtuvo un error promedio del 100%.

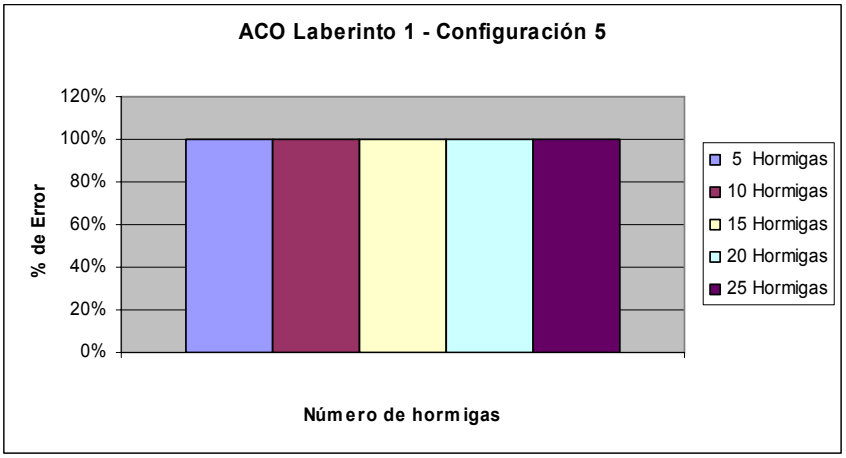


Figura 6. 12 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 5.

Experimento 1.1.6

En la Tabla 6.6, se muestra el resultado de aplicar el algoritmo 5.2 al laberinto 1, al utilizar la configuración 6 ($\gamma > \beta$), donde $\alpha = 2.5$, $\beta = 1.7$ y $\gamma = 3.4$.

En esta tabla de resultados se puede observar que el mayor porcentaje de error es de 81.81% y que en algunos casos no se encontraron soluciones.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
10	X	0.59%	81.84%	81.84%	81.84%	81.84%	81.84%	81.84%	48.28%	48.28%
15	X	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
20	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
25	X	X	X	X	X	14.14%	14.14%	14.14%	14.14%	14.14%

Tabla 6. 6 Porcentajes de error en el laberinto 1, al utilizar la configuración 6.

La Figura 6.13 muestra que el mayor porcentaje de error promedio, fue provocado en las primeras 1000 épocas, sin embargo al aumentarlas disminuye el error.

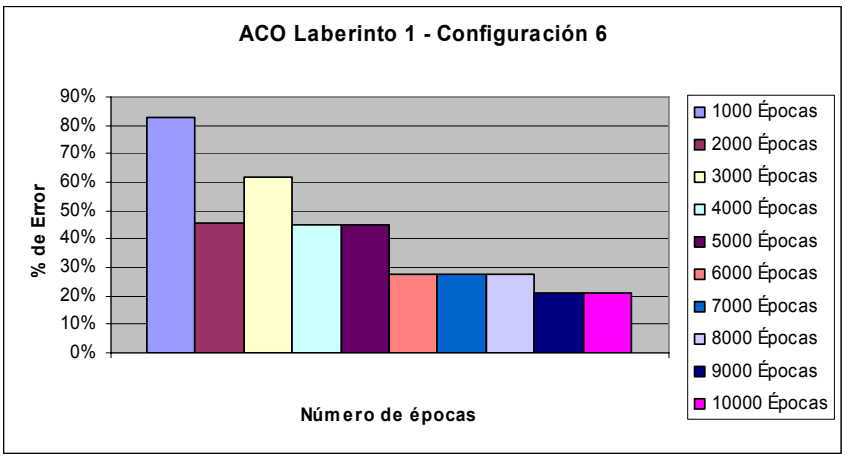


Figura 6. 13 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 6.

La Figura 6.14, muestra que con 15 hormigas se obtiene el porcentaje promedio de error máximo de 69%, siguiéndole el de 25 hormigas.

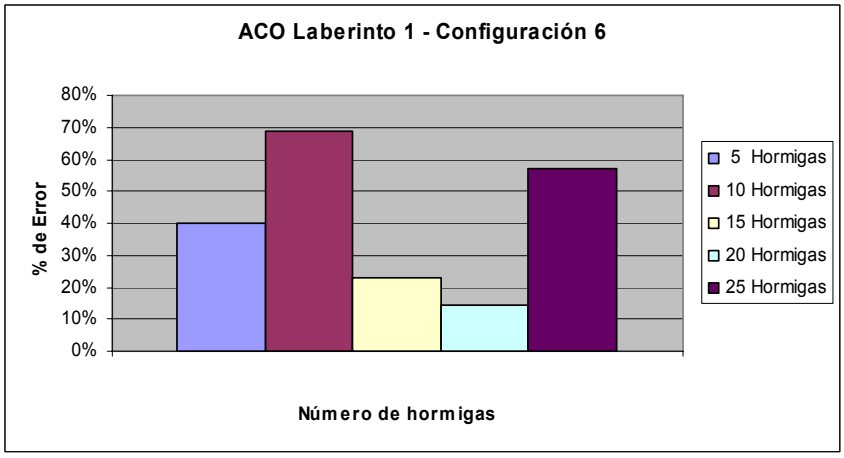


Figura 6. 14 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 6.

Experimento 1.1.7

Finalmente la Tabla 6.7 es el resultado de aplicar el algoritmo 5.2 al laberinto 1, al utilizar la última configuración, la configuración 7 ($\gamma > \beta$), donde $\alpha = \beta = \gamma = 0.8$.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	X	X	X	X	X	X	X
10	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%
15	14.14%	14.14%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%	13.55%
20	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%	14.14%
25	X	X	70.55%	14.61%	14.61%	44.61%	14.61%	44.61%	44.61%	44.61%

Tabla 6. 7 Porcentajes de error en el laberinto 1, al utilizar la configuración 7.

La Figura 6.15 es un ejemplo del comportamiento de un algoritmo heurístico; como ya se ha mencionado en otros casos, se tiene un comportamiento variable del error promedio.

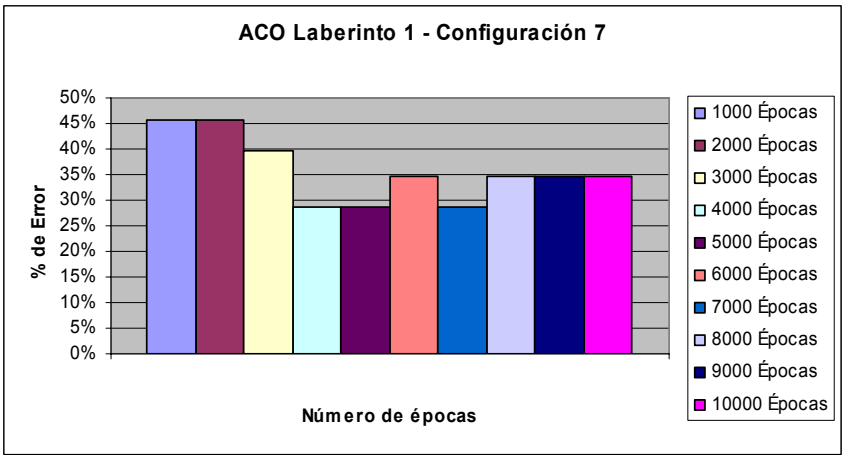


Figura 6. 15 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de épocas, al utilizar la configuración 7.

A diferencia de otras Figuras, se puede que los valores de la Figura 6.16 tienden a subir al incrementar el número de hormigas (a partir de 15 hormigas); sin embargo su

porcentaje de error no es más grande que el obtenido por 5 hormigas. También se puede observar que hay una caída brusca en 10 hormigas, casi del 0.0% del error promedio.

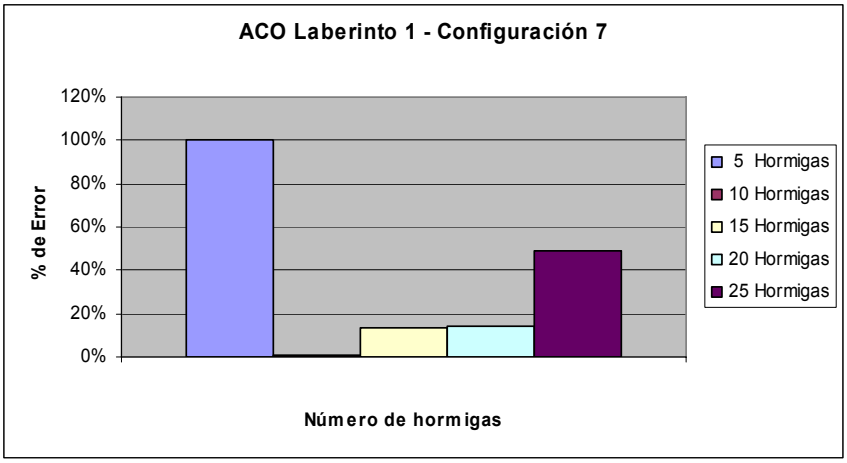


Figura 6. 16 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 1, al incrementar el número de hormigas, al utilizar la configuración 7.

Las siguientes dos Figuras (6.17 y 6.18), muestran el comportamiento del algoritmo en el laberinto 1, al tomar el promedio de las 7 configuraciones.

Como se puede observar en la Figura 6.17, se presenta el porcentaje de error promedio al aumentar el número de épocas. Al tomar en cuenta el promedio de los resultados de las 7 configuraciones para cada una de las épocas, el porcentaje promedio de error disminuye al aumentar el número de épocas.

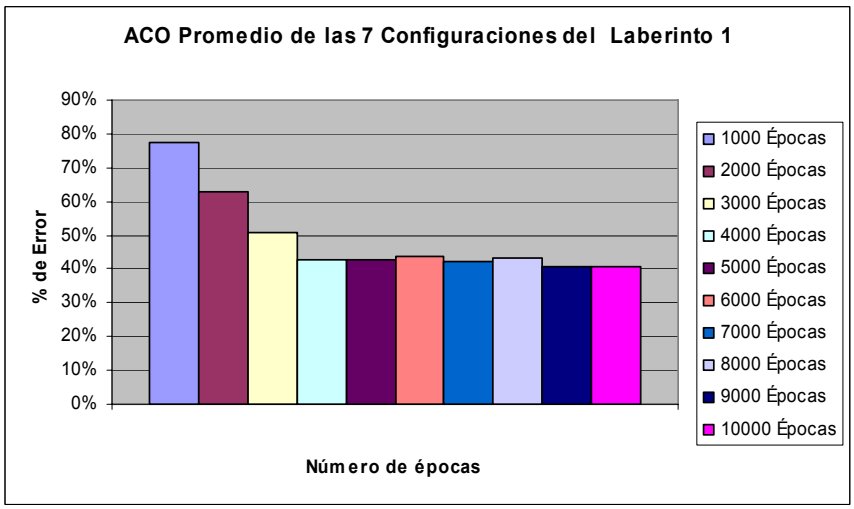


Figura 6. 17 Comportamiento del algoritmo propuesto en el laberinto 1 al incrementar el número de épocas, al utilizar el promedio de las 7 configuraciones.

La Figura 6.18, muestra el promedio de los resultados de las 7 configuraciones para cada conjunto de hormigas. El porcentaje promedio de error disminuye al aumentar el número de hormigas, aunque con 25 hormigas el error se vuelva a incrementar. Esto se debe a que el algoritmo es una heurística y no se asegura que siempre encuentre un óptimo.

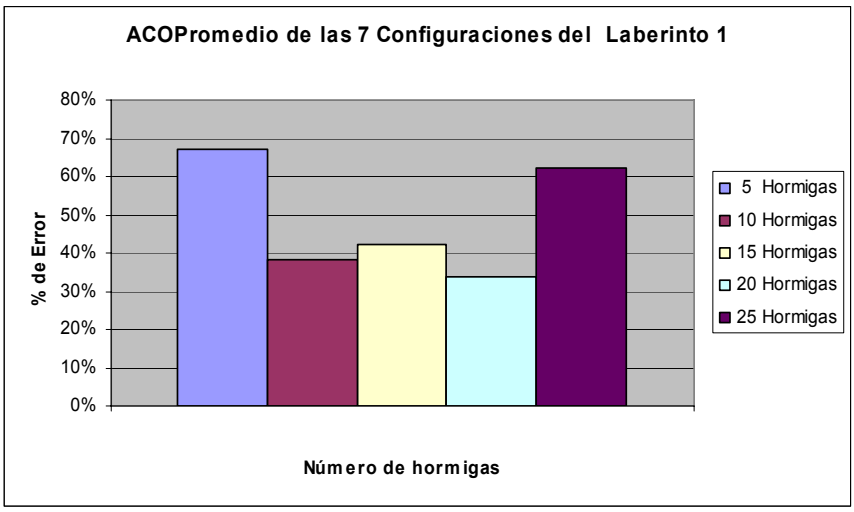


Figura 6. 18 Comportamiento del algoritmo propuesto en el laberinto 1 al incrementar el número de hormigas, al utilizar el promedio de las 7 configuraciones.

Experimento 1.2.1

Los valores de porcentaje de la Tabla 6.8, son el resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 1 ($\alpha > \gamma$), donde $\alpha = 2.5$, $\beta = 1.3$ y $\gamma = 0.23$.

Con ésta configuración, y debido al tipo de laberinto se obtuvieron porcentajes de error del 0.0%, lo que nos indica que sí se encontró el camino más corto.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	26.56%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	26.56%	25.08%	25.08%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	25.56%	25.08%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 8 Porcentajes de error en el laberinto 2, al utilizar la configuración 1.

La Figura 6.19, muestra que el aumento de épocas es un buen indicador para disminuir el porcentaje de error.

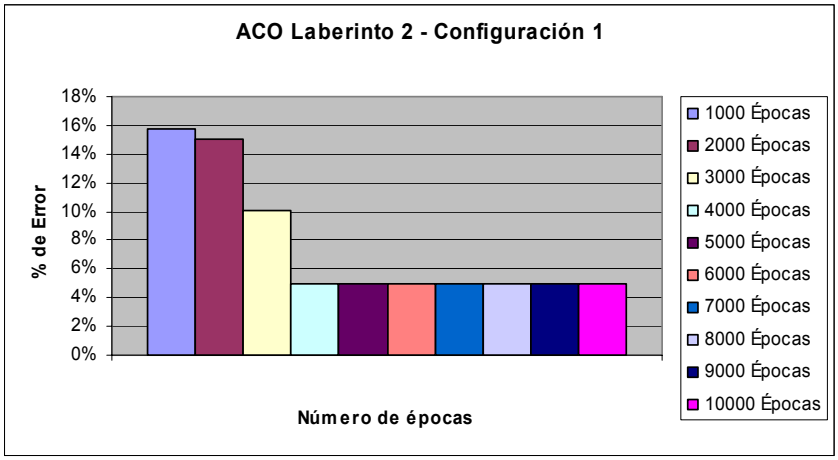


Figura 6. 19 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 1 .

A pesar de que el aumento de épocas disminuyó el error, se puede ver que la Figura 6.20, con 10 y 15 hormigas, el porcentaje de error promedio es de 0.0%, pero después sube un poco. Sin embargo en general, se tiene una baja del error promedio.

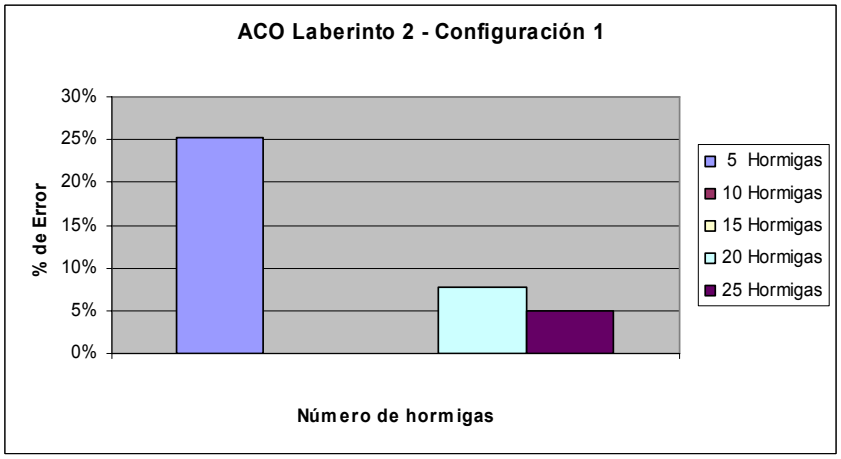


Figura 6. 20 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 1.

Experimento 1.2.2

Los valores de los porcentajes de la Tabla 6.9, son el resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 2 ($\alpha > \beta$), donde $\alpha = 1.8$, $\beta = 1.0$ y $\gamma = 2.2$.

Esta tabla muestra valores mejores que las anteriores. Esto quiere decir que la configuración elegida se aproxima más a la solución óptima (para todos los casos, tanto para el número de épocas como para el número de hormigas).

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	25.08%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 9 Porcentajes de error en el laberinto 2, al utilizar la configuración 2.

La Figura 6.21, muestra que el único error promedio obtenido se dio con 1000 épocas. En los otros casos se obtuvo el camino más corto.

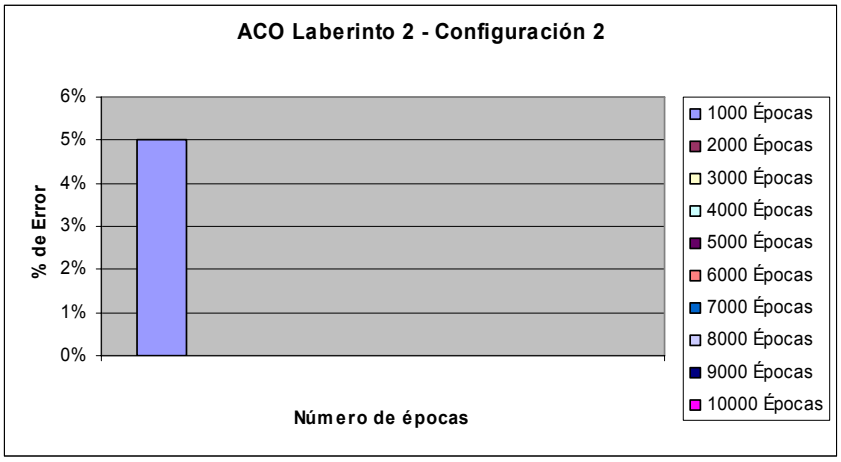


Figura 6. 21 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 2.

La Figura 6.22 muestra que para el caso con 5 hormigas, se obtuvo un 0.0% de error (el máximo porcentaje de error promedio se obtuvo con 15 hormigas).

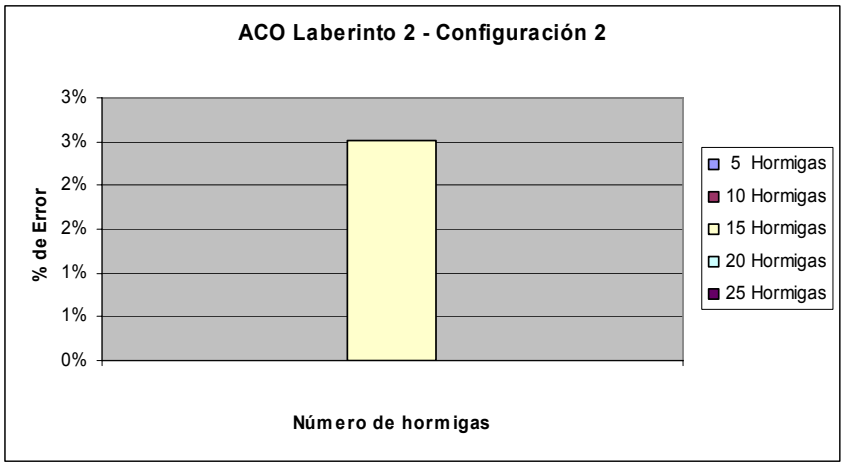


Figura 6. 22 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 2.

Experimento 1.2.3

En la Tabla 6.10, se tienen los porcentajes de error, resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 3 ($\beta > \alpha$), donde $\alpha = 2.4$, $\beta = 3.0$ y $\gamma = 0.5$.

Como se puede ver, en todos los casos se obtuvo un porcentaje de error promedio del 0.0%, es decir, que en todos los casos encontramos siempre el camino más corto, por lo que acertamos con los valores dados a los parámetros α , β y γ .

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 10 Porcentajes de error en el laberinto 2, al utilizar la configuración 3.

La Figura 6.23, es sólo una comprobación de los valores que se encuentran en la Tabla 6.10.

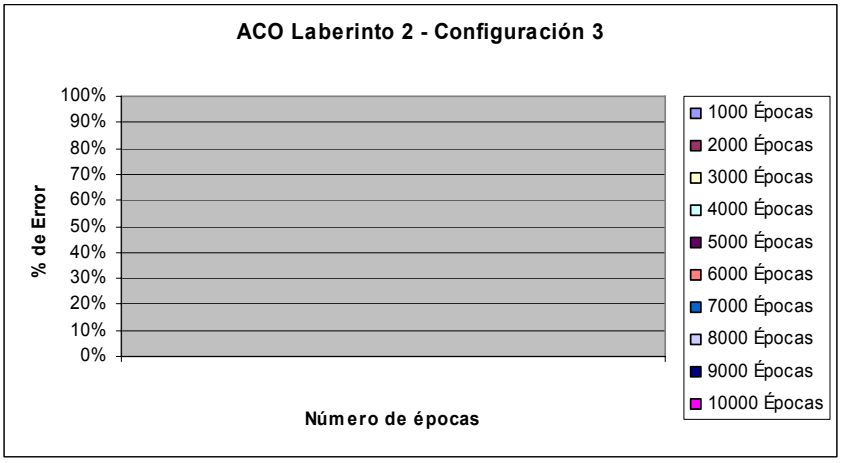


Figura 6. 23 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 3.

No importa cuantas hormigas se utilizaron en este ejemplo, pues con todas se logró acertar a la solución óptima, véase la Figura 6.24.

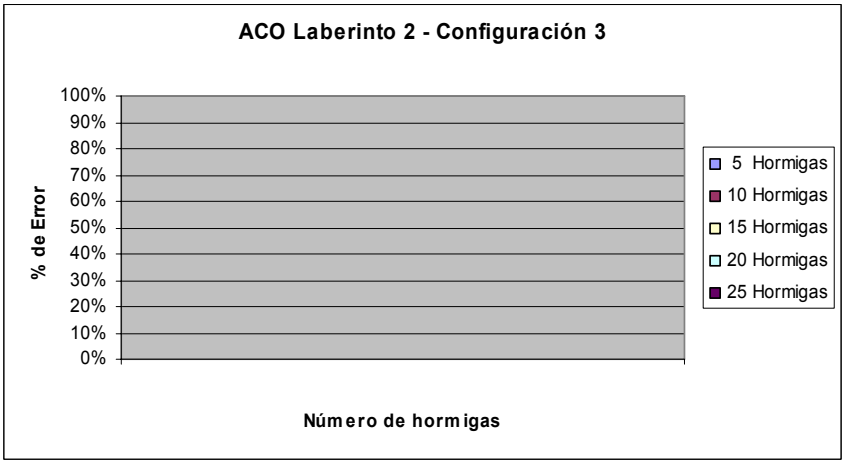


Figura 6. 24 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 3.

Experimento 1.2.4

Los porcentajes de error mostrados en la Tabla 6.11, son el resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 4 ($\beta > \gamma$), donde $\alpha = 0.9$, $\beta = 2.81$ y $\gamma = 1.14$.

Como podemos observar, se obtuvo en su mayoría la solución óptima, y las soluciones restantes se mantuvieron en todo el experimento.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	26.56%	26.56%	26.56%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	0.0%	25.08%	25.08%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	25.08%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 11 Porcentajes de error en el laberinto 2, al utilizar la configuración 4.

La Figura 6.25 muestra la argumentación de que, entre mayor es el número de épocas, menor es el porcentaje de error promedio.

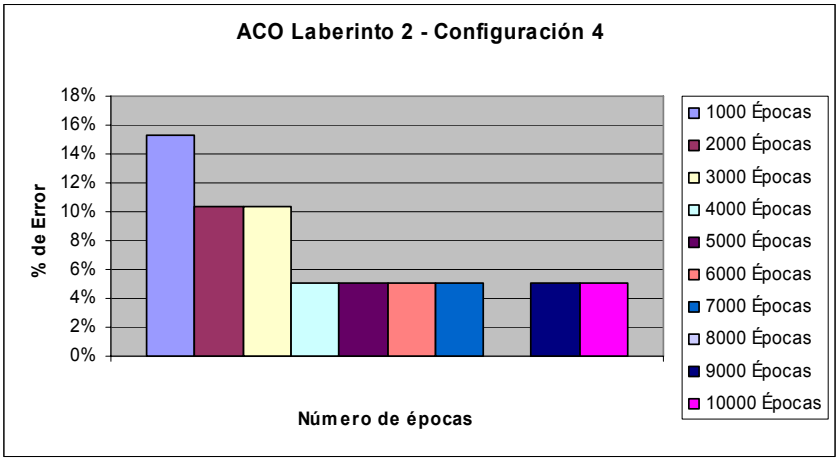


Figura 6. 25 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 4.

A pesar de obtener un porcentaje alto de error al utilizar 10 hormigas, los resultados tienden a dar soluciones cercanas a la solución óptima, ver Figura 6.26.

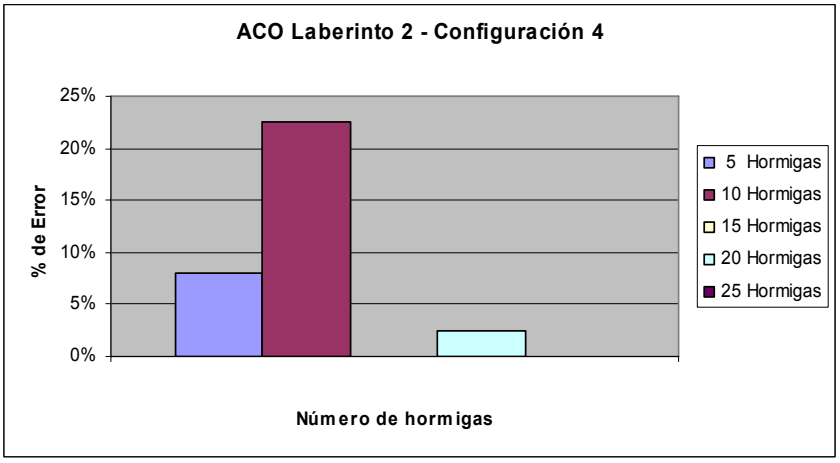


Figura 6. 26 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 4.

Experimento 1.2.5

Los porcentajes de error de la Tabla 6.12, son el resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 5 ($\gamma > \alpha$), donde $\alpha = 0.13$, $\beta = 1.66$ y $\gamma = 0.97$.

Como se puede observar en la Tabla 6.12, el único porcentaje de error diferente a 0.0% es el que se da al utilizar 10 hormigas en 8000 épocas.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	25.08%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 12 Porcentajes de error en el laberinto 2, al utilizar la configuración 5.

En la Figura 6.27, se pude observar que en 8000 épocas no se obtuvo la solución óptima, o no se encontró el camino más corto, a diferencia de las épocas restantes.

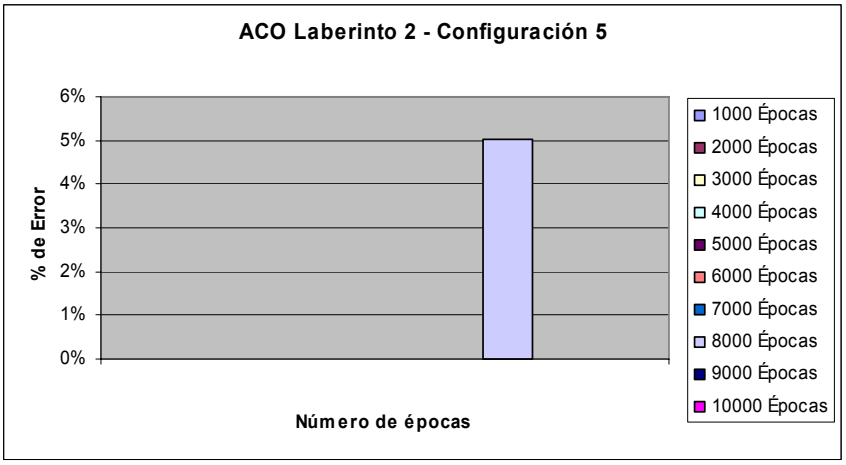


Figura 6. 27 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 5.

Se puede observar, como ya se mencionó en la Tabla 6.12, que el porcentaje de error promedio del 25.08% se dio con 10 hormigas. Para el laberinto dos, con 10 hormigas, 8000 épocas y la configuración adecuada, se obtendría una solución óptima, ver Figura 6.28.

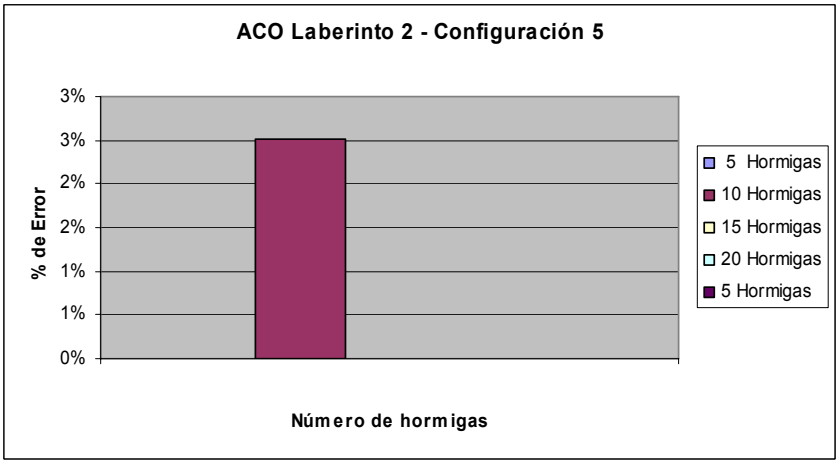


Figura 6. 28 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 5.

Experimento 1.2.6

Los porcentajes de error mostrados en la Tabla 6.13, son el resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 6 ($\gamma > \beta$), donde $\alpha = 2.5$, $\beta = 1.7$ y $\gamma = 3.4$.

Este es otro ejemplo de obtener constantemente los valores de los parámetros dados, el camino más corto.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 13 Porcentajes de error en el laberinto 2, al utilizar la configuración 6.

Sólo se concluye de la Figura 6.29, que siempre se obtuvo un porcentaje de error del 0.0% a pesar de aumentar el número de épocas, lo cual indica el buen desempeño del algoritmo 5.2.

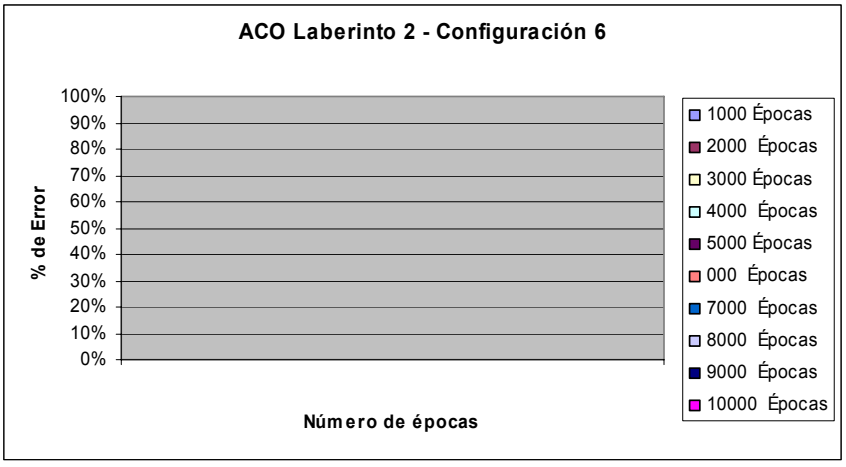


Figura 6. 29 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 6.

No importa el aumento del número de hormigas que se de en el algoritmo, ya que el error promedio se mantiene de manera nula; esto sólo para este caso, bajo las condiciones ya mencionadas.

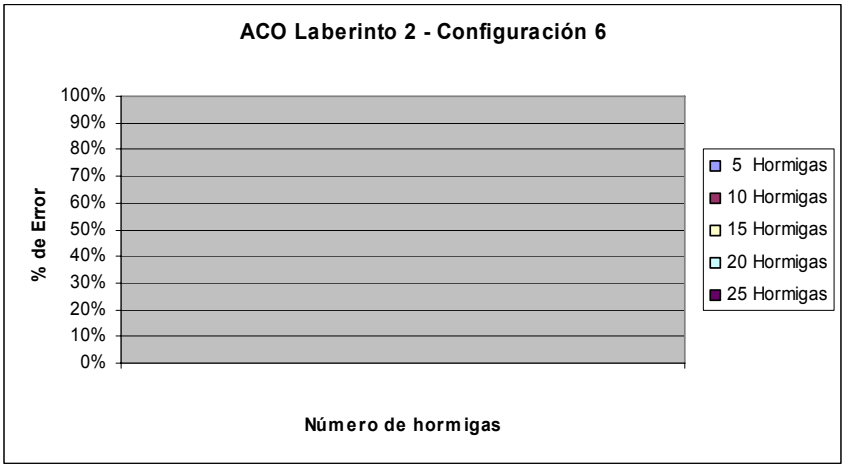


Figura 6. 30 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 6.

Experimento 1.2.7

Los porcentajes de error de la Tabla 6.14 son el resultado de aplicar el algoritmo 5.2 al laberinto 2, al utilizar la configuración 7 ($\gamma > \beta$), donde $\alpha = \beta = \gamma = 0.8$.

Los siguientes porcentajes, en su mayoría son excelentes, y tienden a repetirse en su defecto, por ejemplo para 15 hormigas, no importa con cuántas épocas se trabaje, ya que presentó siempre el mismo porcentaje de error.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	26.56%	26.56%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%	25.08%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 14 Porcentajes de error en el laberinto 2, al utilizar la configuración7.

Los valores mostrados en la Figura 6.31 tienen el comportamiento insistente de ésta experimentación. Al ir aumentando el número de épocas (a partir de 3000 épocas) se puede observar una disminución del porcentaje de error de un 10% a un 5% (no olvidar que se habla de porcentajes de error promedio).

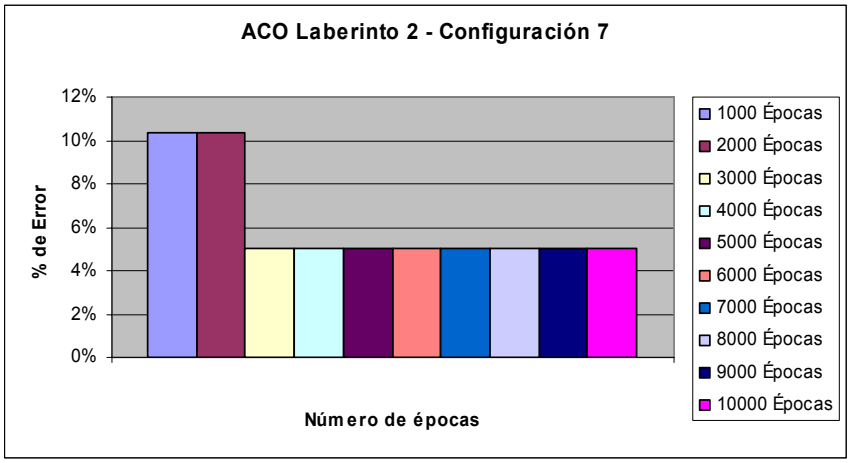


Figura 6. 31 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de épocas, al utilizar la configuración 7.

Solamente con 3 cantidades de hormigas diferentes, se logró encontrar el camino más corto (10, 20 y 25 hormigas), véase la Figura 6.32.

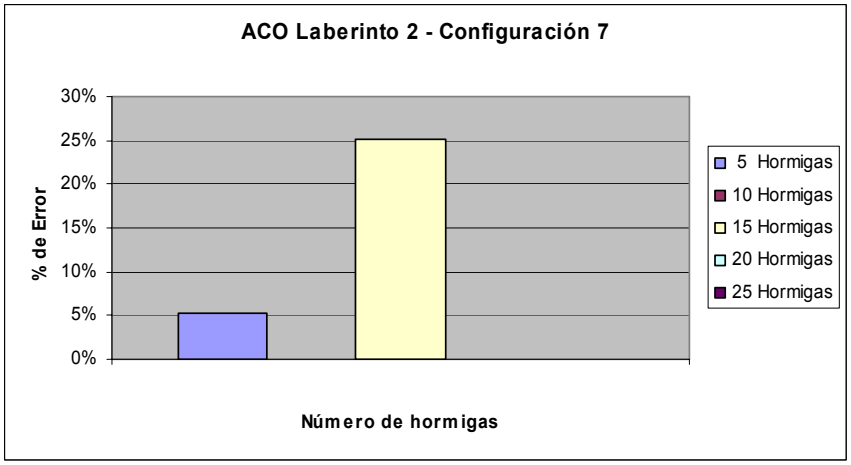


Figura 6. 32 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 2, al incrementar el número de hormigas, al utilizar la configuración 7.

Para tener una idea general del comportamiento del algoritmo 5.2 sobre el laberinto 2, se tomó en cuenta el promedio de los resultados de las 7 configuraciones para cada una de las épocas. Como se puede ver en la Figura 6.33, el porcentaje promedio de error disminuye al aumentar el número de épocas.

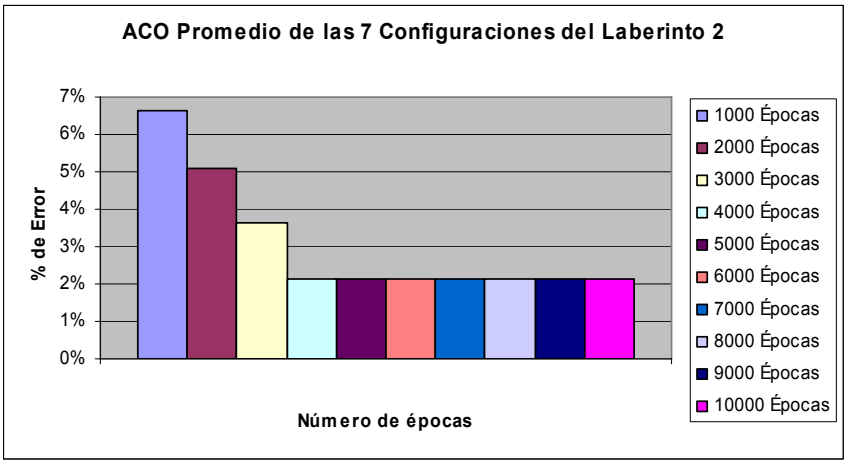


Figura 6. 33 Comportamiento del algoritmo propuesto en el laberinto 2 al incrementar el número de épocas, al utilizar el promedio de las 7 configuraciones.

Un comportamiento parecido aparece en la Figura 6.34, donde, de igual manera se obtiene el error promedio de los resultados de las 7 configuraciones para cada número de hormigas. El resultado es que el porcentaje promedio de error disminuye al aumentar el número de hormigas.

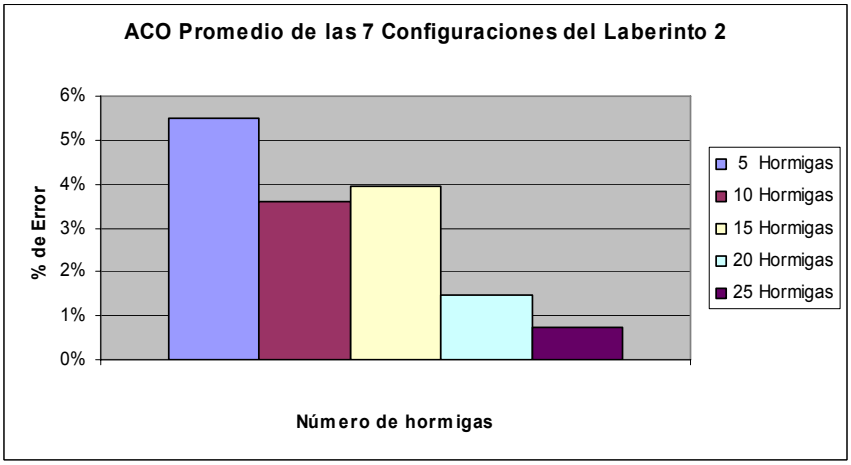


Figura 6. 34 Comportamiento del algoritmo propuesto en el laberinto 2 al incrementar el número de hormigas, al utilizar el promedio de las 7 configuraciones.

Experimento 1.3.1

Los porcentajes de error de la Tabla 6.15, son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 1 ($\alpha > \gamma$), donde $\alpha = 2.5$, $\beta = 1.3$ y $\gamma = 0.23$.

Como se puede observar sólo 3 porcentajes de error nos indican una solución alternativa a la solución óptima.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	11.86%	11.86%	11.86%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 15 Porcentajes de error en el laberinto 3, al utilizar la configuración 1.

En éste ejemplo, con 1000, 2000, y 3000 épocas se obtiene una de las posibles soluciones. Para estos 3 casos un valor máximo del error promedio es del 2%.

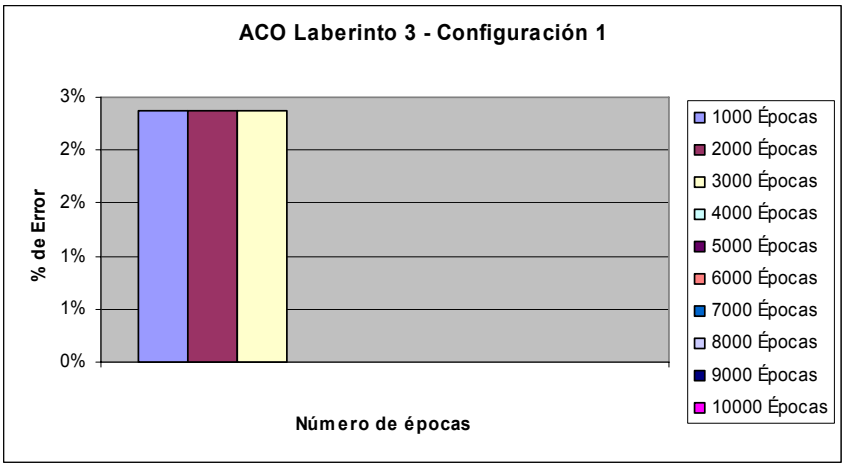


Figura 6. 35 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 1.

En la Figura 6.36, 4 números de hormigas diferentes (5, 10, 20 y 25) dieron un porcentaje de error promedio del 0.0%; *i.e.* la solución óptima. Con 15 hormigas el error promedio fue del 4.1%.

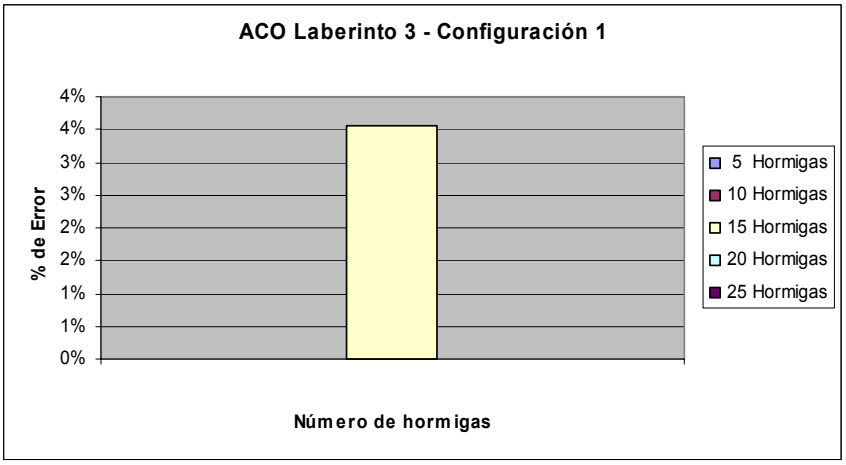


Figura 6. 36 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 1.

Experimento 1.3.2

Los porcentajes mostrados en la Tabla 6.16 son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 2 ($\alpha > \beta$), donde $\alpha = 1.8$, $\beta = 1.0$ y $\gamma = 2.2$.

La Tabla, muestra un porcentaje de error, casi en su totalidad del 0.0% y un porcentaje del 11.86% para el caso de 5 hormigas y 1000 épocas. Este porcentaje es el mismo que el de la tabla anterior. Esto sin embargo, no significa que tenga una relación ya que para este ejercicio, los valores de los parámetros cambian.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	11.86%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 16 Porcentajes de error en el laberinto 3, al utilizar la configuración 2.

Las siguientes dos Figuras (6.37 y 6.38), muestran el porcentaje de error promedio obtenido al aumentar el número de épocas. En todos los casos, excepto para 1000 épocas se encontró el camino más corto.

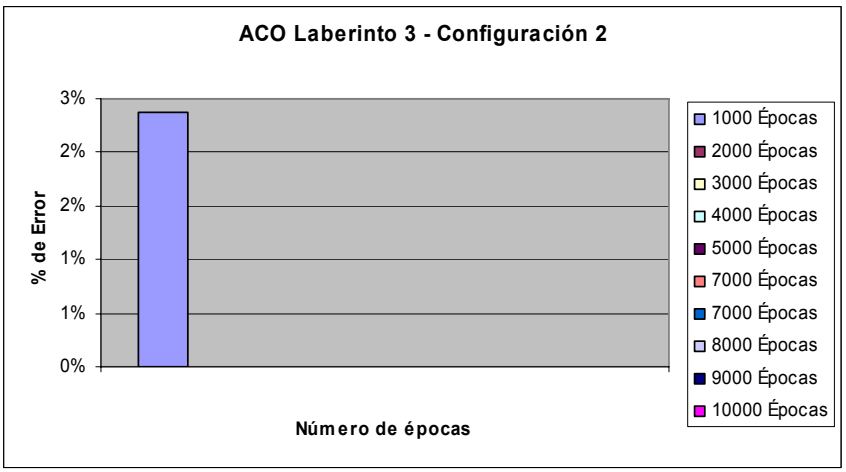


Figura 6. 37 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 2.

Como se puede ver el porcentaje de error es muy pequeño y se presenta sólo para 5 hormigas, en los demás casos se tiene un porcentaje de error promedio del 0.0%.

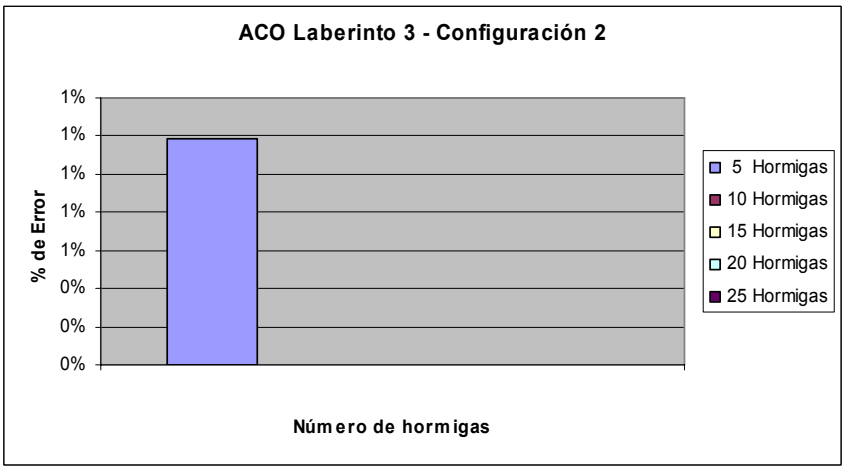


Figura 6. 38 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 2.

Experimento 1.3.3

Los porcentajes mostrados en Tabla 6.17 son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 3 ($\beta > \alpha$), donde $\alpha = 2.4$, $\beta = 3.0$ y $\gamma = 0.5$.

La Tabla 6.17 muestra la eficiencia de los valores de los parámetros para éste ejemplo. Como se pude ver en todos los casos se logró obtener un porcentaje de error nulo.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 17 Porcentajes de error en el laberinto 3, al utilizar la configuración 3.

En las Figuras (6.39 y 6.40) se muestra que al aumentar el número de épocas y el número de hormigas, en todos los casos, se obtuvo la solución óptima, es decir el camino más corto.

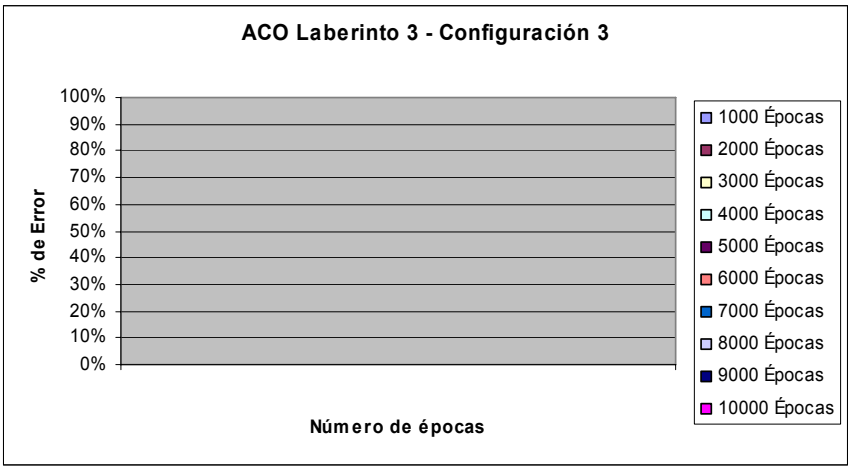


Figura 6. 39 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 3.

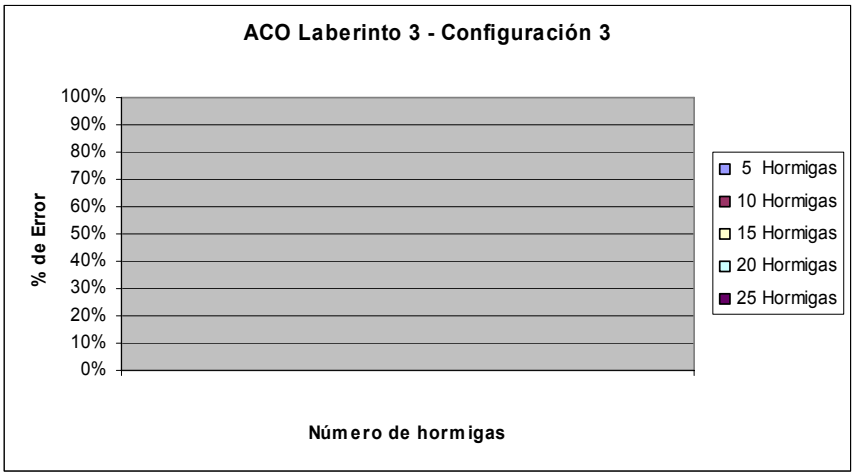


Figura 6. 40 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 3.

Experimento 1.3.4

Los porcentajes mostrados en la Tabla 6.18 son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 4 ($\beta > \gamma$), donde $\alpha = 0.9$, $\beta = 2.81$ y $\gamma = 1.14$.

Como en el ejemplo anterior, los resultados mostrados en la Tabla 6.18 evidencian que en todos los casos, con los parámetros anteriores se pudo conseguir la solución óptima.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 18 Porcentajes de error en el laberinto 3, al utilizar la configuración 4.

Los resultados mostrados en las Figuras (6.41 y 6.42) son reflejo del excelente resultado de utilizar los parámetros mencionados anteriormente, al grafo representativo del laberinto 3, tanto al aumentar el número de épocas como el número de hormigas.

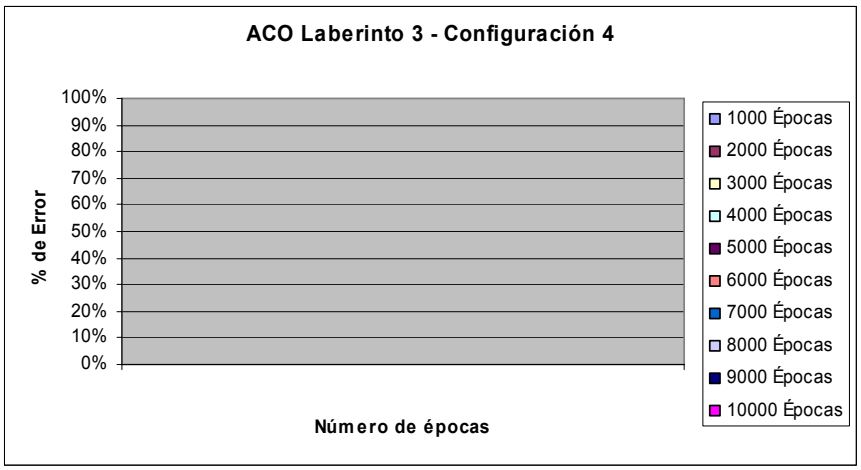


Figura 6. 41 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 4.

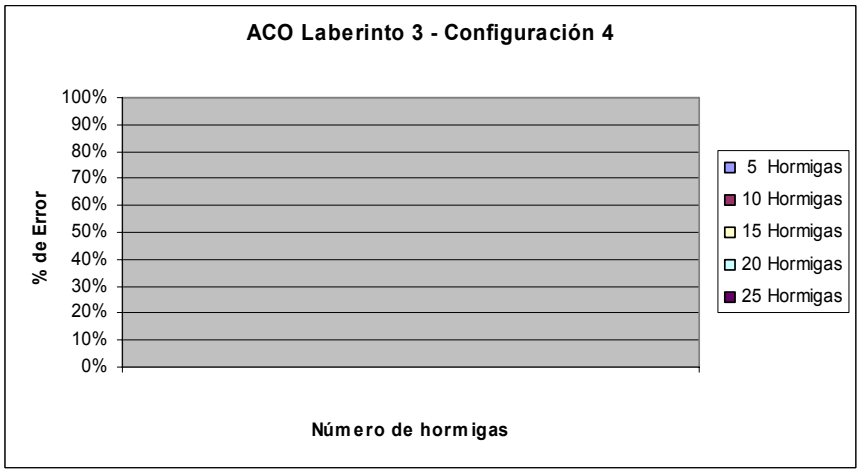


Figura 6. 42 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 4.

Experimento 1.3.5

Los porcentajes de error mostrados en la Tabla 6.19 son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 5 ($\gamma > \alpha$), donde $\alpha = 0.13$, $\beta = 1.66$ y $\gamma = 0.97$.

El único caso desconcertante en éste ejemplo (después de observar que en casi todos los casos se encuentra la solución óptima), se da en 2000 épocas con 5 hormigas. En este caso no se encontró ninguna solución.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	X	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	11.86%	11.86%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	11.86%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 19 Porcentajes de error en el laberinto 3, al utilizar la configuración 5.

Como se observa en la Tabla 6.19, sólo con 1000 y 2000 épocas se obtuvieron porcentajes de error promedio diferentes al 0.0%, al obtener una X entre los valores aumenta dicho error véase la Figura 6.43.

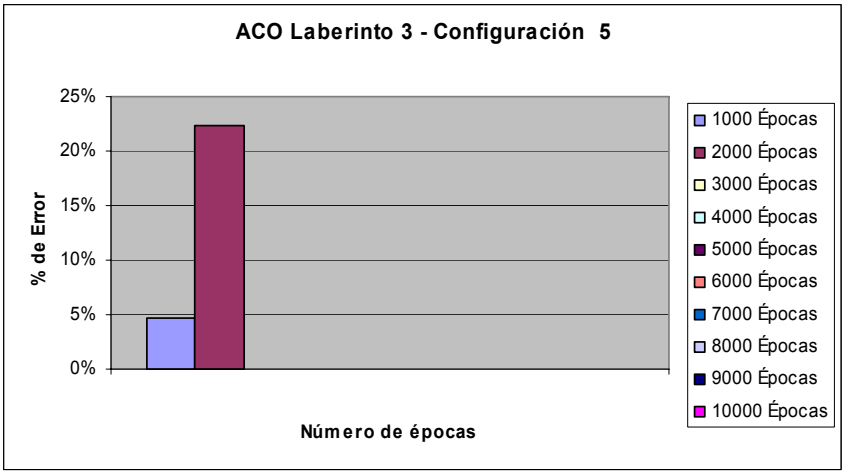


Figura 6. 43 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 5.

La Figura 6.44 muestra que entre mayor es el número de hormigas, menor es el error promedio.

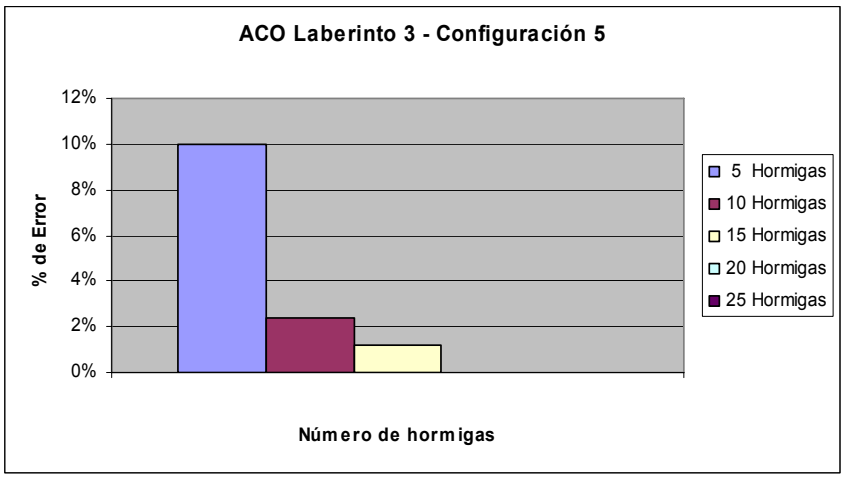


Figura 6. 44 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 5.

Experimento 1.3.6

Los porcentajes mostrados en la Tabla 6.20 son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 6 ($\gamma > \beta$), donde $\alpha = 2.5$, $\beta = 1.7$ y $\gamma = 3.4$.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	11.86%	11.86%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 20 Porcentajes de error en el laberinto 3, al utilizar la configuración 6.

Al graficar el número de épocas con respecto al porcentaje de error obtenemos un error promedio. En esta Tabla se muestra que al aumentar el número de épocas, el error tiende a disminuir notablemente, ver Figura 6.45.

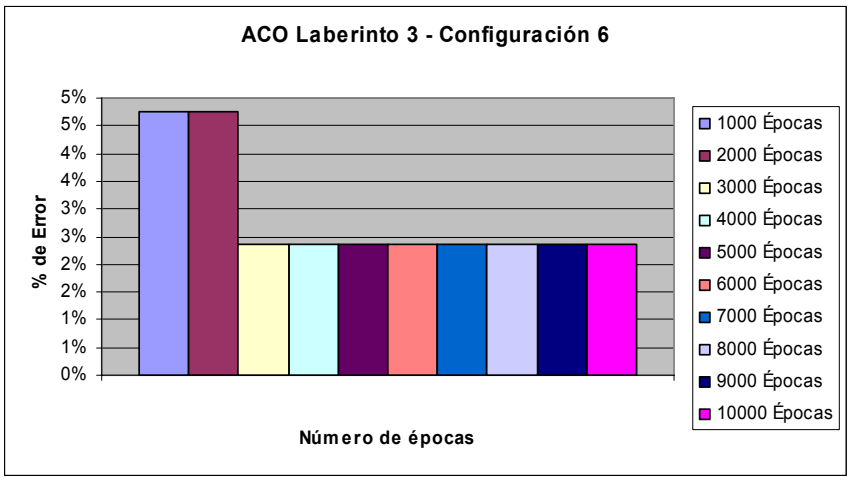


Figura 6. 45 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 6.

El error promedio mostrados en la Figura 6.46 aumenta del 0.0% al 2%. Después se eleva hasta un 12% y cae nuevamente al 0.0%. Este hecho es un comportamiento normal, obtenido ya en casos anteriores, dado a la naturaleza heurística del algoritmo utilizado.

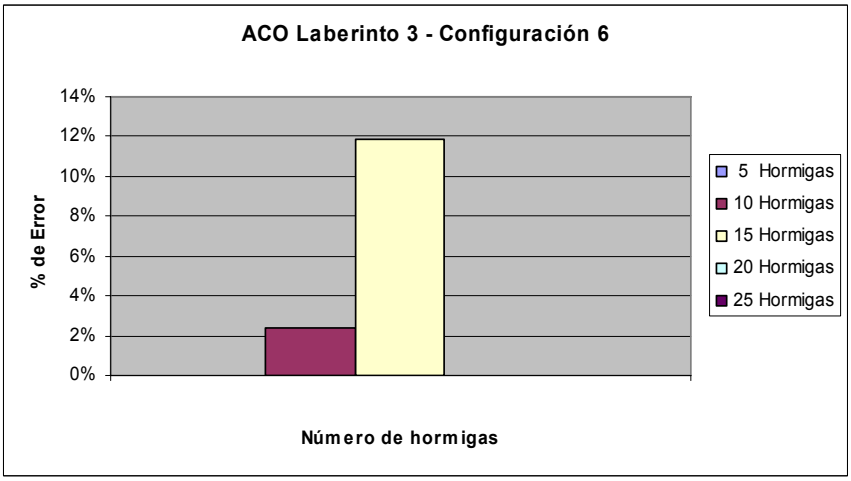


Figura 6. 46 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 6.

Experimento 1.3.7

Los porcentajes mostrados en la Tabla 6.21 son el resultado de aplicar el algoritmo 5.2 al laberinto 3, al utilizar la configuración 7 ($\gamma > \beta$), donde $\alpha = \beta = \gamma = 0.8$.

En la Tabla 6.21, se pude observar que los porcentajes de error más altos se encuentran en las primeras columnas (1000 a 3000 épocas), sin embargo el porcentaje de error que se mantiene constante se da con 15 hormigas.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%	11.86%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	11.86%	11.86%	11.86%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 21 Porcentajes de error en el laberinto 3, al utilizar la configuración 7.

En la Figura 6.47, se muestra que al aumentar el número de épocas a 4000, el porcentaje de error promedio disminuye de un 5% a un 2%.

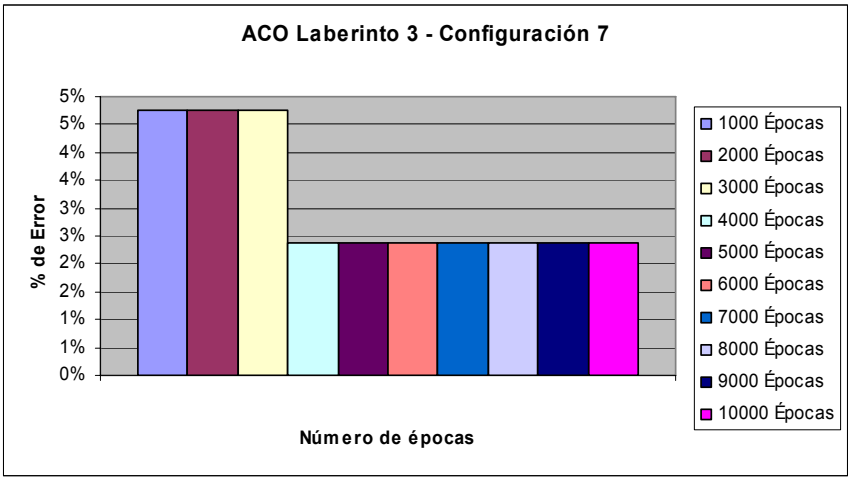


Figura 6. 47 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de épocas, al utilizar la configuración 7.

En la Figura 6.48 se tiene que los únicos valores de error diferentes al porcentaje óptimo ocurrieron con 15 hormigas (con un error máximo del 11.86%), y 25 hormigas.

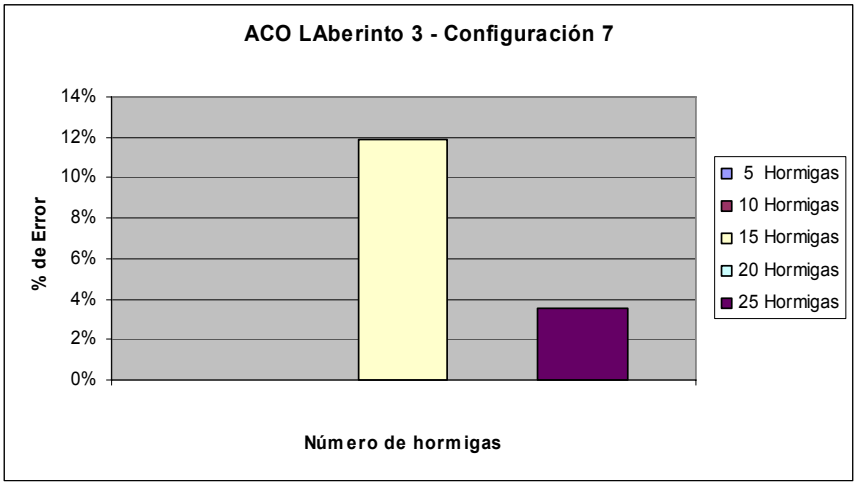


Figura 6. 48 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 3, al incrementar el número de hormigas, al utilizar la configuración 7.

Para ver el comportamiento general del algoritmo, bajo las 7 configuraciones se obtuvo un porcentaje de error promedio para cada una de las épocas. El error promedio disminuye al aumentar el número de épocas, como se puede ver en la Figura 6.49.

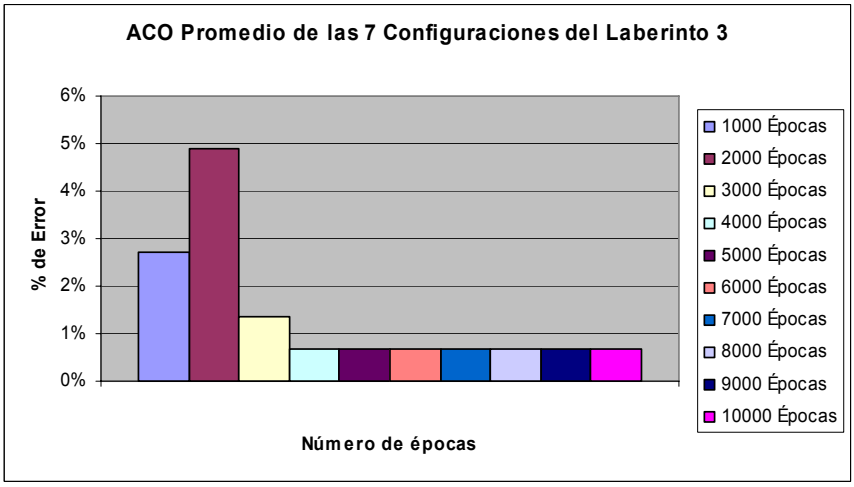


Figura 6. 49 Comportamiento del algoritmo propuesto en el laberinto 3 al incrementar el número de épocas, al utilizar el promedio de las 7 configuraciones.

La Figura 6.50, se realizó bajo las mismas condiciones que la anterior, se obtuvo el porcentaje de error al aumentar el número de hormigas. El error, en general, disminuye a pesar de tener para 15 hormigas. El máximo porcentaje de error promedio fue del 4%.

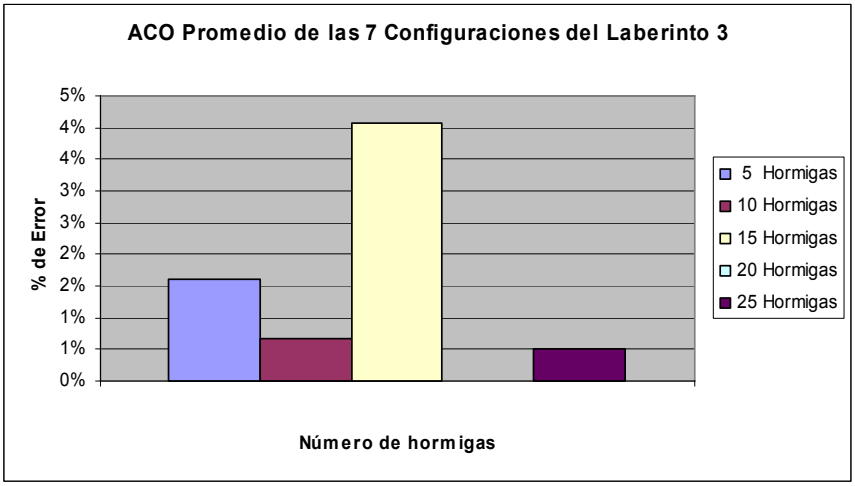


Figura 6. 50 Comportamiento del algoritmo propuesto en el laberinto 3 al incrementar el número de hormigas, al utilizar el promedio de las 7 configuraciones.

Experimento 1.4.1

Los porcentajes mostrados en la Tabla 6.22, son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al utilizar la configuración 1 ($\alpha > \gamma$), donde $\alpha = 2.5$, $\beta = 1.3$ y $\gamma = 0.23$.

Esta tabla muestra valores impresionantes, en el sentido de que el mismo valor del porcentaje de error (8.49%) se manifiesta en casi su totalidad en dicha tabla. La solución óptima se mantiene para 5 hormigas y dos casos más para 20 hormigas en 8000 y 10000 épocas. Un valor diferente se obtuvo en el máximo numero de hormigas (25) y 1000 épocas arrojando un valor del 12.06%.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
15	8.49%	8.46%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
20	8.49%	8.46%	8.49%	8.49%	8.49%	8.49%	8.49%	0.0%	8.49%	0.0%
25	12.06%	8.46%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%

Tabla 6. 22 Porcentajes de error en el laberinto 4, al utilizar la configuración 1.

A pesar de que los porcentajes de error promedio se mantienen constantes en algunas épocas (ver Figura 6.51), se puede ver que la tendencia del error es disminuir mientras se aumenta el número de iteraciones.

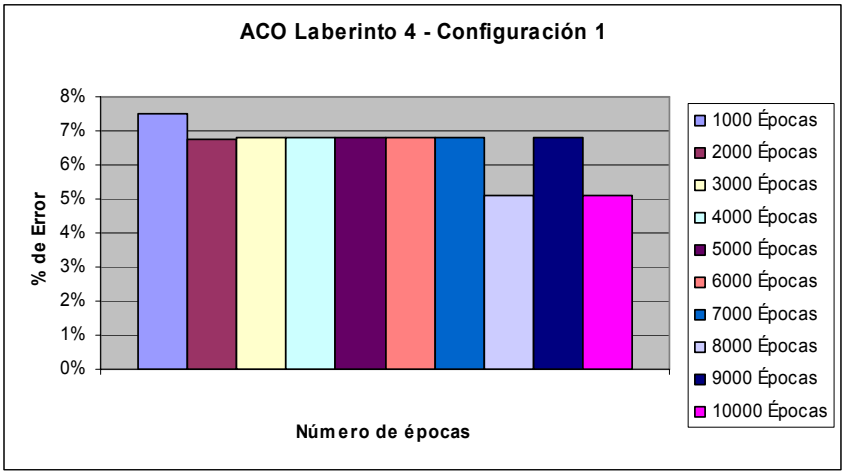


Figura 6. 51 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 1.

A pesar de tener porcentajes de error no óptimos, éstos son muy pequeños ya que se encuentran por debajo del 9%. Véase la Figura 6.52.

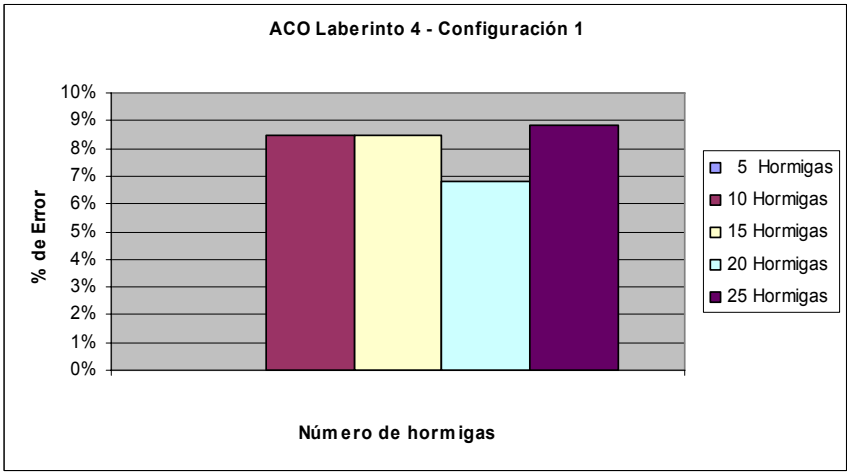


Figura 6. 52 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 1.

Experimento 1.4.2

Los porcentajes de error mostradas en la Tabla 6.23, son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al utilizar la configuración 2 ($\alpha > \beta$), donde $\alpha = 1.8$, $\beta = 1.0$ y $\gamma = 2.2$.

Para este caso se puede decir que con cada número de hormigas se obtuvo un porcentaje de error repetitivo o constante en todo el experimento a pesar de los dos valores en 15 hormigas - 1000 épocas y 10 hormigas y 10000 épocas.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
10	12.06%	12.06%	12.06 %	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	8.75%
15	8.49%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%
25	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%

Tabla 6. 23 Porcentajes de error en el laberinto 4, al utilizar la configuración 2.

En la Figura 6.53, se muestran los porcentajes de error promedio. Se puede ver también que al aumentar el número de épocas el error se reduce.

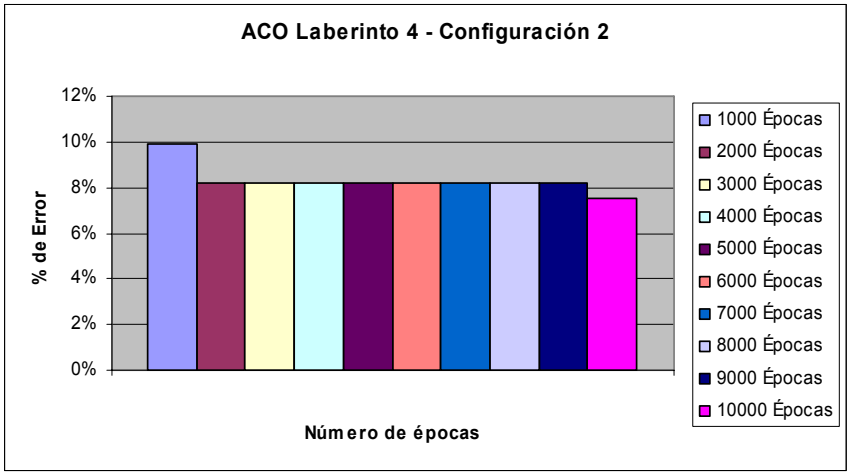


Figura 6. 53 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 2.

En la Figura 6.54 se pude ver que los porcentajes de error promedio varían mucho, mientras que con 15 hormigas el error es el mínimo.

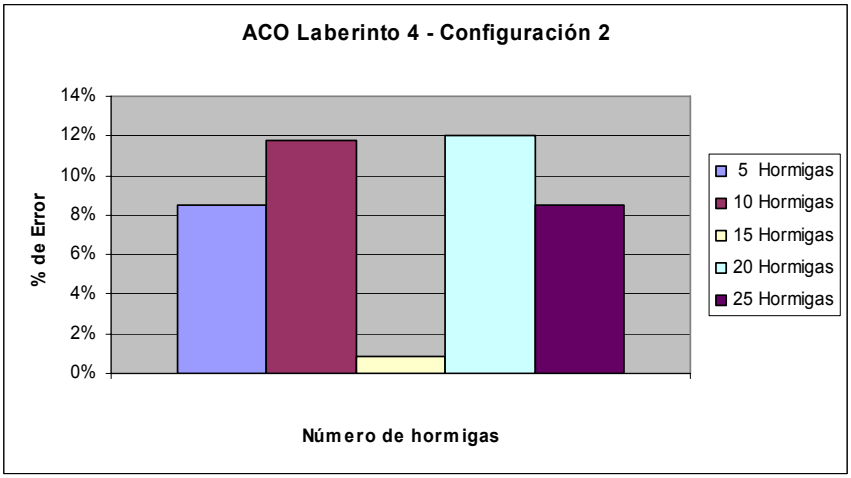


Figura 6. 54 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 2.

Experimento 1.4.3

Los porcentajes mostrados en la Tabla 6.24, son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al aplicar la configuración 3 ($\beta > \alpha$), donde $\alpha = 2.4$, $\beta = 3.0$ y $\gamma = 0.5$.

En la Tabla 6.24 se muestran los casos donde se obtuvo la solución óptima y donde se encontraron soluciones alternativas.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	29.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
15	0.0%	0.0%	8.49%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
25	29.49%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 24 Porcentajes de error en el laberinto 4, al utilizar la configuración 3.

La Figura 6.55 muestra valores que disminuyen al ir incrementando el número de épocas.

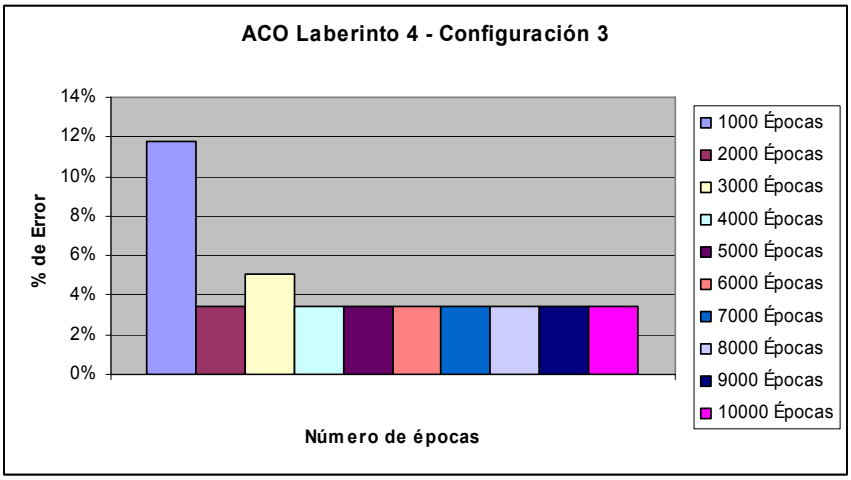


Figura 6. 55 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 3.

En la Figura 6.56 se puede observar que los mejores porcentajes de error se obtienen para el caso de 5 hormigas (el error promedio es 0.0%). Le sigue el error promedio para 15 y 25 hormigas.

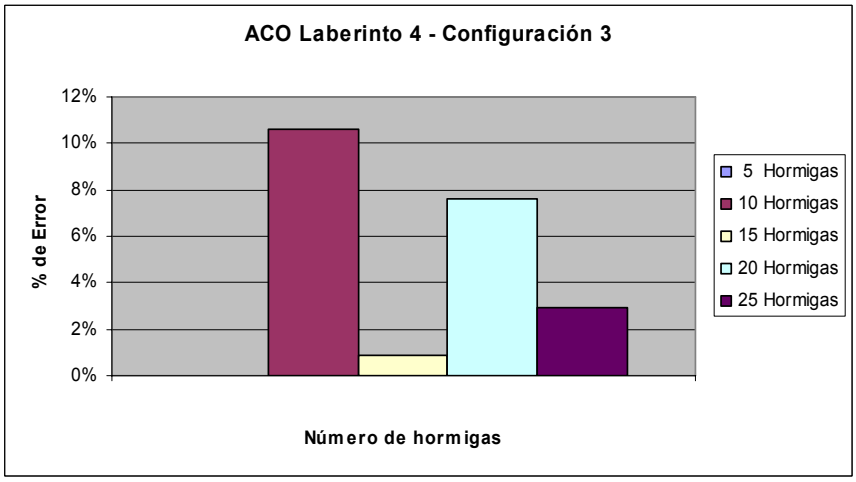


Figura 6. 56 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 3.

Experimento 1.4.4

Los porcentajes de error mostrados en la Tabla 6.25 son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al utilizar la configuración 4 ($\beta > \gamma$), donde $\alpha = 0.9$, $\beta = 2.81$ y $\gamma = 1.14$.

La Tabla 6.25 contiene buenos resultados, ya que en su mayoría indican haber encontrado el camino más corto, mostrando también el porcentaje 12.06 que no es muy alto. Nótese que este porcentaje se da en todos los casos al utilizar 10 hormigas, por el contrario en el caso de 5 hormigas con 1000 épocas, sé tiene el valor más alto de error.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	81.29%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 25 Porcentajes de error en el laberinto 4, al utilizar la configuración 4.

La Figura 6.57 muestra los porcentajes de error, los cuales disminuyen al aumentar el número de épocas a 2000.

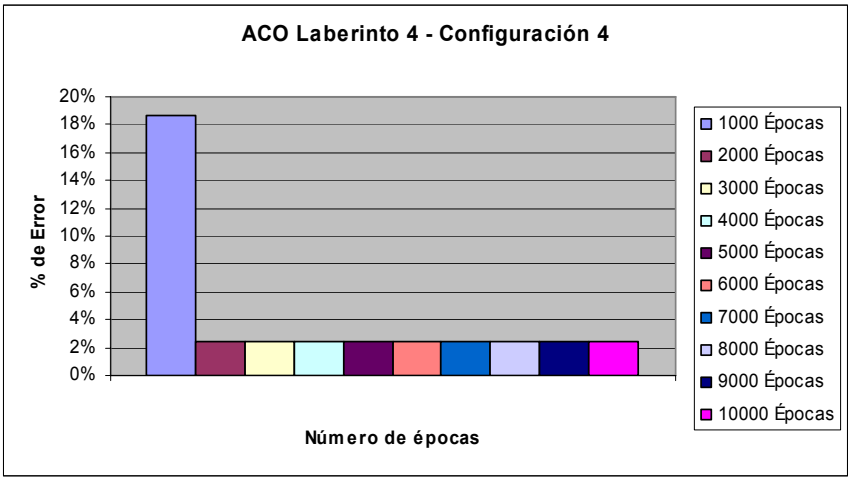


Figura 6. 57 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 4.

Los únicos porcentajes de error promedio registrados se dieron al utilizar 5 y 10 hormigas, en los demás casos se obtuvo una solución óptima, ver Figura 6.58.

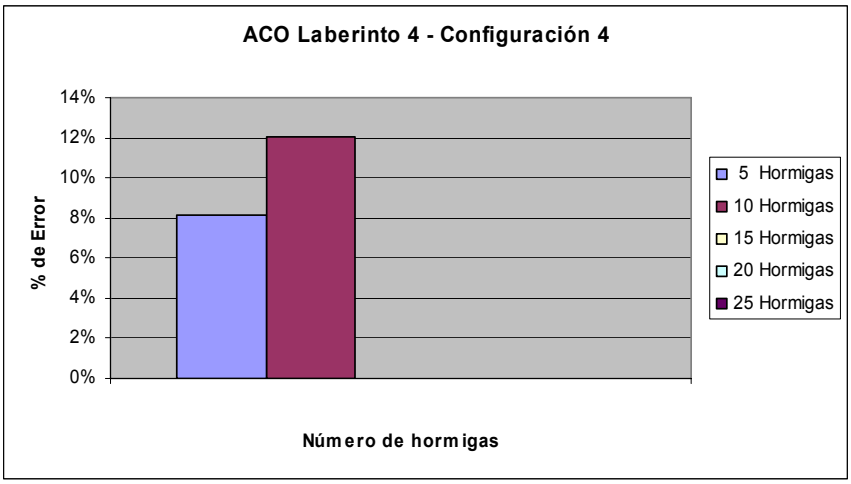


Figura 6. 58 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 4.

Experimento 1.4.5

Los porcentajes mostrados en la Tabla 6.26 son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al utilizar la configuración 5 ($\gamma > \alpha$), donde $\alpha = 0.13$, $\beta = 1.66$ y $\gamma = 0.97$.

Los porcentajes son muy variados ya que en el caso para 5 hormigas 1000 y 2000 épocas no se pudo encontrar alguna solución y para el resto de las épocas hubo un error del 100%. También se obtuvieron valores del 0.0% para el caso de 10 y 25 hormigas y otros valores de error que se mantuvieron constantes como es el caso de 15 y 20 hormigas.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	100%	100%	100%	100%	100%	100%	100%	100%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	0.0%
20	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%
25	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 26 Porcentajes de error en el laberinto 4, al utilizar la configuración 5.

La Figura 6.59 muestra que sólo con 10000 épocas el porcentaje de error promedio se pudo reducir.

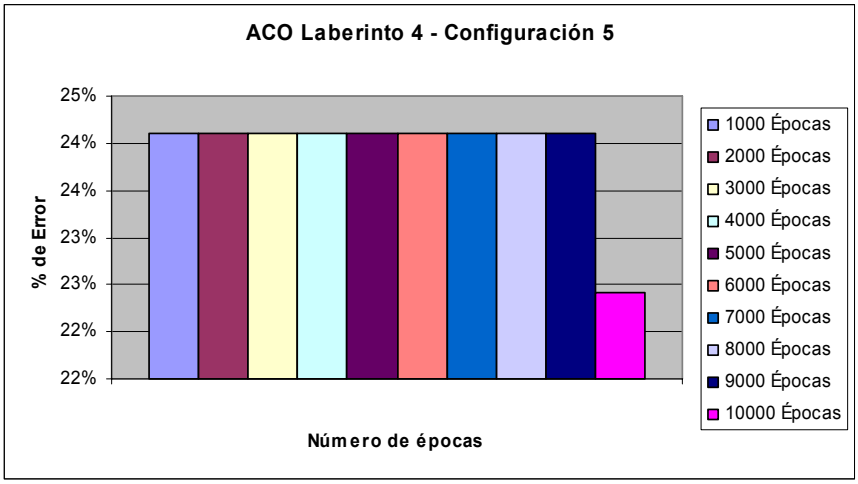


Figura 6. 59 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 5.

De la Figura 6.60, se puede observar que el porcentaje de error promedio se redujo al aumentar el número de hormigas, aún cuando para 15 y 20 hormigas se encontró un error pequeño.

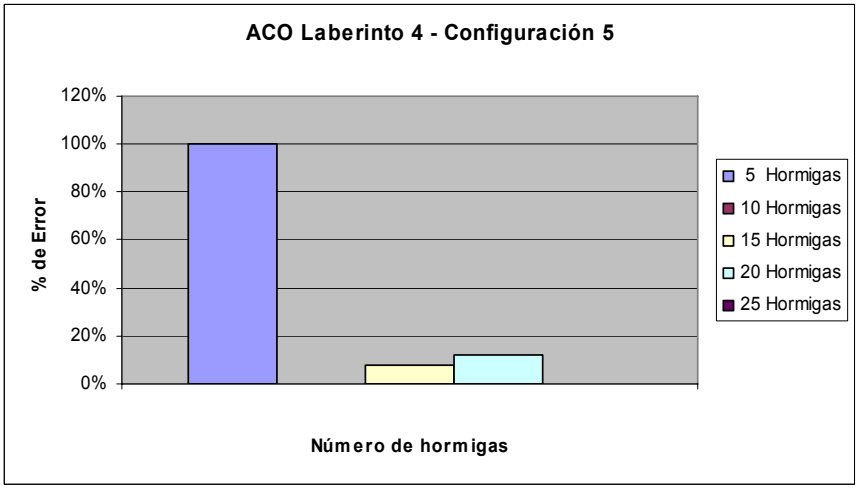


Figura 6. 60 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 5.

Experimento 1.4.6

Los porcentajes mostrados en la Tabla 6.27, son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al utilizar la configuración 6 ($\gamma > \beta$), donde $\alpha = 2.5$, $\beta = 1.7$ y $\gamma = 3.4$.

La Tabla 6.27 muestra mejores porcentajes que los mostrados en la tabla 6.26, lo cual indica que estos parámetros fueron mejores.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	31.04%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	8.49%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	12.06%	12.06%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	14.44%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%	12.06%

Tabla 6. 27 Porcentajes de error en el laberinto 4, al utilizar la configuración 6.

Dicha mejoría se pude observar en la Figura 6.61, donde se aprecian valores bajos en el error promedio. En la mayoría de las épocas, el error promedio no pasa del 2.41%.

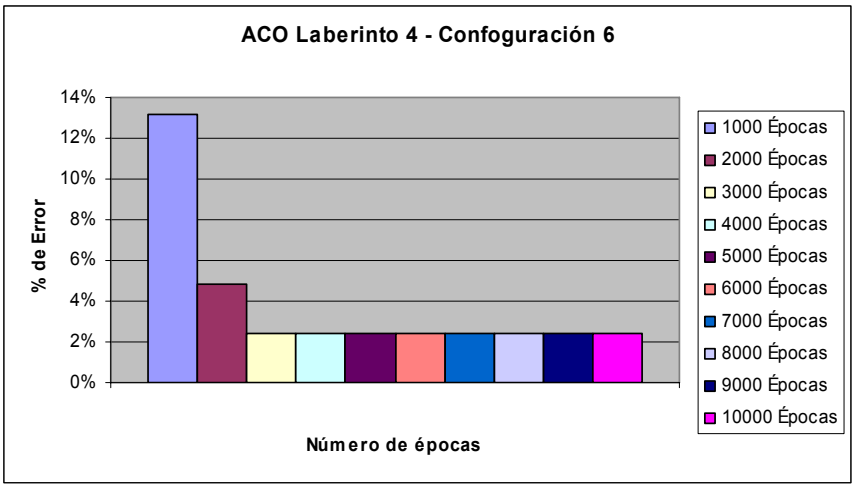


Figura 6. 61 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 6.

Nótese como también la Figura 6.62 muestra, valores bajos de error. Sin embargo, al utilizar el máximo número de hormigas (25) el error aumenta drásticamente.

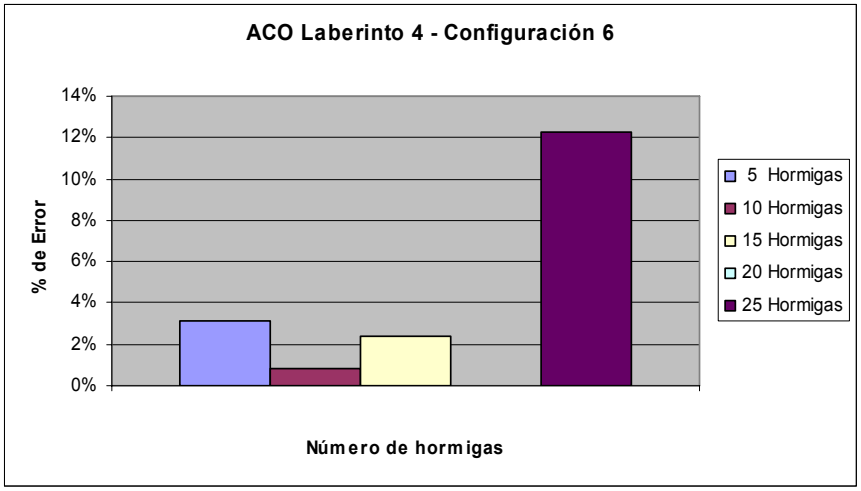


Figura 6. 62 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 6.

Experimento 1.4.7

Los porcentajes de error mostrados en la Tabla 6.28 son el resultado de aplicar el algoritmo 5.2 al laberinto 4, al utilizar la configuración 7 ($\gamma > \beta$), donde $\alpha = \beta = \gamma = 0.8$.

Los valores mostrados en esta tabla son bastante aceptables por su porcentaje de error bajo.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%	8.49%
25	12.06%	12.06%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 28 Porcentajes de error en el laberinto 4, al utilizar la configuración 7.

Al comparar los porcentajes de error con el número de épocas, se tienen valores bajos como resultado de aumentar el número de épocas, excepto el valor dado con 2000 épocas, ver Figura 6.63.

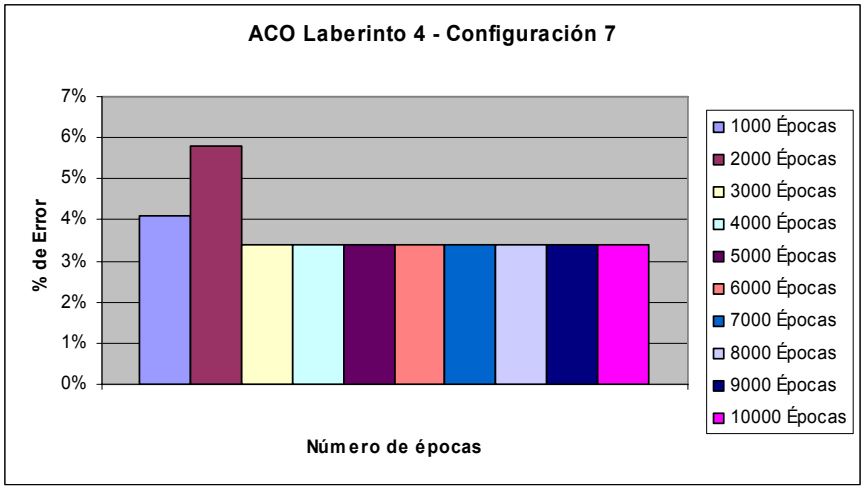


Figura 6. 63 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de épocas, al utilizar la configuración 7.

Como se pude ver en la Figura 6.64, los porcentajes de error promedio son muy variables. Con 5 hormigas se tiene un error promedio del 7.5%, Este error disminuye después hasta 0.0% volviendo aumentar con 20 hormigas al máximo error (8.49%). Al final vuelve a disminuir hasta un 2.5%. Esta variabilidad se da porque nuestro algoritmo es heurístico no mostrando el mismo comportamiento durante su ejecución.

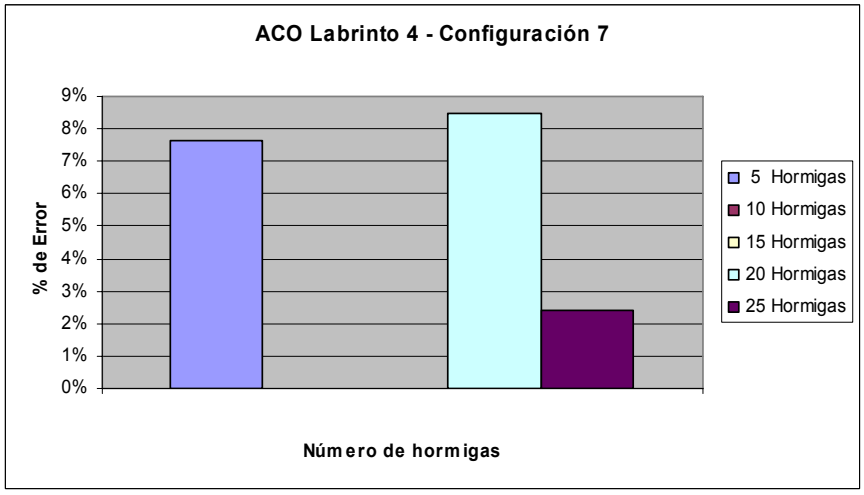


Figura 6. 64 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 4, al incrementar el número de hormigas, al utilizar la configuración 7.

Tomando en cuenta el promedio de los resultados de las 7 configuraciones para cada una de las épocas, se realizaron los siguientes experimentos.

Como se puede ver en la Figura 6.65, el porcentaje promedio de error disminuye al aumentar el número de épocas, manteniéndose así la tendencia de dicha comparación.

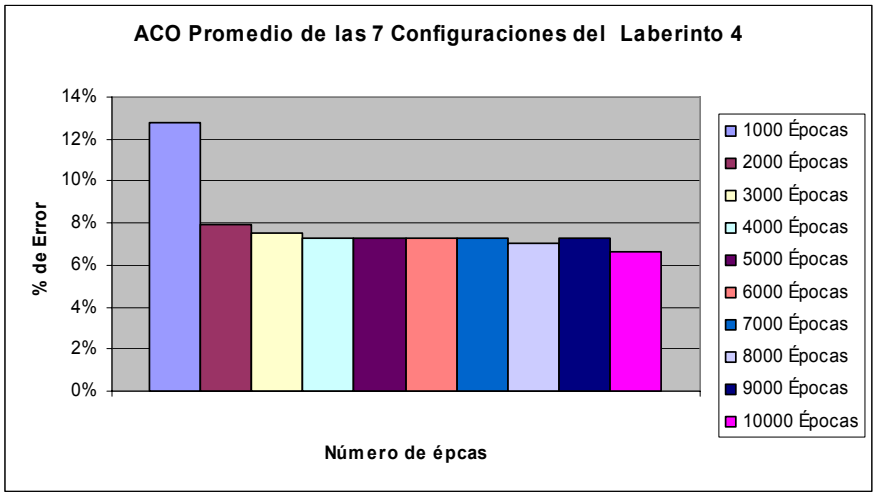


Figura 6. 65 Comportamiento del algoritmo propuesto en el laberinto 4 al incrementar el número de épocas, al utilizar el promedio de las 7 configuraciones.

La Figura 6.66 muestra también la disminución general, al aumentar el número de hormigas.

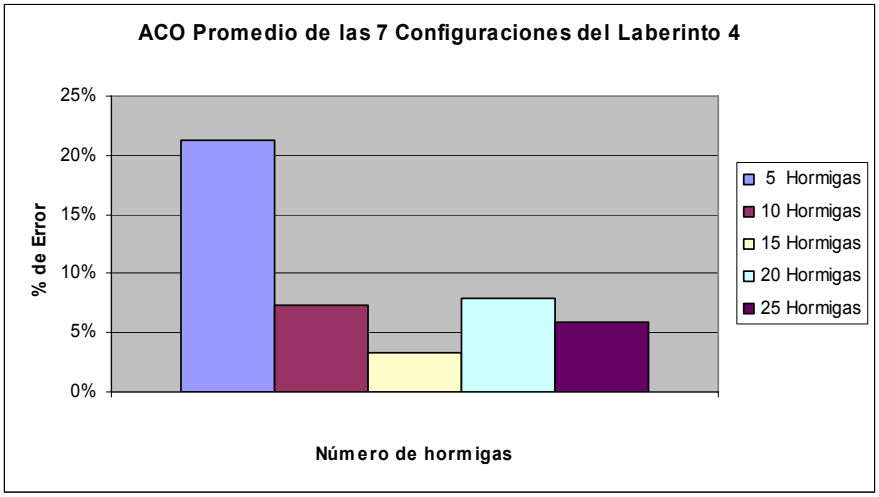


Figura 6. 66 Comportamiento del algoritmo propuesto en el laberinto 4 al incrementar el número de hormigas, al utilizar el promedio de las 7 configuraciones.

Experimento 1.5.1

Los porcentajes de error mostrados en la Tabla 6.29 son el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 1 ($\alpha > \gamma$), donde $\alpha = 2.5$, $\beta = 1.3$ y $\gamma = 0.23$. Los porcentajes muestran que los parámetros elegidos para este caso no ayudan a encontrar siempre la solución óptima. En algunos casos no se encontró una solución o el error fue diferente al óptimo.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	0.0%	X	X	X	X	X	X
10	X	8.19%	8.19%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	X
15	X	8.19%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	8.19%	0.0%	0.0%	0.0%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%
25	X	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 29 Porcentajes de error en el laberinto 5, al utilizar la configuración 1.

El peor porcentaje de error se obtuvo con 1000 épocas. Esta fue de un poco más del 80% de error. Los valores de los parámetros ayudan satisfactoriamente, al aumentar el número de épocas; a pesar de que trabajar con 10000 épocas vuelva a aumentar el error promedio.

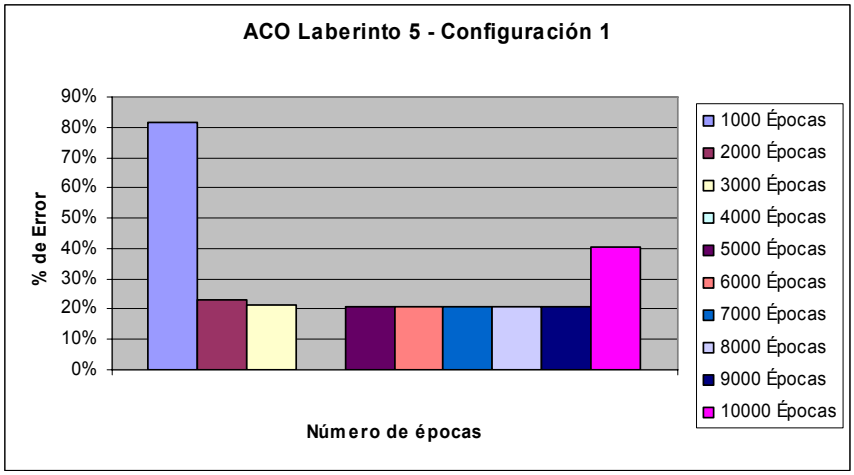


Figura 6. 67 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 1.

Los peores resultados se obtuvieron al utilizar 5 hormigas. El error promedio fue de un 90%. En los demás casos el error se mantuvo por debajo del 21%, véase la Figura 6.68.

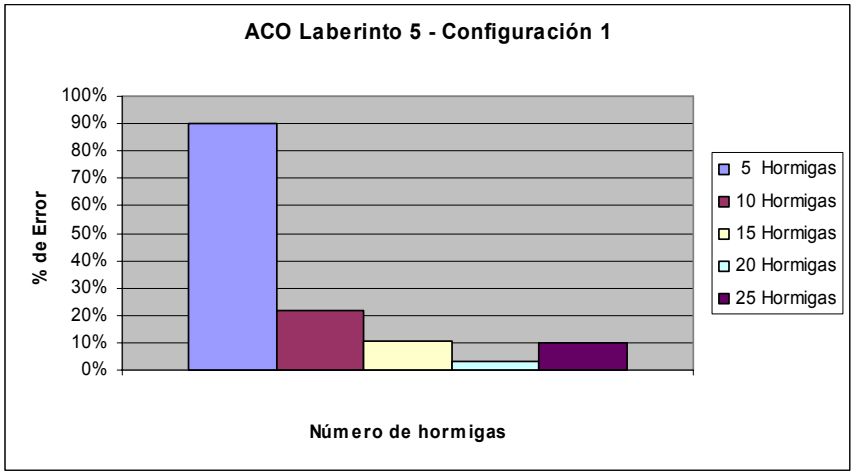


Figura 6. 68 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 1.

Experimento 1.5.2

Los porcentajes de error mostrados en la Tabla 6.30 son el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 2 ($\alpha > \beta$), donde $\alpha = 1.8$, $\beta = 1.0$ y $\gamma = 2.2$.

Con estos parámetros se obtuvieron los valores de la siguiente tabla, los cuales muestran un comportamiento muy variable.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	X	3.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	17.58%	17.58%	0.0%	17.58%	17.58%	17.58%	17.58%	17.58%	17.58%	17.58%
25	X	8.19%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%

Tabla 6. 30 Porcentajes de error en el laberinto 5, al utilizar la configuración 2.

La Figura 6.69 muestra que con 1000 épocas se dio el máximo error. Con el número de épocas restantes el error promedio disminuyó drásticamente en toda la experimentación.

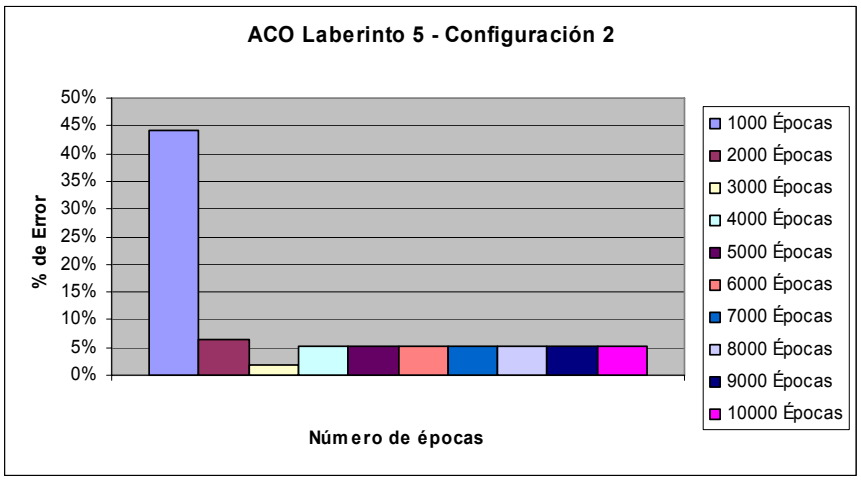


Figura 6. 69 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 2.

A diferencia de otras Figuras, en la Figura 6.70 se muestra un comportamiento inverso, ya que al aumentar el número de hormigas en la experimentación, el error promedio aumentó.

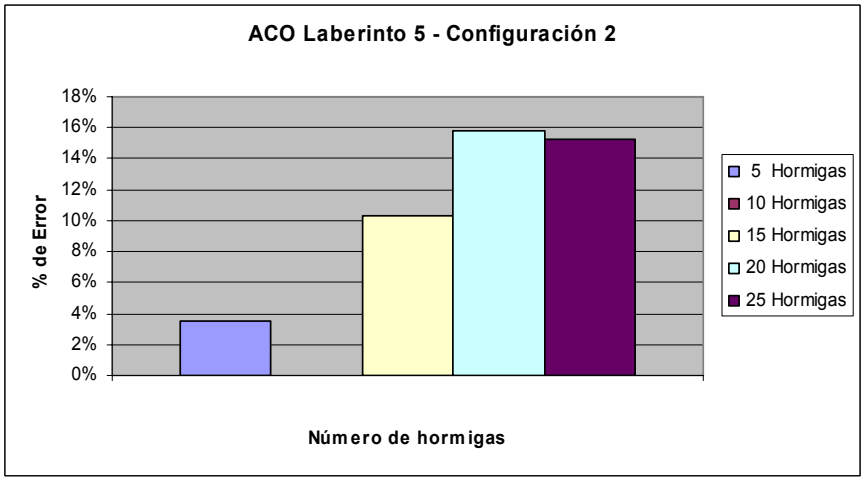


Figura 6. 70 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 2.

Experimento 1.5.3

Los porcentajes de mostrados en la Tabla 6.31 son el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 3 ($\beta > \alpha$), donde $\alpha = 2.4$, $\beta = 3.0$ y $\gamma = 0.5$.

La Tabla 6.31 muestra porcentajes de error que se repiten durante todo el experimento y son bajos. También para 1000 épocas se puede ver que en dos ocasiones no se encontró una solución (al utilizar 5 y 25 hormigas).

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	3.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	3.46%	0.0%
10	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	0.0%	3.46%
15	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	8.19%	8.19%	8.19%	8.19%	8.19%	8.19%	8.19%	8.19%	0.0%	8.19%
25	X	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 31 Porcentajes de error en el laberinto 5, al utilizar la configuración 3.

La Figura 6.71 muestra que el valor a partir de 2000 épocas en adelante ayuda a mantener y disminuir bastante el porcentaje de error promedio.

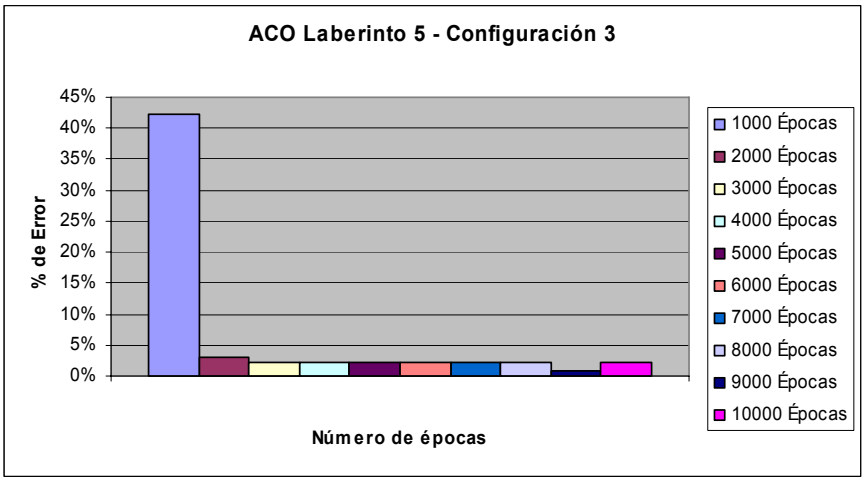


Figura 6. 71 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 3.

Un comportamiento variable se obtuvo con el algoritmo al emplear los parámetros dados al principio de este experimento, ya que al alterar el número de hormigas a utilizar, el

algoritmo arrojó valores muy variados, pero que se mantuvieron por debajo del 10% de error promedio, ver Figura 6.72.

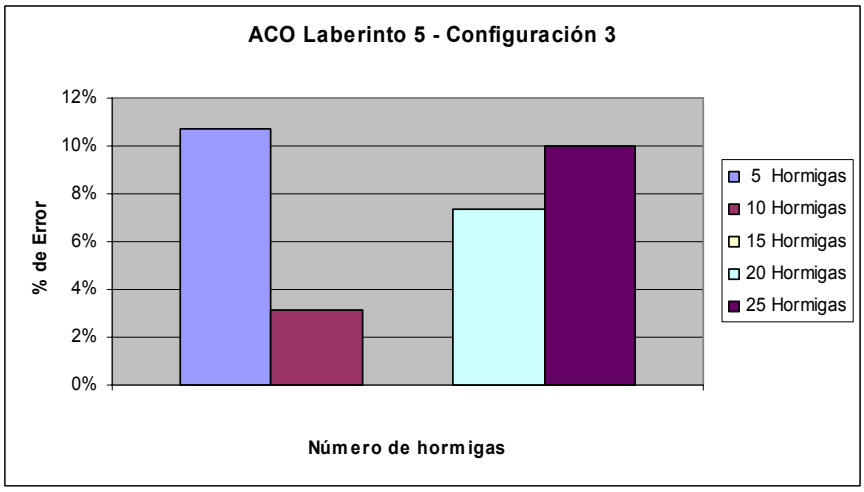


Figura 6. 72 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 3.

Experimento 1.5.4

Los porcentajes de error mostrados en la Tabla 6.32 son el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 4 ($\beta > \gamma$), donde $\alpha = 0.9$, $\beta = 2.81$ y $\gamma = 1.14$. Lo interesante de esta tabla es que a pesar de haber encontrado la solución óptima y otras soluciones, en el caso de 5 hormigas no se pudo obtener una solución con ninguna de las épocas con las que se experimentó.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	X	X	X	X	X	X	X
10	X	8.19%	8.19%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%	5.61%
20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
25	5.61%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 32 Porcentajes de error en el laberinto 5, al utilizar la configuración 4.

Nuevamente el comportamiento del porcentaje de error promedio tiende a disminuir conforme aumenta el número de épocas. Este fenómeno sucede porque se le da más tiempo

al algoritmo de encontrar una solución óptima; claro está que también depende de los valores que se le dieron a los parámetros, ver Figura 6.73.

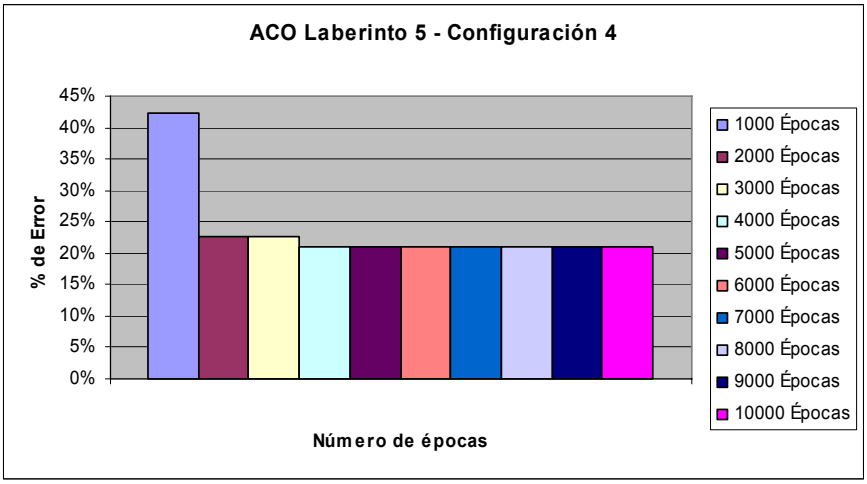


Figura 6. 73 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 4.

La Figura 6.74 muestra el error promedio obtenido con el número de hormigas indicado. Como se puede ver para 5 hormigas el error promedio es 100%, debido a las X obtenidas en la tabla correspondiente (ver Tabla 6.32), sin embargo el error promedio disminuye al aumentar la cantidad de hormigas.

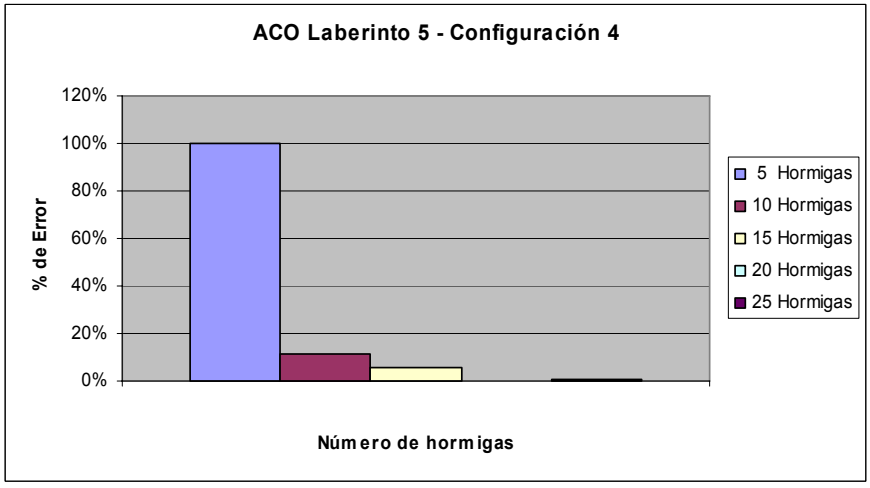


Figura 6. 74 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 4.

Experimento 1.5.5

Los porcentajes de error mostrados en la Tabla 6.33 son el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 5 ($\gamma > \alpha$), donde $\alpha = 0.13$, $\beta = 1.66$ y $\gamma = 0.97$.

La Tabla muestra la gran cantidad de error en la búsqueda del camino más corto, sin embargo, como se puede ver, se obtiene la solución óptima.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	X	X	X	X	X	X	X	X	X
10	X	X	8.19%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%
15	X	X	5.61%	X	X	X	X	X	3.46%	3.46%
20	X	X	0.0%	X	17.58%	17.58%	17.58%	17.58%	17.58%	17.58%
25	3.46%	0.0%	3.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 33 Porcentajes de error en el laberinto 5, al utilizar la configuración 5.

Como se ha mencionado en experimentos anteriores, se puede ver que la tendencia de los porcentajes de error promedio disminuye al aumentar el número de épocas. Sin embargo en algunos casos se altera esta tendencia teniendo altos valores en diferentes casos; como los mostrados en la Figura 6.75.

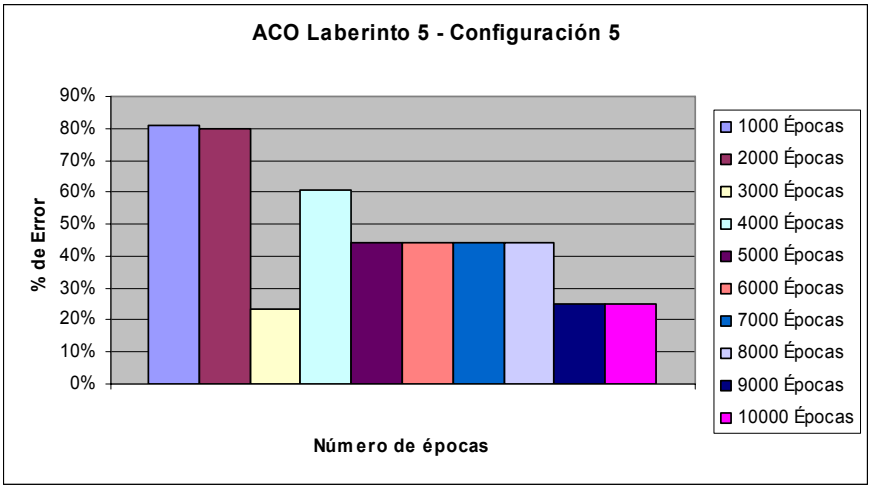


Figura 6. 75 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 5.

El mejor número de hormigas que ayudó a tener una mayor eficiencia fue 25 hormigas, con un 1% de error promedio, ver Figura 6.76.

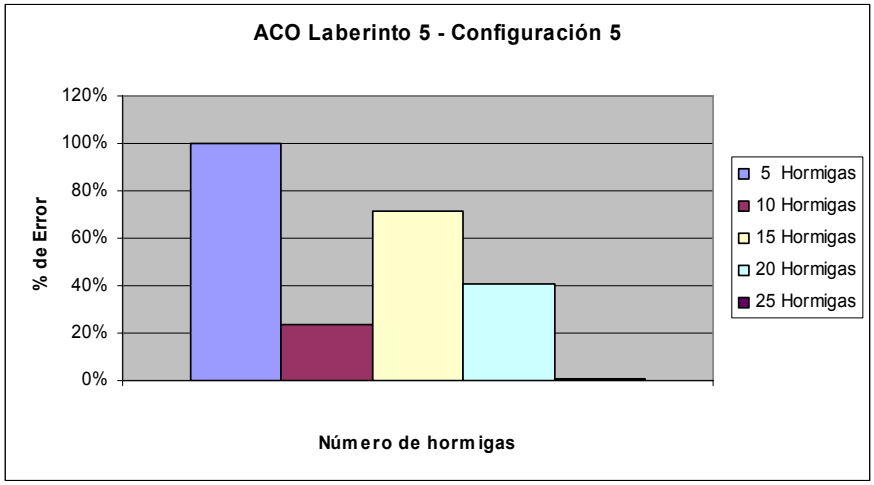


Figura 6. 76 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 5.

Experimento 1.5.6

Los porcentajes de error de la Tabla 6.34 son el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 6 ($\gamma > \beta$), donde $\alpha = 2.5$, $\beta = 1.7$ y $\gamma = 3.4$.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	X	3.46%	3.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
15	X	X	29.26%	29.26%	29.26%	29.26%	29.26%	29.26%	29.26%	29.26%
20	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%	3.46%
25	0.0%	12.06%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Tabla 6. 34 Porcentajes de error en el laberinto 5, al utilizar la configuración 6.

Como se puede ver en la Figura 6.77, el error promedio máximo fue de 41% para 1000 épocas, después se redujo a 24% en 2000. Finalmente se mantuvo constante en un 6.5%.

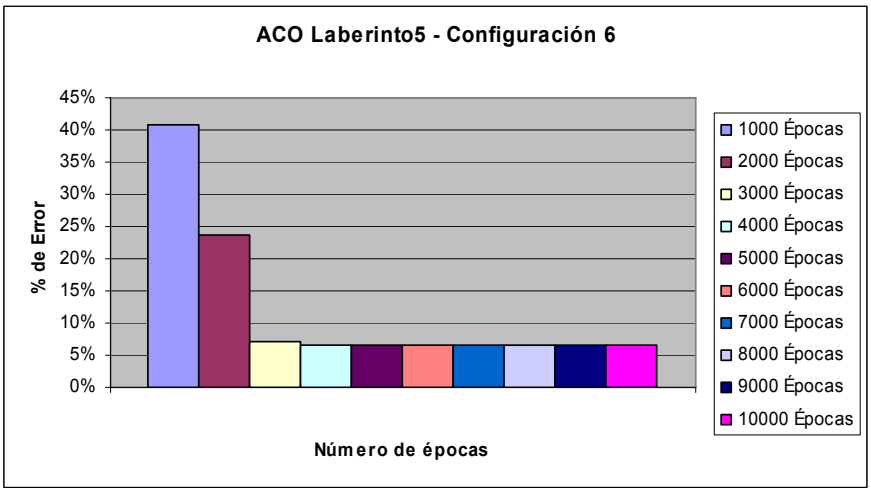


Figura 6. 77 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 6.

La Figura 6.78 muestra porcentajes de error variables, los cuales se obtuvieron con los diferentes números de hormigas. Se puede apreciar un porcentaje del 0.0% para 10 hormigas y uno de 42% para 15, los demás errores promedio fueron bajos.

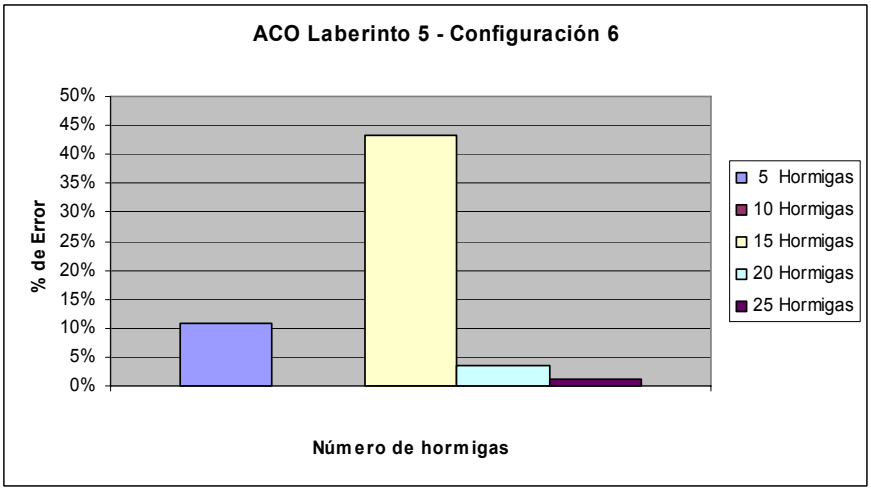


Figura 6. 78 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 6.

Experimento 1.5.7

Los porcentajes de error mostrados en la Tabla 6.35 se obtuvieron como el resultado de aplicar el algoritmo 5.2 al laberinto 5, al utilizar la configuración 7 ($\gamma > \beta$), donde $\alpha = \beta = \gamma = 0.8$.

La Tabla 6.35 muestra errores, en su mayoría del 0.0% lo cual indica que se encontró el camino más corto.

M	1000 épocas	2000 épocas	3000 épocas	4000 épocas	5000 épocas	6000 épocas	7000 épocas	8000 épocas	9000 épocas	10000 épocas
5	0.0%	3.46%	3.46%	3.46%	0.0%	3.46%	3.46%	3.46%	0.0%	3.46%
10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	12.38%	0.0%
15	3.46%	3.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20	5.61%	5.61%	5.61%	5.61%	3.46%	3.46%	0.0%	0.0%	0.0%	0.0%
25	X	3.46%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	X	0.0%

Tabla 6. 35 Porcentajes de error en el laberinto 5, al utilizar la configuración 7.

La Figura 6.79 muestra que después de las 1000 épocas, existe una disminución notable en el porcentaje de error, sin embargo para el caso de 9000 épocas el error nuevamente se dispara hasta un 22%.

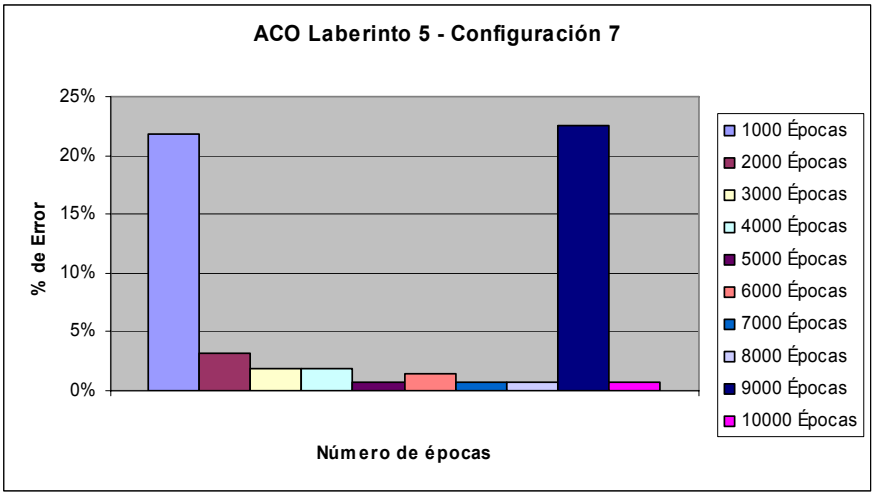


Figura 6. 79 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de épocas, al utilizar la configuración 7.

La Figura 6.80 muestra cómo el error promedio se encuentra en un 2% con 5 hormigas, luego disminuye hasta las 20 hormigas, después vuelve a aumentar hasta un 2.94% y con 25 hormigas el error promedio es de 20%. El error promedio nuevamente varia como en anteriores experimentos.

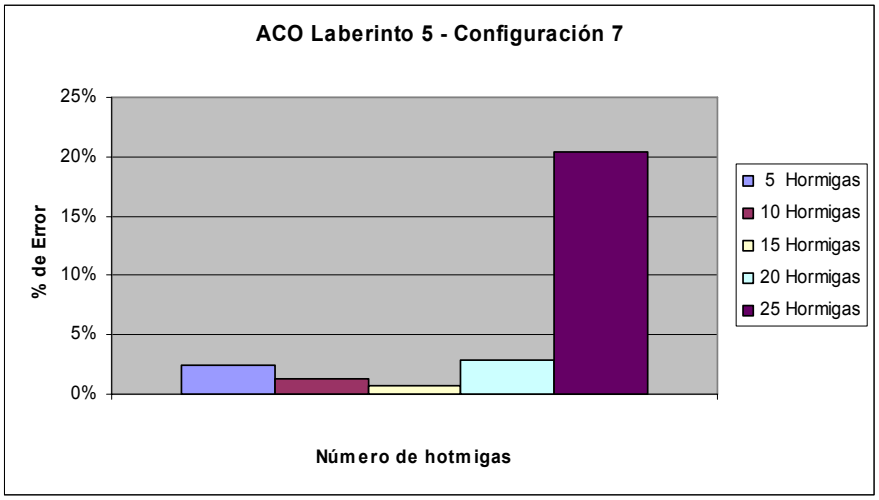


Figura 6. 80 Comportamiento general del algoritmo propuesto (ACO) en el laberinto 5, al incrementar el número de hormigas, al utilizar la configuración 7.

Tomando en cuenta el promedio de los resultados de las 7 configuraciones para cada una de las épocas, el porcentaje promedio de error disminuye al aumentar el número de épocas de un 50% hasta un 15%, ver Figura 6.81.

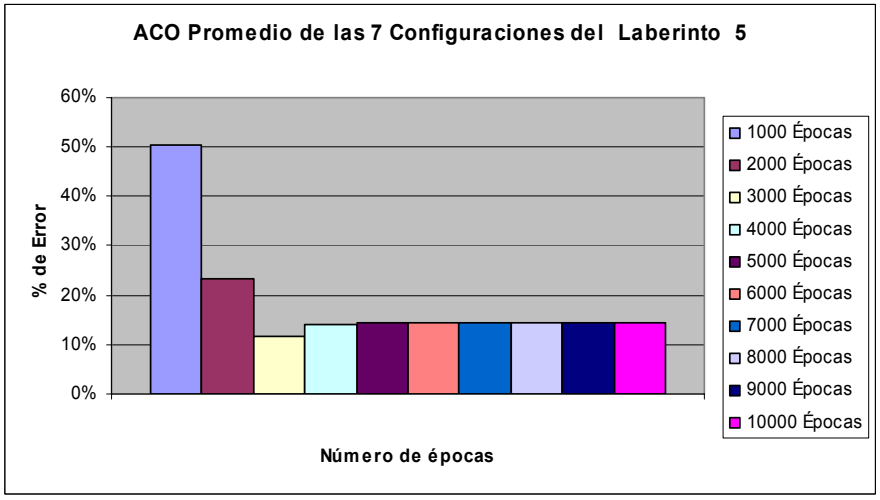


Figura 6. 81 Comportamiento del algoritmo propuesto en el laberinto 5 al incrementar el número de épocas, al utilizar el promedio de las 7 configuraciones.

Al trabajar con una cantidad mayor de hormigas se tiene que el porcentaje de error promedio disminuye; este porcentaje es el promedio de error de las 7 configuraciones para el laberinto 5, véase Figura 6.82.

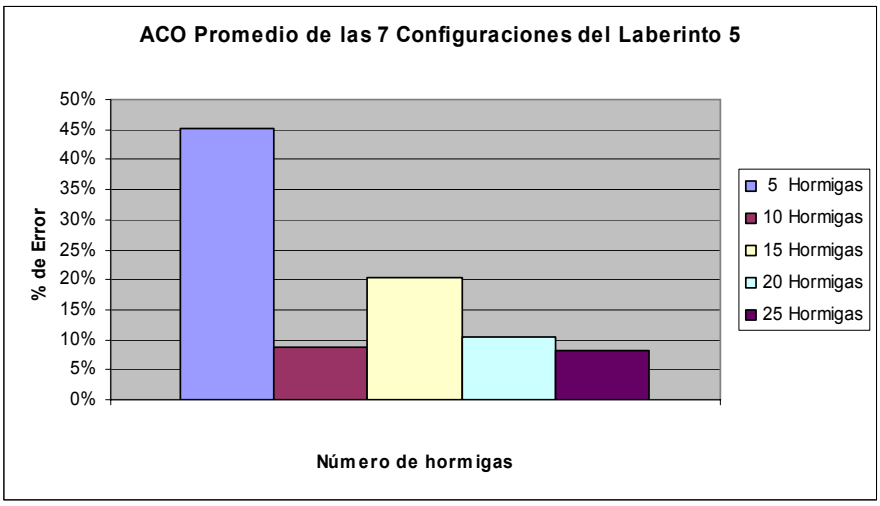


Figura 6. 82 Comportamiento del algoritmo propuesto en el laberinto 5 al incrementar el número de hormigas, al utilizar el promedio de las 7 configuraciones.

A continuación en las Figuras6.83 y 6.84 se muestra el resultado de promediar los porcentajes promedio de error de los 5 laberintos con sus respectivas 7 configuraciones. La Figura 6.83 muestra nuevamente que al aumentar el número de épocas en los 5 laberintos, el porcentaje de error disminuye notablemente.

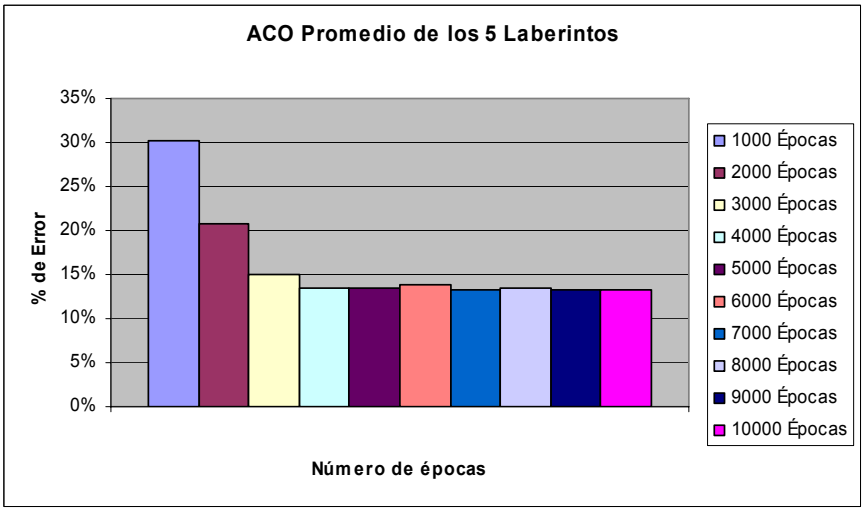


Figura 6. 83 Comportamiento del algoritmo propuesto en los 5 laberintos al incrementar el número de épocas.

Al promediar el porcentaje de error de los 5 laberintos y comparar con el número de hormigas se obtuvo que disminuyó el error de manera general, sin embargo para 15 y 25 hormigas aumentó el error promedio. Para los 5 laberintos la mejor configuración fue trabajar con 20 hormigas ya que el error fue de un 10%, ver Figura 6.84.

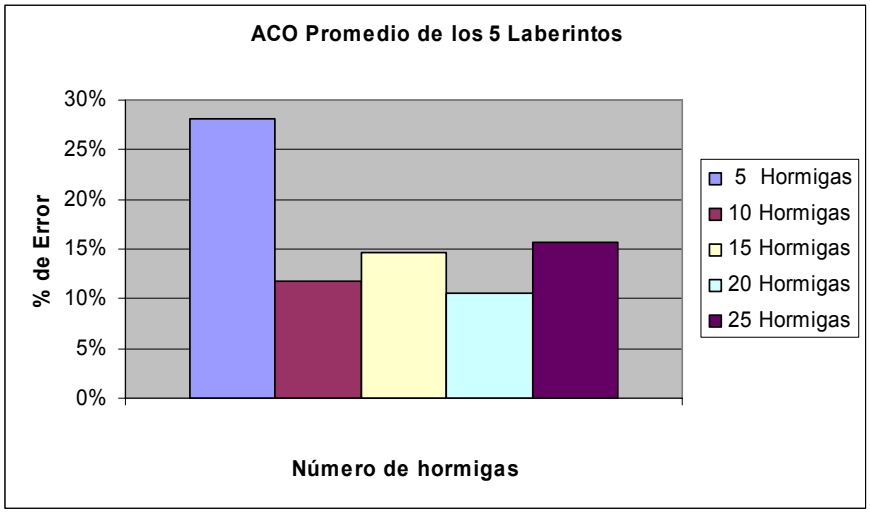


Figura 6. 84 Comportamiento del algoritmo propuesto en los 5 laberintos al incrementar el número de hormigas.

Como se observó a lo largo de la primera experimentación, el algoritmo propuesto 5.2 trabaja de manera eficiente debido a dos características: a los valores de los parámetros y el entorno en el que se trabajó.

Si los valores de los parámetros no son adecuados para el tipo de entorno, se pueden encontrar porcentajes de error altos o quizá ninguna solución. Para esta primer experimentación, algunos valores de parámetros fueron excelentes candidatos ya que con ellos se encontraba el camino más corto. En algunos casos los porcentajes de error eran muy bajos, o ni siquiera se pudo encontrar una solución alterna.

Los resultados mostrados en las Figuras 6.3 a la 6.84 dan evidencia importante de que, en general, el algoritmo 5.2 presenta un desempeño aceptable para los 5 laberintos con las 7 configuraciones diferentes ya mencionadas.

Dos comportamientos importantes se descubrieron en ésta primera experimentación:

- En general, al aumentar el número de épocas el porcentaje de error tiende a disminuir. Esto se debe a que se le da más tiempo al algoritmo de realizar su búsqueda o mejor dicho las “hormigas” tienen más tiempo de explorar el entorno y escoger la mejor solución, (en este caso encontrar el camino más corto y no tomar soluciones sub-óptimas).

- Al aumentar el número de hormigas, el porcentaje de error disminuye, ya que entre más hormigas, más rápido se explora el entorno, y más si es un entorno grande y con muchas conexiones posibles.

Claro que hubo casos en el que estas dos observaciones no se cumplieron. En uno que otro valor rompió con la tendencia. Esto no significa que esté mal. Como ya se mencionó anteriormente los valores de los parámetros combinada con un adecuado número de épocas pueden formar una buena o mala solución. Esto se da ya que el algoritmo es completamente heurístico y simula un comportamiento natural.

6.1.2 Aplicación del algoritmo ACO-AG

El segundo experimento consistió en utilizar el algoritmo 5.3 (la combinación del algoritmo de optimización por colonia de hormigas con el algoritmo genético).

Como se explicó en el sección 5.5, el algoritmo 5.3 ayuda a optimizar los valores de los parámetros α , β y γ . En este experimento se aplica el algoritmo 5.3 a cada uno de los 5 laberintos con el fin de obtener las mejores configuraciones y el número de hormigas necesarias que harán que encontremos el camino más corto en cada uno de los laberintos en un tiempo más reducido.

A continuación se presentan las Tablas con las mejores 5 configuraciones para cada uno de los entornos evaluados. En la primera columna se puede ver los parámetros de las configuraciones α , β y γ con las cuales se garantiza encontrar el camino más corto, *m* es el número de hormigas que se requirió para encontrar la solución óptima, *Ep* es el número de épocas que se requirió para llegar a la solución, es decir el número de iteraciones y *mseg* es el tiempo en milisegundos que tarda nuestro algoritmo en encontrar la solución óptima.

Experimento 2.1

La Tabla 6.36 muestra las mejores 5 configuraciones para el laberinto 1

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
α	-3.5	-1.6	-0.8	-3.7	1.1
β	1.79	1.5	2.599	1.6	2.29
γ	1	-0.3	1.7	-2.5	0.3
m	22	16	20	16	10
Ep	657	2184	1293	2428	3370
mseg.	1090	1020	1196	2229	5292

Tabla 6. 36 Algunas configuraciones óptimas para el Laberinto 1.

La Figura 6.37 muestra que entre mayor número de hormigas, el número de épocas se reduce considerablemente. Con el mínimo de hormigas (10), se obtiene 3370 *Ep* y con el máximo número de hormigas (22) el número de épocas se reduce a 657.

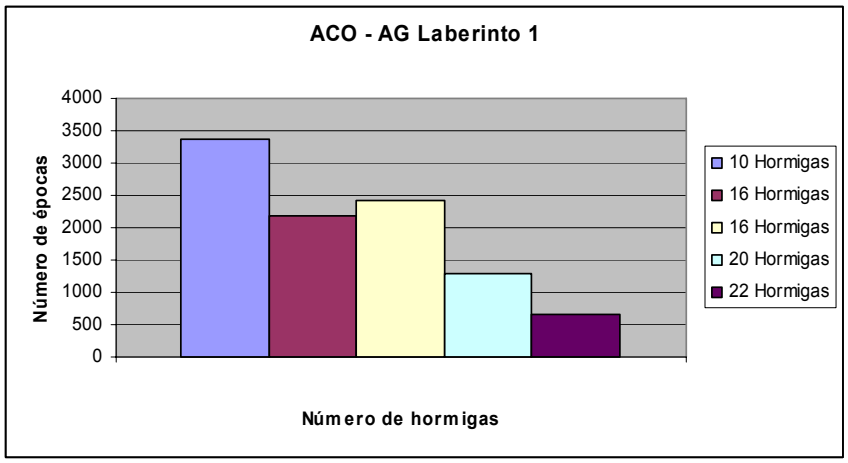


Figura 6. 85 Tiempo que tarda el algoritmo propuesto combinado con el Algoritmo Genético en el laberinto 1 al incrementar el número de hormigas.

Experimento 2.2

La siguiente Tabla muestra las mejores 5 configuraciones para el laberinto 2

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
α	-4.599	-3.7	0.1	4	-1.6
β	1.1	-2.7	-3.299	2.599	1.299
γ	-1.6	0.699	-2	1.7	3.5
m	29	15	15	17	16
Ep	105	643	794	196	2166
mseg.	13	273	277	40	860

Tabla 6. 37 Algunas configuraciones óptimas para el Laberinto 2.

Como se puede ver en la Figura 6.86, al utilizar los valores de la tabla anterior, y al recurrir al mayor número de hormigas, el resultado se encontró en un número de épocas muy pequeño siguiéndole el de 17 hormigas; las demás cantidades de hormigas que son menores se puede ver que el número de iteraciones aumenta.

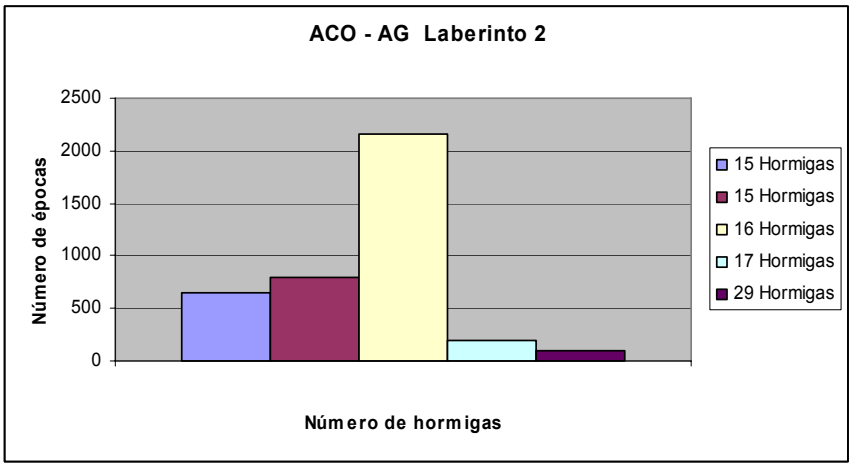


Figura 6. 86 Tiempo que tarda el algoritmo propuesto combinado con el Algoritmo Genético en el laberinto 2 al incrementar el número de hormigas.

Experimento 2.3

La Tabla 6.38 muestra las mejores 5 configuraciones que se obtuvieron de realizar respectivamente 5 corridas del algoritmo sobre el laberinto 3. En todos los casos se encontró el camino más corto al utilizar las configuraciones correspondientes.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
α	1.1	3.2	1.899	-2.9	2.599
β	1	-1.899	-1.799	-3	-3
γ	4.4	2.9	-2.099	2	-0.899
m	21	17	22	15	19
Ep	127	2217	138	6000	129
mseg.	70	414	79	3225	67

Tabla 6. 38 Algunas configuraciones óptimas para el Laberinto 3.

La Figura 6.87 muestra la cantidad de número de épocas (*Ep*) que se requirió para encontrar la solución óptima al variar el número de hormigas (m). Los datos presentados corresponden a la tabla anterior en el orden correspondiente.

Lo que se observa es que al aumentar el número de hormigas, el número de épocas disminuye, por ejemplo, con 21 hormigas (cuarta barra), la presenta un valor de 127 épocas

(corrida 1); con 19 hormigas (tercera barra) hay 129 épocas, con 15 hormigas (primera barra) el número de épocas asciende a 6000.

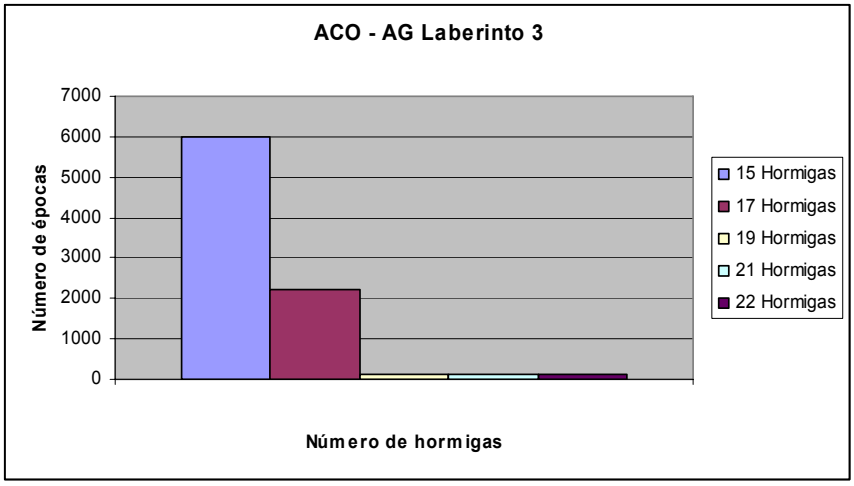


Figura 6. 87 Tiempo que tarda el algoritmo propuesto combinado con el Algoritmo Genético en el laberinto 3 al incrementar el número de hormigas.

Experimento 2.4

La Tabla 6.39 muestra el resultado de aplicar el algoritmo 5.3 al laberinto 4. Para las 5 corridas se encontraron las mejores 5 configuraciones respectivamente, tal como lo muestra esta Tabla.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
α	0.4	1	2.2	4.5	1.299
β	0.8	5	-4.1	-2	2.299
γ	3	3	0.98	2.79	0.8
m	16	20	18	14	13
Ep	210	3239	10437	3261	260
mseg.	170	433	3555	587	207

Tabla 6. 39 Algunas configuraciones óptimas para el Laberinto 4.

La Figura 6.88 muestra un comportamiento muy variable y contrario a las demás. En este caso se tiene que con una de las mayores cantidades de hormigas (18), el número de épocas aumenta y al disminuir la cantidad de hormigas, también el número de épocas baja

(con 13 hormigas); con un valor intermedio (16 hormigas) se obtiene la menor cantidad de épocas.

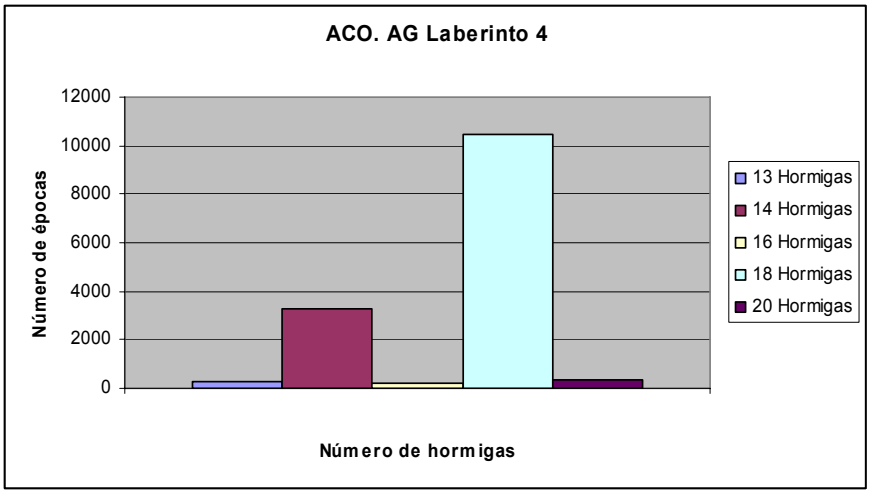


Figura 6. 88 Tiempo que tarda el algoritmo propuesto combinado con el Algoritmo Genético en el laberinto 4 al incrementar el número de hormigas.

Experimento 2.5

La Tabla 6.40 muestra el resultado de aplicar el algoritmo 5.3 al laberinto 5. Para las 5 corridas se encontraron, las mejores 5 configuraciones respectivamente tal como lo muestra la Tabla 6.40.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
α	-2.4	1.899	-2.9	3.099	-3.09
β	1.7	-0.5	2.5	-2.799	4.699
γ	-3.9	1.1	-2.5	-0.8	-2.9
m	8	10	10	11	10
Ep	27887	11811	23477	5352	4285
mseg.	37470	16858	33871	8210	6641

Tabla 6. 40 Algunas configuraciones óptimas para el Laberinto 5.

La Figura 6.89 que se presenta a continuación, muestra cómo varía el número de épocas al cambiar el número de hormigas. En éste caso, para 8 hormigas (valor mínimo encontrado), se obtuvo un número de épocas de 27887, el cual es la cantidad máxima para todo este experimento (véase Tabla 6.40, corrida 1). Los valores pequeños de hormigas corresponden a números de épocas altos. Para el caso de 10 hormigas (segunda barra) el número de épocas disminuyó al igual que con 11 hormigas.

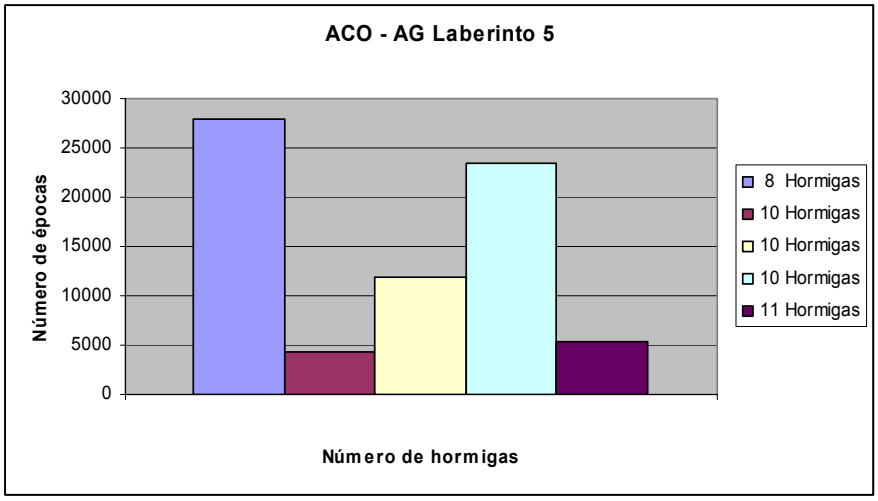


Figura 6. 89 Tiempo que tarda el algoritmo propuesto combinado con el Algoritmo Genético en el laberinto 5 al incrementar el número de hormigas.

Después de haber realizado la segunda experimentación al utilizar el algoritmo 5.3, se observó que con ayuda de éste, se logró encontrar para cada una de las 5 corridas, cinco configuraciones diferentes, en las que el resultado fue óptimo.

Dichas configuraciones son ideales para encontrar la solución óptima, (parámetros óptimos fueron α, β, γ y m =Número de hormigas). Con dichos parámetros se encontró un número de épocas máximo para obtener el camino más corto en tiempos pequeños, aunque en algunos casos fueron altos. Esto se debió a los valores iniciales con los que se trabajó el algoritmo genético ya que los parámetros α, β y γ se inicializaron de manera aleatoria, ocasionando que el algoritmo se tardará más en buscar la mejor solución.

6.1.3 Aplicación de las mejores configuraciones

La tercera experimentación consiste en tomar las configuraciones óptimas del experimento anterior y proporcionarlas al algoritmo 5.2 (utilizado en la primera experimentación). Esto con el fin de comprobar que con parámetros no aleatorios el algoritmo 5.2 puede encontrar el camino más corto y en un tiempo menor.

Experimento 3.1

La Tabla 6.41 muestra el resultado de aplicar el algoritmo 5.2 al laberinto 1, con las 5 configuraciones óptimas encontradas por el algoritmo 5.3.

Comparando con el experimento 2.1 se observa que el número de épocas y el tiempo en milisegundos del algoritmo 5.2, se redujo satisfactoriamente, obteniéndose la mejor solución en un tiempo menor, véase la Tabla 6.41.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
Ep	261	623	465	176	2032
mseg.	99	544	378	86	55

Tabla 6. 41 Tiempo empleado al implementar las mejores configuraciones en el laberinto 1 al utilizar el Algoritmo propuesto 5.2.

Experimento 3.2

La Tabla 6.42 muestra el resultado de aplicar el algoritmo 5.2 al laberinto 2, con las 5 configuraciones óptimas encontradas por el algoritmo 5.3. Ya que el tiempo se redujo en comparación al experimento 2.2, se concluye que las configuraciones obtenidas por el algoritmo ACO – AG son excelentes para encontrar el resultado en un tiempo reducido.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
Ep	105	259	145	141	205
mseg.	52	80	62	62	93

Tabla 6. 42 Tiempo empleado al implementar las mejores configuraciones en el laberinto 2 al utilizar el Algoritmo propuesto 5.2.

Experimento 3.3

La Tabla 6.43 muestra el resultado de aplicar el algoritmo 5.2 al laberinto 3, con las 5 configuraciones óptimas encontradas por el algoritmo 5.3.

Como se observa en la Tabla 6.43, la tendencia del tiempo tanto en épocas como en milisegundos disminuye, aunque el resultado para la Corrida 1 sea mayor que en la del experimento 2.3.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
Ep	1748	215	221	337	122
mseg.	407	48	52	154	65

Tabla 6. 43 Tiempo empleado al implementar las mejores configuraciones en el laberinto 3 al utilizar el Algoritmo propuesto 5.2.

Experimento 3.4

La Tabla 6.44 muestra el resultado de aplicar el algoritmo 5.2 al laberinto 4, con las 5 configuraciones óptimas encontradas por el algoritmo 5.3.

Esta tabla es otro ejemplo de que se reduce el tiempo para encontrar el camino más corto con ayuda de los dos algoritmos propuestos.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
Ep	381	492	2063	1589	133
mseg.	354	547	345	397	62

Tabla 6. 44 Tiempo empleado al implementar las mejores configuraciones en el laberinto 4 al utilizar el Algoritmo propuesto 5.2.

Experimento 3.5

La Tabla 6.45 muestra el resultado de aplicar el algoritmo 5.2 al laberinto 5, con las 5 configuraciones óptimas encontradas por el algoritmo 5.3.

A pesar de que en el experimento 2.5 los tiempos fueron muy grandes, en ésta Tabla se observa cómo disminuye drásticamente el número de épocas y el tiempo.

Configuración	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5
Ep	12369	816	1289	154	336
mseg.	3325	262	635	122	412

Tabla 6. 45 Tiempo empleado al implementar las mejores configuraciones en el laberinto 5 al utilizar el Algoritmo propuesto 5.2.

Como se observó, al utilizar el algoritmo propuesto 5.2 (Modificación de ACO) se necesitaba saber los valores correctos de los parámetros, ya que éstos influyen en la búsqueda de la solución óptima. Por ello, se necesitaron encontrar varias configuraciones óptimas o las mejores configuraciones, al utilizar el algoritmo propuesto 5.3 (ACO – AG). Al utilizar dichas configuraciones, el algoritmo 5.2 reduce su tiempo, tanto el número de épocas como en milisegundos, encontrando la solución óptima.

6.2 Aplicación en entornos reales

Para poder aplicar los algoritmos propuestos en entornos reales se construyeron varios laberintos. Se enfocaron por una cámara, la cual observaba el posicionamiento del origen de un robot móvil (Khepera II). La cámara envía una imagen del entorno a una PC en donde se procesará la imagen con el algoritmo 5.1 para obtener su grafo correspondiente, después se busca la solución óptima (camino más corto) al utilizar los algoritmos 5.2 y 5.3. Al obtener un camino, el robot móvil recibe las coordenadas de la ruta que debe tomar para llegar a su destino. En la Figura 6.90 se muestra la configuración utilizada.

En la Figura 6.91 se muestra el proceso para que un robot móvil pueda encontrar el camino más corto dentro de un entorno, en éste caso un laberinto. Se crearon 3 laberintos diferentes (Figura 6.91(a)), donde el robot Khepera II se colocó en un punto (su punto de origen). Después se obtuvo su grafo correspondiente (Figura 6.91(b)) para poder aplicar los algoritmos 5.2 y 5.3 con el objetivo de encontrar el camino más corto. En (Figura 6.91(c)) se puede ver en color rojo la ruta encontrada por los algoritmos. Finalmente en (Figura 6.91(d)) el robot avanza por la ruta indicada, llegando al punto destino.

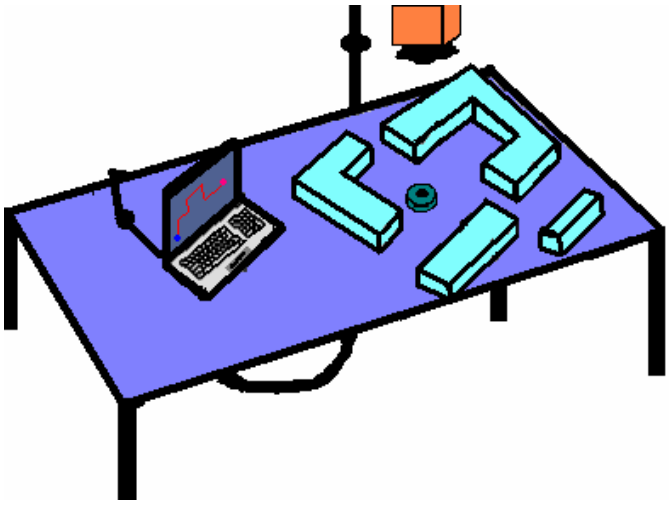


Figura 6. 90 Aplicación real.

6.3 Resumen

En este capítulo se mostraron los resultados obtenidos en tres diferentes experimentaciones.

La primera experimentación o aplicación del algoritmo Modificación de ACO, se probó bajo siete configuraciones de parámetros diferentes y se aplicó a los 5 laberintos

propuestos. En dicha experimentación se obtuvieron tanto buenos como malos resultados, debido a la elección aleatoria de los parámetros de los algoritmos 5.2 y 5.3.

El porcentaje de error del algoritmo se obtuvo al aumentar tanto el número de épocas como el de hormigas para cada laberinto y cada una de las configuraciones. Se promediaron los resultados con las 7 configuraciones de cada laberinto y se obtuvo un promedio de los 5 laberintos.

La segunda experimentación o la aplicación del algoritmo ACO-AG arrojó 5 configuraciones óptimas, ya que con los parámetros obtenidos se encuentra el camino más corto. El motivo de encontrar los parámetros es para mejorar la eficiencia del algoritmo 5.2. El tiempo en épocas que tarda dicho algoritmo, se compara con el aumento del número de hormigas.

El tercer experimento o la Aplicación de las mejores configuraciones contribuyó a demostrar que con las configuraciones óptimas, el algoritmo 5.2 encuentra el camino más corto en un tiempo menor que el obtenido del algoritmo 5.3 tanto en el número de épocas como en mili-segundos.

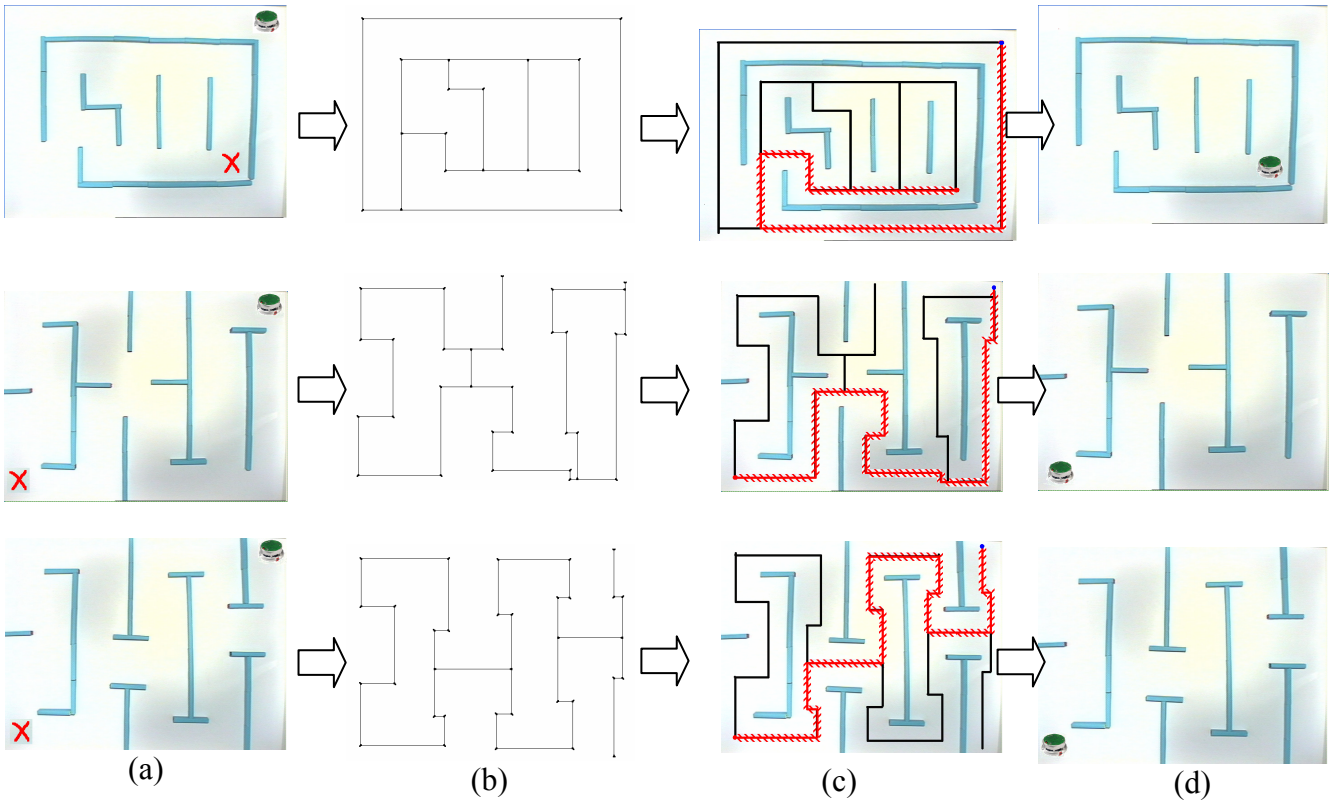


Figura 6. 91 Pasos realizados para la obtención del camino más corto. (a) Posicionamiento inicial del robot Khepera II en 3 entornos diferentes. (b) Obtención del grafo correspondiente al entorno. (c) Obtención del camino más corto al utilizar los algoritmos propuestos. (d) Posicionamiento del robot móvil el punto destino.

CAPÍTULO 7

CONCLUSIONES Y TRABAJOS A FUTURO

En este capítulo se presentan las conclusiones, comentarios sobre la metodología desarrollada durante esta investigación así como los resultados obtenidos. De igual forma, se plantean futuras líneas de investigación que complementan este trabajo o que utilizan sus algoritmos para resolver otros problemas.

Por último se da una lista de publicaciones, fruto del desarrollo de esta investigación, donde cada una de ellas representa un gran avance y desarrollo de la misma.

7.1 Conclusiones

Para poder resolver problemas de optimización, en algunos casos se pueden utilizar métodos clásicos que permitan darnos una solución óptima, como por ejemplo el Método Simplex; sin embargo en problemas donde se requiere optimizar varias variables (Optimización multi-objetivo) suele recurrirse a las Heurísticas.

En ésta investigación se ha propuesto una metodología y sus respetivos algoritmos basados en heurísticas, los cuales ayudan a dar solución al problema de determinar una ruta entre dos puntos dentro de un entorno dado.

Las heurísticas utilizadas, son técnicas basadas en inspiraciones biológicas, como la búsqueda de alimentos de una colonia de hormigas y los Algoritmos genéticos basados en el Neo-Darwinismo; donde la selección natural es el factor dirigente y creativo, la cruza es la oportunidad de obtener mejores individuos y la herencia es parte fundamental de la evolución.

Estas técnicas bio-inspiradas se pudieron comparar durante el desarrollo de la experimentación, obteniéndose características importantes y buenos resultados.

Uno de los primeros logros consistió en modificar el algoritmo original del Sistema de Colonia de hormigas para implementarlo en nuestra metodología. El algoritmo original solo resolvía el problema del agente viajero (TSP); de aquí surge nuestro primer algoritmo propuesto (Algoritmo 5.2).

Para resolver el problema de planificación de rutas en un entorno dado, se propuso que todas las hormigas creadas, iniciaran su exploración desde un mismo nodo; a este nodo se le llamó nodo de origen, así las hormigas cumplirán con el criterio de moverse de un punto de origen a uno destino para encontrar un camino.

Otra modificación fue el de crear hormigas con una frecuencia f, ya que la actualización de la longitud de cada uno de los caminos se evalúa con un costo de longitud máximo de la ruta, es decir, si la longitud actual supera la longitud máxima (significa que alguna ciudad fue visitada más de una vez), entonces la hormiga tiene que morir.

Se implementó una nueva función de transición, la cual permite ir de un punto a otro dentro del grafo. La función de transición permite escoger los posibles caminos de manera aleatoria como lo hacen las hormigas, al utilizar la feromona y su instinto, sin usar las distancias de los caminos como lo hace la función original.

Para asegurar que el algoritmo propuesto con sus respectivas modificaciones encuentra la solución óptima (camino más corto), se obtuvo la distancia del camino óptimo por medio del Algoritmo de Dijkstra, dicha distancia era comparada con la distancia encontrada por el algoritmo propuesto, de donde se obtuvo un porcentaje de error.

Los resultados de la primera experimentación, (usando el algoritmo 5.2) fueron muy interesantes, ya que los entornos propuestos (5 laberintos) fueron probados con el algoritmo 5.2, sus 7 diferentes configuraciones (utilizando α , β y γ). Los resultados mostraron que al aumentar el número de épocas, disminuye el error promedio (se acerca a la solución óptima). Esto se da ya que se le da más tiempo al algoritmo de realizar una búsqueda que se expanda por todo el entorno, teniendo más posibilidades de encontrar la mejor solución; aunque ésto depende de los valores que tomen los parámetros de las configuraciones utilizadas ver Figura 6.81 y también de las características del entorno dado ver Figura 6.83.

También se observó que al aumentar el número de hormigas, el resultado mejora, ya que entre mayor número de hormigas, se da el efecto de cooperatividad entre ellas. Las hormigas exploran diferentes caminos y se proporciona mayor información a las demás, las cuales decidirán si les sirve o no la información recolectada por todas. En algunos casos con alguna cantidad específica de hormigas el error aumenta. Esto se debe a la naturaleza del algoritmo ya que es una heurística y como tal puede generar diferentes resultados acercándose a la solución óptima o no dar ni una solución ver Figuras 6.82 y 6.84.

Ya que los resultados anteriores se obtuvieron con valores de configuraciones aleatorias y esto provocaba encontrar o no el camino más corto o simplemente el acercarse a la solución, se decidió implementar un algoritmo genético, el cual ayudaría a encontrar los valores de los parámetros. Estos parámetros influyen en el comportamiento de las hormigas, dando como resultado el desarrollo del algoritmo 5.3.

Para este algoritmo se implementó la llamada selección y evolución natural, proponiéndose dos tipos de hormigas: las exploradoras y las trabajadoras. Para desarrollar dicho algoritmo se seleccionaron las mejores dos hormigas exploradoras y las mejores dos hormigas trabajadoras las cuales cuentan con una aptitud alta. La aptitud depende del tipo de hormiga; si es trabajadora, entonces se tomará a las dos hormigas que tengan recolectada la mayor cantidad de comida y si es exploradora, se tomarán las dos hormigas que no se hayan perdido más de 5 veces en el entorno.

Cada hormiga cuenta con un cromosoma compuesto por 3 genes, los parámetros α, β y γ de la función de transición. Cada pareja de hormigas seleccionadas se cruza, generando un hijo, un nuevo individuo que podrá ser mutado con cierta probabilidad para asegurar la variedad de individuos. Por último se realizó una extinción de los individuos que no benefician a la colonia o a la población.

El algoritmo 5.3 se implementó sobre los 5 laberintos. En cada caso se obtuvieron 5 configuraciones diferentes que arrojaron la solución óptima (el camino más corto). Los resultados muestran que al aumentar el número de hormigas, el número de épocas disminuye (ver sección 6.1.2). Este fenómeno se debe a que entre mayor cantidad de hormigas, el tiempo para encontrar la mejor solución disminuye, gracias a que se explora todo el entorno.

Por último las configuraciones arrojadas por el algoritmo 5.3 se utilizaron en el algoritmo 5.2 para comprobar que se obtiene el camino más corto en un tiempo menor. Lo que confirma que los resultados dependen de los valores de los parámetros a utilizar en los entornos propuestos.

La metodología propuesta se logró aplicar a un problema real de robótica (ver Figura 6.90). Un robot móvil Khepera II recibe las coordenadas de los puntos de inicio y el final de la trayectoria, y por medio de los algoritmos propuestos , el robot puede vencer los obstáculos y llegar al punto destino, logrando optimizar la ruta dentro del entorno creado.

En general, los experimentos tanto en la teoría como en la práctica dieron muy buenos resultados; indicando que los algoritmos bio- inspirados pueden resolver problemas en el campo de la robótica, de tráfico de vehículos, ruteo, etc. En resumen, los algoritmos bio-inspirados son una vía fiable para resolver el problema de la planeación de una ruta, en donde se encuentra una trayectoria óptima que pueda seguir un robot móvil.

7.2 Trabajos a futuro

Un trabajo que surge de éste, es la implementación de "hormigas-robots", *i.e.* que cada robot móvil represente una hormiga que explore su entorno para encontrar una solución óptima. La ruta a recorrer ya no se daría a conocer desde la computadora, si no que cada robot móvil, (en tiempo real) buscaría los posibles caminos y encontraría la solución adecuada.

Cabe destacar que se tendría que implementar una "feromona" que ayude a comunicarse entre hormigas-robots, que permita el intercambio de información y se puedan así orientar hacia la solución óptima, ver Figura 7.1.

Figura 7. 1 Hormiga-Robot buscando una ruta para realizar su tarea.

Otra investigación posible es el implementar el algoritmo de hormigas propuesto en un problema de análisis de imágenes clásico; el de segmentación, mediante clusterización, ver Figura 7.2.

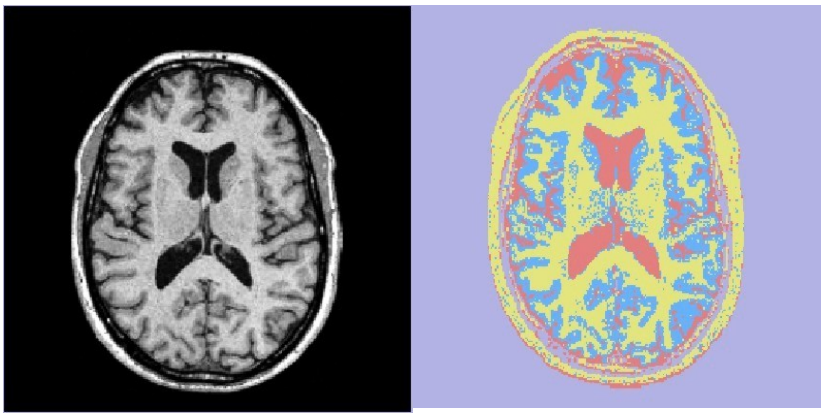


Figura 7. 2 Ejemplo de segmentación de una imagen.

7.3 Publicaciones en congresos

Como resultado de la presente investigación, se obtuvieron las siguientes publicaciones en congresos:

1. **Beatriz A. Garro**, *et al*. (2005). *Swarm intelligence applied to path planning problems*. Proceedings of XVI Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición Industrial IEEE ROC&C 2005.

2. **Beatriz A. Garro**, H. Sossa and Roberto A. Vázquez (2006). *Path planning optimization using bio-inspirited algorithms*. Proceedings of 5th Mexican International Conference on Artificial Intelligence (MICAI 2006) Special Session, November 13-17, 2004, Apizaco, Mexico. IEEE Computer Society, edited by Alexander F. Gelbukh, Carlos A. Reyes García, pp 319-328, ISBN 0-7695-2722-1.

3. **Beatriz A. Garro**, Humberto Sossa and Roberto A. Vázquez (2007). *Evolving ant colony system for optimizing path planning in mobile robots*. Enviado al Cuarto Congreso de Electrónica, Robótica y Mecánica Automotriz (CERMA 2007).

REFERENCIAS

[1] Ulrico Nehmzow. *Mobile Robotics: A practical Introduction*, Springer-Verlag, London 2000.

[2] David Kortenkamp, *et. al*. Artificial Intelligence and Mobile Robots: case studies of successful robot systems, *The AAAI Press/The MIT Press*, 1998.

[3] Moshe Sipper, *Machine Nature: The coming Age of Bio-Inspired Computing*, McGraw-Hill, 2002.

[4] Tse Min Chen and Ren C. Luo. Integrated Multi-behavior Mobile Robot Navigation Using Decentralized Control. *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*. 1998.

[5] Martin D. Adams. Adaptive Motor Control to Aid Mobile Robot Trajectory Execution in the Presence of Changing System Parameters. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, December, 1998.

[6] Atsushi Yamashita, *et. al*. Planning Method for Cooperative Manipulation by Multiple Mobile Robots using Tools with Motion Errors. *IEEVRSJ International Conference on Intelligent Robots and Systems*. 1999.

[7] Igor E. Paromtchik, Hajime Asama. Laser-Based Guidance of Multiple Mobile Robots. *Proceedings IEEUASME International Conference on Advanced Intelligent Mechatronics*. 6-12 Julio 2001.

[8] Yoshihiko Nakamura and Akinori Sekiguchi. *The Chaotic Mobile Robot. IEEE Transactions on Robotics and Automation*, Vol. 17, No. 6, december 2001.

[9] Chomchana Trevai, *et.al*. Exploration-Path Generation of Multiple Mobile Robots using Reaction-Diffusion Equation on a Graph. *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2003.

[10] Jeffrey A. Goldman. Path Planning Problems and Solutions. IEEE 1994.

[11] John S. Zelek. Dynamic Path Planning. IEEE 1995.

[12] Cem Hocaoglu and Arthur C. Sanderson. Multi-dimensional Path Planning using Evolutionary Computation. IEEE 1998.

[13] Zhong Yu, *el. al*. An Implementation of Evolutionary Computation for Path Planning of Cooperative Mobile Robots. *Proceedings 4th World Congress on Intelligent Control and Automation***,** 2002.

[14] Yanrong Hu, *et .al*. A Knowledge Based Genetic Algorithm for Path Planning in Unstructured Mobile Robot Environments. *Proceedings IEEE International Conference on Robotics and Biomimetics*, 2004.

[15] Guo Tong-ying, *et. al*. Research of Path Planning for Polishing Robot Based on Improved Genetic Algorithm. *Proceedings IEEE International Conference on Robotics and Biomimetics,* 2004.

[16] Daniel A. Ashlock, *et. al*. Evolving A Diverse Collection of Robot Path Planning Problems. *Proceedings IEEE Congress on Evolutionary Computation*, 2006.

[17] Luca Maria Gambardella and Marco Dorigo. Solving Symmetric and Asymmetric TSPs by Ant Colonies. *Proceedings IEEE Conference on Evolutionary Computation (ICEC'96)*, May 20-22, 1996.

[18] Marco Dorigo, *et .al*. The Ant System: Optimization by a colony of cooperating agents. *Proceedings IEEE Transactions on Systems, Man, and Cybernetics*–Part B, Vol.26, No.1, pp.1-13, 1996.

[19] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the traveling salesman problem. *BioSystems*, 1997.

[20] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *Proceedings IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, 1997.

[21] Tony White, *et .al.* Connection Management by Ants: An Application of Mobile Agents in Network Management. Systems and Computer Engineering, 1997.

[22] Hozefa M. Botee and Eric Bonabeau. Evolving Ant Colony Optimization. Adv. *Complex Systems* 1, 149-159, 1998.

[23] Pierre Delisle, *et. al.* Parallel Implementation of an Ant Colony Optimization Metaheuristic with OPENMP. *IEEE IPPS/SPDP'99 (Int. Parallel Processing Symposium / Symposium on Parallel and Distributed Processing*, Springer-Verlag, 1999.

[24] Tony White, *et. al*. Revisiting Elitism in Ant Colony Optimization. GECCO 2003.

[25] Hong Zheng, *et. al.* The Application of Ant Colony System to Image Textute Classification. *Proceedings Second International Conference on Machine Learning and Cybernetics,* Xi'an, 2-5 November 2003.

[26] BoLiu, Hussein A. Abbass and Bob McKey. Classification Rule discovery with Ant Colony Optimization. *IEEE Computational Intelligence Bulletin*. Vol. 3 No. 1, Febrero 2004.

[27] E. Salari and K. Eshghi. An ACO Algorithm for Graph Coloring Problem.2005.

[28] Dandan Zhang, *et. al.* An Adaptive Task Assignment Method for Multiple Mobile Robots via Swarm Intelligence Approach. *Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation* Junio 27-30, 2005.

[29] Thomas A. Runkler. Ant Colony Optimization of Clustering Models. *International Journal of Intelligence Systems* Vol. 20. 1233-1251, 2005.

[30] Simone Pimont and Christine Solnon A. Generic Ant Algorithm for Solving Constraint Satisfaction Problems. Pages 100-108. ANTS´2000.

[31] Rafael S. Parpinelli, *et. al.* Data Mining with an Ant Colony Optimization Algorithm. Evolutionary Computation, IEEE Transactions. Vol. 6, pages 321-332. Agosto 2002.

[32] Jose Aguilar. A General Ant Colony Model to solve Combinatorial Optimization Problems. *Revista Colombiana de Computación* Vol.2, No 1, págs. 7-18, 2001.

[33] James Smaldon and Alex A. Freitas. New Version of the Ant-Miner Algorithm Discovering Unordered Rule Sets. GECCO, 2006.

[34] Thang N. Bui and Catherine M. Zrncic. An Ant-Based Algorithm for Finding Degree-Constrained Minimum Spanning Tree. GECCO, 2006.

[35] Thang N. Bui and ThanhVu H. Nguyen. An Agent-Based Algorithm for Generalized Graph Colorings. GECCO, 2006.

[36] José Santos Reyes & Richard J. Duro. *Evolución Artificial y Robótica Autónoma*. Alfaomega Ra-Ma. 2005.

[37] Ulrich Nehmzow. *Mobile Robotics: A Practical Introduction*. Springer. 2000.

[38] Robin R. Murphy. *Introduction to AI Robotics*. The MIT Press Cambridge, Massachusetts London, England. 2000.

[39] A.Newell, J.C. Shaw, and H.A. Simon. *The processes of creative thinking*. In H.E. Gruber, G. Terrel and Wertheimer, editors Conteporary approaches to creative thinking, pages 63-119, Artherton Press, New York, 1193.

[40] Colin B. Reeves, editor. *Modern Heuristic Techniques of Combinatorial Problems*. John Wiley & sons, Great Britain, 1993.

[41] Stuart J. Russell and Meter Norvig. *Artificial Intelligence. A modern Approach.* Prentice hall, Upper Saddle River, New Jersey, 1995.

[42] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer–Verlag Berlin Heidelberg, 2000.

[43] S. Kirkpatrick, Jr. C. D. Gelatt and M.P. Vecchi. *Optimization by Simulated Annealing*. Science, 220:671-680, 1983.

[44] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell Massachusetts, 1998.

[45] Pierre Jaisson, *La Hormiga y el Sociobiólogo*, Fondo de Cultura Económica, Éditions Odile Jacob 1ª edicion, 2000.

[46] G. Beni and J. Wang. Swarm Intelligence in Cellular Robotic Systems. *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*, I1 Ciocco, Tuscany, Italy. 1989.

[47] P.P. Grassé. La reconstruccion du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La théorie de la stigmergie: essai d´interprétation du comportement des termites constructeurs. Insectes Sociaux, 6:41-81, 1959.

[48] V. Maniezzo, A. Colori and M. Dorigo, *The Ant System Applied to the Quad-ratic Assignmet Problem*. *Proceedings IEEE Trans. Knowledge and Data Engineering*, 11(5):769–778. 1998.

[49] B. Bullnheimer (1998). Applying the Ant System to the Vehicle routing prob-lem. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academics.

[50] Eric Bonabeau, Marco Dorigo, Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press 1999.

[51] Dorigo, M. Maniezzo, and A. Colorni. The ant System: Optimization by a Colony of Cooperating Agents. *Proceedings IEEE Trans. Syst. Man Cybern*. B26: 29-41. 1996.

[52] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press, 1975.

[53] Charles Robert Darwin. *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life.* Cambridge University Press, Cambridge, UK, sixth edition, 1964. Originally published in 1859.

[54] A.Hoffman. *Arguments of Evolution: a Paleontologist´s Perspective*. Oxford University Press, New York, 1989.

[55] David B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence.* The Institute of Electrical and Electronics Engineers, New York, 1995.

[56] Carlos A. Coello Coello. *Introducción la Computación Evolutiva.* CINVESTAV-IPN, enero 2004.

[57] Lawrence J. Fogel. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, 1966.

[58] Lawrence J. Fogel. *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., New York. 1999.

[59] Thomas Bäck. Evolutionary *Algorithms in Theory and Practice*. Oxford University Press, New York, 1966.

[60] J. H. Holland, *Adaptation in Natural and Artificial Systems*. 2ª ed., MIT Press, 1992.

[61] John H. Holland. *Concerning efficient adaptive systems*. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, editors, Self-Organizing Systems-1962, pages 215-230. Spartan Books, Washington, D.C., 1962.

[62] John H. Holland. *Outline for a logical theory of adaptive systems.* Journal of the Association for Computing Machinery, 9:297-314, 1962.

[63] Günter Rudolph. Convergente Análisis of Canonical Genetic Algorithmms. *Proceedings IEEE Transactions on Neural Networks*, 5:96-101, January 1994.

[64] Ángel Kuri y José Galviz. *Algoritmos Genéticos*. Instituto Politécnico Nacional, Universidad Nacional Autónoma de México y Fondo de Cultura Económica, Primera edición, México 2002.

[65] A.K. De Jong. *An Análisis of the behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[66] David H. Ackey. *A Connectionist Machine for Genetic Hillclimbig*. Kluwer Academic Publishers, Boston, Massachusetts, 1987.

[67] Beatriz A. Garro, *et.al*. Swarm intelligence applied to path planning problems. *Proceedings of XVI Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición Industrial IEEE* ROC&C 2005.

[68] Gonzalo Pajares y Jesús M. de la Cruz. *Visión por Computador: imágenes digitales y aflicciones*. Alfaomega-Ra-Ma, 2002.

[69] R. C. Gonzalez and R. E. Woods, *Digital Image processing*. Second edition. Prentice hall, Inc 2002.

[70] R. Jain *et. al*. *Machine Vision*. McGraw-Hill. 1995.

[71] Pierre Sollé. Morfological Image Analysis: Principles and Aplications. 2nd Edition Springer-Verlag, Heidelberg 2003.

[72] Barry A. Cipra. The Best of the 20th Century: Editors Name Top 10 Algorithms. SIAM News, Volume 33, Number 4, page 1, May 2000.

[73] Jean-Claude Latombe. *Robot Motion Planning*. The Springer International Series in Engineering and Computer Science. 1991.