Evaluación e implementación de algoritmos genéticos para aplicaciones en robótica y otras áreas de Ingeniería Mecatrónica

José Pablo Petion Rivas

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Evaluación e implementación de algoritmos genéticos para aplicaciones en robótica y otras áreas de Ingeniería Mecatrónica

Trabajo de graduación presentado por José Pablo Petion Rivas para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Evaluación e implementación de algoritmos genéticos para aplicaciones en robótica y otras áreas de Ingeniería Mecatrónica

Trabajo de graduación presentado por José Pablo Petion Rivas para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Es de mi orgullo y satisfacción presentar este trabajo como reflejo del esfuerzo y aprendizaje durante los 5 años de mi trayectoria universitaria. La elaboración de este trabajo surgió de mi interés por las herramientas de inteligencia computacional. Siempre creí que estas herramientas tendrían un potencial alto en un futuro y harían grandes aportes a la ciencia. Hoy en día me siento orgulloso de haber sido participe del desarrollo de estas herramientas afines a mi vocación, contribuyendo de cierta manera a la comunidad ingenieril. De este modo presento esta investigación estando satisfecho con cumplir mi sueño de estudiar una carrera orientada a la robótica.

Quiero expresar mi más profundo agradecimiento a todas las personas que me acompañaron durante esta etapa de mi vida. Agradezco en primer lugar a mi familia, en especial a mi papá y a mi mamá, por haber hecho esto posible y por su apoyo incondicional. De igual forma agradezco a la Fundación Amivalle, en especial al Ing. Hugo Elvira Ramos y el Ing. Edgar Ramírez, quienes me otorgaron la beca que me permitió llevar a cabo mis estudios universitarios. También quiero expresarlo al Dr. Luis Alberto Rivera, por compartir con mi persona su conocimiento y seguimiento durante el desarrollo de esta investigación. Adicionalmente, quiero agradecerle a mis compañeros quienes estuvieron en cada momento a mi lado durante la carrera, en especial a Sergio Alejandro Boch, quien me acompañó todo el tiempo, gracias a todos por las memorias que creamos juntos.

Quiero dedicar este trabajo de graduación a mi familia, como representación del logro académico. Al igual que se lo dedico a la comunidad científica, esperando que este trabajo sirva de base para futuras investigaciones y contribuya al desarrollo en diferentes áreas.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo aborda la evaluación e implementación de algoritmos genéticos (AG) para aplicaciones específicas en robótica y la Ingeniería Mecatrónica. Se plantearon objetivos enfocados en la planificación de trayectorias, evasión de obstáculos y segmentación de imágenes, desarrollando escenarios prácticos para validar la eficacia de estos algoritmos. Mediante una revisión exhaustiva del marco teórico de la computación evolutiva, se diseñaron AG adaptados que fueron implementados en simulaciones realistas utilizando plataformas como MATLAB y Webots.

En el ámbito de la planificación de trayectorias, se evaluaron cinco escenarios en mapas 2D y cuatro en entornos 3D. En los mapas 2D, los AG generaron rutas eficientes tras un número controlado de iteraciones, evitando colisiones con obstáculos estáticos y optimizando las trayectorias en términos de distancia recorrida. En los entornos 3D, los algoritmos demostraron su capacidad para la navegación de drones, adaptándose a restricciones espaciales y obstáculos complejos, y generando trayectorias precisas. La implementación en simulaciones permitió validar el desempeño de los AG en entornos controlados, asegurando la viabilidad de las soluciones propuestas.

En el procesamiento de imágenes, los AG fueron aplicados a la segmentación de imágenes complejas, como imágenes médicas y escenarios con ruido. Los algoritmos lograron un desempeño superior en métricas de calidad de segmentación frente a métodos tradicionales como Otsu y K-means, destacándose en la agrupación de píxeles y la optimización de regiones homogéneas. Adicionalmente, se desarrolló un escenario específico enfocado en la segmentación para la identificación de regiones útiles en visión por computadora, mostrando resultados altamente satisfactorios.

Con estas implementaciones, el trabajo cumplió con el objetivo general de evaluar e implementar algoritmos genéticos para aplicaciones en robótica y otras áreas de la Ingeniería Mecatrónica, validando su capacidad de optimización en diversos escenarios y consolidando su utilidad como herramienta versátil en la resolución de problemas complejos.

Abstract

This work addresses the evaluation and implementation of genetic algorithms (GA) for specific applications in robotics and Mechatronics Engineering. The objectives focused on trajectory planning, obstacle avoidance, and image segmentation, developing practical scenarios to validate the effectiveness of these algorithms. Through an exhaustive review of the theoretical framework of evolutionary computation, adapted GAs were designed and implemented in realistic simulations using platforms such as MATLAB and Webots.

In the area of trajectory planning, five scenarios were evaluated on 2D maps and four in 3D environments. On 2D maps, the GAs generated efficient routes after a controlled number of iterations, avoiding collisions with static obstacles and optimizing trajectories in terms of travel distance. In 3D environments, the algorithms demonstrated their ability to navigate drones, adapting to spatial constraints and complex obstacles, and generating precise trajectories. The implementation in simulations allowed for validating the performance of GAs in controlled environments, ensuring the feasibility of the proposed solutions.

For image processing, GAs were applied to the segmentation of complex images, including medical images and scenarios with noise. The algorithms achieved superior performance in segmentation quality metrics compared to traditional methods such as Otsu and K-means, excelling in pixel grouping and the optimization of homogeneous regions. Additionally, a specific scenario was developed for segmentation aimed at identifying regions useful in computer vision, yielding highly satisfactory results.

Through these implementations, the work fulfilled the general objective of evaluating and implementing genetic algorithms for applications in robotics and other areas of Mechatronics Engineering, validating their optimization capabilities in diverse scenarios and consolidating their utility as a versatile tool for solving complex problems.

CAPÍTULO 1

Introducción

La creciente demanda de soluciones eficientes en el campo de la robótica y la Ingeniería Mecatrónica ha impulsado el desarrollo de algoritmos inteligentes que permiten abordar problemas complejos, como la planificación de trayectorias y la optimización de rutas. Entre las metodologías más destacadas se encuentran los algoritmos genéticos (AG), un tipo de algoritmo evolutivo basado en los principios de la selección natural, los cuales han demostrado ser altamente efectivos para la resolución de problemas de optimización en entornos diversos. Este trabajo de graduación se enfoca en la evaluación e implementación de algoritmos genéticos para aplicaciones en robótica, un área en constante evolución que plantea retos como la evasión de obstáculos, planificación de trayectorias y la optimización de movimiento.

El estudio no solo aborda las aplicaciones de los AG en la planificación de trayectorias para robots móviles, sino que también explora su potencial en otras áreas de la Ingeniería Mecatrónica, como el procesamiento de imágenes y planificación de movimiento para drones en el espacio 3D. Esto responde a la necesidad de contar con herramientas computacionales que optimicen el comportamiento de sistemas autónomos en entornos simulados y, potencialmente, en escenarios reales. En la literatura revisada, los algoritmos genéticos han mostrado un comportamiento favorable para la planificación de rutas libres de colisiones, especialmente cuando se integran en sistemas de navegación para robots con movimiento diferencial.

El objetivo principal de este trabajo es evaluar e implementar algoritmos genéticos para optimización en aplicaciones robóticas y otras áreas de la Ingeniería Mecatrónica. Se desarrollaron diferentes escenarios prácticos, que incluyen simulaciones realistas, donde se validó la eficacia de los algoritmos propuestos. Entre los objetivos específicos, se destaca la investigación y evaluación de los AG dentro del área de inteligencia computacional, así como su implementación en planificación de trayectorias y segmentación de imágenes.

Este estudio busca sentar un precedente en la Universidad del Valle de Guatemala, expandiendo el campo de investigación sobre algoritmos genéticos y su aplicación en ingeniería. A través de simulaciones y pruebas en entornos controlados, se validaron los resultados obtenidos para ofrecer una referencia sólida a futuras investigaciones en este campo. Los resultados

obtenidos podrían ser el punto de partida para futuras implementaciones en escenarios más complejos, como entornos dinámicos y aplicaciones en la vida real.

CAPÍTULO 2

Antecedentes

Según Engelbrecht, la inteligencia computacional se define como el estudio de mecanismos adaptativos para facilitar el comportamiento inteligente. Este es un avance muy importante en el desarrollo algorítmico para resolver problemas complejos. Estos algoritmos inteligentes incluyen redes neuronales artificiales, computación evolutiva e inteligencia de enjambre, combinado con lógica y razonamiento deductivo. Todo este tipo de algoritmos inteligentes forman parte del campo de la inteligencia artificial (IA) [1].

2.1. Inteligencia de enjambre para aplicaciones robóticas y primer avance de algoritmos genéticos

Anteriormente se desarrollaron y validaron los algoritmos de enjambre para adaptarse a aplicaciones de robótica. En los avances de robótica, para la navegación de un robot móvil, se debe contar con un planificador de trayectorias. Por lo que esta investigación [2] se basó en implementar y verificar algoritmos de inteligencia de enjambre, además, computación evolutiva (algoritmos genéticos), como alternativa de los desarrollados con aterioridad. Dichos algoritmos son el *Ant Colony*, basado en comportamiento de hormigas, con el objetivo de econtrar sus parámetros de ecuaciones y validarlo en robots simulados. Para lograrlo, se implementó el algoritmo *Simple Ant Colony*, seguido de *Ant System*. Estos se validaron por medio de simulaciones para vizualizar el comportamiento en la colonia, para luego adaptarlos a los modelos de movimiento necesarios para comparar su desempeño en el software de *Webots*. Adicionalmente, se indagó en los algoritmos genéticos, adhiriendo su implementación en diferentes funciones de costo, para minimizar trayectorias.

El resultado de la investigación, demostró el funcionamiento del algoritmo AS como alternativa de optimización. Otra deducción, se trató sobre qué algoritmo es mejor, cuando se habla de controlar enjambres de robots en forma simultanea. De esta forma se dedujo que *Modified Particle Swarm Optimization* (MPSO) es mejor que AS. Además, se demostró que los algoritmos genéticos no fueron la alternativa más óptima para la implementación

en las funciones de minimización. Sin embargo no se realizaron otras modificaciones, esto se quedó a un nivel muy superficial. Cabe a resaltar que el alcance que obtuvo esta investigación, abarcó la implementación de los algoritmos a nivel de simulación. En cuanto a los algoritmos genéticos, no se realizaron modificaciones para su mejora, dejando la posibilidad de avances en futuro, los resultados de aplicar algoritmos genéticos a funciones de costo se muestran en la Figura 1. Se afirmó que, unicamente se probó con robots de tipo E-puck, considerando sus limitaciones físicas y orientado a lenguaje de programación MATLAB.

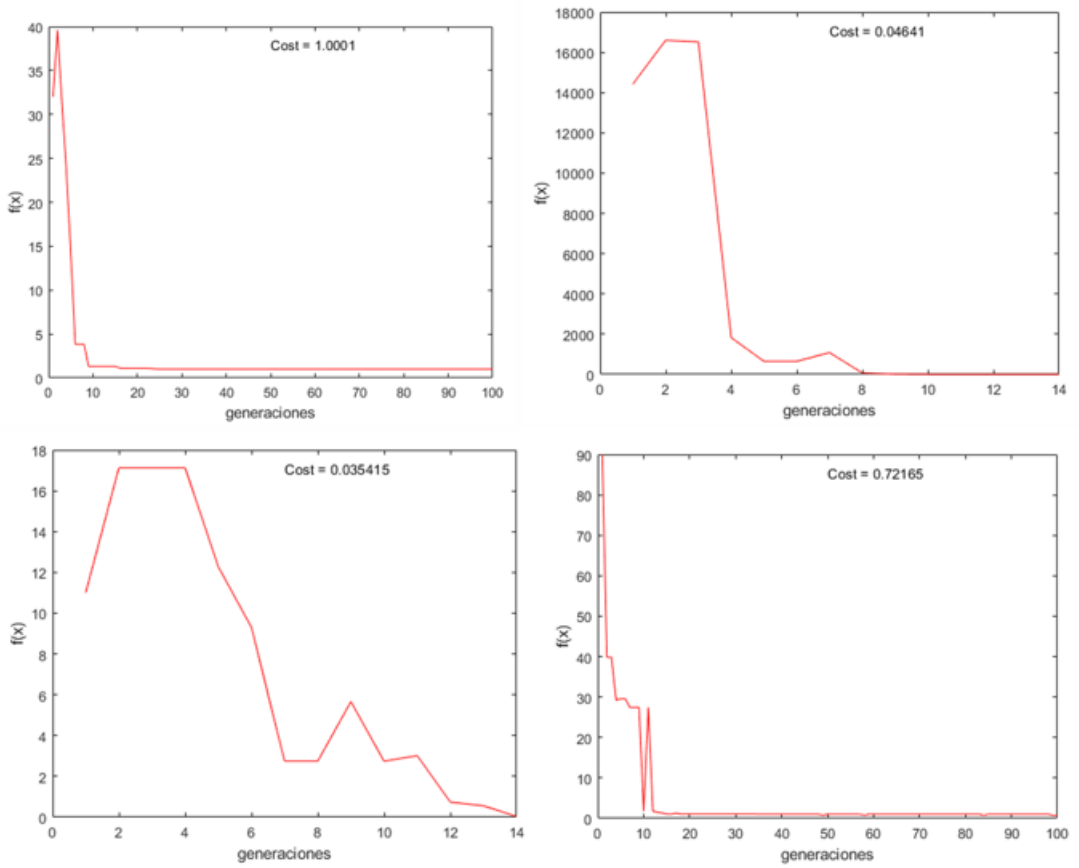


Figura 1: Resultados de cuatro funciones de costo aplicando algoritmos genéticos [2]

2.2. Algoritmo *Ant Colony Optimization* (ACO) para aplicaciones de robótica y biomédica

El objetivo principal de este proyecto [3] se basó en implementar estos algorimos de inteligencia computacional y robótica de enjambre, para desarrollar aplicaciones funcionales. Los algoritmos se utilizaron para exploración de terrenos, planificación de trayectorias y evasión de obstáculos, en la rama de robótica. Por otro lado, se procedió a un estudio en la implementación del algoritmo para aplicaciones biomédicas. Para los problemas en la rama de robótica mencionados, se propusieron dos algoritmos *Ant Colony Optimization* y *Ant System*. Ambos algoritmos fueron implementados en MATLAB, validando su funcionamiento

en tres diferentes mapas. Para la validación en la rama biomédica, se propuso procesamiento de imágenes médicas, siendo implementado en MATLAB. La finalidad de este algoritmo es transformar imágenes distorsionadas a imágenes de referencia.

Como punto concluyente, se validó el funcionamiento del algoritmo de planificación de trayectorias utilizando robots diferenciales, logrando que estas evadan obstáculos. Además, se validó la implementación utilizando imágenes de prueba e imágenes médicas. Se lograron transformar las imágenes distorsionadas en las imágenes de referencia. El alcance de este trabajo abarcó la implementación, a nivel de simualción, de dichos algoritmos.

De manera similar, en [4] se implementó y validó el algoritmo *Ant Colony Optimization* y *Particle Swarm Optimization* en aplicaciones de robótica, a nivel físico. Esta investigación buscó migrar los algoritmos desarrollados anteriormente (ACO y PSO) a los microcontroladores físicos, de esta forma validar el desempeño. De este modo fué posible validar el algoritmo en robots móviles y en un entorno real. Los resultados que más destacan son la características que posee el algoritmo ACO de crear rutas óptimas, en una implementación real.

2.3. Algoritmos genéticos (AG) para planificación de rutas aplicados en robots móviles

En las aplicaciones de robótica, hablando de una manera global, el trabajo presentado en [5] abraca el concepto de los sistemas de navegación en un robot. El objetivo de la investigación, es aplicar algoritmos genéticos cruzados en robots móviles, para que estos puedan planificar rutas libres de colisiones. Este estudio utilizó un robot móvil de accionamiento diferencial como objeto de investigación (DMRR). Se utilizaron cinco puntos de referencia como genes del algoritmo diseñado, el algoritmo genético consta de cromosomas, que representa una solución vectorial. Luego, se verificó si los 30 cromosomas de la población estan libres de colisión. El resultado del experimento con cinco iteraciones evidenció una trayectoria libre de obstáculos, que mientras más iteraciones posee, muestra una convergencia correcta, como se muestra en la Figura 2. Por lo que la investigación concluyó en que el algoritmo genético propuesto modifica el proceso de cruce, mediante el cromosoma que tenga mejor aptitud para ser seleccionado. Esta investigación tuvo un alcance a nivel de simulación, unicamente acotando el espacio de búsqueda y con obstáculos estáticos. Se espera que en un futuro, se pueda aplicar esta solución para planificación de rutas en un entorno dinámico.

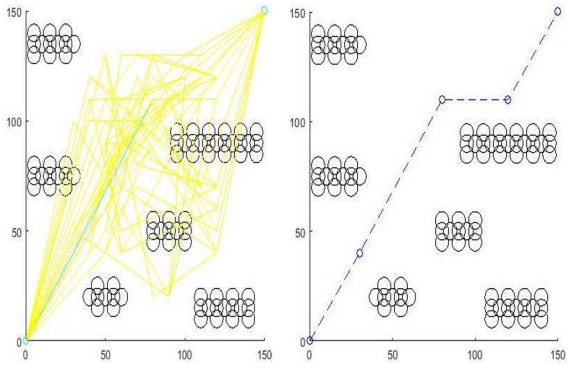


Figura 2: Resultados de planificación de trayectorias con algoritmos genéticos [5]

2.4. Algoritmos genéticos para optimización multiobjetivo

En el artículo presentado en [6] se propone un nuevo algoritmo genético multiagente para optimización multiobjetivo (MAGAMO). El algoritmo se basa en la interacción dinámica de agentes sincronizados AG interdependientes, que tienen evoluciones propias y separadas de sus poblaciones. Los agentes inteligentes buscan soluciones locales subóptimas para la optimización global, que se completará como resultado de la interaccion de todos los agentes. Como resultado de esto, se puede reducir significativamente la necesidad de volver a calcular funciones optimización a gran escala. Se puede inferir por medio de esta investigación, que el algoritmo genético multiagente desarrollado para la optimización multiobjeto es un algoritmo novedoso basado en la interacción dinámica. Como resultado de esto, se pueden resolver problemas de optimización multiobjetivo que tienen un número demasiado grande de dismensiones, como en la simulación de sistemas grandes reales.

2.5. Aplicación de algoritmos genéticos para evasión de obstáculos

En la investigación [7], con el objetivo de mejorar la capacidad de evasión de obstáculos en el espacio de un robot, se propuso un método basado en el algoritmo genético mejorado. Para implementarlo, se construyó el modelo con parámetros definidos para restricciones de control y la trayectoria de evasión de obstáculos se define como el problema de optimización. La feromona liberada en la evolución genética se utilizó como regla de guía de control de los espacios liberados. Como resultado, se obtuvieron los parámetros mecánicos de movimiento para la evasión de obstáculos, al igual que los de seguimiento de trayectoria. El resultado más importante en la simulación demostró que el algoritmo propuesto es efectivo para evitar obstáculos y mejorar el redimiento de control para el robot.

CAPÍTULO 3

Justificación

Anteriormente, en investigaciones realizadas en Universidad del Valle de Guatemala, se indagó de manera muy superficial, en la rama de algoritmos genéticos y respecto a sus posibles aplicaciones en ingeniería. En la investigación de Iriarte (2021), se incursionó en la aplicación de estos algoritmos, como una propuesta para métodos de optimización de funciones. Como producto, se demostró que los resultados preliminares de la implementación del AG no fueron óptimos, brindando la posibilidad de hacer cambios en él para su futura mejora. Por ende, esta investigación se basó en desarrollar de una manera amplia el campo de aplicaciones para este tipo de algoritmos. Se validaron algunas aplicaciones funcionales en robótica que puedan ser útiles para proponer alternativas en el uso de algoritmos. Además, poder tener una alternativa sobre herramientas de inteligencia computacional que sea capaz de mejorar la eficiencia de robots móviles.

El objetivo se enfocó en las aplicaciones de algoritmos genéticos para optimización en la planeación de trayectorias y evasión de obstáculos para robots móviles, a nivel de simulación. Además, también se desarrollaron aplicaciones funcionales para procesamiento de señales e imágenes. De esta forma, se amplió la rama de investigación sobre dichos algoritmos en la Universidad del Valle de Guatemala, como una alternativa en el tema de inteligencia computacional. De este modo, se proporcionó un panorama más amplio de la investigación realizada en el tema y sus resultados, con el fin de que futuros estudiantes interesados puedan continuar y expandir este campo

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Evaluar e implementar algoritmos genéticos para optimización en aplicaciones de robótica y otras áreas de Ingeniería Mecatrónica.

4.2. Objetivos específicos

- Investigar y evaluar los algoritmos genéticos, dentro del área de inteligencia computacional, y sus aplicaciones.
- Evaluar e implementar algoritmos genéticos en aplicaciones de robótica como planificación de trayectorias y evasión de obstáculos.
- Evaluar los algoritmos genéticos en otras aplicaciones de Ingeniería Mecatrónica.
- Desarrollar escenarios prácticos para las distintas aplicaciones, y validar la implementación de los algoritmos genéticos mediante simulaciones realistas.

CAPÍTULO 5

Alcance

Este trabajo de graduación abarca la evaluación e implementación de algoritmos genéticos (AG) en aplicaciones de robótica y otras áreas de la Ingeniería Mecatrónica, con un enfoque específico en la planificación de trayectorias y la evasión de obstáculos. El alcance de esta investigación se centra en el desarrollo y validación de estos algoritmos a nivel de simulación, utilizando entornos 2D y 3D para garantizar la correcta funcionalidad en robots móviles con control de pose lineal.

En el contexto de la robótica, los algoritmos genéticos han sido implementados para generar rutas que permiten a los robots evitar colisiones con obstáculos estáticos. El algoritmo se adapta a mapas de diversas configuraciones y obstáculos de diferentes tamaños y posiciones, validando su desempeño a través de múltiples iteraciones y evaluando la calidad de las soluciones obtenidas. El presente trabajo no sólo busca optimizar las rutas generadas, sino también explorar diferentes configuraciones de parámetros, como la probabilidad de cruce y mutación, para entender cómo estos influyen en la diversidad poblacional y el tiempo de convergencia de los algoritmos.

Además de la planificación de movimiento en robots móviles, se ha extendido el uso de algoritmos genéticos a aplicaciones como el procesamiento de imágenes. En esta área, se evalúa la capacidad del algoritmo para agrupar píxeles en función de sus características, optimizando parámetros que permitan mejorar la precisión de la segmentación en imágenes complejas, como las que se encuentran en el análisis de regiones de diferentes colores.

Es importante resaltar que, aunque la mayor parte del trabajo se ha realizado en entornos simulados, los resultados obtenidos sientan las bases para futuras implementaciones en entornos físicos. Si bien no se ha contemplado una aplicación directa en hardware durante esta investigación, el uso de simulaciones realistas en plataformas como Webots demuestra la viabilidad de estos algoritmos en escenarios más complejos y dinámicos.

Finalmente, el trabajo no está limitado exclusivamente al ámbito de la robótica. Dado que los algoritmos genéticos se pueden aplicar a una amplia variedad de problemas como optimización multiobjetivo, procesamiento de señales, sistemas de control, entre otros. Esta

investigación abre las puertas a futuras implementaciones en otras áreas de la Ingeniería Mecatrónica, donde la búsqueda de soluciones óptimas sea un desafío recurrente.

CAPÍTULO 6

Marco teórico

6.1. Inteligencia computacional

La inteligencia computacional (IC) pertenece a una rama de la inteligencia artificial (IA). Este concepto se define como el estudio de mecanismos adaptativos que permiten o facilitan el comportamiento inteligente en ambientes complejos y cambiantes. Este tipo de mecanismos incluye los paradigmas de IA que demuestran la habilidad de aprender o adaptarse a nuevas situaciones. Entre los paradigmas cubiertos resaltan la computación evolutiva y la inteligencia de enjambre [1].

6.1.1. Computación evolutiva

El objetivo de la computación evolutiva es imitar el proceso de la evolución natural, basándose en el concepto de la supervivencia del más fuerte. Aquellos individuos que son más débiles mueren y las características se descartan. Los algoritmos evolutivos usan una población de ejemplares donde el individuo se denomina cromosoma. El cromosoma definirá las características de cada individuo en la población. Las características se denominan genes. El valor de un gen se denomina alelo. La descendencia se genera combinando partes de los padres, en un proceso conocido como cruce. La fuerza de supervivencia de un individuo se mide mediante una función de aptitud que refleja los objetivos y limitaciones del problema a resolver. Además, las características de comportamiento se pueden utilizar para influir en el proceso evolutivo en cambios genéticos y características que evolucionan por separado [1]. Dependiendo de la aplicación, se pueden implementar los diferentes paradigmas:

- Algoritmos genéticos (AG)
- Programación genética (GP)
- Programación evolutiva (PE)

- Estrategias evolutivas
- Evolución diferencial
- Evolución cultural
- Coevolución

6.2. Algoritmo evolutivo genérico

La evolución mediante selección natural de una población de individuos elegida al azar puede considerarse como una búsqueda en el espacio de posibles valores cromosómicos. En ese sentido, un algoritmo evolutivo (AE) es una búsqueda estocástica de una solución óptima a un problema determinado. El proceso de búsqueda evolutiva esta influenciado por los siguientes componentes principales de un AE: una codificación de soluciones al problema como cromosoma, una función para evaluar la aptitud física o la fuerza de supervivencia de los individuos, inicialización de la población inicial, operadores de selección y operadores de reproducción. En el Algoritmo 6.1 se muestra una la combinación de los componentes para formar un algoritmo genético.

Algoritmo 6.1: Algoritmo evolutivo genérico

```
1  Let t = 0 be the generation counter;
2  Create and initialize an n_x dimensional population, C_(0), to
       consist of n_s individuals;
3  while stopping condition(s) not true do
4  Evaluate the fitness, f(x_r_(t)), of each individual, x_i_(t);
5  Perform reproductions to create offspring;
6  Select the new population, C_(t+1);
7  Advance to the new generation, i.e. t=t+1;
8  end
```

Los pasos de la implementación del algoritmo se aplica de forma iterativa hasta que se cumpla la condición de parada. Cada iteración del AE se denomina generación, dependiendo la forma en que se implemente el componente del algoritmo dará como resultado los paradigmas como algoritmos genéticos y entre los ya mencionados con anterioridad [1].

6.3. Representación del cromosoma

En la naturaleza los cromosomas son estructuras de moléculas compactadas de ADN entrelazadas que se encuentran en el núcleo de las células orgánicas. Cada cromosoma contiene una gran cantidad de genes (unidad de herencia), como se ilustra en la Figura 3. En el contexto de CE, cada individuo representa una solución candidata a un problema de optimización. Las características de un individuo están presentadas por un cromosoma. Estas características se refieren a las variables del problema de optimización, para las cuales se

busca una asignación óptima. Cada variable que debe optimizarse se denomina gen, la unidad más pequeña de información. Las características de un individuo se dividen en genotipos y fenotipos. Un genotipo describe la composición genética de un individuo y un fenotipo son solo los rasgos u comportamiento.

En la implementación de algoritmos genéticos, la representación de los cromosomas puede variar según el tipo de datos y la naturaleza del problema. Las Figuras 4 y 5 ilustran dos formas comunes de codificación de cromosomas: la codificación entera y la codificación en punto flotante (o real). En la codificación entera (Figura 4]), cada gen del cromosoma toma valores discretos, típicamente representando estados específicos o decisiones binarias. Esta codificación es útil en problemas de combinatoria o cuando las soluciones deben estar limitadas a valores discretos. Por otro lado, la codificación en punto flotante (Figura 5) permite que los genes adopten valores continuos, siendo ideal para problemas de optimización que requieren una mayor precisión en los parámetros. Estas representaciones son fundamentales en la aplicación de algoritmos genéticos, pues afectan tanto la convergencia del algoritmo como la precisión de los resultados obtenidos.

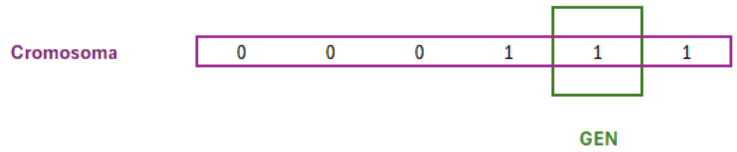


Figura 3: Representación de gen y cromosoma con codificación binaria [1]

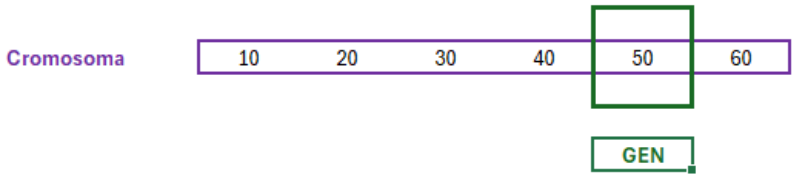


Figura 4: Representación de gen y cromosoma con codificación entera [1]

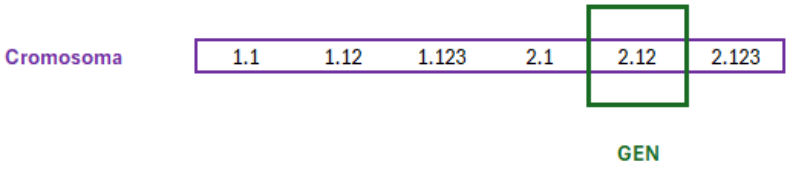


Figura 5: Representación de gen y cromosoma con codificación de punto flotante [1]

En el diseño de AE, el esquema de representación clásico para AG, son vectores binarios de longitud fija. En el caso de una espacio de búsqueda de n dimensiones, cada individuo consta de n variables y cada variable está codificada como una cadena de bits [1].

6.4. Población inicial

Debido a que los AE son algoritmos de búsqueda estocástica basados en la población, cada uno mantiene una población de soluciones candidatas. El primer paso en aplicaciones de un AE para resolver un problema de optimización es generar una población inicial. La forma estándar de generar una población inicial es asignar un valor aleatorio del dominio permitido a cada uno de los genes del cromosoma. Si las regiones de búsqueda no están cubiertas por la población inicial, es probable que el proceso de búsqueda descuide esas partes.

Un gran número de individuos aumenta la diversidad, mejorando así las capacidades de exploración de la población. Cuantos más individuos mayor será la complejidad computacional por generación [1].

6.5. Función de aptitud

Para determinar la capacidad de supervivencia de un individuo en un AE, se utiliza una función matemática para cuantificar qué tan buena es la solución representada por un cromosoma. La función de aptitud asigna una representación cromosómica a un valor escalar. La función aptitud se expresa en la ecuación (1).

$$f : \Gamma^{n_x} \rightarrow \mathbb{R} \tag{1}$$

Normalmente, la función de aptitud proporciona una medida absoluta de aptitud. Es decir, la solución representada por un cromosoma se evalúa directamente utilizando la función objetivo. Es importante comprender que existen diferentes tipos de problemas de optimización que influyen en la formulación de la función aptitud [1]. Por lo que deben mencionarse los diferentes tipos de problemas de optimización:

- Problemas de optimización sin restricciones
- Problemas de optimización restringido
- Problemas de optimización multiobjeto
- Problemas dinámicos y ruidosos

6.6. Selección

El objetivo principal de los operadores de selección es destacar mejores soluciones. Esto se logra en dos de los pasos principales de un AE, siendo la primera selección de la nueva población y la segunda reproducción. A continuación, se proporciona un resumen de los operadores más utilizados.

6.6.1. Precisión selectiva

Los operadores de precisión selectiva, se caracterizan por su tiempo de adquisición. Se relaciona con el tiempo que se requiere para producir una población uniforme. Se define como la velocidad a la que la mejor solución ocupará a toda la población mediante la aplicación repetida del operador de selección. Este tipo de operador disminuye la diversidad en la población rápidamente llevando a una convergencia prematura.

6.6.2. Selección aleatoria

La selección aleatoria es el operador de selección más simple, donde cada individuo tiene la misma probabilidad $\frac{1}{n_s}$ de ser seleccionado. No se utiliza información de aptitud, lo que significa que los mejores y los peores individuos tienen exactamente la misma posibilidad de sobrevivir hasta la siguiente generación. Sin embargo, la selección aleatoria tiene la precisión más baja entre todos los operadores mencionados [1].

6.6.3. Selección proporcional

La selección proporcional, propuesta por Holland, sesga la selección hacia los individuos más aptos. Se crea una distribución de probabilidad proporcional a la aptitud y se seleccionan los individuos muestreando la distribución [1].

$$\varphi_s(x_i(t)) = \frac{f_\Upsilon(x_t(t))}{\sum_{l=1}^{n_s} f_\Upsilon(x_l(t))} \tag{2}$$

donde n_s es el numero total de individuos y $\varphi_s(x_i(t))$ es la probabilidad de que x_i sea seleccionado. $f_\Upsilon(x_t(t))$ es la aptitud escalada de x_i, para producir un valor de punto flotante positivo. Suponiendo valores de maximizan y aptitud normalizados, la selección de la rueda de la ruleta se resume al algoritmo mostrado en el Algoritmo 6.2 siendo un ejemplo de la estructura del operador de selección proporcional.

Algoritmo 6.2: Estructura del algoritmo de selección proporcional

```
1  Let i = 1, where i denotes the chromosome index;
2  Calculate φ_(x_i) using equiation 1;
3  sum = φ_x_i;
4  Choose r ~ U(0,1);
5  while sum < r do
6  i = i+1, i.e. advance to the next chromosome;
7  sum = sum + φ_x_i;
8  end
9  Return x_i as the selected individual;
```

6.6.4. Selección por competencia (torneo)

La selección por torneo selecciona un grupo de n_ts individuos de manera aleatorio dentro de la población, donde $n_ts < n_s$ (n_s es el numero total de individuos). Se compara el desempeño de los individuos seleccionados y el operador selecciona y devuelve al mejor individuo de este grupo [1].

6.6.5. Selección basada en *Rank*

La selección basada en *Rank*, utiliza el ordenamiento de los valores de aptitud para determinar la probabilidad de selección y no los valores absolutos de aptitud. Por lo tanto, la selección es independiente de los valores reales de aptitud, con la ventaja de que el mejor individuo no dominará el proceso de selección [1].

6.7. Operadores de reproducción

La reproducción es el proceso de producir descendencia a partir de padres seleccionados mediante la aplicación de operadores de cruce y/o mutación. El cruce es el proceso de creación de uno o más individuos nuevos mediante la combinación de material genético seleccionado aleatoriamente de dos o más padres. Si la selección se centra en los individuos más aptos, la precisión de selección pueden causar una convergencia prematura debido a la reducción de la diversidad de las nuevas poblaciones [1].

La mutación es el proceso de cambiar aleatoriamente los valores de los genes en un cromosoma. El principal objetivo de la mutación es introducir nuevo material genético en la población, aumentando así la diversidad genética. La mutación debe aplicarse con cuidado para no distorsionar el material genético bueno en individuos muy aptos [1].

6.8. Condición de detención

Los operadores evolutivos se aplican iterativamente en un AE hasta que se cumple una condición de parada. La condición de parada más simple es limitar el número de generaciones que el AE puede ejecutar o, alternativamente, se impone un límite de número de evaluaciones de la función de aptitud. Este límite no debería ser demasiado pequeño, de lo contrario el asesor no tendrá tiempo suficiente para explorar el espacio de búsqueda.

Además de un límite en e tiempo de ejecución, generalmente se utiliza un criterio de convergencia para detectar si la población converge al valor esperado. La convergencia se define vagamente como el evento en el que la población se estanca. Es decir, cuando no existe ningún cambio genotípico o fenotípico en la población [1]. Se pueden utilizar los siguientes criterios de convergencia:

- Terminar cuando no se observe ninguna mejora durante varias generaciones consecu-

tivas.

- Terminar cuando no haya cambios en la población.
- Terminar cuando se haya encontrado una solución aceptable.
- Terminar cuando la pendiente de la función objetivo sea aproximadamente cero.

6.9. Decodificación de variables

La decodificación en algoritmos genéticos es un proceso fundamental que convierte las soluciones representadas en los cromosomas a un formato adecuado para evaluar su aptitud en el contexto del problema específico. Según Engelbrecht [1], la decodificación depende de la representación del cromosoma y del dominio de la solución. La representación de los cromosomas puede variar considerablemente, incluyendo codificación binaria, entera, y en punto flotante (real), cada una con ventajas y aplicaciones particulares.

La elección de la estrategia de decodificación influye significativamente en el rendimiento del algoritmo genético. Engelbrecht señala que la decodificación debe ser compatible con la función de aptitud y el tipo de datos requeridos para asegurar que los cromosomas generen soluciones válidas y eficaces. Un claro ejemplo de decodificación sería considerando una cadena binaria de L bits, $C = c_1c_2...c_3$, que representa un valor en el intervalo $[a, b]$. Luego se calcula el valor decimal como se muestra en la Ecuación (3).

$$D = \sum_{i=1}^{L} c_i \cdot 2^{L-i} \tag{3}$$

Luego el valor del intervalo D se mapea al intervalo $[a, b]$ usando la Ecuación de escalado lineal (4]).

El tipo de decodificación

$$x = a + \frac{D}{2^L - 1} \cdot (b - a) \tag{4}$$

6.10. Algoritmos genéticos (AG)

Los algoritmos genéticos modelan la evolución genética, donde las características de los individuos se expresan mediante genotipos. Los principales operadores impulsores de un AG son la selección (para moldear la supervivencia del mas apto) y la recombinación mediante la aplicación de un operador cruzado (para modelar la reproducción) [1]. El algoritmo genético propuesto por Engelbrecht, contiene los siguientes detalles.

- Utiliza una representación de cadena de bits.

- Utiliza selección proporcional para seleccionar padres para recombinación.
- Utiliza el cruce de un punto como método principal para producir descendencia.
- Se propuso la mutación uniforme como un operador de fondo.

Los algoritmos genéticos combinan la explotación de soluciones conocidas con la exploración de nuevas áreas del espacio de posibles soluciones. Su eficacia depende de una apropiada combinación entre la explotación y la exploración. La selección, de acuerdo a las características y aptitudes de individuos de una población en combinación con el operador genético de cruza, constituyen la tasa de la explotación. Mientras que el operador de mutación es la base de la exploración de nuevas regiones del espacio de búsqueda. Un algoritmo genético, simula el comportamiento dinámico de una población genética generando una base de conocimientos formada por estructuras que se desarrollan en respuesta al desempeño observado en su medio ambiente operacional [8]. En términos generales, los algoritmos genéticos están constituidos por:

- La codificación de una población de individuos
- La decodificación de la población de individuos
- La selección de individuos para producir los descendientes
- La reproducción de dichos individuos por medio de operadores genéticos

Los algoritmos genéticos difieren de los métodos tradicionales de búsqueda y optimización en los siguientes puntos: los AG's tratan el problema como una caja negra, usan codificación, buscan en una población de posibles soluciones y por último emplean operadores aleatorios. Cabe resaltar que muchos métodos de búsqueda requieren de ciertos antecedentes o información adicional en el problema de estudio. En el caso de los algoritmos genéticos no necesitan de información adicional. Por lo tanto, estos tratan el problema como una caja negra y solo requieren evaluar la función objetivo. En la Figura 6, se muestra un esquema que explica cómo debe funcionar un AG, desde la selección por medio de la función aptitud hasta el proceso de cruza para generación de nuevos individuos.

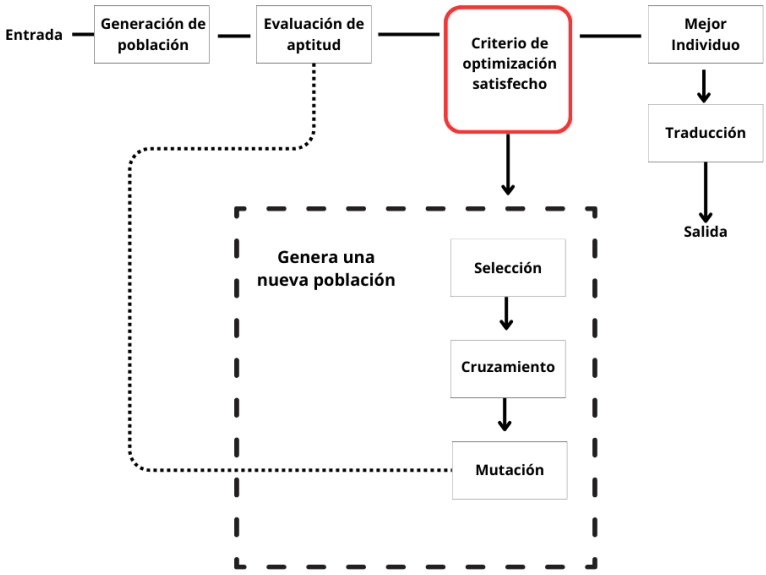


Figura 6: Diagrama de funcionamiento general de un AG [1]

6.10.1. Cruce

Los operadores cruzados se dividen en tres categorías dependiendo el número de padres utilizados. Las categorías son asexual, sexual y recombinación. Los padres se seleccionan utilizando cualquiera de los esquemas de selección discutidos anteriormente. Al seleccionar a los padres deben considerase ciertas cuestiones como seleccionar al mismo individuo, como ambos padres, o que este mismo participe en más de una aplicación del cruce. Cabe a mencionar que el operador de cruce considera política de remplazo, si se genera una descendencia que sea mejor que el padre puede remplazarlo [1].

6.10.2. Representaciones binarias

La mayoría de los operadores de cruce para representaciones binarias son sexuales y se aplican con dos padres seleccionados. Si $x_1(t)$ y $x_2(t)$ denotan a los dos padres seleccionados, entonces el proceso de recombinación se resume en el Algoritmo 6.3. Entre los operadores desarrollados se encuentran los siguientes puntos para computadorizar la mascara:

- **Cruce de un punto:** los segmentos de genes se intercambian entre padres para crear su descendencia, y no genes simples. En la Figura 7 se muestra un esquema de este tipo de cruce.

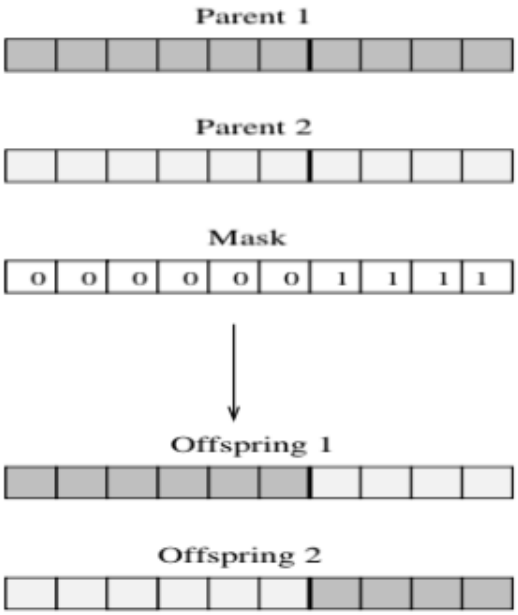


Figura 7: Esquema de cruce por medio de un punto [1]

- **Cruce de dos puntos:** en este caso se seleccionan aleatoriamente dos posiciones de bits y las cadenas de bits entre estos puntos se intercambian como se ilustra en la Figura 8.

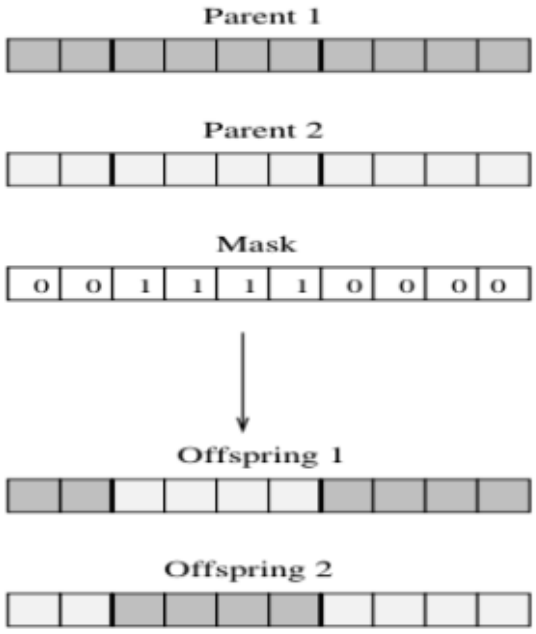


Figura 8: Esquema de cruce por medio de dos puntos [1]

- **Cruce uniforme:** el cruce uniforme es una técnica completamente diferente de las

vistas hasta el momento. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre. Aunque se puede implementar de muy diversas formas, la técnica implica la generación de una máscara de cruce con valores binarios. Si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copia del primer padre. Si por el contrario hay un 0 el gen se copia del segundo padre. Para producir el segundo descendiente se intercambian los papeles de los padres, o bien se intercambia la interpretación de los unos y los ceros de la máscara de cruce [9]. Esta representación se muestra en la Figura 9.

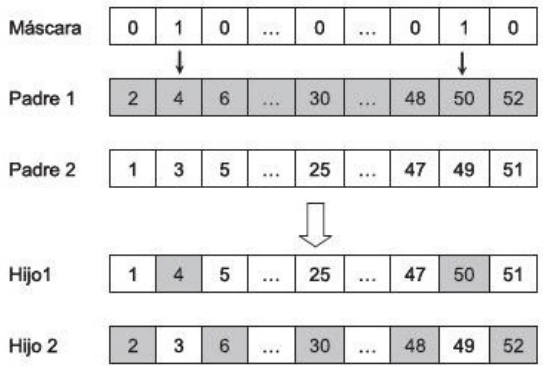


Figura 9: Esquema de cruce uniforme [1]

Algoritmo 6.3: Estructura del algoritmo de cruza por representación binaria

```
Let $\widehat{x}_1(t) = x_1(t)$ and $\widehat{x}_2(t) = x_2(t)$;
if U(0,1) $\leq p_c$ then
Compute the binary mask, m(t);
dor j = 1,..., $n_x$ do
if $m_j = 1$ then
// swap the bits
$\widehat{x}_1 j(t) = x_1 j(t)$
$\widehat{x}_2 j(t) = x_2 j(t)$
end
end
end
```

6.11. Planificación de movimiento y trayectorias

En los robots móviles, el problema de planificar una trayectoria se puede resolver encontrando una ruta y definiendo una ley de sincronización en el camino. Se puede utilizar cualquier esquema de interpolación para planificar la trayectoria de tal manera que se satisfagan las condiciones de frontera apropiadas. La evolución de otras variables de configuración, junto con las entradas de control asociadas, se pueden calcular algebraicamente. La ruta de espacio de configuración satisfará automáticamente las restricciones no holonómicas.

En cuanto a la planificación de trayectorias, una vez determinado el un camino, es posible

elegir una ley de tiempo con la que el robot deba seguirla. En general, estos se integran en el procedimiento de diseño como la optimización de un criterio de coste adecuado a lo largo de la trayectoria. Una técnica sencilla para atacar el problema de planificación óptima consiste en parametrizar excesivamente el esquema de interpolación adoptado, a fin de perseguir la mejora. Normalmente, se adoptan técnicas numéricas del criterio de coste eligiendo adecuadamente los parámetros redundantes [10].

6.11.1. Aplicaciones de AG para planificación de movimiento

Se han utilizado algoritmos genéticos, aprovechando sus capacidades para resolver problemas de optimización, para encontrar el camino más seguro y evitar la colisión con obstáculos. En el Artículo [5], se aplica un algoritmo genético para generar el mejor camino para un robot móvil con accionamiento diferencial y que este evite colisiones con obstáculos estáticos en el entorno. El método propuesto propone un proceso de mutación para ampliar el espacio de búsqueda.

El método propuesto para la planificación consta de un individuo, en este caso el cromosoma, como una celda de 5 puntos de referencia, es decir, $(x_0, y_0), (x_1, y_1)...(x_4, y_4)$. Estos puntos de referencia actuarán como genes en el algoritmo genético diseñado. El cuanto al modelo ambiental, las condiciones se necesita información sobre obstáculos, como forma, número tamaña y posición. En este estudio el entorno tiene un tamaño de matriz 150×150 como se muestra en la Figura 10. Este artículo propone un algoritmo genético cruzado modificado que tiene como objetivo obtener un nuevo cromosoma que tenga buenos valores de aptitud [5].

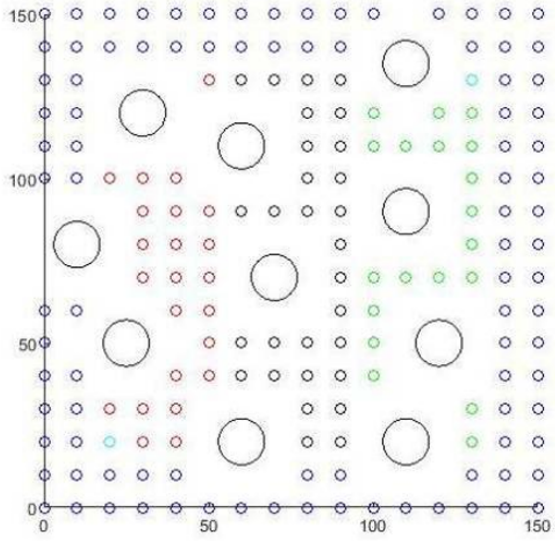


Figura 10: Entorno de prueba para algoritmo genético [5]

Para la verificación, se evalúa la colisión de la generación de la población inicial para que el camino sea factible o no. Los pasos comunes para comprobarlo es mediante el cálculo de la distancia euclidiana con la Ecuación (5).

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{5}$$

6.11.2. Codificación de variables de tipo flotante en *path-planning*

El cromosoma en este problema de búsqueda se representa como un vector de solución. La codificación que se utiliza en esta técnica son enteros de coordenadas cartesianas. En la Figura 11 se muestra el número de genes de un cromosoma como la representación general, es decir, la forma de expresar la codificación. Dos genes representan el punto inicial y final, los puntos fijos, siendo los intermedios los que cambian durante el proceso [5].

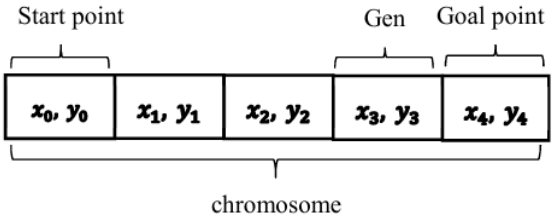


Figura 11: Codificación de solución, representación del cromosoma [5]

6.11.3. Función de aptitud para trayectorias

Para encontrar la calidad del cromosoma en este problema, se debe tener un punto de referencia claro, es decir ¿qué cualidad se debe evaluar? Para esto se utiliza la función de aptitud de la Ecuación (6), que en el problema de planificación evalúa la distancia más corta y libre de colisiones. Basado en la ecuación, se puede deducir que el mejor cromosoma, con distancia más corta, tendrá el máximo valor de aptitud.

$$f(i, j) = \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{6}$$

6.11.4. Cruce aritmético

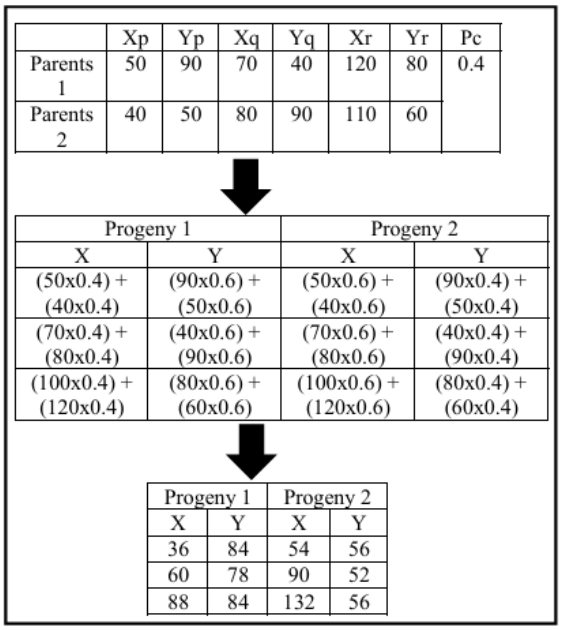
En este problema en especial, se tiene un cromosoma en el cual se comparara la aptitud del padre con el hijo. Ya que se definió una representación de tipo entero, se seguirá la regla de cruce aritmético. Con el siguiente criterio:

- Si el valor de aptitud del progenitor es mejor que el del padre, el el progenitor pasa a la nueva generación. Luego se realiza mutación aleatoriamente.
- Si el valor de aptitud del padre sigue siendo mejor que el del progenitor, se mantiene al padre en la nueva generación. Luego se realiza la mutación aleatoriamente.

El padre principal que se usará principalmente para el cruce aritmético será el mejor individuo de la generación. Este será cruzado con otro cromosoma de la misma generación tomado al azar. El propósito de este procedimiento es mantener los buenos genes y heredarlos. Hay varios métodos de cruza, como fueron mencionados anteriormente, pero en este caso, se utilizará el cruce aritmético representado por la Ecuación 7. Este método se explica mejor en la Figura 12 como una suma aritmética de las componentes del gen [5].

$$\begin{aligned} C_1 &= \lambda_1 X + \lambda_2 Y \\ C_2 &= \lambda_2 X + \lambda_1 Y \end{aligned} \tag{7}$$

Donde C_1 y C_2 son los dos progenitores resultantes del cruce aritmético entre los padres con la mejor aptitud.



	Xp	Yp	Xq	Yq	Xr	Yr	Pc
Parents 1	50	90	70	40	120	80	0.4
Parents 2	40	50	80	90	110	60	

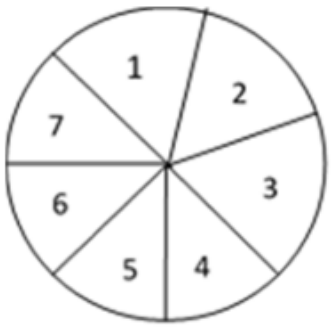
Progeny 1		Progeny 2	
X	Y	X	Y
(50x0.4) + (40x0.4)	(90x0.6) + (50x0.6)	(50x0.6) + (40x0.6)	(90x0.4) + (50x0.4)
(70x0.4) + (80x0.4)	(40x0.6) + (90x0.6)	(70x0.6) + (80x0.6)	(40x0.4) + (90x0.4)
(100x0.4) + (120x0.4)	(80x0.6) + (60x0.6)	(100x0.6) + (120x0.6)	(80x0.4) + (60x0.4)

Progeny 1		Progeny 2	
X	Y	X	Y
36	84	54	56
60	78	90	52
88	84	132	56

Figura 12: Ejemplo de cruce aritmético para cromosomas de 2 dimensiones cartesianas [5]

6.11.5. Ruleta para cruce y mutación

Debido a que en este caso se pretende utilizar el método de "cruce uniforme", se debe tener una máscara binaria. La máscara permitirá indicar los puntos que se cruzan a favor del progenitor 1 y el progenitor 2. Los puntos de cruce serán seleccionados de manera aleatoria usando el método de la ruleta giratoria. Luego de seleccionar los puntos de cruza se realizará el cruce aritmético [5]. El método de la ruleta se muestra gráficamente en la Figura 13, dependiendo de la longitud del cromosoma, igual será la longitud de la cadena binaria.



0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Figura 13: Representación del método de ruleta giratoria [5]

6.12. Otras aplicaciones de algoritmos genéticos para problemas de planificación de movimiento

El AG debe ser capaz de negociar su ruta desde su punto de partida hasta el punto final. También es necesario incluir dentro de la solución la capacidad de optimizar estas rutas para múltiples puntos de inicio [11]. Para codificar las soluciones, la población es representada por una cadena de valores en la Figura 14. En este esquema se codifican con base en la dirección de movimiento como *North, West, East,* etc.

Value	Interpretation
0	N
1	E
2	S
3	W
4	NE
5	SE
6	SW
7	NW
8	WAIT

Figura 14: Solución a la codificación de variables de movimiento [11]

Esta es otra manera de representar las variables para el problema de búsqueda para trayectoria libre de obstáculos. Sin embargo, el problema sera la codificar las cadenas del cromosoma, es posible pero es una tarea larga. Además, tiene un tiempo computacional muy alto según los resultados descritos en [11].

6.13. Otras aplicaciones exitosas de AG en diferente área

6.13.1. Algoritmos genéticos para segmentación de imágenes

Se propone utilizar un algoritmo genético para explorar el espacio de soluciones, en el sentido de que cada pixel se agrupa en otros pixeles mediante una función de distancia basada en segmentos ya calculados, tanto locales como globales. Las dos cuestiones que son abordadas para adaptar un AG para la segmentación de imágenes son:

- Seleccionar una representación apropiada para cada solución potencial.
- Formular la función de aptitud adecuada en términos de tamaño.

Los algoritmos de segmentación de imágenes contienen parámetros que se utilizan para controlar los resultados de la segmentación. El sistema genético puede cambiar dinámicamente los parámetros para lograr un mejor rendimiento [12]. El proceso del AG basado en segmentación se puede resumir de la manera que se muestra en la Figura 15. En este esquema se representa la segmentación como valores aleatorios de una población inicial, después una evaluación de resultados que termina si satisface el resultado, de lo contrario se repite el ciclo por medio de los operadores de reproducción.

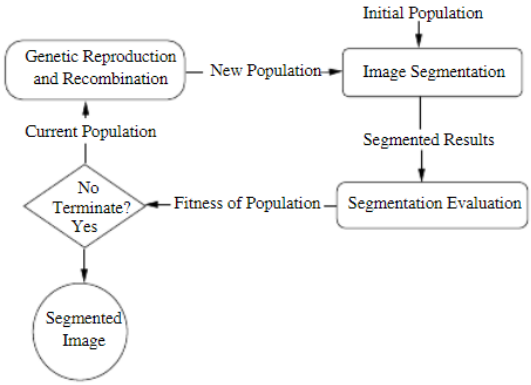


Figura 15: Proceso de algoritmo genético para segmentación de imágenes [12]

Función de aptitud

La calidad de cada cromosoma se evaluó mediante una función de aptitud basada en la entropía intrarregión (H_{intra}) y la entropía interregión (H_{inter}). La función se definió como:

$$f(C) = -\left(\frac{H_{\text{intra}}}{H_{\text{inter}} + \epsilon}\right)$$

Donde:

- H_{intra}: entropía promedio de las intensidades dentro de cada región.
- H_{inter}: entropía de las intensidades entre regiones.
- ϵ: término pequeño para evitar divisiones por cero.

La entropía intrarregión se calculó como:

$$H_{\text{intra}} = \frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{n_i} -p_{ij} \log(p_{ij})$$

Donde:

- k: número de regiones segmentadas.
- n_i: número de píxeles en la región R_i.
- p_{ij}: probabilidad de que un píxel en la región R_i tenga el nivel de intensidad j.

La entropía interregión se definió en términos de la distribución global de intensidades, calculada como:

$$H_{\text{inter}} = -\sum_{j=1}^{n} P_j \log(P_j)$$

Donde:

- n: número total de niveles de intensidad.
- P_j: probabilidad de intensidad global para el nivel j en toda la imagen.

$$f(C) = -\left(\frac{H_{\text{intra}}}{H_{\text{inter}} + \epsilon}\right)$$

Figura 16: Función de aptitud basada en la entropía intra e interregión [13]

6.14. Procesamiento de imágenes

6.14.1. Segmentación de imágenes

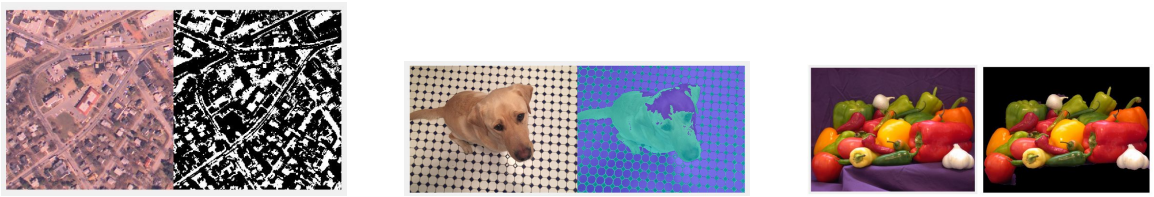
La segmentación de imágenes es una técnica utilizada en el procesamiento digital para dividir una imagen en múltiples regiones basándose en las características de los píxeles, como

color o forma. Esto permite separar el fondo del primer plano o agrupar áreas similares, por ejemplo, en aplicaciones médicas para detectar tumores en imágenes cerebrales o de otros órganos. Su objetivo principal es identificar y resaltar las partes relevantes de la imagen para un análisis más detallado.

Se han desarrollado numerosos algoritmos para la segmentación, adaptados a aplicaciones específicas como la conducción autónoma, la video vigilancia y la visión artificial. Este proceso comienza convirtiendo la imagen en regiones o máscaras etiquetadas, lo que facilita seleccionar y procesar únicamente las áreas de interés, optimizando los recursos y mejorando la eficiencia del análisis.

Técnicas de segmentación de imágenes

- ***Thresholding*:** la umbralización es una técnica de segmentación basada en el establecimiento de un valor límite para separar regiones en una imagen. Los píxeles se asignan a diferentes grupos según si sus valores son mayores o menores al umbral. Es útil en imágenes con contrastes claros entre fondo y objeto, y puede implementarse de manera global o local.
- ***Clustering*:** el agrupamiento clasifica los píxeles en grupos homogéneos basados en características como color, intensidad o textura. Técnicas como *K-means* agrupan píxeles según su similitud, generando regiones segmentadas. Es adecuada para imágenes con múltiples colores o texturas.
- **Segmentación basada en grafos:** la segmentación basada en grafos representa la imagen como un grafo donde los píxeles son nodos y las conexiones indican similitud. Los algoritmos dividen el grafo en subgrafos o regiones mediante criterios de minimización de cortes, logrando una segmentación eficiente en imágenes complejas.
- **Basada en crecimiento:** el crecimiento de regiones comienza con un conjunto inicial de píxeles o semillas, expandiendo las regiones al agregar píxeles vecinos que cumplan ciertos criterios de similitud. Es útil para segmentar objetos con límites bien definidos y propiedades homogéneas.
- ***Deep learning*:** la segmentación basada en aprendizaje profundo, utiliza redes neuronales convolucionales (*CNNs*) para aprender patrones y características directamente de las imágenes. Modelos como *U-Net* han demostrado ser efectivos para segmentar objetos en escenarios complejos, como imágenes médicas o escenas naturales [14].

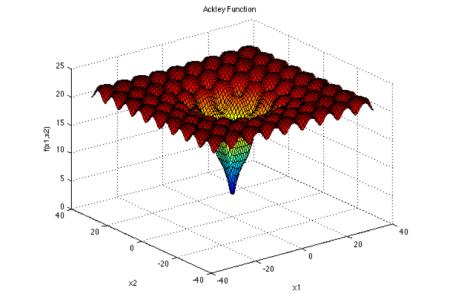


(a) *Thresholding*. (b) *Clustering* (*K-means*) (c) Agrupación por grafos

Figura 17: Ejemplos de segmentación de imágenes utilizando diferentes técnicas [14]

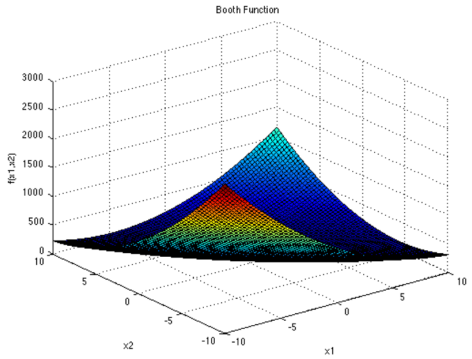
6.15. Funciones de costo

Las funciones de costo son un elemento central en problemas de optimización. Estas funciones cuantifican la calidad de una solución en relación con el problema que se está resolviendo. En términos simples, una función de costo asigna un valor numérico a cada posible solución, representando qué tan "buena" o "mala" es esa solución [15]. En [2], se utilizan para evaluar los costos que requiere el algoritmo *Particle Swarm Optimization*. Las funciones que se muestran a continuación, son algunas de las funciones y conjuntos de datos comunes que se utilizan para probar algoritmos de optimización.



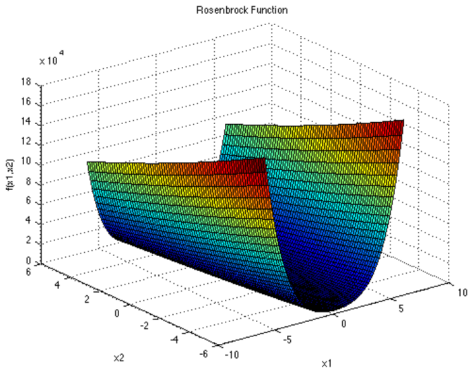
$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Figura 18: Función de Ackley [15]



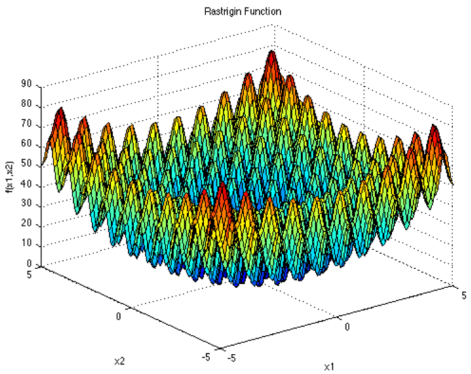
$$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Figura 19: Función de Booth [15]



$$f(\mathbf{x}) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Figura 20: Función de Rosenbrock [15]



$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Figura 21: Función de Rastrigin [15]

6.16. Funciones de superficies

Las superficies cuadráticas son superficies en un espacio tridimensional que se describen mediante ecuaciones cuadráticas en dos variables. Estas superficies son útiles en problemas

de optimización porque suelen tener formas bien definidas, lo que permite identificar fácilmente los puntos máximos o mínimos en problemas de optimización [16]. A continuación, se presentan 3 diferentes tipos de superficies.

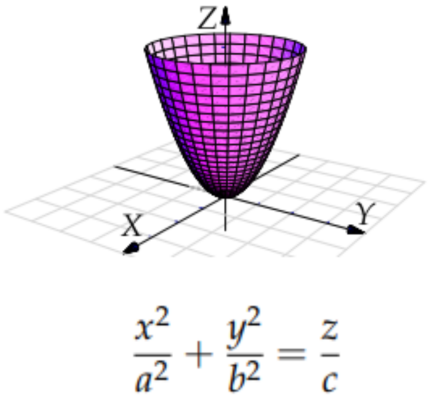


Figura 22: Función de paraboloide elíptico [15]

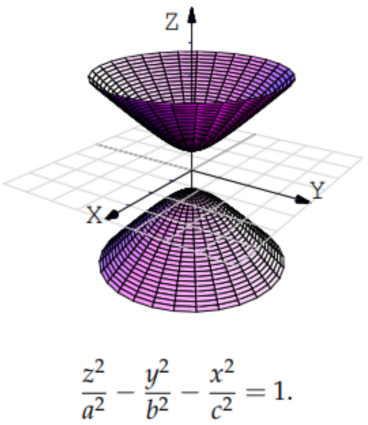


Figura 23: Función de hiperboloide de 1 hoja [15]

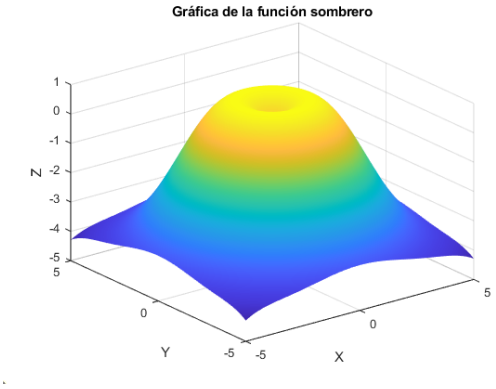


Figura 24: Función de campana de Gauss [15]

6.17. Controlador no lineal de pose para robots móviles

El controlador no lineal de pose para robots móviles utiliza un enfoque basado en la cinemática diferencial para controlar el robot y hacer que siga una trayectoria deseada. Este controlador típicamente usa tres constantes:

- k_ρ: ganancia para la distancia objetivo
- k_α: ganancia para el ángulo de orientación respecto al objetivo
- k_β: ganancia para el ángulo de orientación relativo al objetivo

Estos controladores están diseñados para garantizar que el robot se acerque a la trayectoria deseada y siga la dirección correcta para minimizar el error en la posición y orientación del robot. Las ecuaciones típicas del controlador no lineal de pose para robots móviles se basan en la cinemática diferencial, expresándose como en la Ecuación 8, que describe el control de velocidad lineal v y velocidad angular ω [17].

$$\begin{aligned} v &= K_\rho \rho \\ \omega &= K_\alpha \alpha + K_\beta \beta \end{aligned} \tag{8}$$

6.18. Intersección de una recta con un plano limitado en un espacio tridimensional

En la planificación de trayectorias 3D es posible encontrar varias formas de medir la colisión con obstáculos. En cuanto la experimentación descrita en esta investigación se requirió de un método que calcule de alguna forma la distancia o intersección de una partícula con un plano. Por lo que, se utilizó el método propuesto por un cálculo algebraico que permite conocer si la recta entre dos puntos interseca el plano. Sea un espacio tridimensional $\mathbb{R}^3$, donde se obtiene que:

- Un plano Π dado por la ecuación $y = c$, limitado por los intervalos:

$$0 < x < n, \quad 0 < z < k.$$

- Una recta parametrizada que conecta dos puntos $\mathbf{p_1} = (x_1, y_1, z_1)$ y $\mathbf{p_2} = (x_2, y_2, z_2)$, descrita en forma paramétrica como:

$$\mathbf{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} (1-t)x_1 + tx_2 \\ (1-t)y_1 + ty_2 \\ (1-t)z_1 + tz_2 \end{pmatrix},$$

  donde $t \in [0, 1]$ [18].

Se determinará si la recta $\mathbf{r}(t)$ intersecta el plano Π dentro de los límites establecidos.

6.18.1. Ecuación para la intersección con el plano $y = c$

Para encontrar el parámetro t en el que la recta intersecta el plano Π, se utiliza la condición:

$$y(t) = c,$$

donde $y(t) = (1-t)y_1 + ty_2$. Sustituyendo esta expresión, se determina:

$$(1-t)y_1 + ty_2 = c.$$

Reorganizando para despejar t:

$$y_1 - ty_1 + ty_2 = c,$$

$$y_1 + t(y_2 - y_1) = c,$$
$$t = \frac{c - y_1}{y_2 - y_1}.$$

Este valor de t representa el punto en la recta donde $y(t) = c$ [18]. Es importante verificar si:

$$t \in [0, 1].$$

- Si $t \in [0, 1]$, la recta intersecta el plano $y = c$ entre los puntos $\mathbf{p_1}$ y $\mathbf{p_2}$.
- Si $t \notin [0, 1]$, la recta no intersecta el plano $y = c$ entre los puntos $\mathbf{p_1}$ y $\mathbf{p_2}$, aunque podría intersectarlo fuera de este segmento.

6.18.2. Condición adicional: verificación de los límites en x y z

Si $t \in [0, 1]$, la recta intersecta el plano $y = c$. Sin embargo, el plano Π está limitado a un rectángulo definido por $0 < x < n$ y $0 < z < k$. Se debe verificar si las coordenadas $x(t)$ y $z(t)$ del punto de intersección también cumplen estas restricciones.

Las coordenadas $x(t)$ y $z(t)$ se calculan sustituyendo t en las expresiones paramétricas:

$$x(t) = (1 - t)x_1 + tx_2, \quad z(t) = (1 - t)z_1 + tz_2$$

Luego se verifican las condiciones:

$$0 < x(t) < n, \quad 0 < z(t) < k.$$

- Si ambas condiciones son satisfechas, el punto de intersección está dentro de la región limitada del plano Π.
- Si cualquiera de las condiciones no se cumple, el punto de intersección está fuera de los límites permitidos en x y z, por lo que la recta no intersecta la región limitada del plano Π [18].

6.18.3. Criterio de verificación

Este criterio se resume a un código computadorizado que realiza los cálculos eficientemente. El procedimiento para determinar si la recta intersecta el plano limitado se resume al siguiente:

- Calcular el valor de t usando:
$$t = \frac{c - y_1}{y_2 - y_1}.$$
- Verificar si $t \in [0, 1]$:
  - Si $t \notin [0, 1]$, la recta no intersecta el plano Π en el segmento $\mathbf{p_1 p_2}$.

- Si $t \in [0,1]$, calcular las coordenadas del punto de intersección:

$$x(t) = (1-t)x_1 + tx_2, \quad z(t) = (1-t)z_1 + tz_2.$$

- Verificar las condiciones:

$$0 < x(t) < n, \quad 0 < z(t) < k.$$

- Concluir:
  - Si ambas condiciones se satisfacen, la recta intersecta la región limitada del plano Π.
  - Si cualquiera de las condiciones no se cumple, el punto de intersección está fuera de la región limitada del plano Π.

CAPÍTULO 7

Validación general de algoritmo genético

En este capítulo se presenta la metodología aplicada para la evaluación de los algoritmos genéticos en su forma general. También, se describe el procedimiento realizado para la obtención de los resultados de optimizar funciones de costo y matemáticas. Estas pruebas se basan en buscar puntos máximos y mínimos dentro de un dominio. Para las simulaciones en este capítulo se utilizó lenguaje MATLAB versión R2020a. Los resultados obtenidos tienen el objetivo de garantizar el funcionamiento general de los AG en aplicaciones básicas. Con el concepto de este tipo de algoritmo, se aplican estructuras similares adaptadas a otro tipo de aplicación en los siguientes capítulos.

7.1. Algoritmos genéticos para optimizar funciones de costo

Previamente se mencionó sobre una aplicación de AG general para evaluar costos en un algoritmo de planificación de trayectorias [2]. Esta investigación tuvo como objetivo incursionar en el área, planteando el problema para optimizar las funciones de costo (mínimo) requerida en el *Particle System Optimization* (PSO). El problema consiste en 4 funciones principales descritas en el marco teórico: Rosenbrock, Ackley, Rastrigin y Booth. Para adaptar el AG a este planteamiento se utiliza la estructura algorítmica de la Figura 6. El proceso consiste en armar la estructura ya mencionada, implementando el algoritmo descrito en 6.1.

Para crear el algoritmo del ciclo principal se desarrollaron las diferentes funciones por separado utilizando MATLAB. Las funciones desarrolladas para este algoritmo: generación de población inicial, codificación de individuos en población, definición de la función objetivo (aptitud), asignación de rango, ruleta, cruce de un punto, mutación, selección y decodificación. Las funciones mencionadas en el marco teórico, están compuestas de los algoritmos descritos para el cruce, para ruleta y codificación. La adaptación del algoritmo genético a este problema se describe en el siguiente diagrama de flujo de la Figura 25.

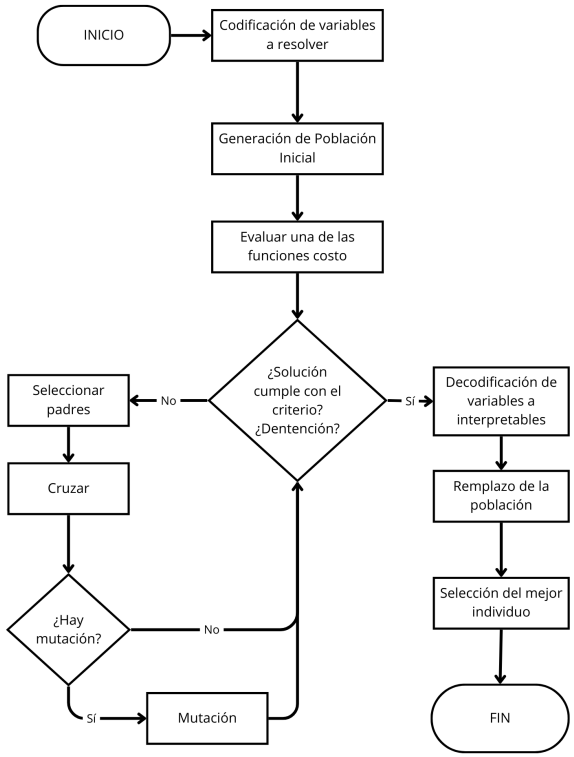


Figura 25: Diagrama de flujo de AG para minimizar costos

Luego de tener listo el algoritmo para la optimización de las funciones costo, se establecieron diferentes parámetros de simulación. El objetivo de variar los parámetros es validar distintos escenarios, para poder implementar esta estructura a otro concepto de búsqueda y optimización. En el Cuadro 1, se describe el intervalo cerrado donde se realiza el espacio de búsqueda del AG, para todas las validaciones.

Función	**Dominio x**	**Dominio y**	Longitud
Rosenbrock	[-5, 10]	[-5, 10]	20
Ackley	[-32.768, 32.768]	[-32.768, 32.768]	24
Rastring	[-5.12, 5.12]	[-5.12, 5.12]	20
Booth	[-10, 10]	[-10, 10]	20

Cuadro 1: Parámetros de simulación en base a su función de costo

Una mejor perspectiva de lo que se habló es ver la representación gráfica de lo que implica codificar las variables del problema. Para codificarlas se utiliza el proceso inverso visto en la Sección 6.11. Es decir, utilizando una representación de cromosoma y codificación binaria. Esta definición se resume en crear una población binaria de longitud l donde los n bits más significativos representan la variable x y los m bits menos significativos la variable y. La Figura 26 representa de manera visual lo descrito anteriormente, con un ejemplo de $n = 8$ y $m = 12$, pues este sería nuestro cromosoma codificado en binario.

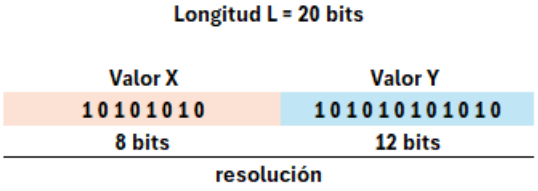


Figura 26: Representación y codificación del cromosoma para problema de optimización de costo

En cuanto a la decodificación de las variables para este problema, primero se define que estas se decodifican únicamente cuando se consigue la población final. Esto es con fines de representar el resultado como una cifra entendible a las variables traducidas, es decir, un valor (x, y). Para lograr la traducción de la variable utilizaremos la Ecuación (9) que contiene la conversión de la parte binaria a decimal para mapear a coordenadas cartesianas la solución.

$$\begin{aligned} x_i &= l_{\text{inf}} + \text{decimal}\,(11001\ldots11_2)\left(\frac{l_{\text{sup}} - l_{\text{inf}}}{2^{l_{\text{ind}}} - 1}\right) \\ y_i &= l_{\text{inf}} + \text{decimal}\,(11001\ldots11_2)\left(\frac{l_{\text{sup}} - l_{\text{inf}}}{2^{l_{\text{ind}}} - 1}\right) \end{aligned} \tag{9}$$

7.1.1. Resultados de primer escenario para optimización de función de costo

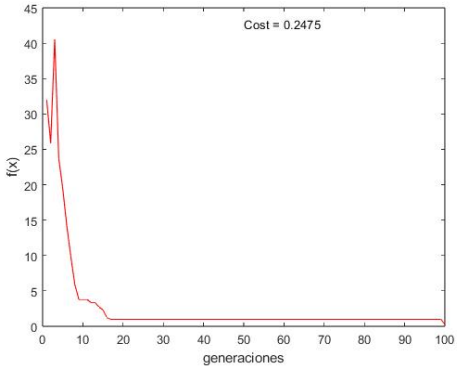
Para el primer escenario, se realizó la validación de las funciones con los siguientes parámetros: probabilidad de cruce 90 %, probabilidad de mutación 80 %, número de individuos 100 por población y la longitud de cada individuo de 20 bits (10 en x y 10 en y). Este escenario se hace con un alto porcentaje de mutación para explorar la capacidad de búsqueda del algoritmo. Al correr el algoritmo en MATLAB, se obtuvieron los resultados que se presentan en el Cuadro 2. Estos resultados describen el costo mínimo encontrado en un intervalo establecido. También se presentan datos relevantes, como tiempo computacional y el número de iteraciones necesarias para el resultado final.

Función	Iteraciones	Tiempo (s)	Costo (x, y)
Rastrigin	100	0.801	0.2475
Rosenbrock	12	0.790	0.0466
Ackley	19	1.015	0.0354
Booth	100	1.049	0.4649

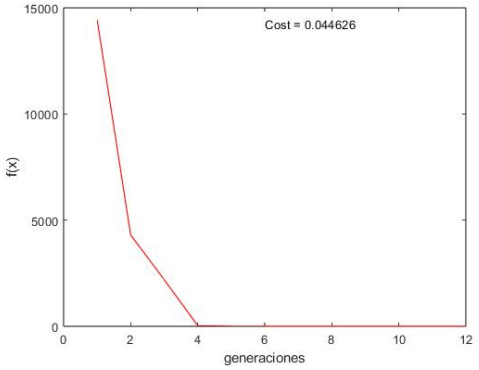
Cuadro 2: Resultados de optimización de funciones costo, primer caso

Adicional, en la siguiente Figura 27 se muestran en conjunto las soluciones (costo) en función de las generaciones (iteraciones) realizadas por el algoritmo. Estos gráficos demuestran cómo evoluciona el valor hasta el más óptimo con el paso de las generaciones. Se detiene

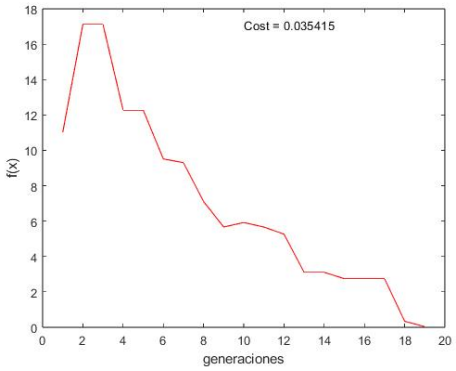
cuando alcanza una tolerancia del 15 % o en todo caso, cuando este llega a su número máximo de iteraciones.



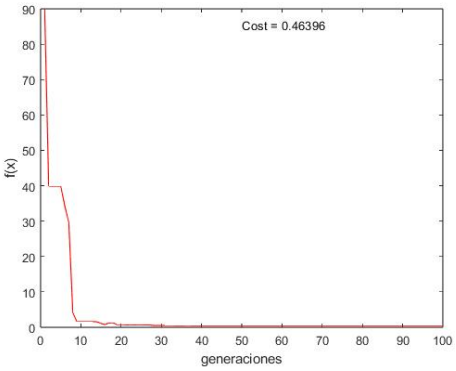
(a) Función costo de Rastrigin



(b) Función costo de Rosenbrock



(c) Función costo de Ackley.



(d) Función costo de Booth

Figura 27: Funciones de costo vs. generaciones, primer caso

La Figura 27, muestra la evolución del algoritmo conforme ocurren las generaciones. Se evidencia una pronta convergencia hacia el valor final estimado. Dado que el porcentaje mutación es un valor grande, se tiene un espacio de búsqueda más alto. En [2] se nos menciona que no son los resultados más óptimos y que estos son preliminares para una primera aplicación. Sin embargo, la estructura algorítmica demostró funcionar para el problema de optimización, independientemente de sus parámetros de funcionamiento. Por lo que esta validación permitió, más adelante, adaptar el algoritmo a otros problemas.

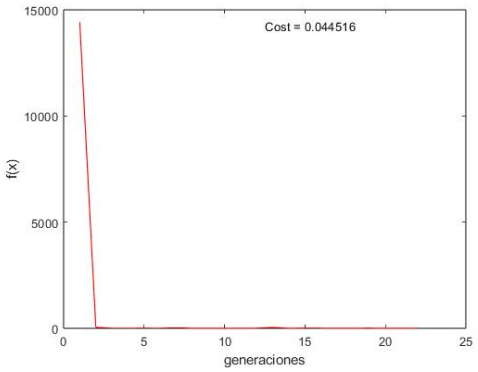
7.1.2. Resultados de segundo escenario para optimización de función de costo

Para el segundo escenario, se realizó la validación de las funciones con los siguientes parámetros: probabilidad de cruce 80 %, probabilidad de mutación 15 %, número de individuos 100 por población y la longitud de cada individuo de 20 bits (10 en x y 10 en y). Este escenario se hace con un bajo porcentaje de mutación para reducir la capacidad de búsqueda del algoritmo y comparar con el de la sección anterior. Al correr el algoritmo en MATLAB,

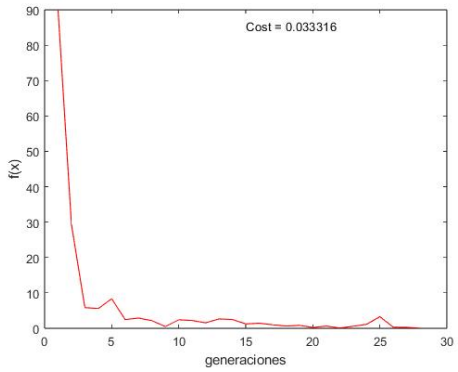
se obtuvieron los resultados que se presentan en el Cuadro 3. Estos resultados describen el costo mínimo encontrado en un intervalo establecido. Al igual que en el caso anterior, se presentan el conjunto de gráficos de costo, Figura 28.

Función	Iteraciones	Tiempo (s)	Costo (x, y)
Rastrigin	75	0.496	0.0445
Rosenbrock	33	0.347	0.0333
Ackley	57	0.586	0.0354
Booth	16	0.521	0.0300

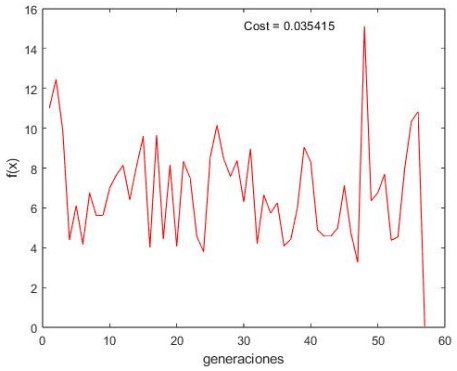
Cuadro 3: Resultados de optimización de funciones costo, segundo caso



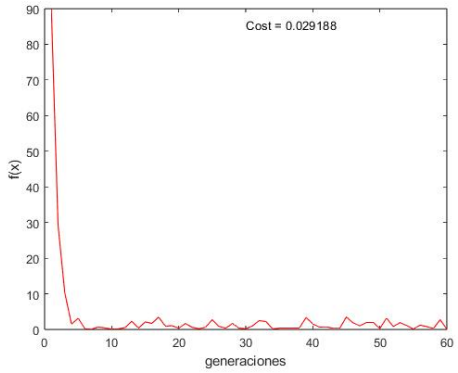
(a) Función costo de Rastrigin



(b) Función costo de Rosenbrock



(c) Función costo de Ackley



(d) Función costo de Booth

Figura 28: Funciones de costo vs. generaciones, segundo caso

Estos resultados nos muestran una convergencia más rápida. Tenemos cierta varianza con el caso anterior, sin embargo, esto se debe a varios factores. El factor principal es que reducimos el espacio de búsqueda aleatoria (mutación). Esto puede ser de beneficio en ciertas ocasiones, debido a que para la función de Rastrigin y Booth, se ajustó mejor y tuvo una convergencia mas rápida. Como se mencionó anteriormente, fueron resultados preliminares para la aplicación que se le dio en PSO. En este caso, se esta satisfecho con el hecho de su funcionalidad para adaptarlo a las aplicaciones descritas en los objetivos.

Con base en la validación de estos antecedentes, podemos inferir que la estructura mencionada para un algoritmo evolutivo, algoritmo genético, es válida para generar soluciones como problema de búsqueda u optimización. Se desarrolló, en la siguiente sección, una implementación par un problema de optimización más simple y comprensible que replique esta estructura.

7.2. Algoritmos genéticos para optimización de máximos y mínimos en funciones algebraicas multivariable

En este caso particular, se validó el algoritmo genético enfocado a otro tipo de problema. Este problema se trata de buscar máximos y mínimos en intervalos cerrados, para funciones de tipo $f(x, y)$ en diferentes escenarios. El objetivo es traducir la estructura mostrada en el Algoritmo 6.1 a problemas de optimización más entendibles, así poder hacer la re-interpretación a conveniencia. Para hacerlo, se utilizaron las funciones propuestas en el marco teórico, sección 6.2, siendo superficies con puntos máximos y mínimos a simple vista.

El proceso para el AG, es el mismo que en la sección anterior, al igual que el diagrama de flujo mostrado en la Figura 25. En esta caso cambia la interpretación del problema, las funciones y los parámetros de simulación. La condición de detención se basa en el criterio mencionado en el marco teórico: terminar cuando no haya cambios en la población. Por lo tanto, la población será interpretada como los posibles puntos máximos o mínimos de la superficie $f(x, y)$, generando posibles puntos en x y y. El algoritmo debe encargarse de buscar la solución más apropiada en el intervalo cerrado $x_b < x < x_b$ y $y_a < y < y_b$.

Las tres principales funciones a evaluar son una hipérbola, un paraboloide y la campana de Gauss. Estas fueron evaluadas en 3 escenarios distintos con el objetivo de notar cambios radicales importantes. Dado que todas las funciones están centradas en el origen, se utilizaran los siguientes parámetros en común para los 3 casos, mostrados en el Cuadro 4.

Función (x,y)	**Dominio x**	**Rango y**	**Longitud**
Hipérbola	[-5, 5]	[-5, 5]	20
Paraboloide	[-5, 5]	[-5, 5]	20
C. de Gauss	[-5, 5]	[-5, 5]	20

Cuadro 4: Parámetros de simulación para optimizar superficies

Se eligieron los intervalos dado por conveniencia, debido a que gráficamente es más fácil identificar los máximos o mínimos. Esto será una ventaja para comparar las pruebas del algoritmo. La resolución o longitud de 20 bits es para tener 10 bits en x y 10 bits en y. Se recomienda usarlo en esa longitud para que no tome mas tiempo computacional. Replicando la representación y codificación binaria para el cromosoma, en este caso, se muestra en la forma de la Figura 29.

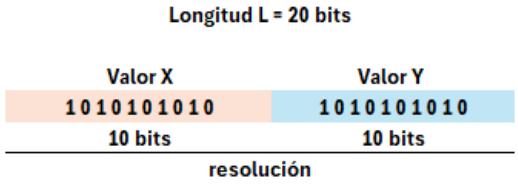


Figura 29: Representación y codificación del cromosoma para problema de optimización de superficies

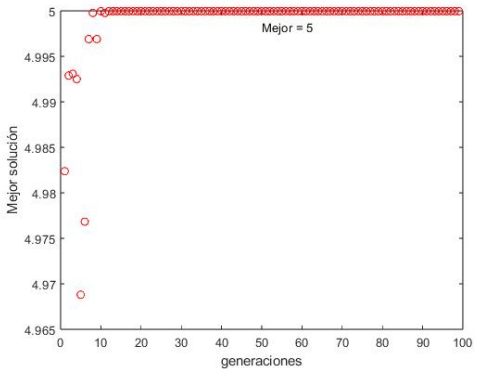
En cuanto a la decodificación de las variables de este problema, se utilizó la Ecuación (9). Considerando los límites de dominio y rango para las variables a traducir como cotas superiores e inferiores en la ecuación. Esta función tomará la solución generada en valor decimal para convertirla a un valor entero flotante interpretable.

7.2.1. Primer escenario para optimización de superficies

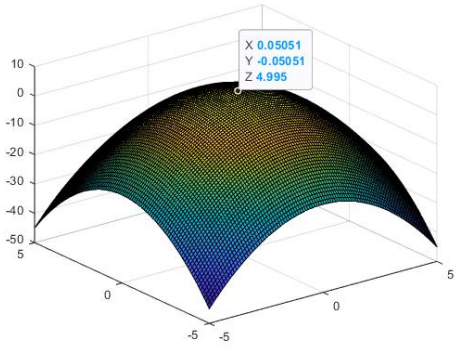
En este caso, se variaron unicamente la probabilidad de cruce, mutación y las iteraciones. Como se mencionó, se usó Pc = 90 % y Pm = 1 %, las iteraciones incrementan para evidenciar cuanto se tarda en converger. El usar un porcentaje de mutación bajo reducirá la probabilidad de búsqueda aleatoria, para comprobar el comportamiento limitado a una región. Los resultados obtenidos en este escenario se muestran en el Cuadro 5. En este cuadro se muestra el valor encontrado luego de variar las iteraciones, comparado con el valor real.

f(x, y)	Iteraciones	Tiempo (s)	Valor AG	Valor real
Hipérbola e.	4	0.7523	72.170	85.0
	30	1.9241	85.000	
	100	5.0325	85.000	
Paraboloide	4	0.4145	4.8985	5.0
	30	2.0577	5.0000	
	100	6.578	5.0000	
C. Gauss	4	0.2544	0.7911	0.795
	30	2.3354	0.7946	
	100	7.7072	0.7950	

Cuadro 5: Resultados de optimización de superficies, primer caso

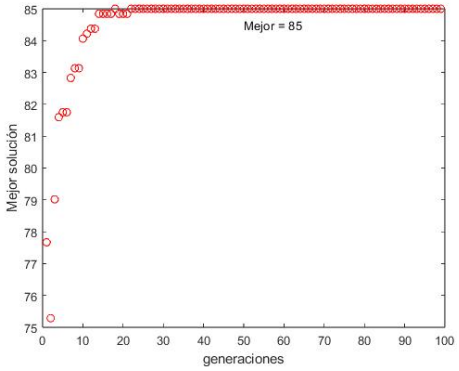


(a) Solución en hipérbola vs. generaciones

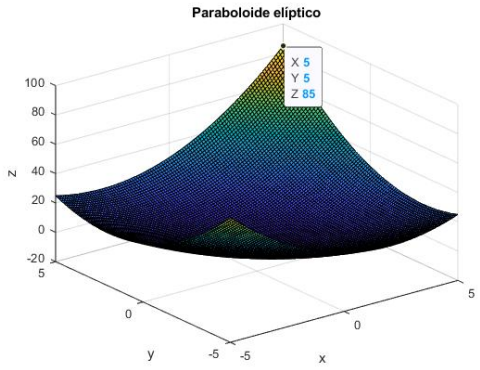


(b) Valor real en función hipérbola

Figura 30: Solución hipérbola $f(x, y)$ vs. generaciones, primer caso

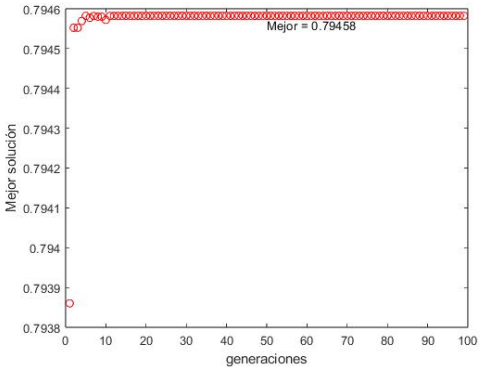


(a) Solución en paraboloide vs. generaciones

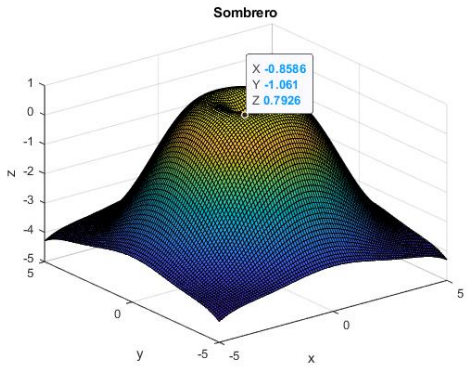


(b) Valor real en función paraboloide

Figura 31: Solución paraboloide $f(x, y)$ vs. generaciones, primer caso



(a) Solución en Gauss vs. generaciones



(b) Valor real en función Gauss

Figura 32: Solución Gauss $f(x, y)$ vs. generaciones, primer caso

Los resultados en las Figuras 30, 31 y 32, explican gráficamente como evoluciona la solución $f(x, y)$ conforme pasan las generaciones. Se puede apreciar que al cabo de 10 gene-

raciones ya converge al valor real para la mayoría de casos. Por lo que se conoce que existe eficiencia en cuanto a convergencia y dio a conocer la característica más importante para los criterios en los parámetros. Ahora considerando lo mencionado, se agregó una condición de detención favorable para reducir el tiempo computacional en la siguiente sección.

7.2.2. Segundo escenario para optimización de superficies

En este caso se realizó una modificación en la condición de detención, que ocurrirá cuando no exista variación en la solución, conforme pasen las generaciones. También se mantuvo el valor de porcentaje de cruza y mutación de Pc = 90 % y Pm = 1 %. El propósito de realizar este cambio es reducir el tiempo computacional, debido a que en el primer caso se mostró una pronta convergencia. Los resultados de este caso se muestran el Cuadro 6, donde se indican las iteraciones tomadas hasta que se cumpla la condición de detención y la solución final.

f(x,y)	Iteraciones	Tiempo (s)	Valor AG	Valor real.
Hipérbola e.	17	0.9293	85.0	85.0
Paraboloide	16	0.7695	4.9902	5.00
C. Gauss	14	1.2821	0.7946	0.795

Cuadro 6: Resultados de optimización de superficies, segundo caso

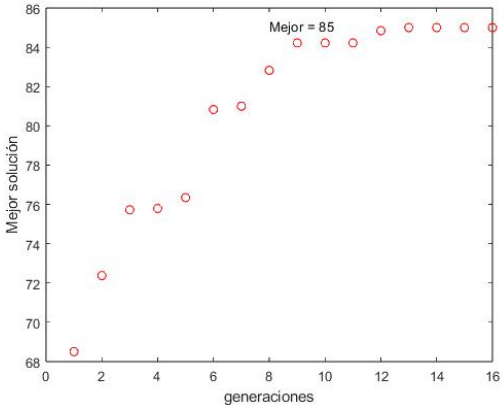


Figura 33: Solución de hipérbola vs. generaciones, segundo caso

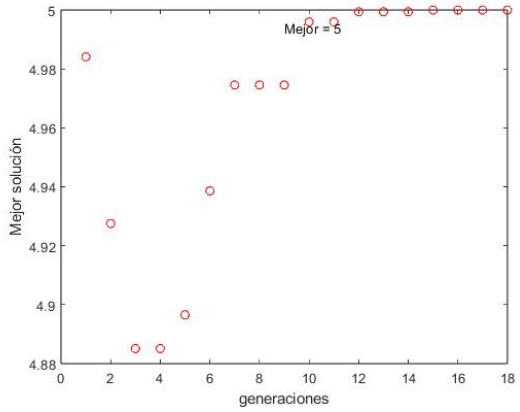


Figura 34: Solución de paraboloide vs. generaciones, segundo caso

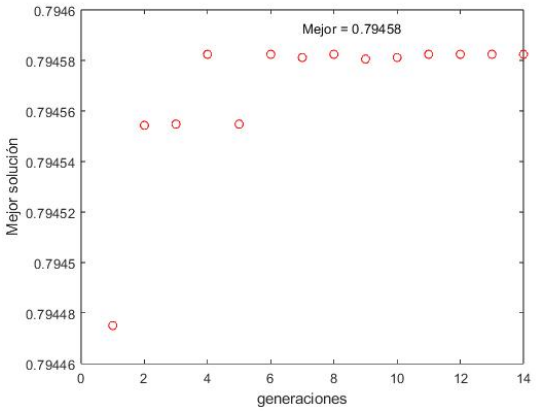


Figura 35: Solución de Gauss vs. generaciones, segundo caso

En las Figuras 33, 34 y 35 se muestran los resultados de la ejecución del algoritmo. Se aprecia cómo la función $f(x, y)$ evoluciona mientras ocurren las generaciones, pero a diferencia del caso anterior, esta se detiene cuando no hay variación. De esta forma, pudimos confirmar que el tiempo de ejecución se reduce y comparado al valor real la diferencia es insignificante. Por último, realizó otra validación incrementando la probabilidad de búsqueda autónoma del algoritmo, es decir, variando la mutación.

7.2.3. Tercer escenario para optimización de superficies

En este tercer caso, se planteó validar el funcionamiento ahora con un parámetro de búsqueda aleatoria más alto. De esta forma se comparó el comportamiento que brindó la perspectiva de funcionamiento que se aplicó más adelante. Por lo tanto, en este caso se varía el Pc a 75 % y el Pm a 5 %. Al igual que el caso anterior, se configuró el mismo criterio en detención. Los resultados de esta implementación se muestran en el Cuadro 7.

f(x,y)	Iteraciones	Tiempo (s)	Valor AG	Valor real.
Hipérbola e.	24	2.2216	85.0	85.0
Paraboloide	32	2.7656	4.9980	5.00
C. Gauss	36	3.0902	0.7946	0.7950

Cuadro 7: Resultados de optimización de superficies, tercer caso

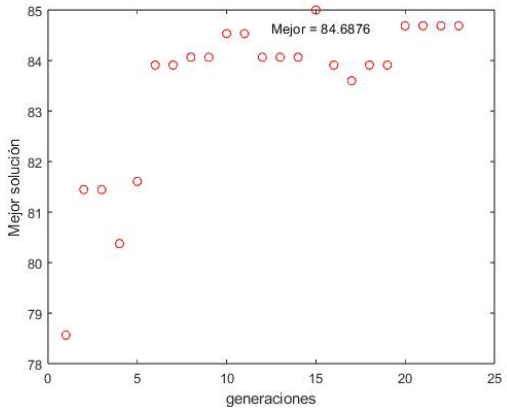


Figura 36: Solución de hipérbola vs. generaciones, tercer caso

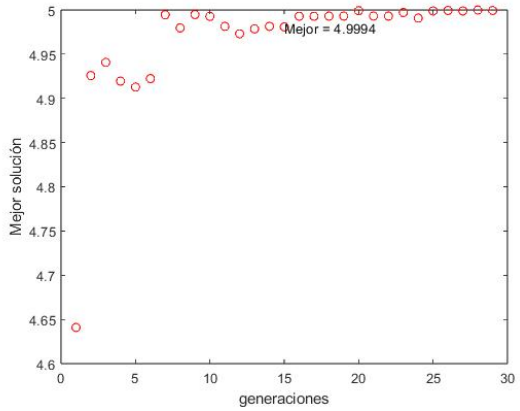


Figura 37: Solución de paraboloide vs. generaciones, tercer caso

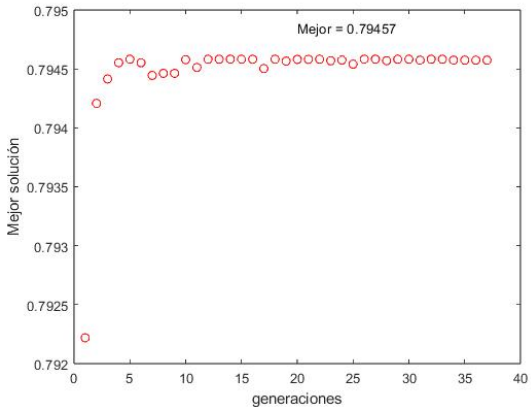


Figura 38: Solución de Gauss vs. generaciones, tercer caso

Los resultados de este tercer escenario demostraron que la exploración y explotación se hace más presente. Esto es debido a que el aumento de mutación hace que las soluciones cambien descartando algunos aspectos de los padres. El método de mutación aveces es bueno, como también puede ser malo. En este caso particular, aumenta el número de iteraciones realizadas como consecuencia. En las Figuras 36, 37 y 38, se ve una dispersión más frecuente en las soluciones conforme pasan las generaciones. Esto nos brinda una noción sobre los criterios a utilizar en las aplicaciones más especificas que se demostraron en los siguientes capítulos.

Este capítulo se centró específicamente en la evaluación de los AG aplicando la estructura investigada. Esta demostró ser funcional y aplicable a problemas básicos y generales de optimización. También evidenció las limitantes y criterios a tomar en cuenta para la selección de parámetros.

CAPÍTULO 8

Algoritmo genético para planificación de trayectorias

En este capítulo se muestra la metodología implementada para adaptar el AG en planificación de trayectorias en un escenario 2D. También se detalla el procedimiento realizado para implementar la estructura, desarrollar los entornos de búsqueda y los parámetros utilizados. En este caso se toma un robot móvil ideal como una masa puntual y los obstáculos como figuras circulares. Los resultados obtenidos en esta sección tienen como objetivo garantizar el funcionamiento para búsqueda trayectorias libres de obstáculos. No buscamos las más optimas, pero sí las mejores soluciones que pueda encontrar el algoritmo.

8.1. Metodología

Utilizando la estructura de un AG descrita en el marco teórico, se basó el procedimiento de esta implementación como lo sugiere [5]. Como primer punto, se tomó el supuesto de utilizar un robot móvil con movimiento de control lineal y angular. Este se asume como una masa puntual que recorrerá el camino, dentro del mapa de $n \times n$ unidades de longitud arbitrarias, que generará el AG. El camino consiste de un una cadena (x_0, y_0),... (x_k, y_k) con k cantidad de puntos intermedios representando el cromosoma. Los valores de punto flotante dentro del punto $p_k(x_k, y_k)$ constituyen un gen, dentro del plano 2D. La codificación de los obstáculos se puede apreciar mejor en la representación vectorial de ejemplo en 10. Esta es la representación del cromosoma como un vector trayectoria $\mathbf{T}$ de coordenadas $< x, y >$ para $n = 5$ puntos intermedios.

$$\mathbf{T} = \begin{bmatrix} 10.5 & 10.5 \\ 25.0 & 30.0 \\ 48.5 & 41.5 \\ 55.5 & 60.0 \\ 70.0 & 70.0 \end{bmatrix} \tag{10}$$

Para esta implementación no deben decodificarse las variables de solución. Esto se debe a que los datos ya son interpretables para el tipo de problema, en todo caso la decodificación vendría siendo la extracción de un elemento independiente del vector. Es decir, extraerlo y operarlo como un escalar en el paso del cruce aritmético, que se presentó en el Marco teórico 6, respecto al cruce aritmético.

El entorno donde se evaluó el algoritmo consta de una cantidad de obstáculos dentro del mapa 2D. Los obstáculos variaron conforme a los casos que se presentan más adelante. También se consideró un radio de obstáculos. Además, se definieron los puntos de inicio y los puntos de final. Estos no deben coincidir inicialmente con el área del obstáculo y deben estar dentro de las coordenadas del mapa.

Utilizando la estructura descrita en el Código 6.1, se realizó el planteamiento y adaptación del algoritmo al problema de optimización. En este caso se tiene un vector con puntos intermedios los cuales actuarán como el cromosoma o solución a optimizar. Se evaluará la aptitud de los cromosomas candidatos por medio de cual sea el camino con menor distancia. El mejor cromosoma será el padre que heredará las mejores características hasta que se cumpla el criterio de optimización. Por lo tanto, el AG funcionará de la manera descrita en el diagrama de flujo en la Figura 39.

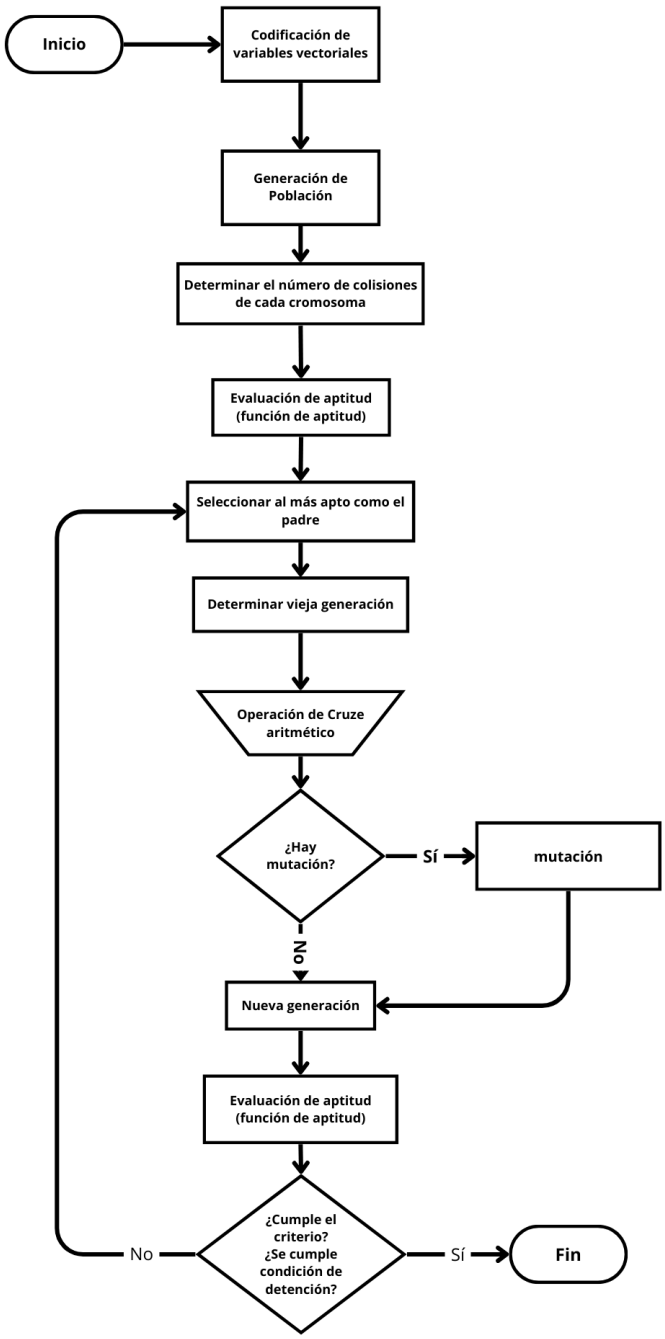


Figura 39: Diagrama de flujo para AG en planificación de trayectorias 2D

Una aclaración importante, es que la mutación ocurre de manera aleatoria para este caso. Es un parámetro independiente. Para todos los casos que se desarrollaron en las siguientes secciones se aplica el mismo proceso descrito en la figura anterior. Unicamente se variaron los obstáculos, número de puntos, tamaño de la población, número de generaciones a crear,

dimensiones del mapa y radio de los obstáculos.

8.1.1. Selección con base en la aptitud

En esta implementación se utilizó la función de aptitud descrita por la Ecuación 6. La ecuación usa la norma euclidiana como medida de calidad. De esta forma se obtiene el mejor valor de aptitud que representa el criterio de nuestra selección, mientras mayor sea la distancia que representan los puntos dentro del cromosoma, la aptitud se reducirá. Cabe resaltar que la aptitud del nuevo individuo se compara directamente con su progenitor. Si este no alcanza mejores cualidades seguirá siendo descartado y se mantendrá al padre.

8.1.2. Función de cruce aritmético

Basando el cruce de esta aplicación en el gráfico de la Figura 12 y la Ecuación 7, se utilizó un criterio modificado. Debido a que en [5], no se especifica cuál es el criterio de selección de padres, se utilizó el descrito en la figura ya mencionada. Este criterio se resume a obtener al primer padre como el mejor individuo de la población y reproducirlo con uno aleatorio. Cuando se aplique el cruce aritmético, basado en cruce uniforme, el hijo heredará características de ambos padres, pero si este no mejora, se descarta y se conserva a los padres.

8.2. Evaluación de algoritmo genético en mapas 2D

En esta sección se pone a prueba la implementación del AG descrito en la sección anterior. Se presentan los diferentes casos evaluados para su validación por medio de los parámetros de simulación utilizados. Lo que tendrán en común es el tipo de codificación de las variables, como se describe en la sección 6.11, al igual que el método de ruleta.

8.2.1. Implementación de AG para planificación de trayectorias, escenario 1

En este escenario se validó el AG en un terreno regular, como lo describe [5] para el espacio de búsqueda. Los parámetros de simulación que se implementaron son los siguientes:

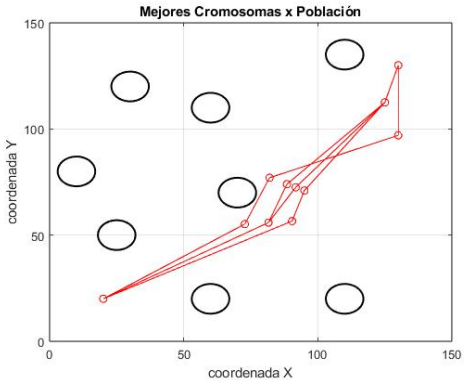
- Dimensiones del mapa: 150×150
- Cantidad de obstáculos: 9
- Centro de los obstáculos: (85,100), (25, 50), (30, 120), (60,110), (60,20), (70,70), (110,20), (110, 135) y (110, 90)
- Población: 30 individuos
- Condición de detención aplica cuando se cumplan todas las iteraciones

- Punto de inicio y punto de meta: (20,20) y (145,145)
- Cantidad de puntos intermedios: 5

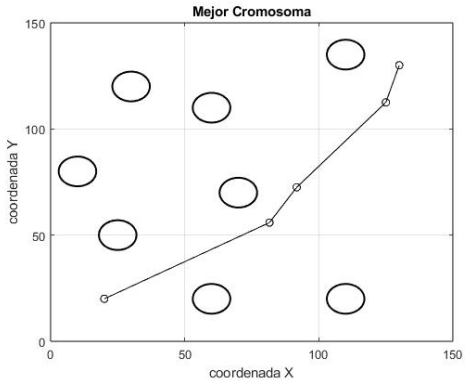
Para este caso se variaron únicamente las generaciones para determinar las mejorías. Aplicando el AG y los parámetros de simulación ya descritos, se obtuvieron los resultados del Cuadro 8 y las Figuras 40, 41 y 42. En el cuadro mencionado, se usaron diferentes iteraciones, que fueron tomadas al azar de manera incremental, para fines prácticos.

No.	Iteraciones	Población	Aptitud	Tiempo (s)	Distancia
1	5	30	0.0062123	0.056504	160.970
2	40	30	0.0063421	0.425596	157.677
3	100	30	0.0063624	1.583522	156.490

Cuadro 8: Resultados de implementación de AG en mapa 2D primer caso

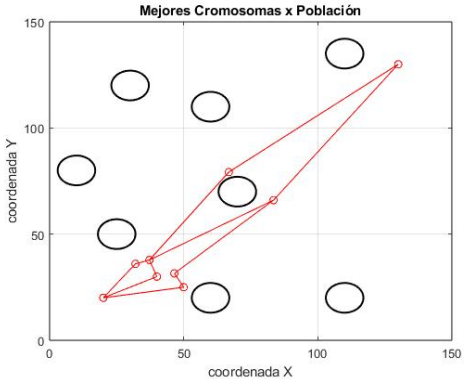


(a) Mejores cromosomas de cada generación

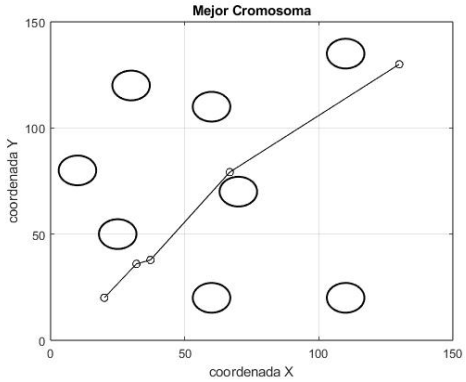


(b) Mejor cromosoma seleccionado

Figura 40: Camino generado con 5 iteraciones por AG, primer caso

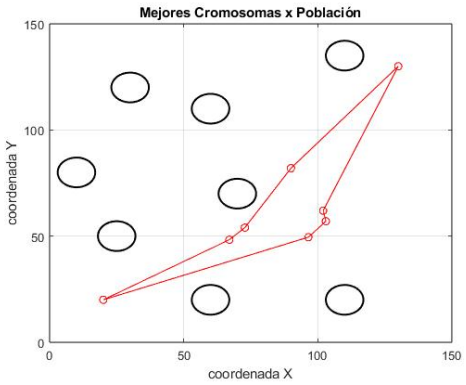


(a) Mejores cromosomas de cada generación

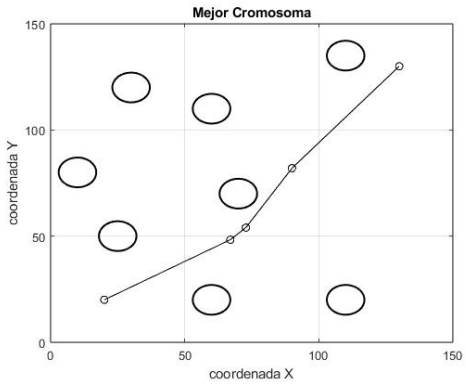


(b) Mejor cromosoma seleccionado

Figura 41: Camino generado con 40 iteraciones por AG, primer caso



(a) Mejores cromosomas de cada generación



(b) Mejor cromosoma seleccionado

Figura 42: Camino generado con 100 iteraciones por AG, primer caso

Los resultados de la implementación demostraron que el algoritmo desarrollado funciona satisfaciendo el criterio propuesto. Al optimizar el problema con el AG se generó una buena trayectoria libre de obstáculos. En este punto no se buscó encontrar la ruta más corta, debido a que el problema esta enfocado optimizar trayectorias libres de colisión. Por lo que se logró inferir que el algoritmo fue capaz de generar rutas que satisfacen el criterio de evitar colisiones, validando su eficacia en el escenario evaluado. No obstante, cabe destacar que el cumplimiento de este objetivo depende significativamente de la correcta configuración de parámetros como la probabilidad de cruce y mutación. Otro factor influyente adicional es el número de individuos, lo cual hace que existan una amplia variedad de soluciones.

Otro dato significativo es el costo computacional que demostró cada prueba. El costo computacional se mide por el tiempo que tarda en procesar y resolver el problema de optimización. Entre las tres simulaciones no se obtienen tiempos mayores a 2 segundos, demostrando un resultado rápido en convergencia. Esto es un dato relevante para la experimentación en este problema de optimización.

Como se menciona al principio de la sección, el propósito principal es evaluar la capacidad del algoritmo para producir trayectorias eficientes, en cuanto a cero colisiones. En este sentido se logró evitar los obstáculos definidos en un pequeño mapa, lo que sugiere una posible evaluación con ajuste de parámetros más robustos. Aumentando su capacidad de exploración del espacio de búsqueda y soluciones. Esto subraya la necesidad de implementar escenarios más complejos que terminen de validar este algoritmo, es decir, un mapa de exploración más grande.

8.2.2. Implementación de AG para planificación de trayectorias, escenario 2

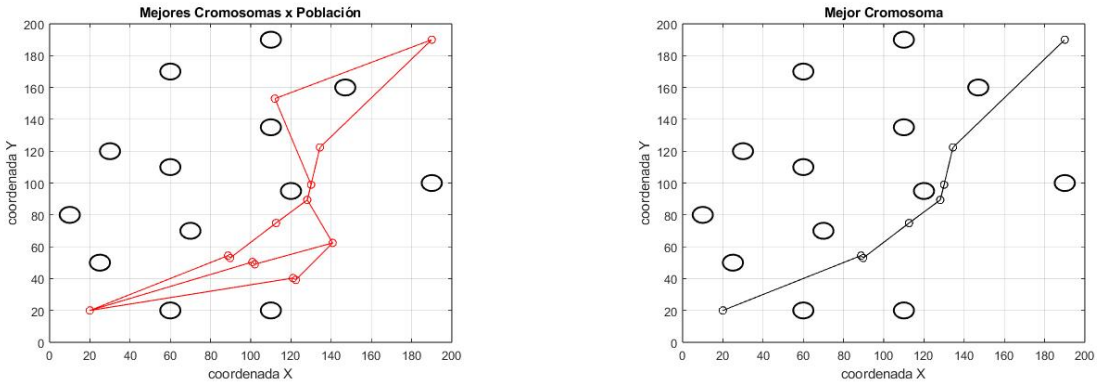
Una vez validado el algoritmo genético en el escenario más básico, se procedió a hacer implementaciones más robusta. En este caso se planteó generar un terreno de dimensiones de búsqueda mayores que sea más desafiante para el algoritmo. Por lo tanto, se hicieron las modificaciones pertinentes que se resumen en los siguientes parámetros:

- Dimensiones del mapa: 200×200
- Cantidad de obstáculos: 13
- Centro de los obstáculos: (85,100), (25, 50), (30, 120), (60,110), (60,20), (70,70), (110,20), (110, 135), (110, 90), (90, 80), (190,100), (147,160) y (140, 80)
- Población: 60 individuos
- Condición de detención aplica cuando se cumplan todas las iteraciones
- Punto de inicio y punto de meta: (20,20) y (195,195)
- Cantidad de puntos intermedios: 8
- Radio de los obstáculos: 5

Estos parámetros fueron elegidos gracias a los criterios que destacaron en el capítulo anterior. Al tener más individuos aumenta la diversidad de búsqueda en regiones más amplias. Una observación importante es que, mientras más puntos intermedios se seleccionen, mayor será el tiempo de ejecución. Se utilizan iteraciones arbitrarias de manera creciente como en el caso anterior, con el objetivo de experimentar cómo influye en el tiempo y en la aptitud. Los resultados de esta prueba se muestran en el Cuadro 9 y las Figuras 43, 44 y 45.

No.	Iteraciones	Población	Aptitud	Tiempo (s)	Distancia
1	5	60	0.0039566	0.236771	252.741
2	40	60	0.0039781	1.503192	251.378
3	100	60	0.0040147	3.573368	249.087

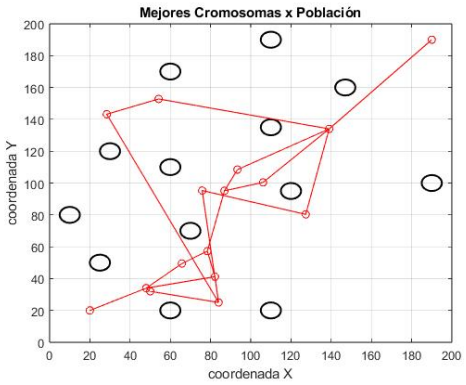
Cuadro 9: Resultados de implementación de AG en mapa 2D segundo caso



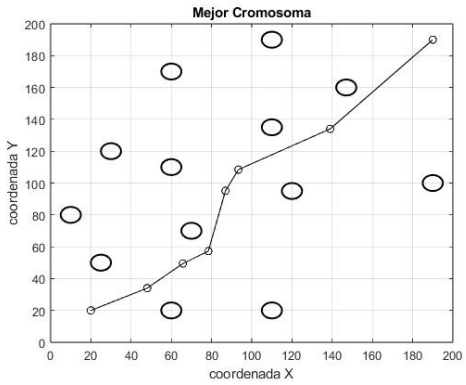
(a) Mejores cromosomas de cada generación

(b) Mejor cromosoma seleccionado

Figura 43: Camino generado con 5 iteraciones por AG, segundo caso

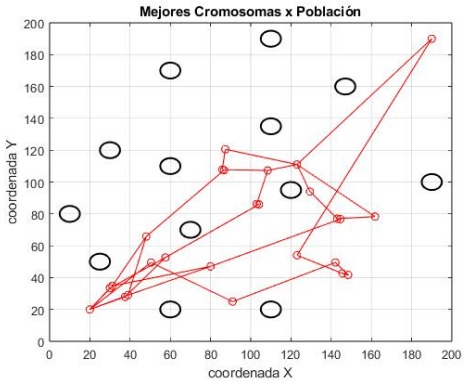


(a) Mejores cromosomas de cada generación

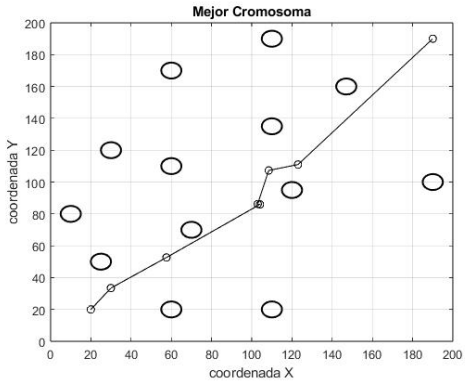


(b) Mejor cromosoma seleccionado

Figura 44: Camino generado con 40 iteraciones por AG, segundo caso



(a) Mejores cromosomas de cada generación



(b) Mejor cromosoma seleccionado

Figura 45: Camino generado con 100 iteraciones por AG, segundo caso

Los resultados evidencian que el algoritmo satisface el criterio deseado. En este segundo escenario, con un espacio de búsqueda más amplio y un mayor número de obstáculos, los resultados muestran que el algoritmo genético propuesto satisface el objetivo específico de planificar trayectorias evitando colisiones. El aumento en la cantidad de individuos y puntos intermedios permite una mayor exploración del espacio, lo que lleva a soluciones más robustas en comparación con el primer escenario. Sin embargo, como se mencionó al inicio de la sección, los tiempos de ejecución aumentan considerablemente, lo que demuestra la relación directa entre la complejidad del mapa y el rendimiento del algoritmo.

Ahora que se demostró su funcionalidad en escenarios más complejos, también es muy importante conocer las limitantes del algoritmo. Hasta ahora se determinó que funciona con un número de individuos y una población considerable. También se conoce que las iteraciones influyen en el tiempo, y que requiere de puntos intermedios para llegar a una trayectoria buena. El siguiente escenario expone lo que sucedió cuando se modificaron estos parámetros de manera obstructora para experimentar las trayectorias generadas por el algoritmo.

8.2.3. Validación de AG para planificación de trayectorias, escenario 3

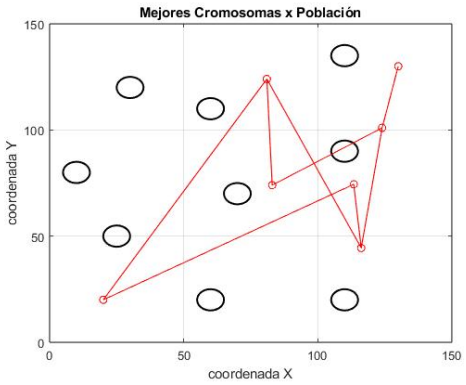
En los primeros escenarios, se evaluó el funcionamiento en condiciones óptimas que cumplieran con los objetivos de resolver el problema de búsqueda. Se obtuvo una pequeña noción hasta ahora de las limitantes. Por lo tanto, este escenario se realizó con el fin de identificar las limitantes del algoritmo y los casos en los que puede dificultarse la implementación. Si se reduce el número de individuos en la población, disminuye la probabilidad de encontrar diversidad, lo cual dificultaría encontrar una buena solución. El parámetro que se varió es la población a menor cantidad de individuos para identificar cómo se comporta esta limitante. Por lo tanto, los parámetros en este caso se muestran de la siguiente forma:

- Dimensiones del mapa: 150×150
- Cantidad de obstáculos: 9
- Centro de los obstáculos: (85,100), (25, 50), (30, 120), (60,110), (60,20), (70,70), (110,20), (110, 135), (110, 90)
- Población: variable
- Condición de detención aplica cuando se cumplan todas las iteraciones
- Punto de inicio y punto de meta: (20,20) y (140,140)
- Cantidad de puntos intermedios: 5
- Radio de los obstáculos: 7

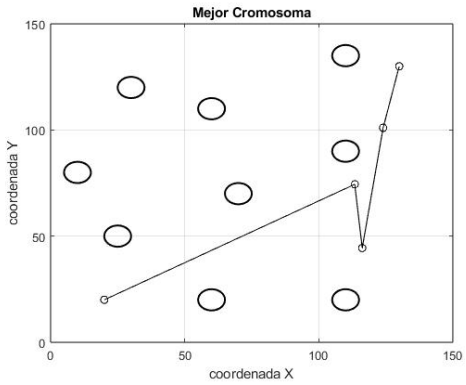
Los resultados de realizar esta implementación, manteniendo las iteraciones, se muestran en el Cuadro 10 y las Figuras 46, 47 y 48. En este caso la cantidad de individuos se redujo considerablemente a comparación de los casos anteriores. Además, se utilizó una cantidad de iteraciones considerable, que, con base en los resultados anteriores, mantendría un tiempo computacional menor que 1 segundo. En este caso no se esta buscando evaluar el tiempo sino que las limitaciones, por lo que se aplicará el mismo número de iteraciones para las 3 corridas.

No.	Iteraciones	Población	Aptitud	Tiempo (s)	Distancia
1	30	2	0.0044451	0.005479	224.967
2	30	4	0.1111080	0.0044648	223.972
3	30	6	0.2332920	0.0045387	214.901

Cuadro 10: Resultados de implementación de AG en mapa 2D tercer caso

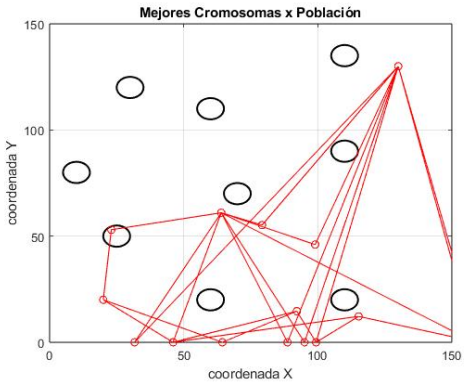


(a) Mejores cromosomas de cada generación

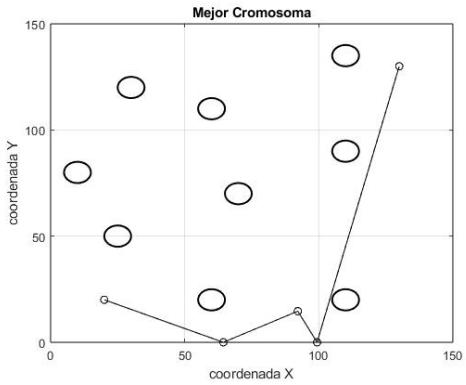


(b) Mejor cromosoma seleccionado

Figura 46: Camino generado con 30 iteraciones por AG, tercer caso

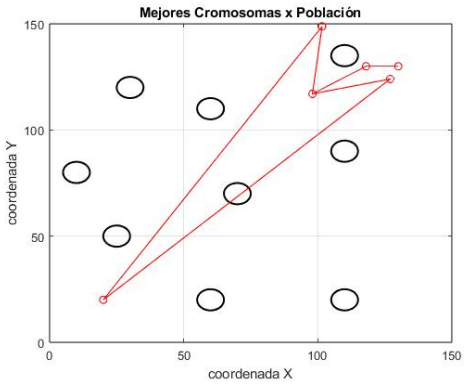


(a) Mejores cromosomas de cada generación

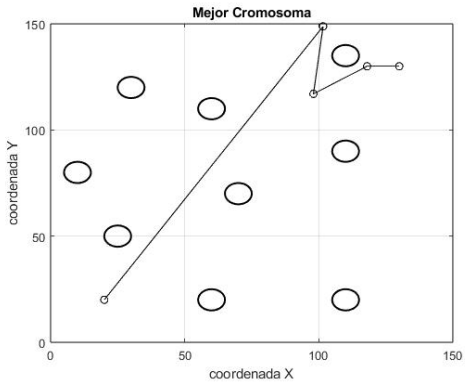


(b) Mejor cromosoma seleccionado

Figura 47: Camino generado con 30 iteraciones por AG, tercer caso



(a) Mejores cromosomas de cada generación



(b) Mejor cromosoma seleccionado

Figura 48: Camino generado con 30 iteraciones por AG, tercer caso

Analizando los resultados de la implementación, se demostraron los detalles de fallo para el algoritmo. En la Figura 46, se muestra que entre los mejores cromosomas, existe

1 que colisiona con los obstáculos, lo cual es riesgoso para un implementación formal. Sin embargo, gracias a los operadores de reproducción, se puede obtener una mejor solución a pesar esto. La experimentación en este caso demuestra una limitante en el algoritmo, lo que es un resultado muy bueno para tener este criterio presente para la funcionalidad del AG. En este escenario, mediante la implementación del algoritmo genético, fue posible evaluar otro escenario de planificación de trayectorias.

La limitante en este algoritmo se da por el factor mencionado al inicio de este escenario. El reducir los individuos en la población se reduce la diversidad y por ende la probabilidad de ampliar la búsqueda. En este caso, simplemente se tuvo una baja concentración de muestras que no podían compararse en grandes cantidades con base en su aptitud. Por lo que, el resultado de esto es trabajar el problema de optimización con poca información que reduce las probabilidades de éxito.

Este caso, simplemente se destaca para saber las formar optimas de manejar el AG. Es importante conocer las formas adecuadas para implementarlo, el conocer las limitantes previene las malas prácticas para esta aplicación. Otra limitante a plantear, característica de este tipo de implementación, es el reducir el numero de puntos intermedios. Esta limitante fue explorada en el siguiente escenario.

8.2.4. Validación de AG para planificación de trayectorias, escenario 4

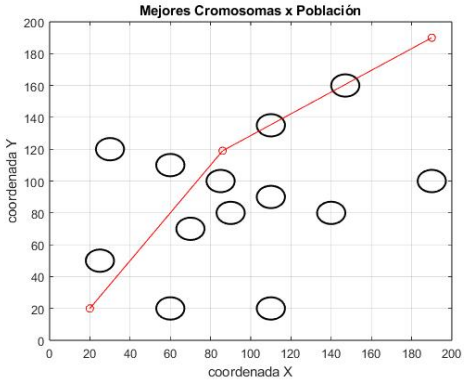
El objetivo de este escenario es demostrar la segunda limitante o mas bien mala práctica para la ejecución del algoritmo. En este caso se planteó un terreno de exploración grande con varios obstáculos, donde el cromosoma debería tener la suficiente longitud L (puntos intermedios) para satisfacer el criterio de búsqueda. Si reducimos los puntos en un mapa grande, puede que no tengamos el camino más apto para llegar sin interceptar con ellos. Por lo tanto, en este caso se amplió las dimensiones del terreno y se aumentó el número de obstáculos, pero a la vez reduciendo los puntos intermedios. Por lo que los parámetros utilizados, se describen de la siguiente manera:

- Dimensiones del mapa: 200×200
- Cantidad de obstáculos: 13
- Centro de los obstáculos: (85,100), (25, 50), (30, 120), (60,110), (60,20), (70,70), (110,20), (110, 135), (110, 90), (90, 80), (190,100), (147,160) y (140, 80)
- Población: variable con pocos individuos
- Condición de detención aplica cuando se cumplan todas las iteraciones
- Punto de inicio y punto de meta: (20,20) y (195,195)
- Cantidad de puntos intermedios: 3
- Radio de los obstáculos: 5

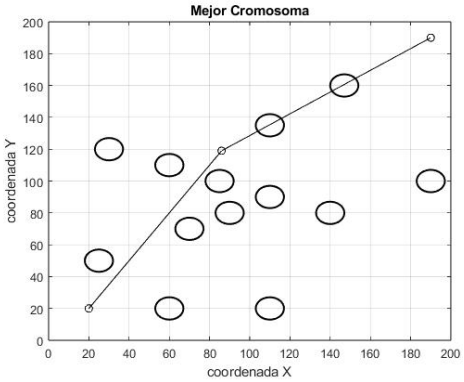
Los resultados de esta implementación se muestran en el Cuadro 11 y las Figuras 49, 50 y 51. Se mantuvieron criterios similares que en los escenarios anteriores, en cuanto a número de iteraciones y población.

No.	Iteraciones	Población	Aptitud	Tiempo (s)	Distancia
1	30	2	0.0005104	0.031791	244.908
2	30	4	0.0037037	0.073850	240.558
3	30	6	0.0032379	0.161587	240.416

Cuadro 11: Resultados de implementación de AG en mapa 2D cuarto caso

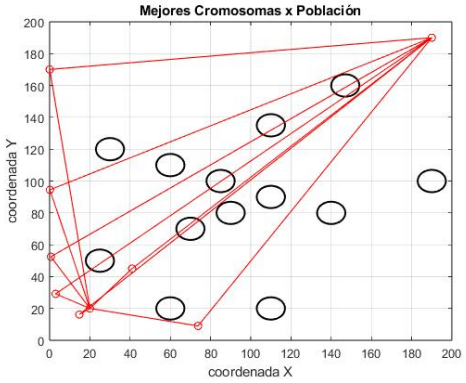


(a) Mejores cromosomas de cada generación

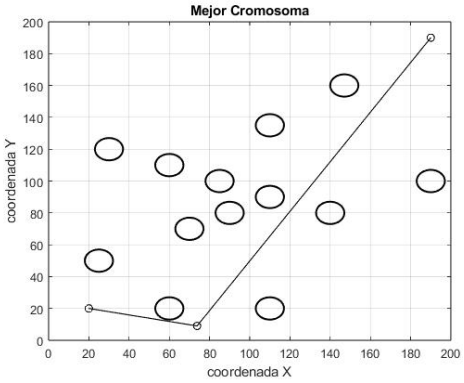


(b) Mejor cromosoma seleccionado

Figura 49: Camino generado con 30 iteraciones por AG, cuarto caso

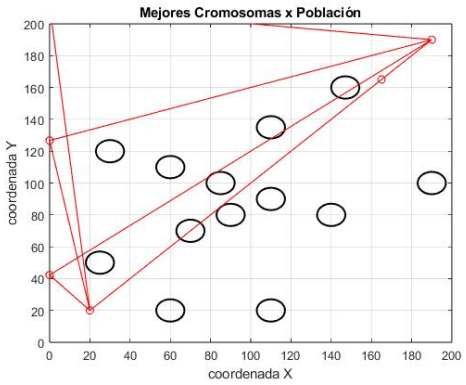


(a) Mejores cromosomas de cada generación

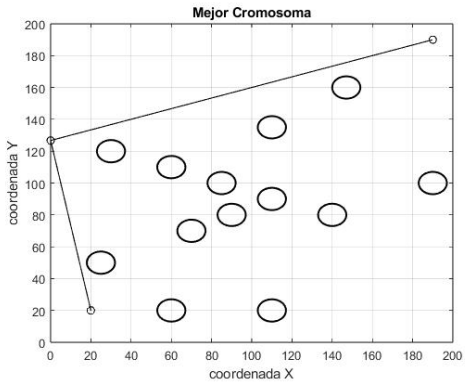


(b) Mejor cromosoma seleccionado

Figura 50: Camino generado con 30 iteraciones por AG, cuarto caso



(a) Mejores cromosomas de cada generación



(b) Mejor cromosoma seleccionado

Figura 51: Camino generado con 30 iteraciones por AG, cuarto caso

Los resultados de la implementación en este escenario satisfacen el criterio planteado al inicio. En este caso el objetivo de demostrar las limitantes del algoritmo, aplicándolo a un mapa complejo con una baja longitud de puntos en la cadena cromosómica, concuerda con los resultados experimentales.

En este cuarto escenario, el enfoque estuvo en evaluar la capacidad del algoritmo genético para generar trayectorias en un entorno con mayor densidad de obstáculos y más restricciones de movimiento. A diferencia de escenarios anteriores, este experimento buscó poner a prueba la flexibilidad del algoritmo al enfrentarse a condiciones más desafiantes. Los resultados muestran que, con un entorno de mayor complejidad y bajas referencias, el algoritmo no es capaz de generar rutas eficientes en sus primeras generaciones. Este comportamiento evidencia las malas formas de aplicar el algoritmo y por otro lado, refuerza la adaptabilidad del AG y su capacidad para manejar variaciones en el entorno.

A partir de este escenario, se tiene evaluado el buen funcionamiento del algoritmo con parámetros consistentes. También se conoció que las limitantes reflejaron las malas formas de variar los parámetros. Por lo que, en los siguientes escenarios y secciones anexadas, se mostraron casos de funcionamiento óptimo y de mayor complejidad, que mostraron resultados congruentes y óptimos.

8.2.5. Validación de AG para planificación de trayectorias, escenario 5

Luego de haber evaluado el algoritmo genético cumpliendo con los criterios deseados, se desarrolló un escenario más realista y con complejidad. En este se utiliza un mapa con paredes representadas por varios círculos esféricos, de manera que se simule una pared sólida. Esto es con el fin de explotar la capacidad de este algoritmo en variaciones de terreno aún más complejas. Por lo tanto, los parámetros utilizados se describen a continuación:

- Dimensiones del mapa: 200×200
- Cantidad de obstáculos: 13
- Obstáculos circulares formando un cuerpo sólido

- Población: 30 individuos
- Condición de detención aplica cuando se cumplan todas las iteraciones
- Punto de inicio y punto de meta: (20,20) y (195,195)
- Cantidad de puntos intermedios: 6
- Radio de los obstáculos: 5

En cuanto a parámetros de iteraciones, se mantiene el criterio de utilizar datos arbitrarios crecientes con fines de tener métricas en el costo computacional. En cuanto a el número de individuos, se mantiene constante con un numero considerable (50 individuos) para que exista diversidad pero no dispare el costo computacional. En la sección de Anexos 14.1, se encontrará el formato del *array* utilizado como código para realizar el mapa como se muestra en la Figura 52. Los resultados de la evaluación en este escenario se muestra a en el Cuadro 12. Las validaciones gráficas se muestran en las Figuras 53, 54 y 55.

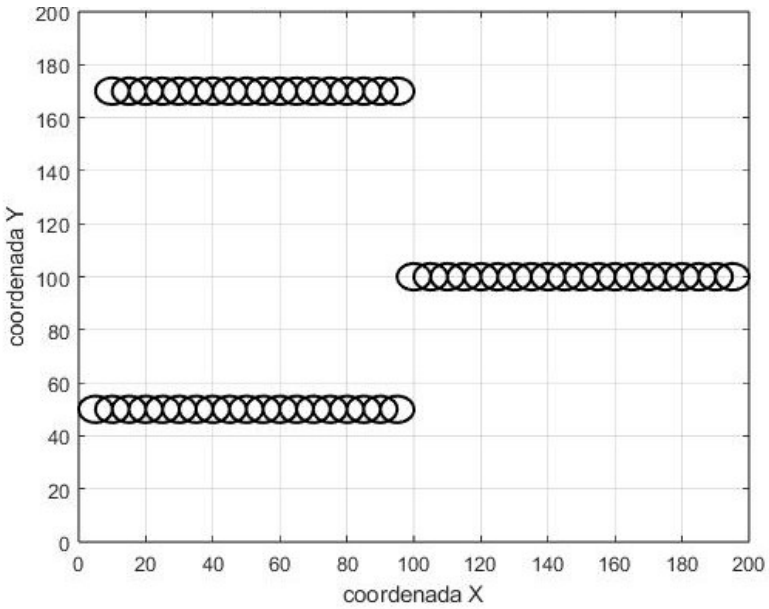
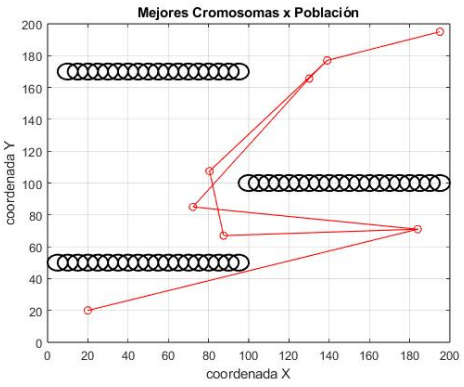


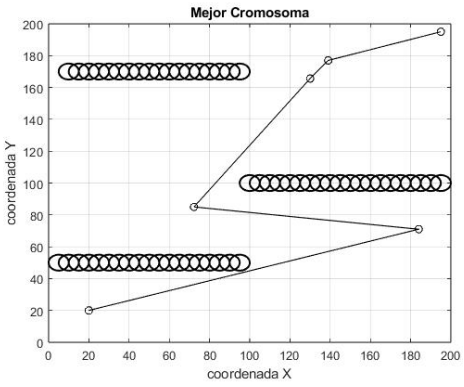
Figura 52: Mapa con estructura compleja para implementación de AG

No.	Iteraciones	Población	Aptitud	Tiempo (s)	Distancia
1	10	50	0.0021888	1.574	456.870
2	20	50	0.0027308	2.088	366.190
3	100	50	0.0029137	9.813	343.210

Cuadro 12: Parámetros de rendimiento de AG en planificación de trayectorias, quinto caso

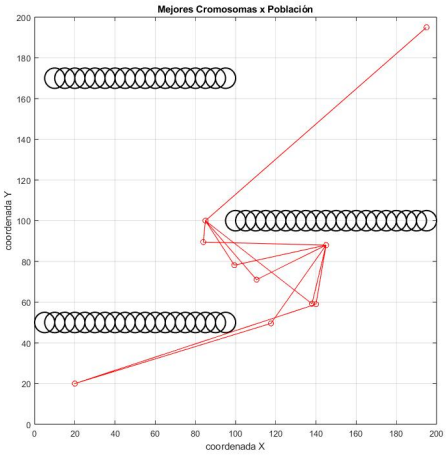


(a) Mejores cromosomas de cada generación

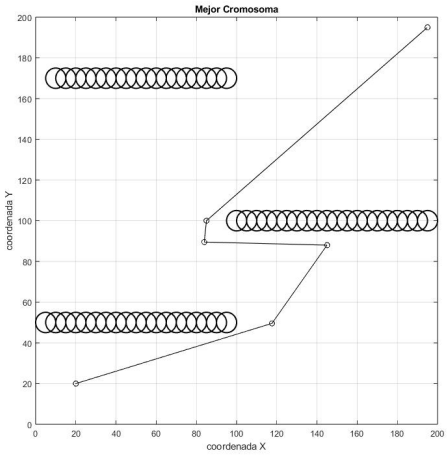


(b) Mejor cromosoma seleccionado

Figura 53: Camino generado con 10 iteraciones por AG, quinto caso

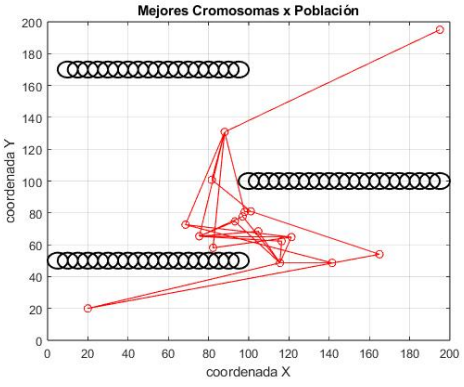


(a) Mejores cromosomas de cada generación

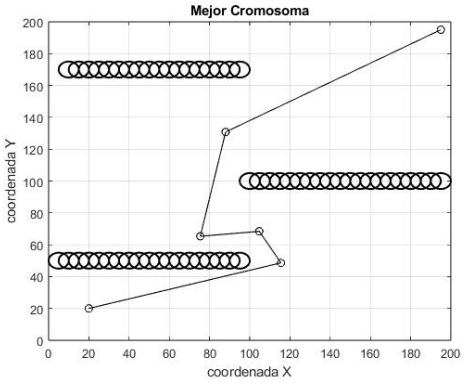


(b) Mejor cromosoma seleccionado

Figura 54: Camino generado con 20 iteraciones por AG, quinto caso



(a) Mejores cromosomas de cada generación



(b) Mejor cromosoma seleccionado

Figura 55: Camino generado con 100 iteraciones por AG, quinto caso

Esta implementación demostró, una vez más, que los algoritmos genéticos aplicados en la rama de robótica para planificación de trayectoria satisfacen los criterios para generar trayectorias libres de colisión. La robustez del escenario evaluado en esta sección demostró que el algoritmo es funcional sin importar la configuración de los obstáculos. Por lo tanto, con base en los 5 escenarios experimentados, se determinó que esta implementación del AG al problema de optimización descrito cumple con el objetivo establecido al principio.

Para este escenario, el entorno simulaba un mapa más realista, donde los obstáculos representaban una pared sólida. Este tipo de configuración permitió evaluar la efectividad del algoritmo en condiciones más cercanas a las que se encontrarían en aplicaciones reales. A pesar de la complejidad añadida, los resultados indicaron que el algoritmo fue capaz de encontrar soluciones que evitaron las colisiones y cumplió con los objetivos planteados en términos de planificación de trayectorias. Sin importar la complejidad, el AG sigue demostrando su funcionalidad en escenarios realistas, alineándose con el objetivo de validar su robustez en distintos entornos. También, el dato que destaca es su eficiencia computacional, teniendo tiempo muy adecuados para un algoritmo computacional.

En el Capítulo 10.1 se demostró la aplicación en simulaciones realistas de este algoritmo. Al igual que con esta implementación existen varias que fueron demostradas en siguientes experimentos. Los experimentos adicionales a esta implementación se encuentran en la sección de Anexos 14.1, donde se detallan más simulaciones que fueron óptimas y similares a las presentadas.

CAPÍTULO 9

Algoritmos genéticos en aplicaciones dentro de la Ingeniería Mecatrónica

Los algoritmos genéticos (AG) han demostrado ser una herramienta versátil y efectiva en diversos campos de la ingeniería, más allá de su uso tradicional en la robótica móvil. En este capítulo, se exploran dos aplicaciones específicas de los AG en el contexto de la Ingeniería Mecatrónica, que destacan su capacidad para resolver problemas complejos de optimización en dominios diversos. La primera aplicación se centra en la planificación de trayectorias en escenarios tridimensionales, una tarea clave para sistemas autónomos como drones o robots manipuladores, donde se requiere navegar en espacios obstaculizados y restringidos. La segunda aplicación aborda el procesamiento de imágenes mediante la segmentación basada en espacios de color, una técnica utilizada para aislar objetos específicos en imágenes complejas. Estas aplicaciones ilustran la flexibilidad y robustez de los AG, ampliando su potencial en la Ingeniería Mecatrónica.

9.1. Evaluación de algoritmos genéticos en planificación de movimiento para drones en tres dimensiones

En esta aplicación, se aborda el problema de planificación de trayectorias en entornos tridimensionales con obstáculos estáticos representados como planos sólidos. Estos obstáculos modelan escenarios reales, como paredes o barreras, que limitan el espacio de movimiento del robot. La principal prioridad de este enfoque es encontrar trayectorias viables que eviten colisiones, priorizando la seguridad sobre la óptima minimización de la distancia.

Los algoritmos genéticos (AG) se seleccionaron como herramienta para resolver este problema debido a su capacidad para explorar espacios de búsqueda complejos y su flexibilidad para integrar restricciones de colisión en la función de aptitud. Además, su enfoque evolutivo permite generar soluciones diversificadas que se adaptan a las características específicas del

entorno. Esta aplicación presenta una evaluación de los AG aplicados a la planificación de trayectorias en 3D, explorando su desempeño en múltiples escenarios, en MATLAB.

9.1.1. Metodología

Descripción del problema a optimizar

El problema abordado consistió en la planificación de trayectorias tridimensionales en un espacio definido por un mapa de dimensiones $n \times m \times k$, como se ejemplifica en la Figura 56. Dentro de este entorno, se incluyeron una cantidad i obstáculos estáticos representados por superficies en el plano zx, descritos por la ecuación $y = C$, con límites definidos en x y z ($L_{\text{infx}} < x < L_{\text{supx}}$ y $0 < z < k$). Estos planos actuaron como barreras sólidas que restringían el movimiento del robot, codificados como se muestra en la Figura 57.

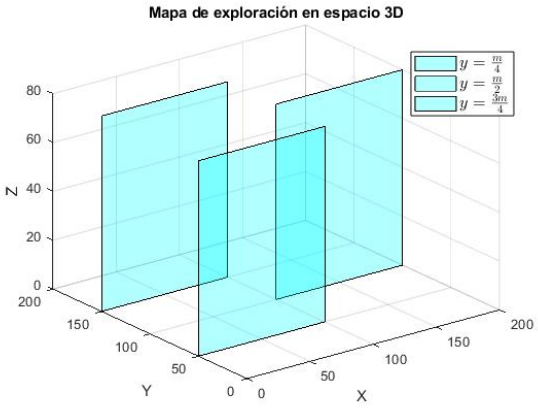


Figura 56: Visualización de mapa genérico 3D desarrollado en MATLAB

$$\mathbf{O_0} = \begin{bmatrix} x_0u & y_0 & z_0u \\ x_0l & y_0 & z_0l \end{bmatrix}, \mathbf{O_1} = \begin{bmatrix} x_1u & y_1 & z_1u \\ x_1l & y_1 & z_1l \end{bmatrix}, ...\mathbf{O_i} = \begin{bmatrix} x_iu & y_i & z_iu \\ x_il & y_i & z_il \end{bmatrix}$$

Figura 57: Codificación de obstáculos en AG para planificación de trayectorias en 3D

La trayectoria del robot, representada como un cromosoma en el algoritmo genético, se modeló como un vector compuesto por L_{crom} puntos en coordenadas tridimensionales (x, y, z), los cuales eran interpolables entre sí para formar segmentos de línea continua. Se asumió que el robot era una masa puntual, lo que permitió simplificar las verificaciones de colisión al considerar únicamente la intersección entre los segmentos de la trayectoria y los planos.

El objetivo principal fue determinar una trayectoria válida que, con base en la distancia recorrida entre el punto de inicio y el punto final, garantizara que ningún segmento de la trayectoria colisionara con los planos. En este contexto, la calidad de una trayectoria se evaluó mediante una función de aptitud que penalizaba las colisiones y premiaba las

trayectorias más cortas. Por lo tanto, el problema, se redujo a determinar qué trayectoria seguir de inicio a fin en un entorno tridimensional restringido. Para lograrlo, se utilizó el algoritmo descrito en el diagrama de flujo de la Figura 39.

Configuración del algoritmo genético

El algoritmo genético implementado para la planificación de trayectorias tridimensionales se estructuró utilizando una representación de cromosomas basada en vectores flotantes. Cada cromosoma representó una trayectoria como un conjunto de L_{crom} puntos interpolables en el espacio tridimensional (x, y, z), donde los puntos se conectaban mediante segmentos de línea continua, como se muestra en la Figura 58.

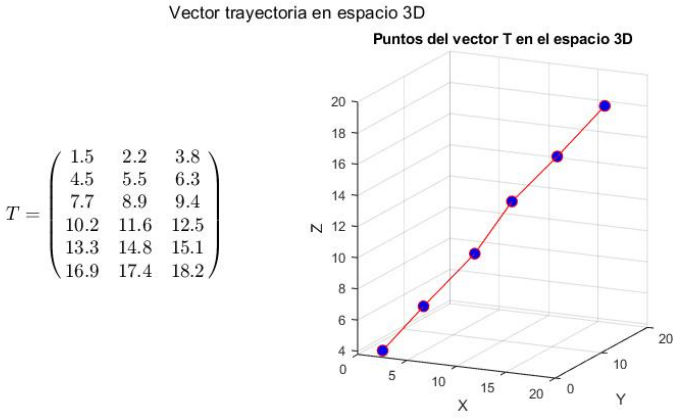


Figura 58: Representación y codificación del cromosoma en espacio 3D

La población inicial se generó de manera aleatoria, distribuyendo una cantidad de N cromosomas en el espacio tridimensional definido por el mapa $n \times m \times k$. Cada cromosoma contenía un conjunto de puntos aleatorios que se generaron respetando las dimensiones del espacio, garantizando así la diversidad inicial en la población. Los parámetros clave utilizados en el algoritmo genético fueron los siguientes:

- **Tamaño de la población**: N cromosomas
- **Número de generaciones**: G generaciones
- **Probabilidad de cruce**: P_c, utilizada para determinar qué cromosomas se sometían al operador de cruce, configurada totalmente aleatoria.
- **Probabilidad de mutación**: P_m, aplicada después del cruce para introducir variaciones en los cromosomas.

Operador de cruce

Los operadores genéticos se implementaron de la siguiente manera:

- **Cruce aritmético**: este operador combinó características de dos cromosomas padres sumando sus componentes de forma ponderada, según una probabilidad de cruce. Este operador se basa en el descrito en el Marco teórico 6.11.4 pero ahora adaptado a 3 dimensiones cartesianas. Este cruce aritmético, adaptado a este problema, se describe con la Ecuación de suma algebraica mostrada en (11).

$$\begin{aligned} C_1 &= \lambda_1 X + \lambda_2 Y + \lambda_1 Y \\ C_2 &= \lambda_2 X + \lambda_1 Y + \lambda_2 Z \end{aligned} \tag{11}$$

- **Mutación**: este operador introdujo variaciones en las trayectorias generadas mediante modificaciones en uno o más puntos de los cromosomas, dependiendo de la probabilidad de mutación. Utilizando el mismo criterio mencionado anteriormente pero sustituyendo el parámetro λ por la probabilidad de mutación μ. Este cambio se expresa en la Ecuación (12), donde C_1 y C_2 son los hijos luego de la mutación.

$$\begin{aligned} C_1 &= \mu X + \mu Y + \mu Y \\ C_2 &= \mu X + \mu Y + \mu Z \end{aligned} \tag{12}$$

Para cada cruce, ya sea por reproducción o mutación, se implementó un método de ruleta basado en 13, que establece la máscara de reproducción o mutación. Esta máscara se genera de forma aleatoria dependiendo del porcentaje de reproducción o mutación. El objetivo de esta máscara es definir qué gen cambia dentro del cromosoma y hereda nuevas características.

Función de aptitud

En esta implementación, la selección de cada individuo se basó en utilizar la función de aptitud descrita en la Ecuación (1). Esta describe la norma euclidiana como medida de la calidad del individuo. En otros términos la función de aptitud depende inversamente de la distancia recorrida y el número de colisiones N. Para este caso en particular, se adaptó el modelo de esta función a una norma euclidiana de 3 dimensiones, descrita en la ecuación (13).

$$f(i,j) = \frac{1}{\sqrt{(x_i + x_j)^2 + (y_i + y_j)^2 + (z_i + z_j)^2} \cdot N} \tag{13}$$

De esta manera, las trayectorias que no colisionaban con los obstáculos y tenían una menor distancia obtenían una mayor aptitud, lo que aumentaba su probabilidad de ser seleccionadas en generaciones futuras.

La calidad de colisiones se mide con N como el factor de penalización por cada colisión. Este se determina como el número de colisiones elevado a una potencia de 9, $N = a^9$, para garantizar que se verá afectada drásticamente en la aptitud. Para determinar si existe colisión entre la partícula y el plano se realizó un procedimiento más complejo que consiste en parametrizar utilizando álgebra vectorial.

El procedimiento consta de obtener dos puntos p_1 y p_2 consecutivos del cromosoma que formen una recta para luego verificar si interseca cada plano $y = c$. Realizando el

procedimiento explicado en la Sección 6.18, se desarrolló en una función computacional que se encargó de determinar el factor de colisión para cada individuo.

Operador de selección

La selección de los mejores individuos se basó en la comparación de la aptitud entre padres e hijos. Los cromosomas hijos reemplazaron a los padres únicamente si su aptitud era mayor. De lo contrario, estos son descartados y se conservan las características de los padres para la siguiente generación.

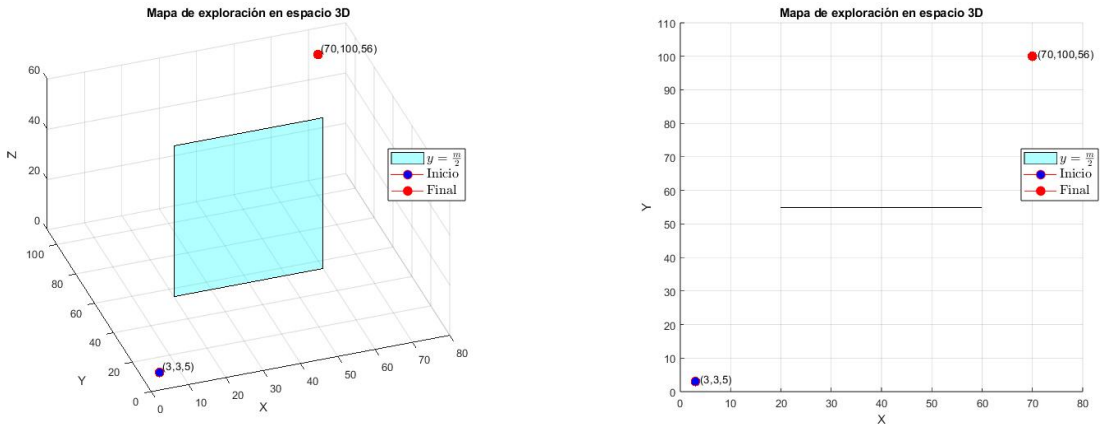
9.1.2. Evaluación de algoritmo genético en mapas 3D

Para evaluar el desempeño del algoritmo genético planteado anteriormente, en la planificación de trayectorias tridimensionales, se diseñaron seis escenarios experimentales con configuraciones variadas. Cada escenario presentó un conjunto de obstáculos estáticos representados como planos, y el objetivo principal fue generar trayectorias que evitaran colisiones. Las trayectorias fueron analizadas en términos de calidad, definida por la longitud recorrida y la penalización por colisiones que este obtenga.

Implementación de AG en planificación de trayectorias 3D, escenario 1

El primer escenario fue diseñado con únicamente un obstáculo para evaluar si el algoritmo genético es capaz de reconocer y evitar colisiones en un entorno sencillo. La simplicidad del escenario permite centrarse en la capacidad del algoritmo para encontrar trayectorias viables sin complejidades adicionales, lo que proporciona una base para validar su funcionamiento antes de aplicarlo a entornos más complejos.

El entorno tridimensional se definió como un mapa de dimensiones $80 \times 110 \times 60$, con un único plano estático ubicado en $y = 55$ y delimitado por los intervalos $20 < x < 60$ y $0 < z < 60$. Este plano dividió el espacio en dos regiones principales, restringiendo el movimiento del robot en el eje y. Los puntos inicial y final de la trayectoria se fijaron en $(3, 3, 5)$ y $(70, 100, 56)$, respectivamente, como se muestra en la Figura 59.



(a) Vista isométrica

(b) Vista en plano xy

Figura 59: Mapa de exploración en espacio 3D para escenario 1

Cada cromosoma consistió en una secuencia de 6 puntos interpolables en el espacio tridimensional (x, y, z), conectados mediante segmentos de línea continua. Esto proporcionó una representación flexible para modelar trayectorias dentro del entorno, permitiendo al algoritmo explorar diversas rutas posibles. Los parámetros del algoritmo genético se configuraron de la siguiente manera:

- **Probabilidad de mutación** (P_m): 10 %
- **Tamaño de la población**: 30 individuos
- **Longitud del cromosoma**: 6 puntos
- **Número de generaciones**: se incrementó de manera creciente en tres corridas independientes para analizar el efecto de este parámetro en el desempeño del algoritmo.

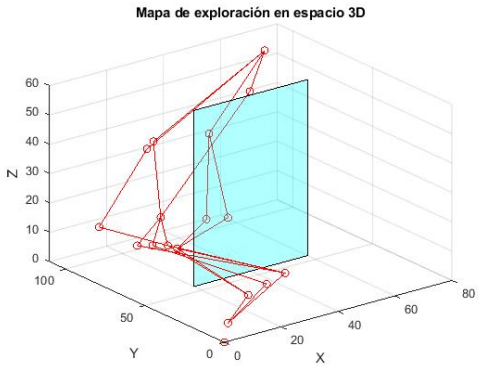
Las métricas evaluadas incluyeron la aptitud del mejor cromosoma, el tiempo computacional de cada corrida y la trayectoria total del mejor cromosoma. Se escogió una cantidad considerable de individuos en la población para garantizar la diversidad en la primera muestra poblacional. Por el momento, la longitud de cromosoma se mantuvo baja debido a simplicidad del terreno de exploración. Los resultados de esta implementación se muestran el Cuadro 13.

No.	Iteraciones	Población	Aptitud	Tiempo (s)	Distancia
1	16	30	0.0010117	0.153198	988.395
2	50	30	0.0014331	0.628481	697.778
3	100	30	0.0013133	0.866768	761.415

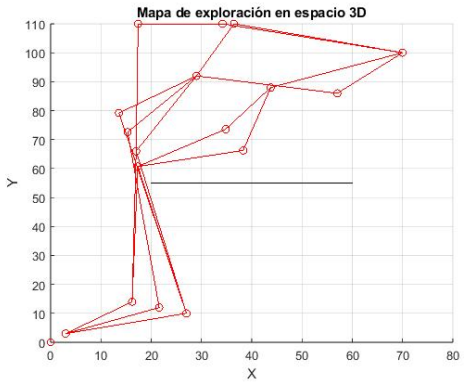
Cuadro 13: Resultados obtenidos para el escenario 1 en las tres corridas del AG en 3D

En las Figuras 60, 61 y 62, se muestran las trayectorias generadas por los mejores individuos de cada generación, en las tres corridas descritas anteriormente. Mientras que, la

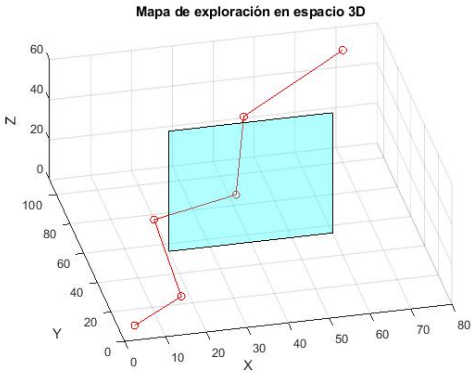
Figura 63, ilustra las trayectorias correspondientes al mejor cromosoma obtenido en cada corrida. Las mejores soluciones obtenidas para cada corrida se presentan en forma de vectores tridimensionales (x, y, z).



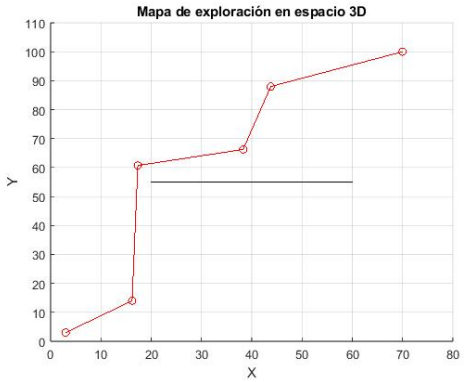
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos

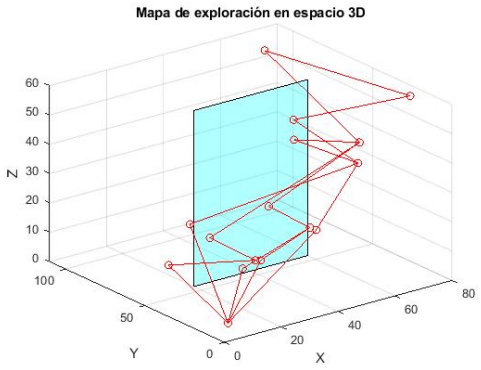


(c) Vista isométrica del mejor individuo seleccionado

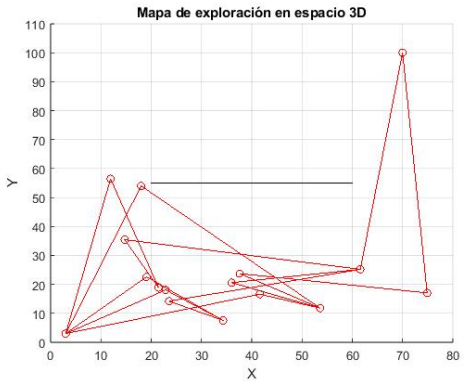


(d) Vista en plano xy del mejor individuo seleccionado

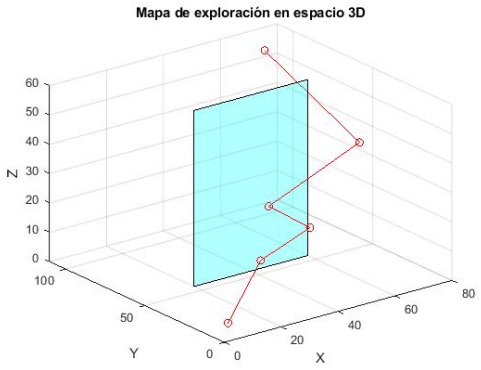
Figura 60: Mejores individuos generados con 16 iteraciones por AG en 3D, primer caso



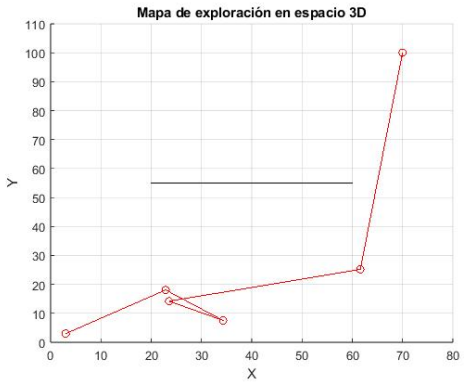
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos

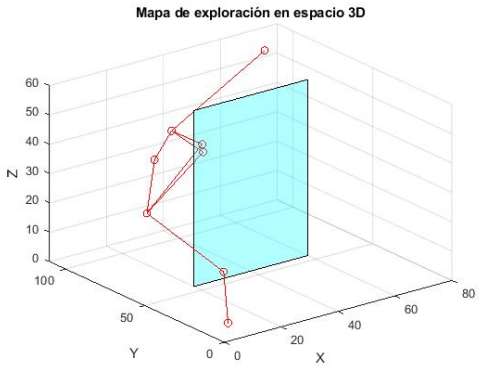


(c) Vista isométrica del mejor individuo seleccionado

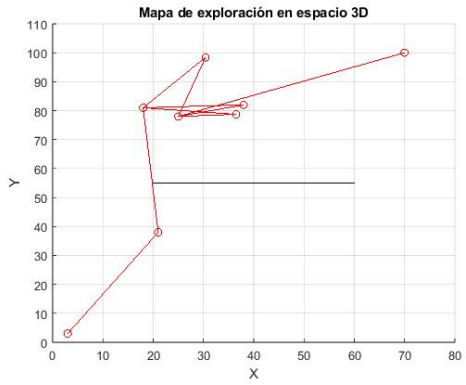


(d) Vista en plano xy del mejor individuo seleccionado

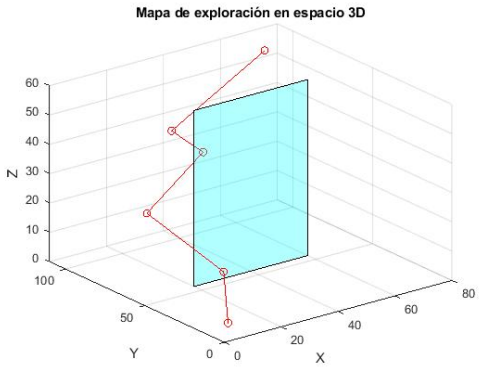
Figura 61: Mejores individuos generados con 50 iteraciones por AG en 3D, primer caso



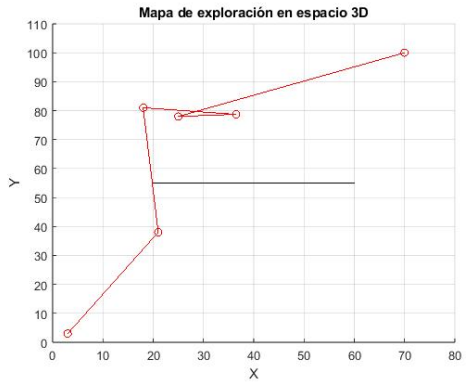
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado



(d) Vista en plano xy del mejor individuo seleccionado

Figura 62: Mejores individuos generados con 100 iteraciones por AG en 3D, primer caso

$$\mathbf{T} = \begin{bmatrix} 3.00 & 3.00 & 5.00 \\ 16.20 & 14.00 & 8.40 \\ 17.34 & 60.72 & 12.24 \\ 38.28 & 66.24 & 15.84 \\ 38.28 & 66.24 & 15.84 \\ 70.00 & 100.0 & 56.0 \end{bmatrix}$$

(a) 16 iteraciones

$$\mathbf{T} = \begin{bmatrix} 3.00 & 3.00 & 5.00 \\ 22.89 & 18.07 & 17.36 \\ 34.30 & 7.55 & 28.16 \\ 23.55 & 14.18 & 36.61 \\ 61.60 & 25.20 & 45.60 \\ 70.00 & 100.00 & 56.00 \end{bmatrix}$$

(b) 50 iteraciones

$$\mathbf{T} = \begin{bmatrix} 3.00 & 3.00 & 5.00 \\ 21.00 & 38.00 & 9.00 \\ 18.00 & 81.00 & 19.00 \\ 36.48 & 78.72 & 35.52 \\ 25.00 & 78.00 & 46.00 \\ 70.00 & 100.00 & 56.00 \end{bmatrix}$$

(c) 100 iteraciones

Figura 63: Representación del cromosoma como mejor solución en escenario 1

Este escenario permitió analizar cómo el algoritmo maneja la optimización de trayectorias mientras respeta las restricciones espaciales. En las tres configuraciones de iteraciones evaluadas (16, 50 y 100), las trayectorias generadas evitaron de manera efectiva las colisiones con el único obstáculo presente. Esto demuestra que el algoritmo implementado es capaz de adaptarse a las restricciones impuestas por el entorno.

Los resultados obtenidos destacan los siguientes comportamientos clave del algoritmo:

- La aptitud del mejor cromosoma mejoró significativamente al incrementar el número de iteraciones de 16 a 50, pasando de 0.0010117 a 0.0014331. Sin embargo, en la configuración de 100 iteraciones, la aptitud disminuyó ligeramente a 0.0013133, lo cual podría deberse a un fenómeno de sobre exploración o a la introducción de soluciones menos eficientes en generaciones posteriores.

- La distancia total de la trayectoria mostró un comportamiento similar. Disminuyó de 988.395 a 697.778 unidades entre 16 y 50 iteraciones, lo que indica que el algoritmo refinó las trayectorias significativamente en este rango. Sin embargo, en 100 iteraciones, la distancia aumentó ligeramente a 761.415, lo que puede reflejar un balance menos óptimo entre exploración y explotación. Esto podría estar relacionado con la introducción de cromosomas menos eficientes en generaciones posteriores debido a la mutación.

- El tiempo computacional incrementó de manera proporcional al número de iteraciones, pasando de 0.153 s en 16 iteraciones a 0.866 s en 100 iteraciones. Este aumento es esperable, y el tiempo total sigue siendo razonable para simulaciones de este tipo.

Estos resultados destacan que 50 iteraciones ofrecieron un equilibrio favorable entre calidad de la solución y tiempo computacional, mientras que 100 iteraciones no proporcionaron mejoras sustanciales en el desempeño.

Finalmente, la ausencia de colisiones en todas las corridas validó que el algoritmo cumple con las restricciones espaciales impuestas por el entorno. Este comportamiento inicial sentó una base sólida para escenarios más complejos, donde se podrá analizar la capacidad del algoritmo en entornos con mayor densidad de obstáculos, lo cual se demostró en los siguientes escenarios.

Implementación de AG en planificación de trayectorias 3D, escenario 2

El escenario 2 consistió en evaluar el desempeño del algoritmo genético en un entorno tridimensional con mayor complejidad que el escenario 1. El mapa se definió con dimensiones $100 \times 100 \times 50$, donde se incorporaron dos planos estáticos como obstáculos. Estos planos se ubicaron en $y = \frac{m}{2}$, con los límites $0 < x < \frac{2n}{5}$ y $\frac{3n}{5} < x < n$, mientras que $0 < z < k$. Los puntos inicial y final de la trayectoria se fijaron en $P_s = (20, 5, 15)$ y $P_g = (195, 385, 50)$, respectivamente. Este entorno presentó múltiples restricciones, dividiendo el espacio en varias regiones que requerían una planificación más sofisticada para generar trayectorias libres de colisiones.

En comparación con el Escenario 1, se realizaron ajustes en los parámetros del algoritmo genético para adaptarse a la complejidad del entorno:

- El número de iteraciones se incrementó progresivamente (50, 100 y 150) para analizar su efecto en un mapa más restrictivo, considerando los resultados del primer escenario, donde se observó que 50 iteraciones ofrecieron un balance favorable, pero 100 iteraciones no siempre mejoraron la solución.

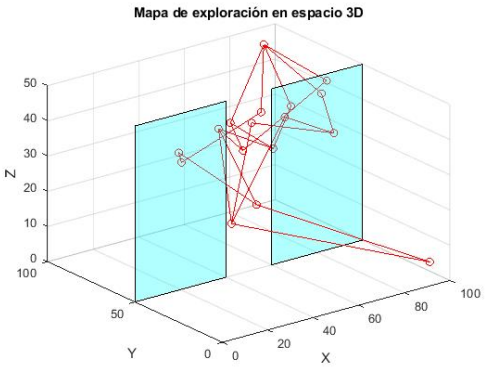
- El tamaño de la población se amplió a 50 individuos en algunas configuraciones para garantizar una mayor diversidad de soluciones, dado que el entorno con dos planos podría requerir una exploración más exhaustiva del espacio de búsqueda.
- La longitud del cromosoma se evaluó en dos configuraciones ($L_{\text{crom}} = 6$ y $L_{\text{crom}} = 10$) para verificar si trayectorias con más puntos ofrecían mejores adaptaciones al entorno.
- La probabilidad de mutación ($\%P_m$) se variaron entre 5 %, 10 % y 20 % para determinar cómo influía la introducción de diversidad en la calidad de las soluciones en este entorno más complejo.

Estos cambios de parámetros fueron justificados por la necesidad de analizar el impacto de la diversidad poblacional, la longitud de las trayectorias y la exploración del espacio de búsqueda en un entorno con restricciones más complejas. La incorporación de un segundo plano como obstáculo añadió nuevas limitaciones que obligaron al algoritmo a evaluar trayectorias alternativas y a refinar el balance entre exploración y explotación, los resultados para este caso se muestran en el Cuadro 14.

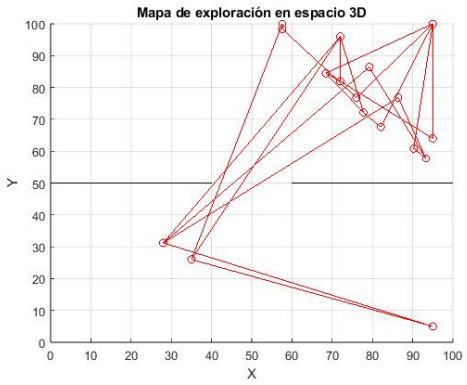
No.	**Iteraciones**	**Población**	$L_{\mathbf{crom}}$	P_m	**Tiempo (s)**	**Aptitud**	**Distancia**
1	50	30	6	10 %	0.394625	0.0014527	688.377
2	100	30	6	10 %	0.528957	0.0017142	583.367
3	100	50	10	10 %	0.871795	0.0008984	1113.148
4	150	50	10	5 %	1.224068	0.0005684	1759.275
5	150	50	10	20 %	1.230925	0.0005017	1993.308

Cuadro 14: Resultados obtenidos para el escenario 2 en las tres corridas del AG en 3D

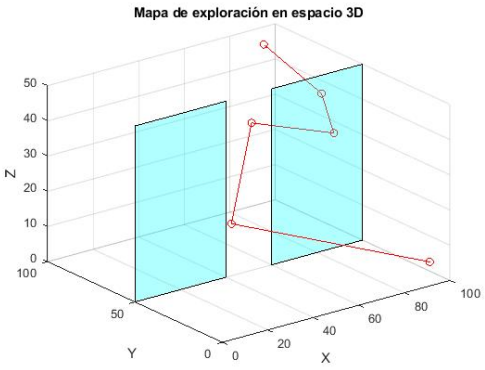
Las Figuras 64, 65, 66, 67 y 68 reflejan los tres resultados más relevantes de las cinco corridas del AG. También se muestra el cromosoma seleccionado como mejor solución. Los resultados gráficos y métricas obtenidas en las cinco corridas destacan varios aspectos clave del comportamiento del algoritmo.



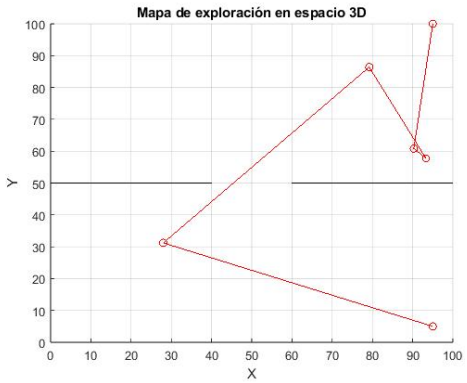
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

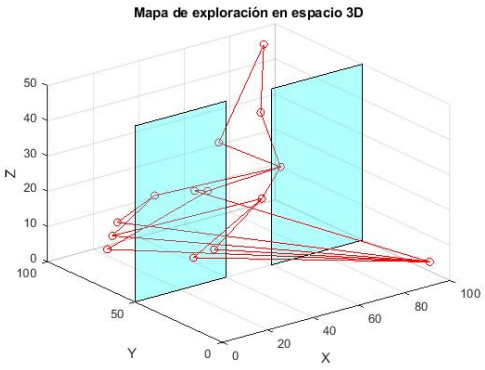


(d) Vista en plano xy del mejor individuo seleccionado

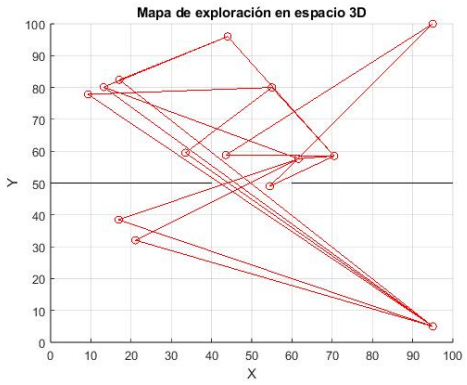
$$\mathbf{T} = \begin{bmatrix} 95.00 & 5.00 & 5.00 \\ 28.00 & 31.20 & 21.60 \\ 79.20 & 86.40 & 28.60 \\ 93.31 & 57.73 & 29.81 \\ 90.29 & 60.83 & 40.87 \\ 95.00 & 100.00 & 45.00 \end{bmatrix}$$

(e) Mejor cromosoma

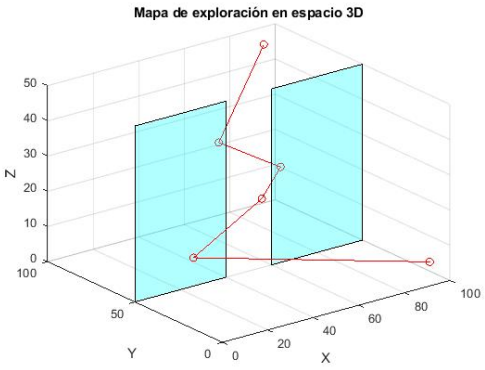
Figura 64: Mejores individuos generados en la primera corrida por AG en 3D, segundo caso



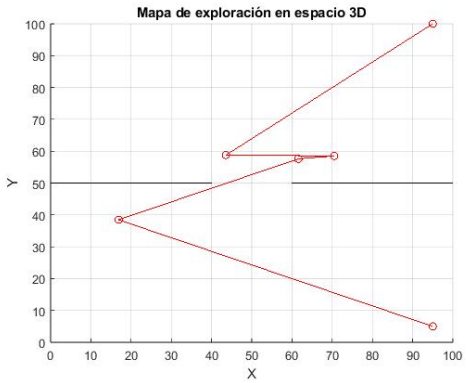
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

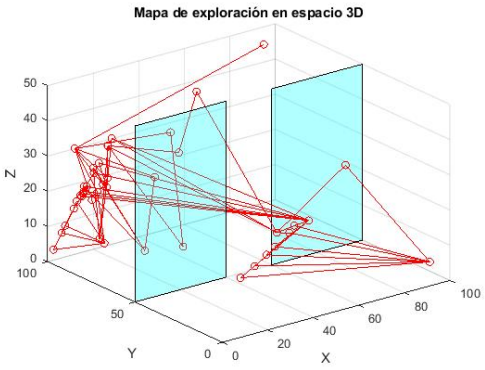


(d) Vista en plano xy del mejor individuo seleccionado

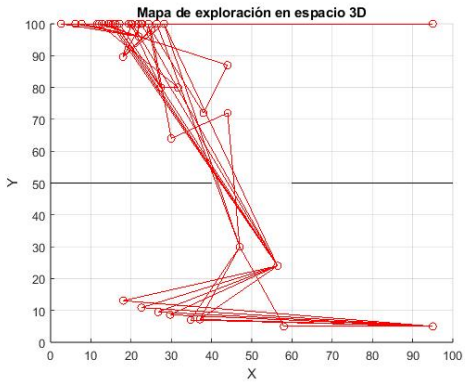
$$\mathbf{T} = \begin{bmatrix} 95.00 & 5.00 & 5.00 \\ 16.86 & 38.46 & 12.24 \\ 61.60 & 57.60 & 16.80 \\ 70.50 & 58.50 & 24.00 \\ 43.60 & 58.80 & 35.60 \\ 95.00 & 100.00 & 45.00 \end{bmatrix}$$

(e) Mejor cromosoma

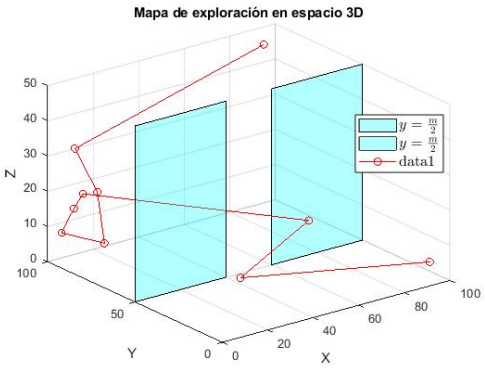
Figura 65: Mejores individuos generados en la segunda corrida por AG en 3D, segundo caso



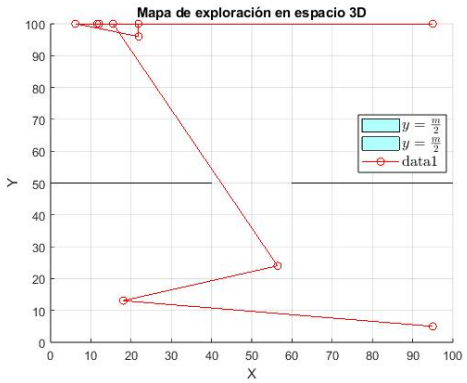
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

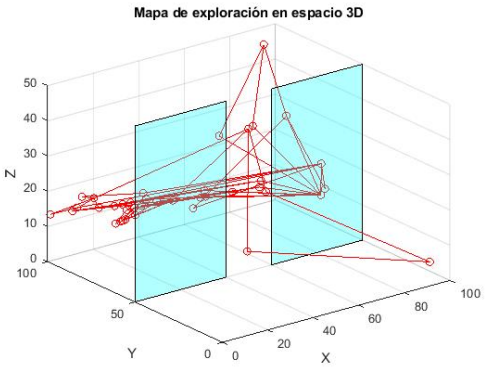


(d) Vista en plano xy del mejor individuo seleccionado

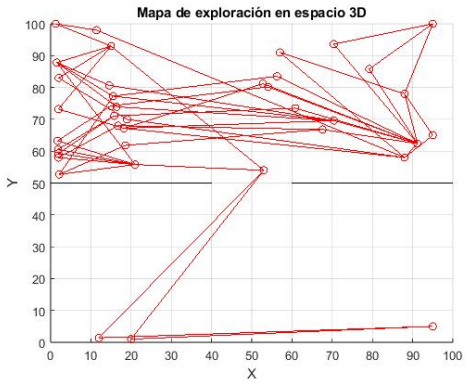
$$\mathbf{T} = \begin{bmatrix} 95.00 & 5.00 & 5.00 \\ 18.06 & 13.06 & 12.14 \\ 56.40 & 24.00 & 19.20 \\ 15.50 & 100.00 & 16.56 \\ 11.52 & 100.00 & 13.06 \\ 6.14 & 100.00 & 7.13 \\ 21.89 & 96.00 & 2.30 \\ 21.86 & 100.00 & 15.90 \\ 12.00 & 100.00 & 30.00 \\ 95.00 & 100.00 & 45.00 \end{bmatrix}$$

(e) Mejor cromosoma

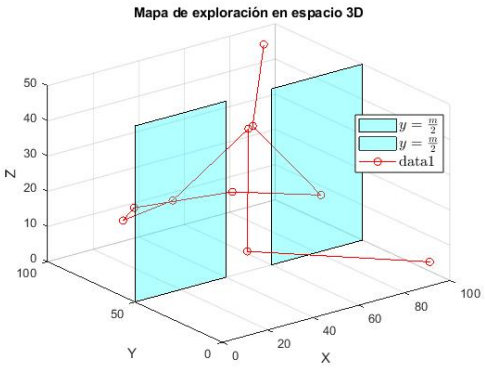
Figura 66: Mejores individuos generados en la tercera corrida por AG en 3D, segundo caso



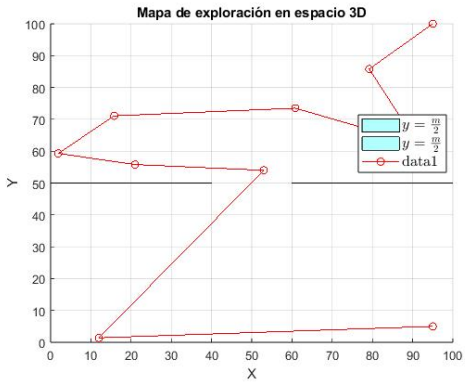
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

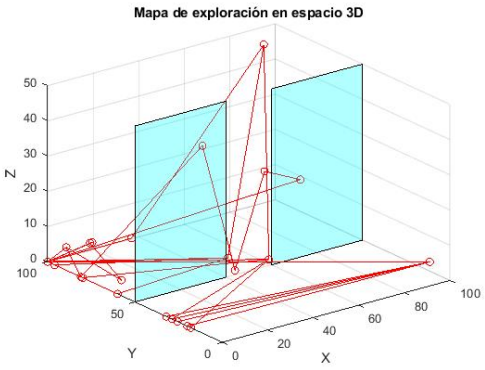


(d) Vista en plano xy del mejor individuo seleccionado

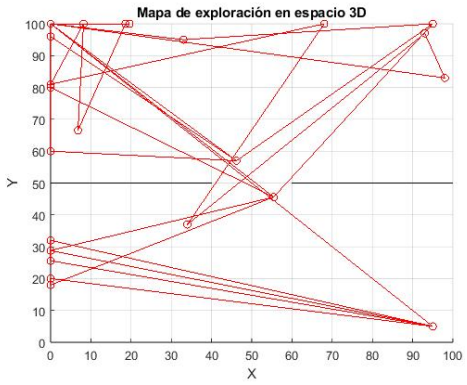
$$\mathbf{T} = \begin{bmatrix} 95.00 & 5.00 & 5.00 \\ 12.00 & 1.40 & 23.40 \\ 53.00 & 54.00 & 39.00 \\ 21.00 & 55.80 & 23.80 \\ 1.88 & 59.29 & 20.70 \\ 15.81 & 71.10 & 19.14 \\ 60.83 & 73.50 & 15.21 \\ 91.20 & 62.40 & 11.52 \\ 79.20 & 85.80 & 27.90 \\ 95.00 & 100.00 & 45.00 \end{bmatrix}$$

(e) Mejor cromosoma

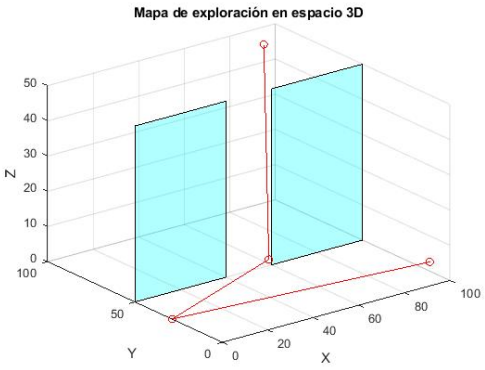
Figura 67: Mejores individuos generados en la cuarta corrida por AG en 3D, segundo caso



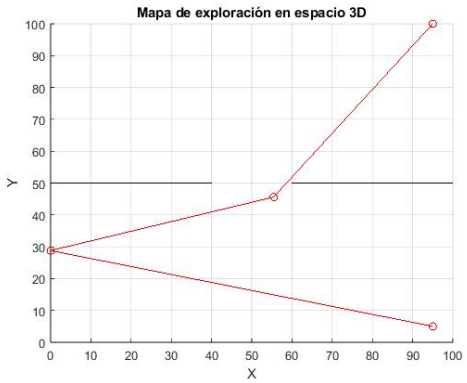
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado



(d) Vista en plano xy del mejor individuo seleccionado

$$\mathbf{A} = \begin{bmatrix} 95.00 & 5.00 & 5.00 \\ 0.00 & 28.80 & 0.00 \\ 0.00 & 28.80 & 0.00 \\ 0.00 & 28.80 & 0.00 \\ 0.00 & 28.80 & 0.00 \\ 0.00 & 28.80 & 0.00 \\ 0.00 & 28.80 & 0.00 \\ 0.00 & 28.80 & 0.00 \\ 55.44 & 45.60 & 3.36 \\ 95.00 & 100.00 & 45.00 \end{bmatrix}$$

(e) Mejor cromosoma

Figura 68: Mejores individuos generados en la quinta corrida por AG en 3D, segundo caso

Resultados gráficos

- En general, todas las corridas generaron trayectorias que evitaron colisiones con los obstáculos, validando la capacidad del algoritmo para respetar las restricciones espaciales.
- La corrida 1 produjo una trayectoria funcional, pero con numerosos quiebres, lo que

refleja limitaciones en la representación de cromosomas ($L_{\text{crom}} = 6$) frente a la complejidad de los obstáculos.

- La corrida 2 generó trayectorias más simplificadas y eficientes, mejorando significativamente la distancia total recorrida en comparación con la corrida 1.
- En las corridas 3 y 4, las trayectorias mostraron redundancias, con caminos complejos y repetitivos, indicando que, aunque el aumento en L_{crom} y población incrementó la capacidad de representación, la exploración del espacio de búsqueda fue insuficiente.
- La corrida 5 destacó como una de las mejores soluciones, logrando trayectorias más directas y simplificadas. Esto se atribuye a la mayor probabilidad de mutación ($P_m = 20\,\%$), que permitió diversificar las generaciones y evitar trayectorias redundantes.

Aunque las iteraciones se incrementaron hasta 150 en algunas corridas, los resultados gráficos mostraron una baja variación en los mejores cromosomas de las generaciones finales. Este comportamiento sugiere que el algoritmo pudo haber convergido prematuramente hacia soluciones similares debido al método de selección, limitando la diversidad en generaciones posteriores. Además, las trayectorias redundantes observadas en las corridas 3 y 4 reflejan un balance inadecuado entre exploración y explotación. A pesar del mayor tamaño de población y longitud del cromosoma, la baja probabilidad de mutación ($P_m = 5\,\%$) parece haber restringido la capacidad del algoritmo para escapar de óptimos locales. Por otro lado, la corrida 5 demostró cómo una mayor probabilidad de mutación ($P_m = 20\,\%$) favoreció la generación de trayectorias más directas y simplificadas, evidenciando la importancia de ajustar este parámetro en escenarios complejos.

En este escenario, se reafirmó que el algoritmo genético es capaz de encontrar soluciones viables sin colisiones, incluso en un entorno con múltiples restricciones. Sin embargo, el análisis de las trayectorias y métricas resalta la necesidad de un ajuste cuidadoso de los parámetros (por ejemplo, mutación y selección) para equilibrar diversidad y convergencia en escenarios más complejos. En el siguiente escenario, se tomaron en cuenta los aspectos observados en este caso para ajustar los parámetros del algoritmo genético. Particularmente, se promovió una mayor diversidad en las generaciones finales para mitigar la convergencia temprana, optimizar la relación entre exploración y explotación, y reducir las redundancias en las trayectorias generadas.

Implementación de AG en planificación de trayectorias 3D, escenario 3

El escenario 3 se desarrolló para analizar el desempeño del algoritmo genético en un entorno tridimensional con tres planos rectos distribuidos en diferentes posiciones del eje y. El mapa se definió con dimensiones $200 \times 200 \times 80$, incrementando la escala del espacio en comparación con los escenarios previos, para evaluar cómo el algoritmo maneja restricciones más amplias. Los planos se ubicaron en:

- $y = \frac{m}{4}$, con $x \in [0, \frac{n}{2}]$ y $z \in [0, k]$
- $y = \frac{m}{2}$, con $x \in [\frac{n}{2}, n]$ y $z \in [0, k]$
- $y = \frac{3m}{4}$, con $x \in [0, \frac{n}{2}]$ y $z \in [0, k]$

Esta configuración permitió distribuir las restricciones en diferentes regiones del mapa, lo que añadió complejidad al problema de planificación de trayectorias.

Los puntos inicial y final se establecieron en $P_s = (20, 5, 15)$ y $P_g = (195, 195, 70)$, respectivamente. Estos puntos se seleccionaron para garantizar que las trayectorias generadas atravesaran múltiples zonas restringidas, evaluando así la capacidad del algoritmo para adaptarse a diferentes regiones del mapa.

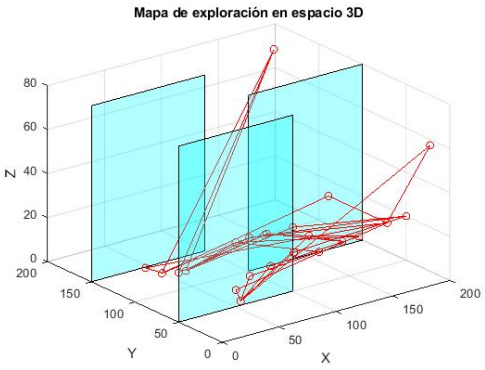
Se limitaron las iteraciones a 50 y 100, dado que en escenarios previos se observó que iteraciones más altas no necesariamente mejoraron la calidad de las trayectorias y redujeron la diversidad gráfica. Los tamaños de población se establecieron en 30 y 50 individuos para analizar el efecto de una mayor diversidad en la exploración del espacio de búsqueda. La longitud del cromosoma se evaluó en 6 y 8 puntos para determinar si un incremento en los puntos de la trayectoria mejora la representación de las soluciones en un entorno más complejo. Finalmente, la probabilidad de mutación (P_m) se varió entre 10 % y 20 % para fomentar la diversidad en generaciones posteriores, mitigando la convergencia prematura observada en el segundo escenario.

En este escenario, se buscó analizar cómo el algoritmo maneja restricciones distribuidas en el mapa, evaluar la calidad de las trayectorias generadas y estudiar la diversidad gráfica de las soluciones. Los resultados con las métricas evaluadas se muestran en el Cuadro 15.

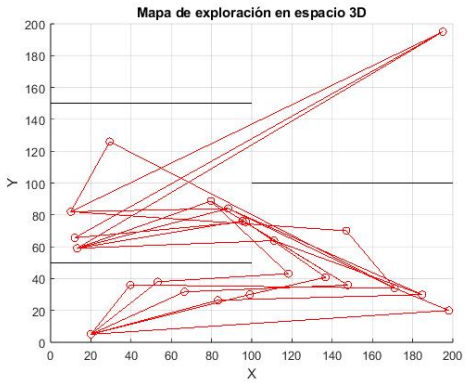
No.	**Iteraciones**	**Población**	$L_{\mathbf{crom}}$	P_m	**Tiempo (s)**	**Aptitud**	**Distancia**
1	50	30	6	10 %	0.350428	0.0003335	2998.256
2	50	50	6	20 %	0.546234	0.0003598	2779.609
3	100	30	8	10 %	0.834233	0.0001474	6782.991
4	100	50	8	20 %	0.852568	0.0006454	1549.372

Cuadro 15: Resultados obtenidos para el escenario 3 en las tres corridas del AG en 3D

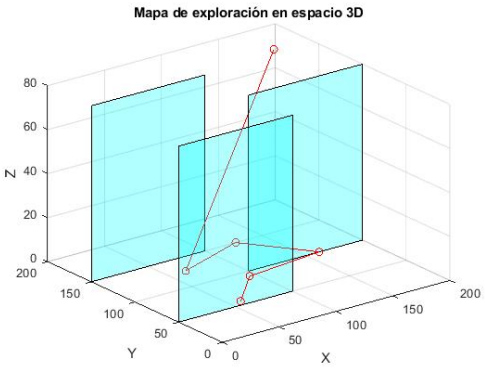
A continuación, en las Figuras 69, 70, 71 y 72 se presentan las observaciones clave obtenidas de las corridas realizadas.



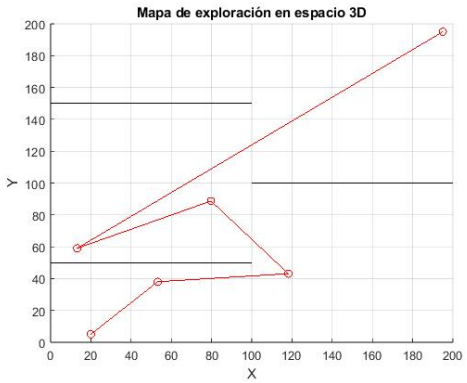
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

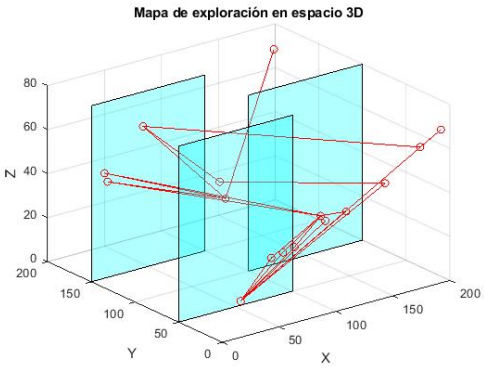


(d) Vista en plano xy del mejor individuo seleccionado

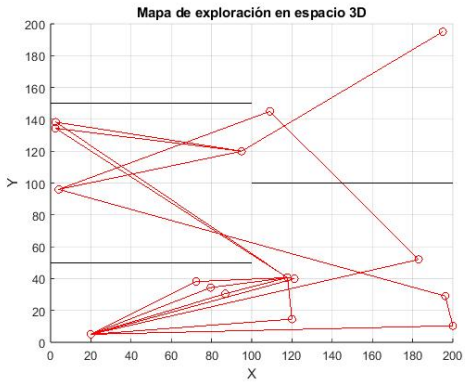
$$\mathbf{T} = \begin{bmatrix} 20.00 & 5.00 & 15.00 \\ 53.22 & 38.02 & 15.59 \\ 118.20 & 43.08 & 16.59 \\ 79.73 & 88.70 & 17.90 \\ 13.20 & 59.04 & 19.80 \\ 195.00 & 195.00 & 70.00 \end{bmatrix}$$

(e) Mejor cromosoma

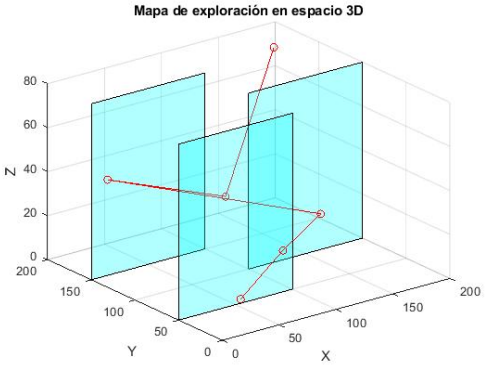
Figura 69: Mejores individuos generados en la primera corrida por AG en 3D, tercer caso



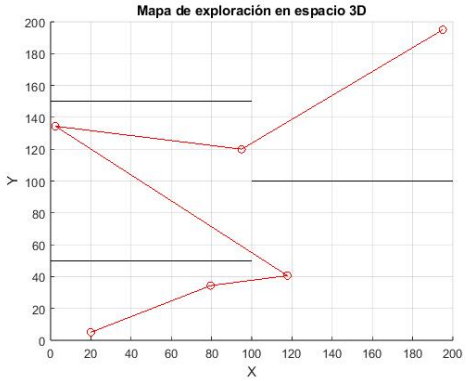
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

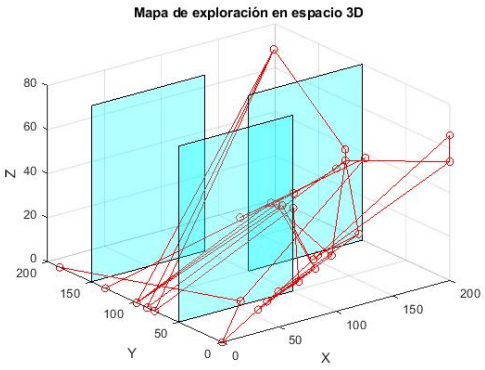


(d) Vista en plano xy del mejor individuo seleccionado

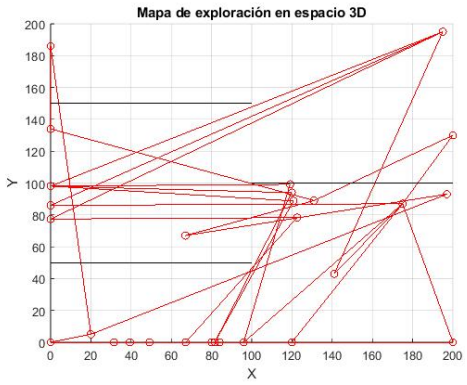
$$\mathbf{T} = \begin{bmatrix} 20.00 & 5.00 & 15.00 \\ 79.60 & 34.33 & 23.35 \\ 117.72 & 40.60 & 33.48 \\ 2.40 & 134.40 & 48.00 \\ 95.00 & 120.00 & 30.00 \\ 195.00 & 195.00 & 70.00 \end{bmatrix}$$

(e) Mejor cromosoma

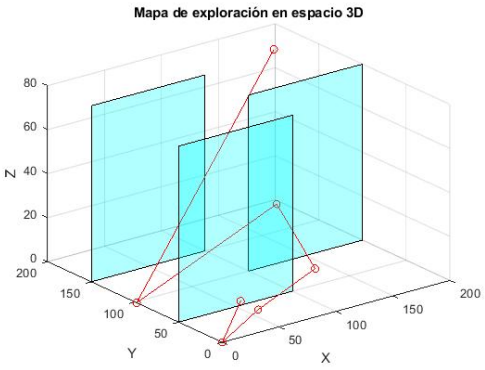
Figura 70: Mejores individuos generados en la segunda corrida por AG en 3D, tercer caso



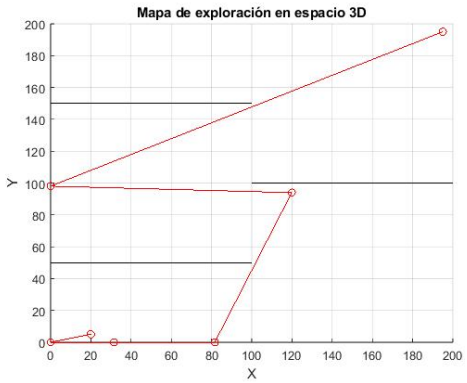
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado

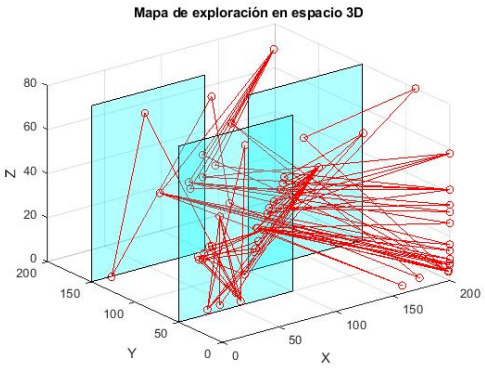


(d) Vista en plano xy del mejor individuo seleccionado

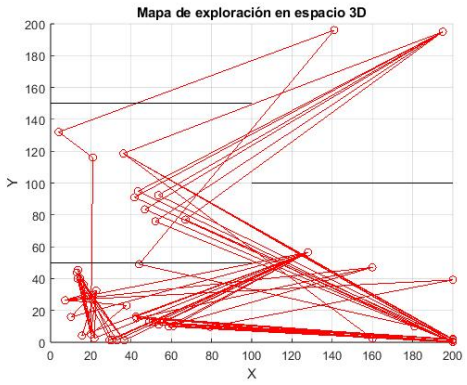
$$\mathbf{T} = \begin{bmatrix} 20.00 & 5.00 & 15.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 31.49 & 0.00 & 10.39 \\ 81.60 & 0.00 & 22.03 \\ 119.88 & 93.96 & 28.77 \\ 0.00 & 98.04 & 0.00 \\ 195.00 & 195.00 & 70.00 \end{bmatrix}$$

(e) Mejor cromosoma

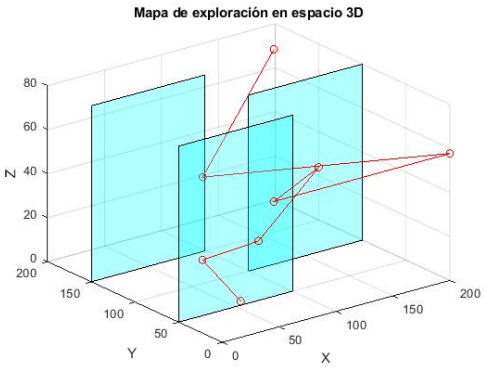
Figura 71: Mejores individuos generados en la tercera corrida por AG en 3D, tercer caso



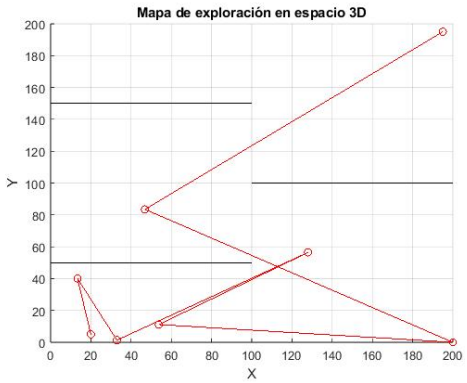
(a) Vista isométrica de mejores individuos



(b) Vista en plano xy de mejores individuos



(c) Vista isométrica del mejor individuo seleccionado



(d) Vista en plano xy del mejor individuo seleccionado

$$\mathbf{T} = \begin{bmatrix} 20.00 & 5.00 & 15.00 \\ 13.33 & 40.16 & 28.20 \\ 32.92 & 1.34 & 41.15 \\ 128.00 & 56.45 & 51.20 \\ 53.79 & 11.14 & 54.34 \\ 200.00 & 0.00 & 57.68 \\ 46.89 & 83.48 & 53.19 \\ 195.00 & 195.00 & 70.00 \end{bmatrix}$$

(e) Mejor cromosoma

Figura 72: Mejores individuos generados en la cuarta corrida por AG en 3D, tercer caso

- **Corrida 1:** la generación de soluciones mostró poca variación entre cromosomas dentro del conjunto evaluado, especialmente en las generaciones finales, donde los individuos de la población variaron mínimamente entre los índices 15 y 50. Esto refleja una convergencia temprana hacia soluciones similares, posiblemente debido al bajo tamaño de población y al valor limitado de la probabilidad de mutación ($P_m = 10\,\%$). Sin embargo, el algoritmo logró generar un cromosoma sin colisiones como solución final.
- **Corrida 2:** en esta corrida se observó una mayor dispersión en el espacio de búsqueda, lo que permitió explorar diferentes trayectorias posibles. A pesar de que algunos indi-

viduos en las generaciones iniciales colisionaron con los obstáculos, el valor aumentado de $P_m = 20\,\%$ promovió suficiente diversidad para obtener una solución sin colisiones al final. Esta corrida destacó por balancear mejor la exploración y explotación del espacio de búsqueda.

- **Corrida 3:** aunque se observó una variación considerable en las trayectorias generadas, muchos cromosomas colisionaron con los obstáculos debido al incremento en la longitud del cromosoma ($L_{\text{crom}} = 8$) y al bajo tamaño de población ($N = 30$). A pesar de estos desafíos, el algoritmo logró generar un individuo libre de colisiones como solución final. Sin embargo, el aumento en la distancia total recorrida (6782.991) sugiere una mayor complejidad en las trayectorias exploradas.
- **Corrida 4:** esta corrida mostró una alta variación y dispersión en el espacio de búsqueda, probando múltiples regiones dentro del mapa. Aunque algunos individuos iniciales colisionaron con los obstáculos, la configuración con mayor tamaño de población ($N = 50$) y $P_m = 20\,\%$ permitió encontrar rápidamente cromosomas que evitaban colisiones. La solución final fue la mejor en términos de aptitud y distancia recorrida (1549.372), destacando la efectividad de los parámetros ajustados.

En este escenario, se observó que el algoritmo genético fue capaz de generar soluciones viables y sin colisiones en todas las corridas, aunque con variaciones significativas en la calidad de las trayectorias generadas. El aumento en la probabilidad de mutación (P_m) demostró ser clave para mejorar la diversidad en las generaciones finales, particularmente en las corridas con $P_m = 20\,\%$. Además, el mayor tamaño de población en las corridas 2 y 4 contribuyó a explorar de manera más efectiva el espacio de búsqueda, reduciendo la convergencia temprana observada en corridas con tamaños más pequeños. Sin embargo, la longitud del cromosoma ($L_{\text{crom}} = 8$) en las corridas 3 y 4 también incrementó la complejidad de las soluciones, generando trayectorias más largas y menos eficientes en algunos casos. Estos resultados resaltan la importancia de ajustar cuidadosamente los parámetros del algoritmo para equilibrar la diversidad y la calidad de las soluciones generadas.

A partir de los resultados obtenidos en los escenarios probados, se realizó una prueba experimental más desafiante para el algoritmo, incrementando la complejidad del mapa y manteniendo los criterios adquiridos previamente.

Implementación de AG en planificación de trayectorias 3D, escenario 4

El cuarto escenario se diseñó para evaluar el desempeño del algoritmo genético en un entorno tridimensional más desafiante, con una mayor densidad de restricciones. El mapa se definió con dimensiones $200 \times 400 \times 100$, incluyendo cuatro planos rectos distribuidos en posiciones fijas del eje y:

- $y = \frac{m}{5}$, con $x \in [0, \frac{n}{2}]$ y $z \in [0, k]$
- $y = \frac{2m}{5}$, con $x \in [\frac{n}{2}, n]$ y $z \in [0, k]$
- $y = \frac{3m}{5}$, con $x \in [0, \frac{n}{2}]$ y $z \in [0, k]$

- $y = \frac{4m}{5}$, con $x \in [\frac{n}{2}, n]$ y $z \in [0, k]$

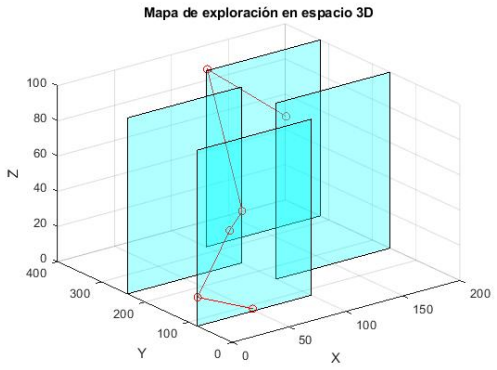
Los puntos inicial y final se establecieron en $P_s = (20, 20, 20)$ y $P_g = (195, 385, 50)$, respectivamente, de forma que las trayectorias generadas debieran atravesar zonas restringidas en diferentes regiones del mapa. Este diseño incrementó significativamente la complejidad del entorno, obligando al algoritmo a buscar trayectorias viables en un espacio con restricciones densas y distribuidas.

Para este escenario, los parámetros del algoritmo se limitaron a 50 iteraciones, manteniendo un tamaño de población constante de 50 individuos y una longitud de cromosoma ($L_{\text{crom}} = 6$) para representar trayectorias detalladas. Se evaluaron ambos casos con valores de probabilidad de mutación $P_m = 20\,\%$ para analizar su impacto en la diversidad y calidad de las soluciones generadas. Esta configuración buscó optimizar el rendimiento computacional mientras se abordaba un entorno más desafiante, con los parámetros mostrados en el Cuadro 16. Estas decisiones se basaron en los resultados de los escenarios previos, donde iteraciones limitadas y poblaciones moderadas ofrecieron un balance favorable entre diversidad gráfica y tiempo computacional. Además, ajustar la probabilidad de mutación permitió explorar el impacto de este parámetro en un entorno con mayor densidad de restricciones.

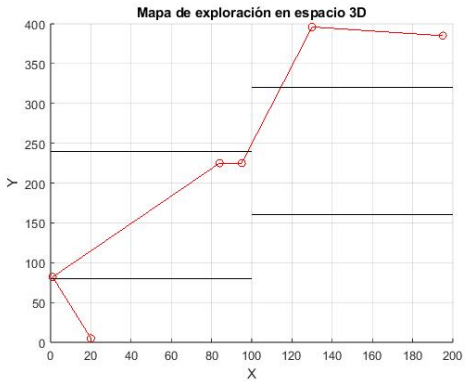
No.	**Población**	**Iteraciones**	**Tiempo (s)**	**Aptitud**	**Distancia**
1	50	50	0.400817	0.0000000	5021.121
2	150	50	1.174234	0.0003888	2572.316
3	500	50	26.972323	0.0004712	2122.234

Cuadro 16: Resultados obtenidos para el escenario 4 en las tres corridas del AG en 3D

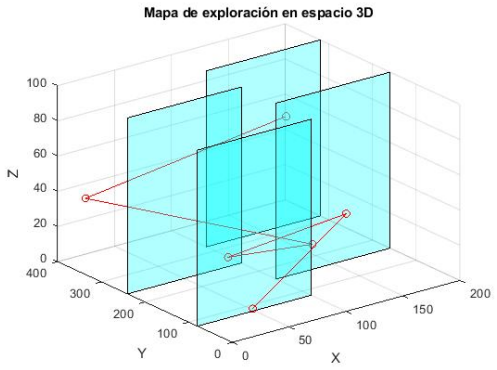
Los resultados de este escenario se muestran en la Figura 73 y las mejores soluciones en 74. En este se encuentran los mejores individuos seleccionados por el AG, como mejor solución al problema.



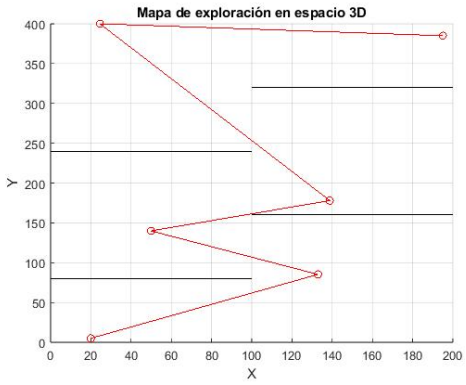
(a) Pob. 50, Mejor individuo



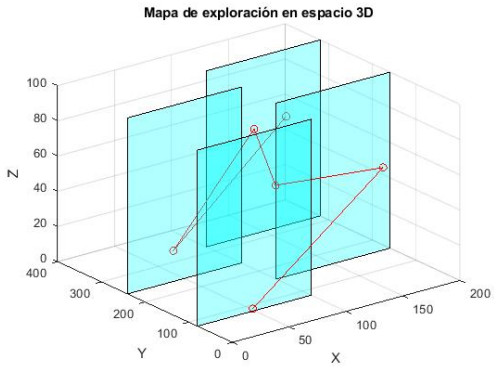
(b) Vista en plano xy



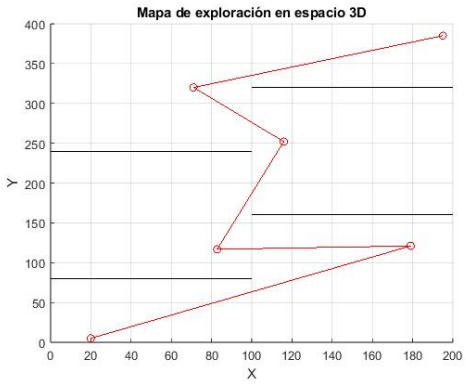
(c) Pob. 150, Mejor individuo



(d) Vista en plano xy



(e) Pob. 500, Mejor individuo



(f) Vista en plano xy

Figura 73: Mejores individuos generados por el AG en 3D, cuarto caso

$$\mathbf{T} = \begin{bmatrix} 20.00 & 20.00 & 20.00 \\ 1.01 & 82.00 & 16.70 \\ 84.50 & 225.00 & 23.00 \\ 95.00 & 225.00 & 32.00 \\ 130.00 & 396.00 & 87.01 \\ 195.00 & 385.00 & 50.00 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 20.00 & 20.00 & 20.00 \\ 132.90 & 85.24 & 39.96 \\ 49.90 & 139.88 & 23.55 \\ 138.82 & 177.98 & 10.94 \\ 24.48 & 400.00 & 31.68 \\ 195.00 & 385.00 & 50.00 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 20.00 & 20.00 & 20.00 \\ 104.00 & 82.80 & 24.80 \\ 27.60 & 191.76 & 40.26 \\ 105.00 & 234.00 & 53.00 \\ 50.40 & 302.40 & 28.80 \\ 195.00 & 385.00 & 50.00 \end{bmatrix}$$

(a) 50 individuos (b) 150 individuos (c) 500 individuos

Figura 74: Representación del cromosoma como mejor solución en escenario 4

- **Corrida 1 ($N = 50$):** los resultados gráficos mostraron que los individuos generados colisionaron con los obstáculos desde las primeras generaciones hasta las últimas, lo que refleja una exploración insuficiente del espacio de búsqueda debido al tamaño reducido de la población. Esto se corroboró con una aptitud de 0.01, lo que indica que no se logró encontrar una solución viable. Aunque se observaron intentos de dispersión para buscar mejores trayectorias, no se obtuvo un cromosoma que cumpliera con las restricciones.
- **Corrida 2 ($N = 150$):** en esta configuración, el algoritmo comenzó generando individuos con ligeras colisiones, pero progresivamente encontró trayectorias más óptimas y libres de obstáculos. Esto permitió alcanzar una aptitud de 0.00039 y una distancia de 2572.32, lo que representa una mejora significativa respecto a la primera corrida. Los resultados gráficos reflejan una exploración más efectiva del espacio de búsqueda gracias al aumento en la población.
- **Corrida 3 ($N = 500$):** con el mayor tamaño de población, el algoritmo comenzó sin generar individuos que colisionaran, aunque inicialmente presentó poca variabilidad en las soluciones. A medida que avanzaron las generaciones, se observó una mejora progresiva, obteniendo una trayectoria final óptima con la menor distancia recorrida (2122.234) y la mejor aptitud (0.00047). Sin embargo, esta configuración tuvo un costo computacional elevado, con un tiempo de ejecución de 26.97 segundos, lo que debe considerarse al evaluar la eficiencia general del algoritmo.

Este escenario mostró que el tamaño de la población tiene un impacto crítico en la capacidad del algoritmo para encontrar soluciones viables en entornos con alta densidad de restricciones. Un tamaño reducido ($N = 50$) resultó insuficiente para explorar adecuadamente el espacio de búsqueda, mientras que un tamaño mayor ($N = 500$) permitió obtener las mejores soluciones, aunque con un costo computacional elevado. La configuración intermedia ($N = 150$) demostró ser un compromiso razonable entre diversidad, tiempo computacional y calidad de las soluciones. Los resultados refuerzan la importancia de ajustar el tamaño de población según la complejidad del entorno, buscando un balance entre rendimiento computacional y calidad de las trayectorias generadas.

A pesar de los resultados positivos obtenidos en las corridas con poblaciones mayores, el alto costo computacional observado en la tercera corrida representa una limitante significativa. El incremento en el tamaño de población a $N = 500$ permitió encontrar soluciones

óptimas y libres de colisiones, pero el tiempo requerido (26.97 segundos) podría resultar inviable para aplicaciones prácticas que requieran respuestas rápidas o sistemas con recursos limitados. Este comportamiento destaca la necesidad de buscar un balance entre la calidad de las soluciones y la eficiencia computacional, priorizando configuraciones intermedias, como las observadas en la segunda corrida ($N = 150$), donde se logró una solución adecuada con un costo computacional razonable. Estas observaciones refuerzan la importancia de ajustar cuidadosamente el tamaño de población según las restricciones y objetivos de cada entorno.

Implementación de AG en planificación de trayectorias 3D, escenario 5

El quinto escenario se diseñó como un caso final para evaluar la capacidad del algoritmo genético para resolver un entorno tridimensional altamente restrictivo, aplicando los criterios optimizados deducidos de los escenarios anteriores. Este escenario buscó consolidar los aprendizajes obtenidos y demostrar cómo los algoritmos genéticos pueden planificar trayectorias eficientes y viables en condiciones desafiantes.

El mapa tridimensional se definió con dimensiones $300 \times 300 \times 150$, incluyendo cinco planos distribuidos en el eje y, con posiciones $y = \frac{m}{6}, \frac{2m}{6}, \frac{3m}{6}, \frac{4m}{6}, \frac{5m}{6}$. Los límites de cada plano se alternaron para crear zonas restringidas complejas y menos áreas despejadas:

- Plano 1: $x \in [0, \frac{n}{3}]$, $z \in [0, \frac{k}{2}]$
- Plano 2: $x \in [\frac{n}{3}, \frac{2n}{3}]$, $z \in [0, k]$
- Plano 3: $x \in [\frac{2n}{3}, n]$, $z \in [0, \frac{k}{2}]$
- Plano 4: $x \in [\frac{n}{3}, \frac{2n}{3}]$, $z \in [0, k]$
- Plano 5: $x \in [0, \frac{n}{3}]$, $z \in [\frac{k}{2}, k]$

Los puntos inicial y final se fijaron en $P_s = (10, 10, 10)$ y $P_g = (290, 290, 140)$, respectivamente, garantizando que las trayectorias generadas atravesaran múltiples regiones restringidas.

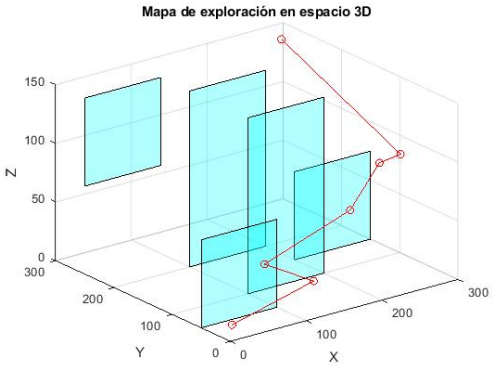
Para este escenario, los parámetros del algoritmo se establecieron en función de los resultados previos:

- Tamaño de población: $N = 150$ para garantizar suficiente diversidad sin excesivo costo computacional.
- Longitud del cromosoma: $L_{\text{crom}} = 7$, representando trayectorias suficientemente detalladas para entornos complejos.
- Iteraciones: 50, para mantener tiempos computacionales razonables.
- Probabilidad de mutación: $P_m = 20\,\%$, favoreciendo la diversidad en generaciones posteriores.

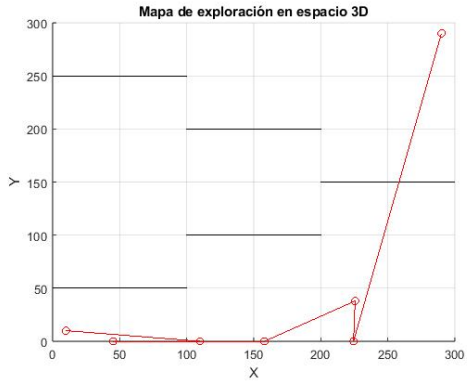
Este escenario permitió analizar cómo los algoritmos genéticos pueden adaptarse y responder en entornos representativos de aplicaciones reales, reforzando los aprendizajes obtenidos en los escenarios previos.

No.	Iteraciones	Población	Tiempo (s)	Aptitud	Distancia
1	150	50	1.236615	0.0002792	3581.133
2	250	50	1.891325	0.0002948	3392.658

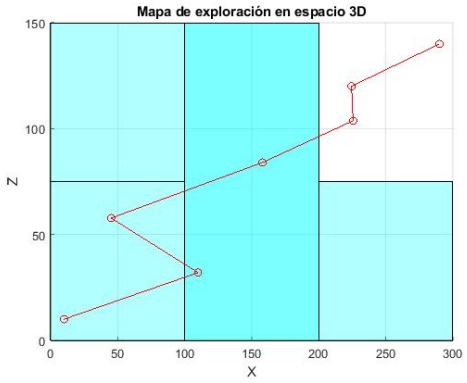
Cuadro 17: Resultados obtenidos para el escenario 5 en las tres corridas del AG en 3D



(a) Vista isométrica del mejor individuo



(b) Vista en plano xy del mejor individuo

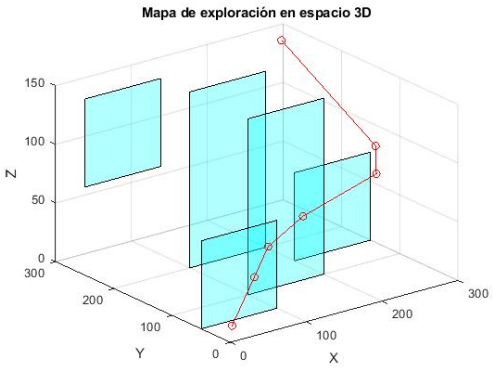


(c) Vista en plano xz del mejor individuo

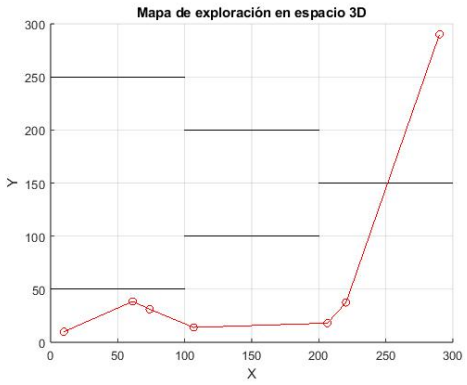
$$\mathbf{T} = \begin{bmatrix} 10.00 & 10.00 & 10.00 \\ 109.93 & 0.00 & 32.00 \\ 45.12 & 0.00 & 57.76 \\ 158.00 & 0.00 & 84.00 \\ 225.60 & 37.95 & 103.74 \\ 224.40 & 0.00 & 120.00 \\ 290.00 & 290.00 & 140.00 \end{bmatrix}$$

(d) Mejor cromosoma

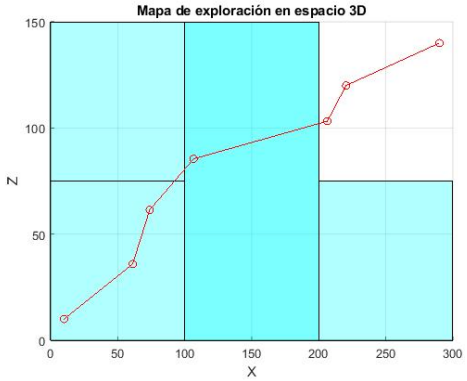
Figura 75: Mejores individuos generados en la primera corrida por AG en 3D, quinto caso



(a) Vista isométrica del mejor individuo



(b) Vista en plano xy del mejor individuo



(c) Vista en plano xz del mejor individuo

$$\mathbf{T} = \begin{bmatrix} 10.00 & 10.00 & 10.00 \\ 61.22 & 38.50 & 35.96 \\ 73.86 & 31.16 & 61.40 \\ 106.63 & 13.97 & 85.44 \\ 206.40 & 18.04 & 103.20 \\ 220.32 & 37.44 & 120.00 \\ 290.00 & 290.00 & 140.00 \end{bmatrix}$$

(d) Mejor cromosoma

Figura 76: Mejores individuos generados en la segunda corrida por AG en 3D, quinto caso

El quinto escenario representó el caso más desafiante evaluado en esta investigación, diseñado para consolidar los criterios optimizados obtenidos de los escenarios previos. Los resultados obtenidos en las dos corridas muestran un desempeño sólido del algoritmo genético en condiciones altamente restrictivas.

- **Corrida 1:** con un tamaño de población de $N = 150$, el algoritmo logró generar trayectorias libres de colisiones en todas las generaciones. Los gráficos muestran un balance adecuado entre exploración y explotación, con una aptitud final de 0.0002792 y una distancia recorrida de 3581.133. El tiempo computacional, de 1.236615 segundos, fue razonable, considerando la densidad de restricciones en el mapa.
- **Corrida 2:** al incrementar el tamaño de la población a $N = 250$, se observó una mejora en la distribución de las soluciones dentro del espacio de búsqueda, lo que permitió encontrar una trayectoria más corta, con una distancia final de 3392.658 y una aptitud de 0.0002948. Los gráficos reflejan una mejor cobertura del mapa, destacando la capacidad del algoritmo para adaptarse y explorar diferentes regiones. El tiempo computacional aumentó ligeramente a 1.891325 s, pero este incremento fue proporcional al tamaño de la población y aceptable para las condiciones del entorno.

Este escenario consolidó los aprendizajes de los casos previos, demostrando que los crite-

rios optimizados permiten al algoritmo genético abordar entornos tridimensionales complejos de manera efectiva. Ambas corridas generaron soluciones viables, libres de colisiones y con buena variación entre generaciones. La comparación entre las dos configuraciones de población destaca que $N = 250$ proporcionó una mejor calidad de solución, con una trayectoria más corta y una distribución más amplia en el espacio de búsqueda. Sin embargo, $N = 150$ ofreció un balance eficiente entre calidad de la solución y costo computacional, lo que refuerza su idoneidad para aplicaciones donde el tiempo de procesamiento es una limitante.

En este último escenario, el algoritmo genético demostró ser una herramienta robusta y adaptable para la planificación de trayectorias en entornos tridimensionales con múltiples restricciones, cumpliendo con los objetivos planteados y resaltando su potencial en aplicaciones dentro de la Ingeniería Mecatrónica.

Conclusiones generales de la aplicación

A través de los cinco escenarios diseñados y evaluados, se demostró que los algoritmos genéticos son una herramienta efectiva y robusta para la planificación de trayectorias en entornos tridimensionales con restricciones complejas. Cada escenario permitió abordar aspectos específicos del desempeño del algoritmo, desde la generación de trayectorias libres de colisiones hasta el análisis del impacto de parámetros clave, como el tamaño de población, la longitud del cromosoma, y la probabilidad de mutación. Los resultados obtenidos evidenciaron una evolución progresiva en la calidad de las soluciones, consolidando criterios optimizados que fueron aplicados exitosamente en el quinto escenario, donde se logró resolver un entorno desafiante con alta densidad de restricciones de manera eficiente. Esto validó la capacidad del algoritmo genético para adaptarse y responder en condiciones representativas de aplicaciones prácticas.

El análisis experimental realizado a lo largo de los escenarios permitió cumplir con el objetivo planteado al principio, evaluando los algoritmos genéticos en aplicaciones de la Ingeniería Mecatrónica, particularmente en la planificación de trayectorias tridimensionales. Los resultados mostraron que estos algoritmos son capaces de adaptarse y generar soluciones de calidad en condiciones diversas, optimizando tanto la diversidad gráfica como la calidad de las trayectorias generadas. Asimismo, los ajustes realizados en los parámetros demostraron que el balance entre exploración del espacio de búsqueda y rendimiento computacional es fundamental para el éxito del algoritmo. Esta implementación consolidó un enfoque sistemático para evaluar y ajustar algoritmos genéticos, alineándose con el objetivo general de la investigación al demostrar su implementación en aplicaciones dentro de la rama de la Ingeniería Mecatrónica.

9.2. Evaluación de algoritmos genéticos en procesamiento de imágenes

En esta implementación se demostró la efectividad de los algoritmos genéticos (AG) para la segmentación de imágenes, validando su aplicabilidad en otras áreas de la Ingeniería Mecatrónica, conforme al tercer objetivo específico. Este objetivo buscó evaluar los AG fuera

del ámbito exclusivo de la robótica, optimizando procesos como la división de imágenes en regiones homogéneas basadas en características de intensidad.

La segmentación de imágenes resolvió la problemática de identificar automáticamente regiones relevantes en imágenes de alto contraste y ruido, un desafío común en aplicaciones industriales, biomédicas y de visión computacional. Mediante el uso de AG, se optimizaron los umbrales de segmentación, permitiendo dividir la imagen en regiones bien definidas, adaptándose a condiciones complejas donde métodos tradicionales suelen fallar.

9.2.1. Metodología

Se implementó un algoritmo genético para la segmentación de imágenes, con el objetivo de optimizar umbrales que permitieran dividir automáticamente una imagen en regiones homogéneas. Este procedimiento permitió evaluar el desempeño del AG en tareas de procesamiento de imágenes. En esta implementación, se utilizaron las herramientas computacionales para lectura de imágenes que posee MATLAB.

Representación de cromosomas

Cada cromosoma representó un conjunto de umbrales, codificados como valores en un vector numérico. Estos valores se ajustaron automáticamente durante el proceso evolutivo, permitiendo una segmentación adaptativa a las características de cada imagen, como se muestra en la Figura 77.

$$\mathbf{C} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \quad \xrightarrow{\text{Segmentación}} \quad \begin{cases} R_1 : I(x,y) \leq t_1 \\ R_2 : t_1 < I(x,y) \leq t_2 \\ R_3 : I(x,y) > t_2 \end{cases}$$

Figura 77: Representación del cromosoma como un vector de umbrales y su codificación en regiones homogéneas

Los valores de los umbrales $(t_1, t_2, \dots)$ se codificaron como números reales dentro del intervalo $[0, 1]$, en correspondencia con las intensidades normalizadas de las imágenes. Esta codificación permitió representar directamente los límites de las regiones segmentadas, evitando conversiones adicionales y facilitando la aplicación de operadores genéticos. Durante las operaciones de selección, cruce y mutación, se garantizó que los umbrales se mantuvieran ordenados $(t_1 < t_2 < \dots)$, preservando la coherencia en la segmentación [13].

Operadores genéticos

Se utilizó selección por torneo, descrita en la Sección 6.6.4, cruce aritmético, utilizado en implementaciones anteriores, y mutación aleatoria para generar diversidad en la población.

La selección favoreció a los cromosomas con mayor aptitud, mientras que el cruce y la mutación introdujeron variabilidad para evitar el estancamiento en óptimos locales.

Función de aptitud

La función de aptitud se definió para maximizar la entropía interregión y minimizar la entropía intrarregión, asegurando que las regiones segmentadas fueran homogéneas internamente y distintas entre sí. Esta métrica justificó el desempeño del AG en imágenes de diferentes características [19].

$$f(C) = -\left(\frac{H_{\text{intra}}}{H_{\text{inter}} + \epsilon}\right)$$

Figura 78: Relación entre la entropía intrarregión (H_{intra}) e interregión (H_{inter}) en la función de aptitud

Configuración experimental

Se evaluaron imágenes en escala de grises, seleccionadas por su contraste y nivel de complejidad, incluyendo imágenes con ruido agregado para medir la robustez del AG. El algoritmo se configuró con una población inicial de 50 individuos, evolucionando durante 200 generaciones con una probabilidad de mutación del 20 %. Las pruebas se realizaron en MATLAB, utilizando imágenes de referencia para validar los resultados.

9.2.2. Resultados

En esta sección se presentan los resultados obtenidos al implementar y evaluar el algoritmo genético para segmentación de imágenes. Los experimentos realizados incluyeron imágenes de características variadas para validar la eficacia, robustez y capacidad de adaptación del AG en comparación con métodos tradicionales como Otsu y K-means.

Los resultados se estructuraron en dos partes: inicialmente, se analizó el desempeño del AG en una imagen de alto contraste, resaltando la segmentación en tres regiones bien definidas. Posteriormente, se evaluaron imágenes con ruido y complejidad creciente, destacando la adaptabilidad del AG en condiciones desafiantes.

Las métricas empleadas incluyen la entropía intra e interregión, el índice de Jaccard y el tiempo de ejecución, permitiendo una comparación cuantitativa entre los diferentes métodos.

Segmentación de imágenes de alto contraste

Se evaluó el algoritmo genético utilizando una imagen de alto contraste con regiones claras y bien definidas. Este escenario permitió analizar la capacidad del AG para segmentar

eficazmente imágenes donde los bordes entre regiones son evidentes. En la Figura 79, se presentan los resultados visuales obtenidos mediante el AG y se comparan con los métodos de Otsu y K-means.

Las métricas cuantitativas se muestran en el Cuadro 18, donde se destacan las bajas entropías intrarregión del AG, indicando regiones homogéneas, y su alto tiempo de ejecución en comparación con métodos tradicionales.

Segmentación en imágenes de alto contraste

El primer experimento utilizó una imagen de alto contraste en escala de grises, ideal para evaluar la capacidad del algoritmo genético de segmentar regiones bien definidas. Se configuraron los parámetros del AG con un tamaño de población de 50 individuos, 200 generaciones y una tasa de mutación del 20 %. Estos valores se establecieron para balancear la diversidad en la población y permitir la convergencia hacia una solución óptima, dado que las regiones de la imagen poseen bordes claros pero requieren una segmentación precisa. Además, se segmentó la imagen en tres regiones homogéneas, objetivo acorde a la naturaleza del problema.

En la Figura 79, se presentan los resultados visuales de la segmentación. El AG generó tres regiones homogéneas utilizando los mejores umbrales obtenidos ($t_1 = 0.2771$, $t_2 = 0.6174$). En comparación, el método de Otsu, al ser binario, no logró dividir la imagen en tres regiones. Por otro lado, K-means presentó transiciones menos definidas entre regiones, destacando la precisión superior del AG.

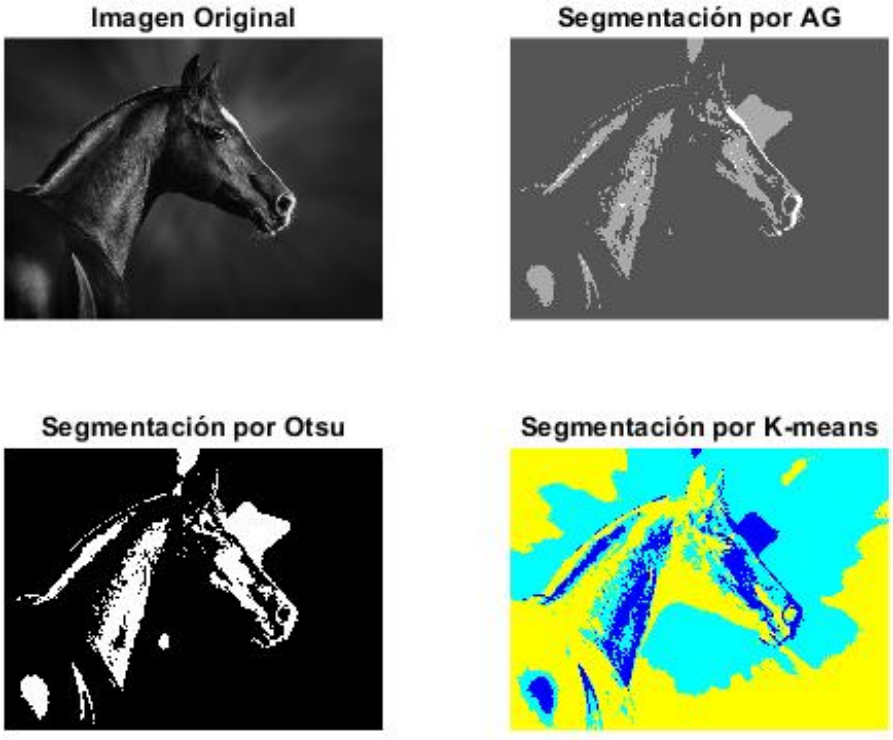


Figura 79: Segmentación en imágenes de alto contraste: (a) imagen original, (b) AG, (c) Otsu, (d) K-means

Las métricas cuantitativas, presentadas en el Cuadro 18, mostraron que el AG logró una entropía intrarregión de 12.7251, la más baja entre los métodos evaluados, lo que reflejó regiones internas más homogéneas. La entropía interregión fue consistente entre los métodos (4.4345), indicando una buena diferenciación global de las regiones segmentadas. Sin embargo, el tiempo de ejecución del AG (946.88 s) fue significativamente mayor, evidenciando su alta complejidad computacional frente a métodos más simples como Otsu (0.2 s) y K-means (1.3 s).

Método	Entropía intrarregión	Entropía interregión	Tiempo (s)
AG	12.7251	4.4345	946.88
Otsu	0.0	4.4345	0.2
K-means	11.2353	4.4345	1.3

Cuadro 18: Métricas obtenidas para la segmentación en imágenes de alto contraste

Este experimento demostró que el algoritmo genético fue capaz de segmentar eficazmente la imagen de alto contraste, generando regiones internas homogéneas y bien diferenciadas. El uso de parámetros como una población de 50 individuos y 200 generaciones permitió alcanzar un equilibrio entre diversidad y convergencia en un problema donde los bordes

entre regiones eran claros pero requerían ajustes finos. Aunque el AG presentó un tiempo computacional mayor, su precisión fue superior frente a métodos tradicionales, como Otsu y K-means, validando su capacidad para resolver problemas de segmentación en imágenes de alto contraste. Estos resultados cumplieron con los objetivos específicos de evaluar la eficacia del AG en problemas de segmentación, destacando sus ventajas en escenarios donde se prioriza la calidad sobre el tiempo de ejecución.

En una segunda prueba, se ajustaron los parámetros del algoritmo genético para analizar su comportamiento bajo condiciones distintas. Se incrementó el tamaño de la población a 100 individuos y el número de regiones a cuatro, mientras que las generaciones se redujeron a 150 y la tasa de mutación a 10 %. Este ajuste buscó mejorar la exploración de soluciones mediante una mayor población, adaptándose a un número más alto de regiones, y acelerar la convergencia reduciendo el número de generaciones. Sin embargo, esto implicó un aumento en el tiempo computacional, como se reflejó en el tiempo de ejecución total (1584.31 s).

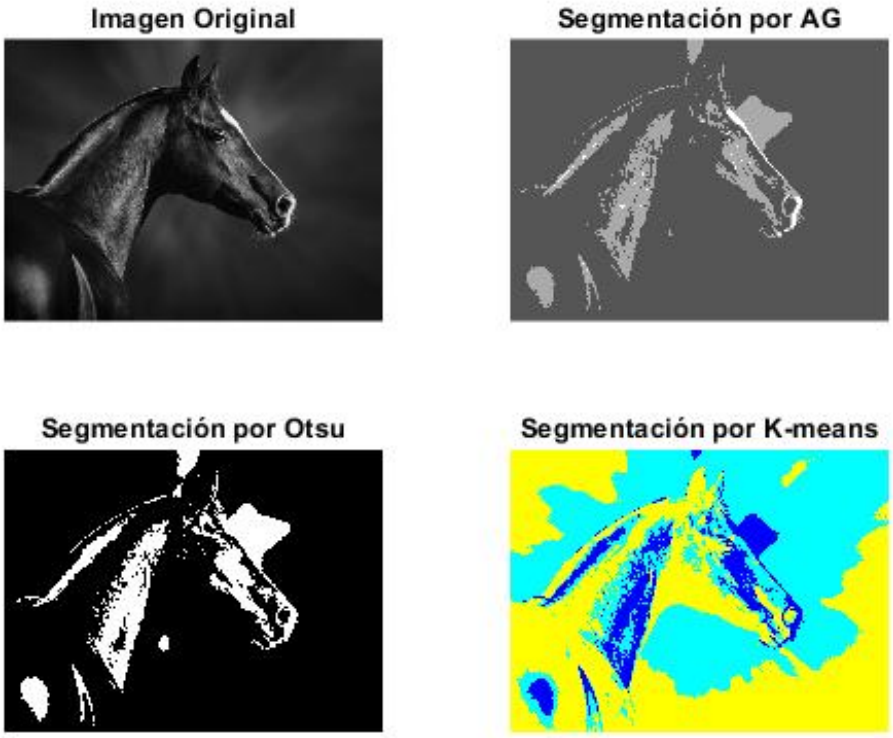


Figura 80: Segmentación en imágenes de alto contraste con parámetros ajustados: (a) imagen original, (b) AG, (c) Otsu, (d) K-means

Método	Entropía intrarregión	Entropía interregión	Tiempo (s)
AG	15.9695	4.4345	1584.31
Otsu	4.3962	4.4345	0.2
K-means	14.6883	4.4345	1.3

Cuadro 19: Métricas obtenidas para la segmentación con parámetros ajustados

Los resultados obtenidos, presentados en la Figura 80 y el Cuadro 19, demostraron que el AG fue capaz de segmentar cuatro regiones homogéneas, aunque con un incremento en la entropía intrarregión (15.9695), indicando una ligera disminución en la homogeneidad interna. Comparativamente, K-means también presentó un aumento en la entropía intrarregión (14.6883), mientras que Otsu mantuvo una baja entropía intrarregión (4.3962) debido a su naturaleza binaria. Aunque el AG continuó mostrando una diferenciación consistente entre regiones (entropía interregión de 4.4345), el incremento en el tiempo de ejecución resaltó la sensibilidad del método a cambios en la configuración de parámetros. Este experimento confirmó la flexibilidad del AG para adaptarse a problemas más complejos, cumpliendo su objetivo de segmentación precisa a expensas de una mayor demanda computacional.

Segmentación en imágenes con ruido Gaussiano

Para analizar la robustez del algoritmo genético, se diseñó un escenario utilizando imágenes afectadas por ruido Gaussiano. Este tipo de ruido, común en sistemas de adquisición de imágenes, introduce perturbaciones aleatorias que pueden dificultar la segmentación precisa. Se generaron imágenes con media cero y una varianza de 0.01, simulando un nivel de ruido moderado que representa condiciones adversas, pero habituales en aplicaciones reales.

Los parámetros del AG se mantuvieron constantes con respecto a los experimentos previos: una población de 50 individuos, 200 generaciones y una tasa de mutación del 20 %. Esto permitió comparar directamente su desempeño en imágenes ruidosas frente a los métodos de Otsu y K-means, utilizando las mismas configuraciones de segmentación en tres regiones homogéneas. Este enfoque buscó validar la capacidad del AG para adaptarse a imágenes degradadas, asegurando que las regiones segmentadas sean coherentes y diferenciadas, incluso en condiciones de ruido.

Los resultados visuales de la segmentación se presentan en las Figuras 82, 83 y 81. El AG generó segmentaciones más homogéneas y adaptativas en todas las imágenes a pesar del ruido, mientras que los métodos tradicionales presentaron limitaciones. Otsu, debido a su enfoque binario, falló en capturar detalles en múltiples regiones, mientras que K-means mostró transiciones menos definidas.

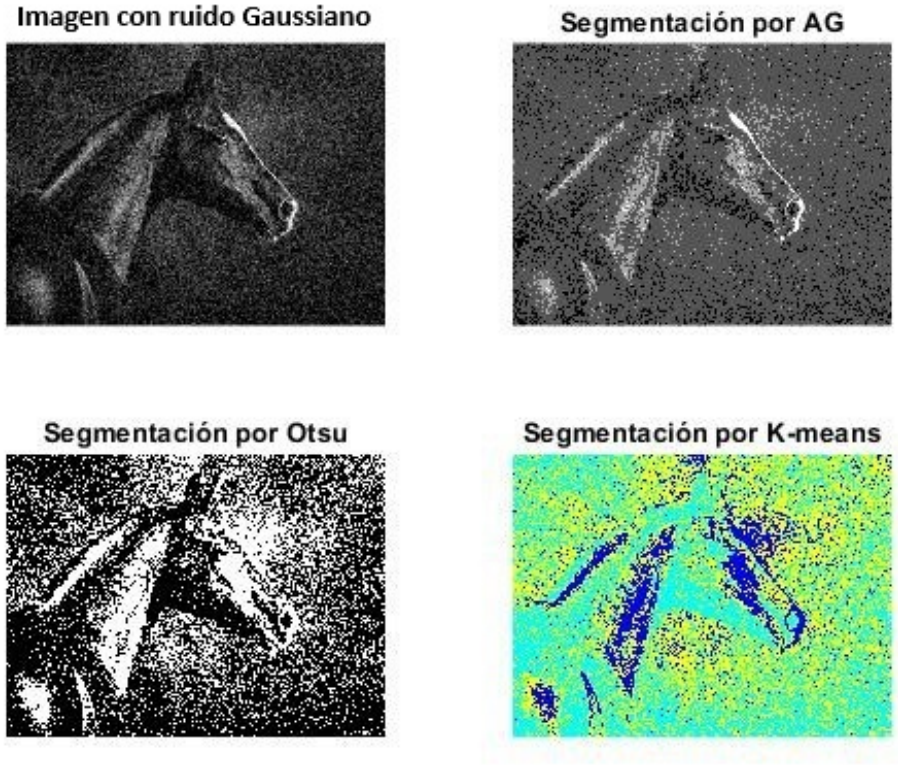


Figura 81: Segmentación en imagen con ruido gaussiano (caballo): (a) imagen original, (b) AG, (c) Otsu, (d) K-means

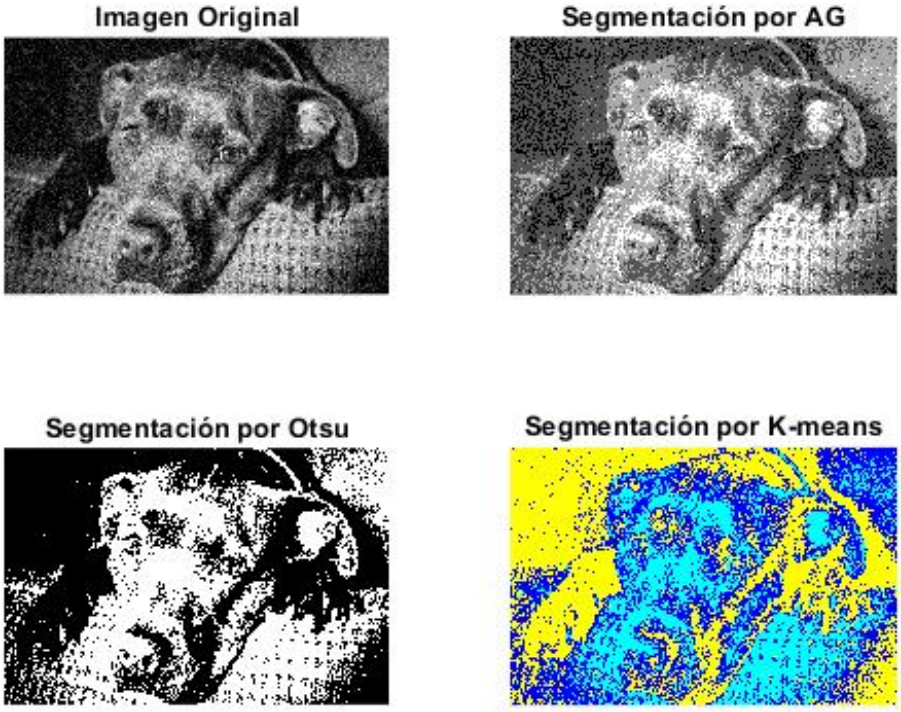


Figura 82: Segmentación en imagen con ruido gaussiano (perro): (a) imagen original, (b) AG, (c) Otsu, (d) K-means

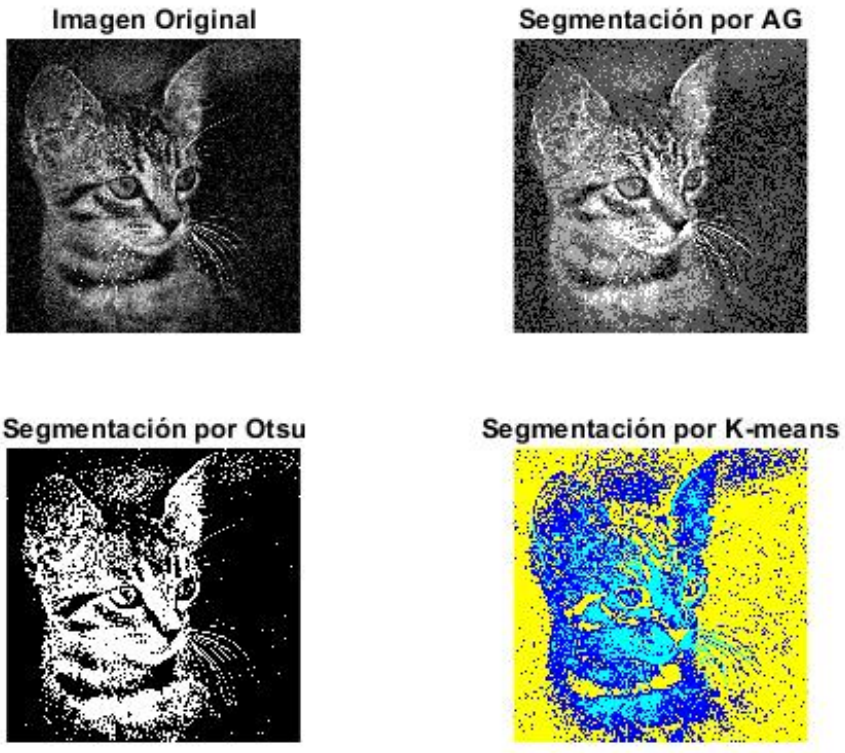


Figura 83: Segmentación en imagen con ruido gaussiano (gato): (a) imagen original, (b) AG, (c) Otsu, (d) K-means

En el Cuadro 20 se presentan las métricas obtenidas. El AG mantuvo una entropía intrarregión más baja (12.7709, 13.0883 y 13.0458 en cada corrida); indicando regiones homogéneas a pesar del ruido. Otsu presentó los valores más bajos en entropía intrarregión debido a su segmentación binaria, pero perdió capacidad para diferenciar múltiples regiones. K-means, aunque computacionalmente más eficiente, mostró una mayor entropía intrarregión, reflejando menos homogeneidad interna. En términos de entropía interregión, todos los métodos mantuvieron valores similares, destacando la capacidad del AG de adaptarse al ruido sin sacrificar diferenciación entre regiones.

Imagen	Método	Entropía intrarregión	Entropía interregión	Tiempo (s)
Caballo	AG	12.7709	4.6075	384.46
	Otsu	4.3182	4.6075	0.2
	K-means	11.2640	4.6075	1.3
Perro	AG	13.0883	5.1768	985.58
	Otsu	4.9042	5.1768	0.2
	K-means	12.5394	5.1768	1.3
Gato	AG	13.0458	4.6472	326.22
	Otsu	4.8873	4.6472	0.2
	K-means	12.1129	4.6472	1.3

Cuadro 20: Métricas obtenidas en imágenes con ruido gaussiano

Este escenario demostró que el algoritmo genético fue robusto frente a ruido Gaussiano, generando segmentaciones precisas y homogéneas incluso en condiciones adversas. Aunque su tiempo de ejecución fue significativamente mayor en comparación con Otsu y K-means, el AG destacó por su capacidad de adaptación y su habilidad para mantener la homogeneidad de las regiones segmentadas, cumpliendo con el objetivo de evaluar su eficacia en escenarios ruidosos. Estos resultados validaron la aplicabilidad del AG en problemas reales de segmentación bajo condiciones no ideales.

Segmentación en imágenes complejas

El tercer escenario se centró en evaluar el desempeño del algoritmo genético (AG) en imágenes con múltiples objetos y regiones heterogéneas, conocidas como imágenes complejas. Este tipo de imágenes presenta desafíos adicionales debido a la presencia de texturas, bordes menos definidos y diferencias de intensidad más sutiles, lo que complica la identificación de regiones homogéneas. Estas características son comunes en aplicaciones prácticas como escenas industriales o biomédicas.

Los parámetros del AG se mantuvieron consistentes con los experimentos previos: 50 individuos, 200 generaciones, una tasa de mutación del 20 % y tres regiones para segmentación. Esto permitió analizar su desempeño comparándolo con métodos tradicionales como Otsu y K-means.

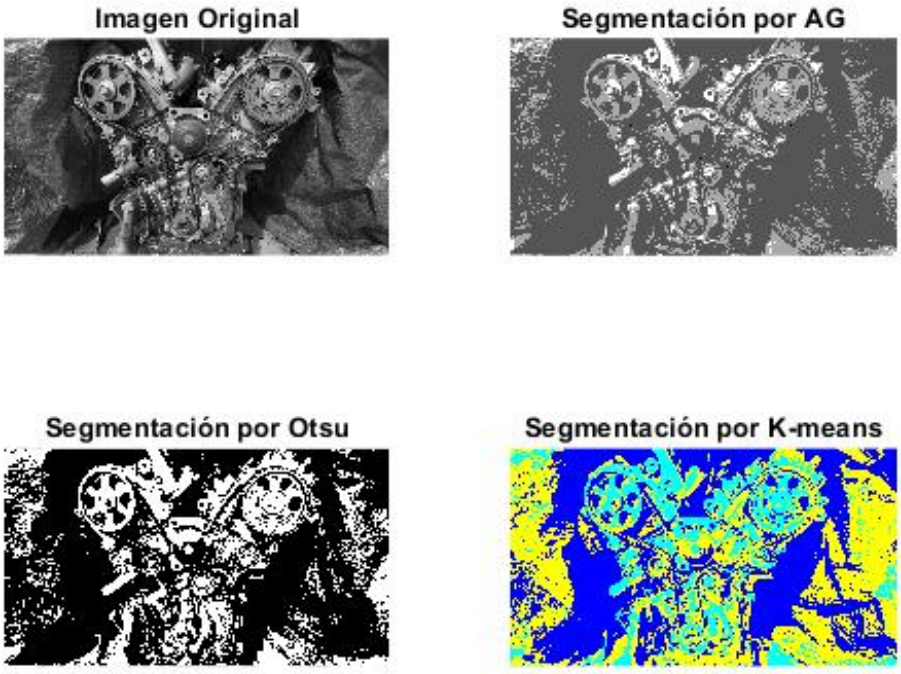


Figura 84: Segmentación en imagen industrial (motor): (a) imagen original, (b) AG, (c) Otsu, (d) K-means

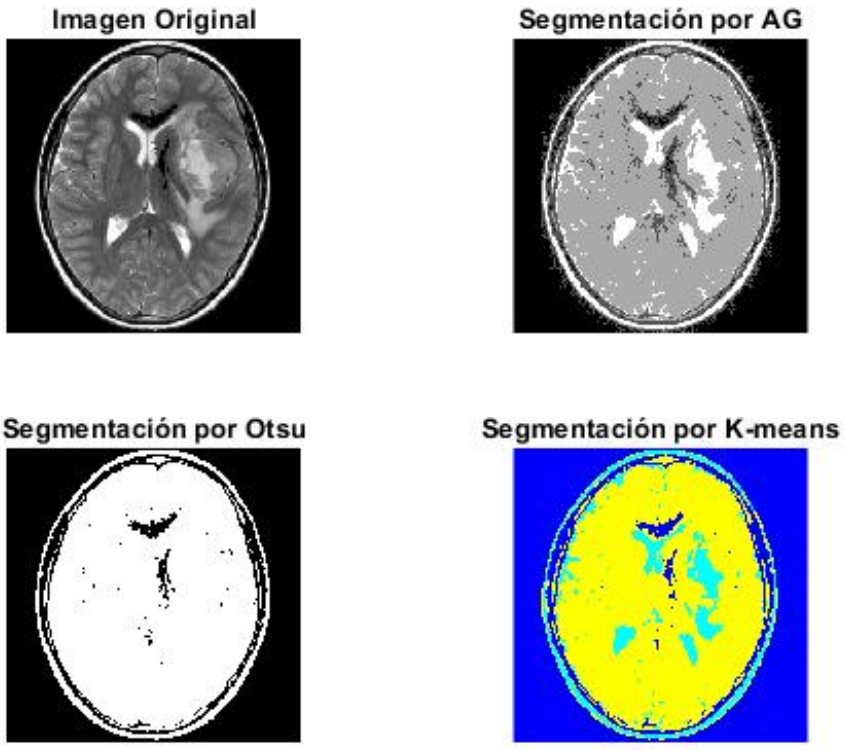


Figura 85: Segmentación en resonancia magnética cerebral con tumor: (a) imagen original, (b) AG, (c) Otsu, (d) K-means

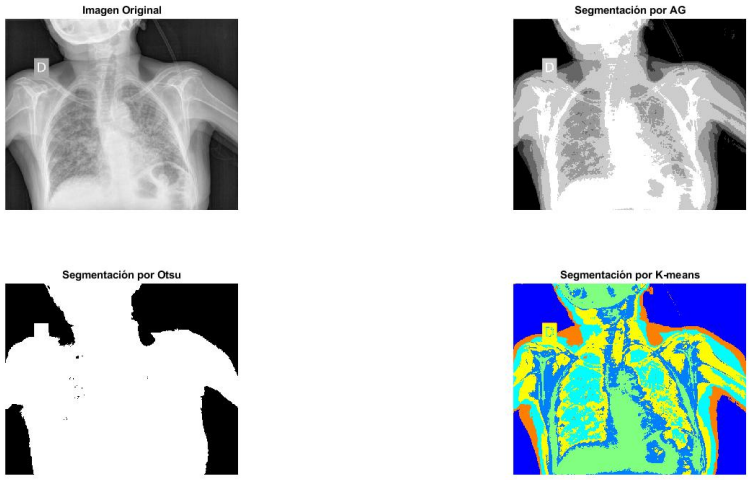


Figura 86: Segmentación en radiografía de tórax: (a) imagen original, (b) AG, (c) Otsu, (d) K-means

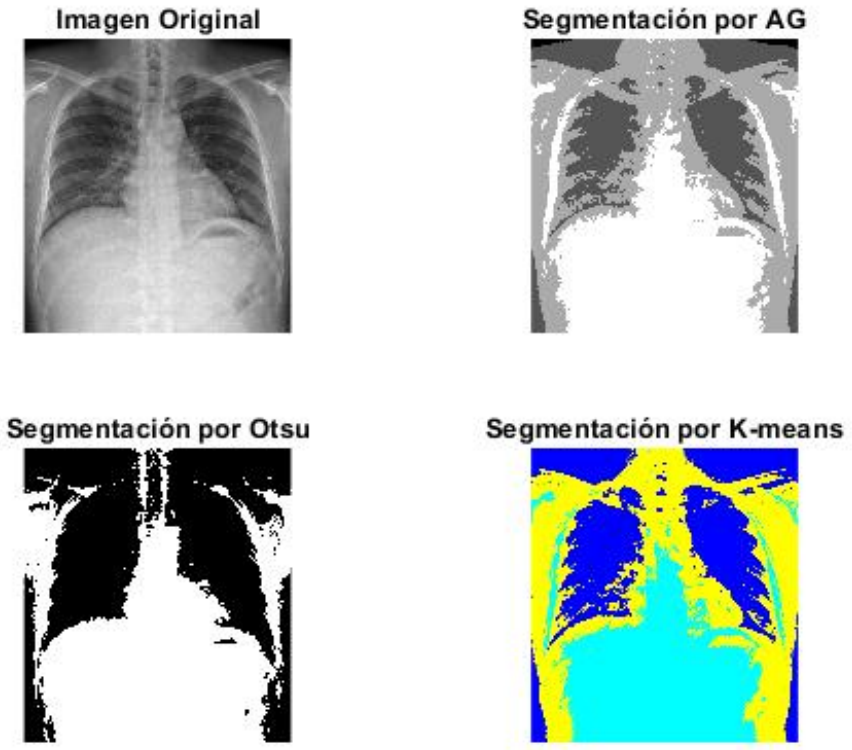


Figura 87: Segmentación en radiografía de tórax más definida: (a) imagen original, (b) AG, (c) Otsu, (d) K-means

En el Cuadro 21 se resumen las métricas obtenidas. El AG mantuvo una entropía intrarregión baja (12.8351, 12.1593 y 12.5187) en comparación con K-means, reflejando regiones más homogéneas en las imágenes biomédicas. Sin embargo, Otsu presentó los valores más bajos debido a su naturaleza binaria, lo que limita su capacidad para segmentar múltiples regiones. En términos de entropía interregión, todos los métodos obtuvieron resultados consistentes, destacando la diferenciación entre regiones. El tiempo de ejecución del AG fue significativamente mayor, como era esperado por su complejidad computacional.

Imagen	Método	Entropía intrarregión	Entropía interregión	Tiempo (s)
Motor	AG	13.0348	5.2159	685.62
	Otsu	4.7502	5.2159	0.2
	K-means	12.8097	5.2159	1.3
Cerebro	AG	12.8351	3.8903	37.10
	Otsu	4.8498	3.8903	0.2
	K-means	10.0716	3.8903	1.3
RX1	AG	12.1593	5.1277	48.95
	Otsu	4.7186	5.1277	0.2
	K-means	12.0673	5.1277	1.3
RX2	AG	12.5187	4.8201	115.34
	Otsu	4.7565	4.8201	0.2
	K-means	12.4320	4.8201	1.3

Cuadro 21: Métricas obtenidas para imágenes complejas

Este escenario demostró que el algoritmo genético es altamente adaptable para segmentar imágenes complejas, generando regiones homogéneas y diferenciadas incluso en escenarios con múltiples texturas o regiones heterogéneas. Aunque Otsu logró valores bajos de entropía intrarregión debido a su enfoque binario, perdió precisión al segmentar múltiples regiones. K-means, aunque computacionalmente más eficiente, mostró mayor variabilidad en la homogeneidad de las regiones. Los resultados validaron que el AG es una herramienta eficaz para problemas reales de segmentación, cumpliendo con el objetivo de evaluar su desempeño en imágenes con características diversas.

9.2.3. Discusión

Los resultados obtenidos en los tres escenarios evaluados (alto contraste, ruido gaussiano e imágenes complejas) demostraron la capacidad del algoritmo genético para segmentar imágenes de manera precisa y robusta, incluso en condiciones adversas o de alta heterogeneidad. En todos los casos, el AG logró una segmentación con entropías intrarregión bajas, destacando su capacidad para generar regiones homogéneas, mientras que la entropía interregión fue consistente entre los métodos, evidenciando una diferenciación adecuada de las regiones. Comparado con los métodos tradicionales, como Otsu y K-means, el AG superó en precisión a costa de un mayor tiempo de ejecución, lo que lo posiciona como una herramienta adecuada para problemas donde la calidad es prioritaria.

Estos resultados validaron el cumplimiento del objetivo descrito al inicio, que consistía en aplicar y evaluar los algoritmos genéticos en el procesamiento de imágenes, al demostrar su eficacia en un área relevante de la Ingeniería Mecatrónica. La capacidad del AG para segmentar imágenes complejas y ruidosas refleja su potencial para aplicaciones prácticas, como diagnóstico biomédico, inspección visual en sistemas industriales y visión artificial en sistemas robóticos. Así, la implementación desarrollada no solo probó la flexibilidad y adaptabilidad del AG, sino también su aplicabilidad como una herramienta valiosa dentro del ámbito de la Ingeniería Mecatrónica.

CAPÍTULO 10

Algoritmos genéticos aplicados a escenarios prácticos

En los escenarios evaluados en MATLAB no se tomó en consideración las limitantes físicas de los cuerpos a controlar. Tampoco se especificó el funcionamiento de un controlador o el modelo de un robot en específico. El objetivo de los experimentos contenidos en este capítulo es demostrar el funcionamiento de los AG en escenarios realistas, considerando las condiciones físicas que están asociadas a su aplicación. Para desarrollar los escenarios prácticos para las distintas aplicaciones y validar las implementaciones del AG se debe utilizar un software. En este capítulo se estableció utilizar el software *Webots* para desarrollar los escenarios correspondientes donde se validó cada respectivo AG desarrollado en las secciones anteriores. La versión de Webots utilizada corresponde a la R2023b, basando su configuración en la documentación establecida en [20].

10.1. Implementación de AG para planificación de trayectorias en Webots

Como se mencionó anteriormente, se planteó la demostración mediante esta implementación que el algoritmo funcione en un escenario realista. Este escenario se apega a todo lo demostrado en el Capítulo 8.2. El entorno de Webots permite configurar objetos cilíndricos y una mapa con superficie cerrada para la simulación. Por lo tanto, se desarrolló la simulación en un mapa convencional que se muestra en la Figura 88. Para esta simulación se utilizó como objeto de prueba un robot móvil Pololu 3 pi, con su respectivo controlador de velocidad lineal y angular, para la implementación. No se brindó un mayor enfoque en el controlador, simplemente se describen los parámetros de su aplicación para formas de uso.

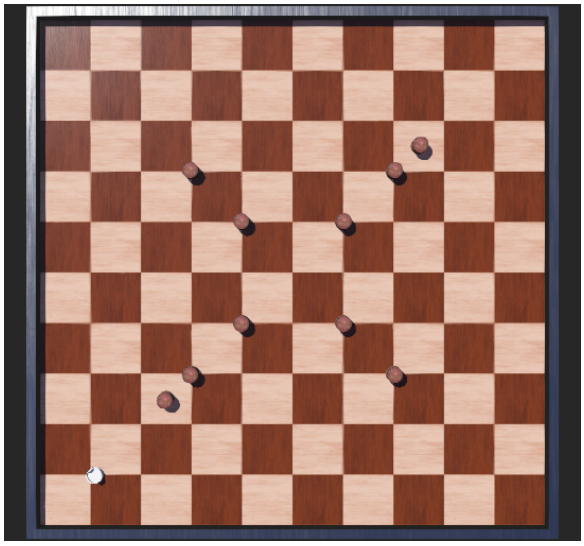


Figura 88: Mapa para simulación de AG en Webots

Como se puede apreciar en la Figura 88, el mapa consta de una cuadrícula con una cantidad de obstáculos cilíndricos. Los obstáculos son barriles que contiene la librería de Webots. Todos los parámetros para la simulación se especifican en el Cuadro 22. Cabe resaltar algunos aspectos a tomar en cuenta para la simulación que posee la plataforma. En el caso de las unidades Webots utiliza metros, por lo que se realizó la modificación de los parámetros de simulación como se muestra en el siguiente Algoritmo 10.1.

Parámetro	**Valor**
Dimensión del mapa	5×5 m
Punto de inicio	(-2.25, -2.25)
Punto de meta	(2.30, 2.30)
Población	30 ind.
Generaciones	20
Longitud del Cromosoma	5
Radio de obstáculos	8 cm
Radio del Robot	6.5 cm

Cuadro 22: Parámetros de simulación de AG en escenario realista

Algoritmo 10.1: Conversión a metros

```
1    Defina las variables
2    Dimension del mapa = Dimension del Mapa * 100;
3    Obstaculos = Obstaculos*100 + Dimension del mapa/2;
4    Punto de incio = punto de incio * 100 + Dimension del mapa/2;
5    Punto de meta = punto de meta * 100 + Dimension del mapa/2;
6    Radio obstaculo = Radio Obstaculo * 100;
7    Radio robot = Radio robot * 100;
```

No se realizaron cambios en la programación del algoritmo, simplemente se adaptó el controlador al mismo lenguaje en MATLAB. El controlador consta de un programa principal

donde se desarrolló el funcionamiento del robot para aplicar la función de planificación del AG y los parámetros de las variables de control. El programa calcula la trayectoria y el controlador se encarga de mandar las condiciones de movimiento al robot móvil.

10.1.1. Controlador para robot móvil

Para poder seguir los puntos de trayectoria que devuelve el AG en forma de vector, es necesario implementar un controlador. En este caso el controlador implementado es el controlador de pose lineal. Es esta compuesto de la manera que se describe en el marco teórico en la sección 6.17. Las constantes implementadas para el controlador se describen en el Cuadro 23, para poder tener un resultado óptimo en la velocidad de movimiento del robot.

Parámetro	**Valor**
k_ρ	175
k_α	0.2
k_β	-0.0015

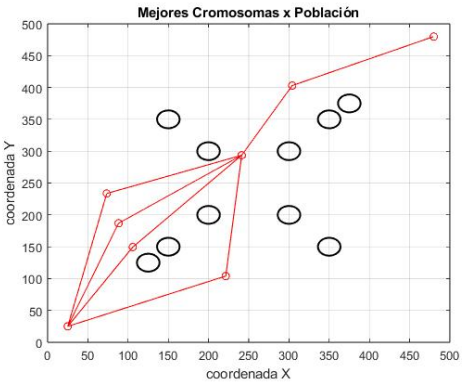
Cuadro 23: Valores de las constantes del controlador de pose lineal

10.1.2. Resultados de la implementación AG en simulación de Webots

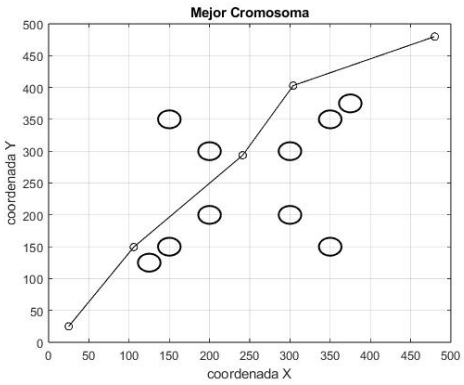
Al implementar el algoritmo genético en Webots, se obtuvo una simulación muy fluida. El tiempo que se tarda el AG en crear la trayectoria es computacionalmente rápido, estos resultados se muestran en el Cuadro 24. El controlador funciona muy bien con los parámetros especificados sin presentar oscilaciones. El resultado brindó trayectorias libres de obstáculos como se muestra en la Figura 89, donde se aprecia la trayectoria que devuelve la función del AG para planificación. En el conjunto de la Figura 90 se muestran los 5 puntos que recorrió el robot, siguiendo la misma trayectoria calculada por el algoritmo a la perfección. El tiempo de la simulación es irrelevante, debido a que no es parte del enfoque medir el rendimiento de la simulación como objetivo principal.

Parámetro	**Valor**
Tiempo (s)	0.946
Aptitud	0.00151
Distancia (m)	6.64
Iteraciones	20

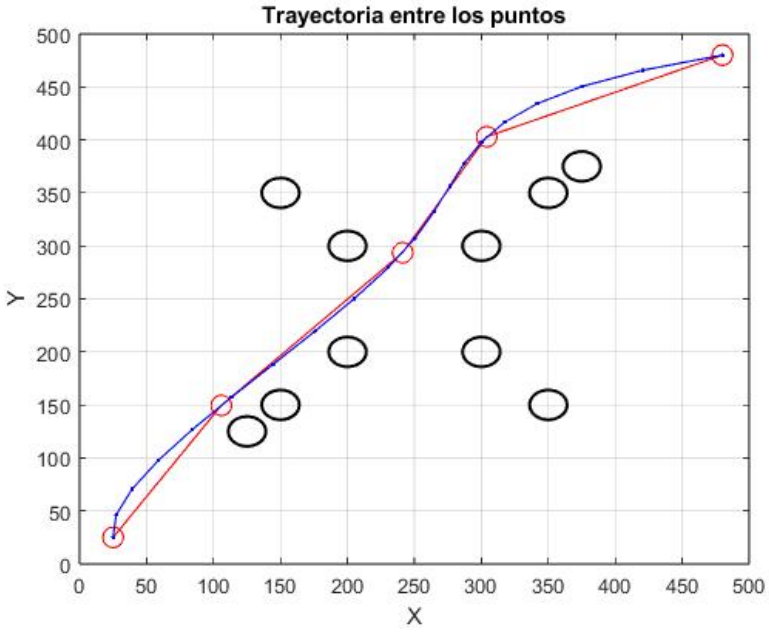
Cuadro 24: Parámetros de rendimiento del AG en simulación realista



(a) Mejores soluciones

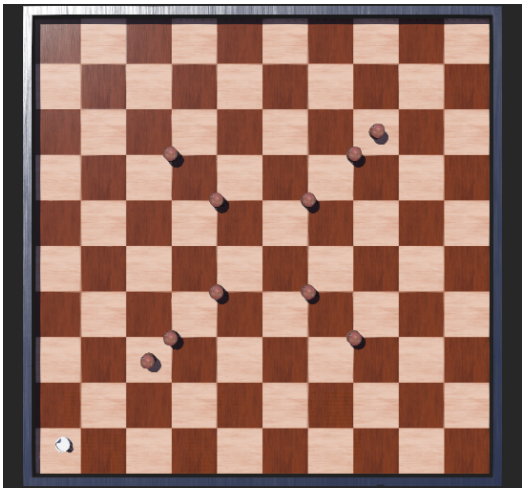


(b) Mejor solución

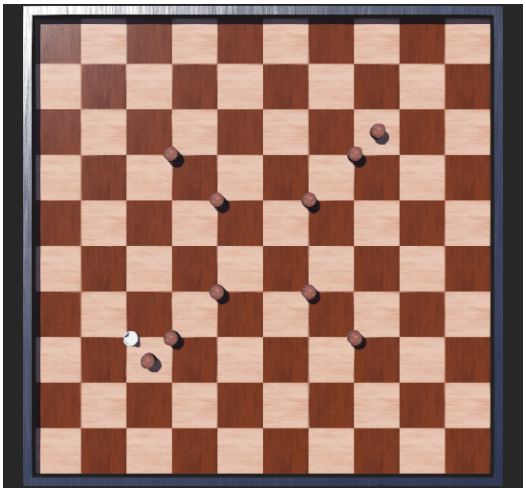


(c) Trayectoria suavizada

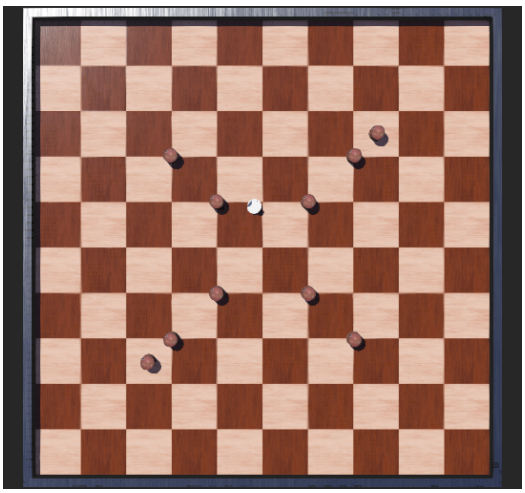
Figura 89: Trayectoria resultante del AG para el mapa en Webots



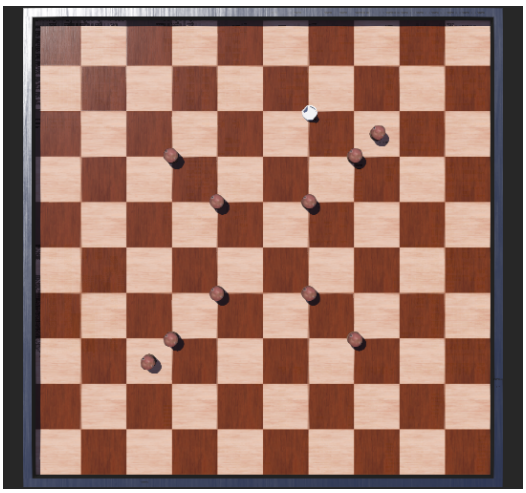
(a) Punto de inicio



(b) Punto número 2

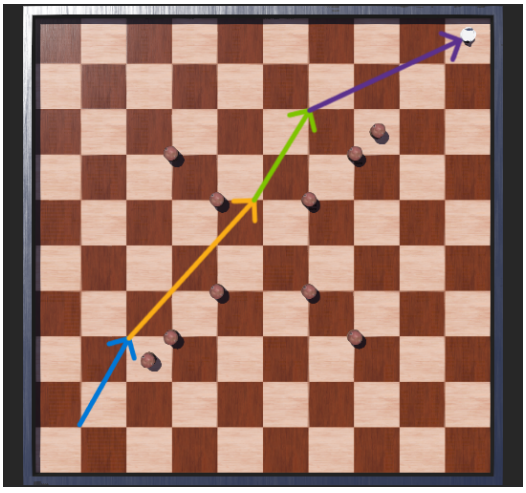


(c) Punto número 2



(d) Punto número 3

(e) Punto de meta



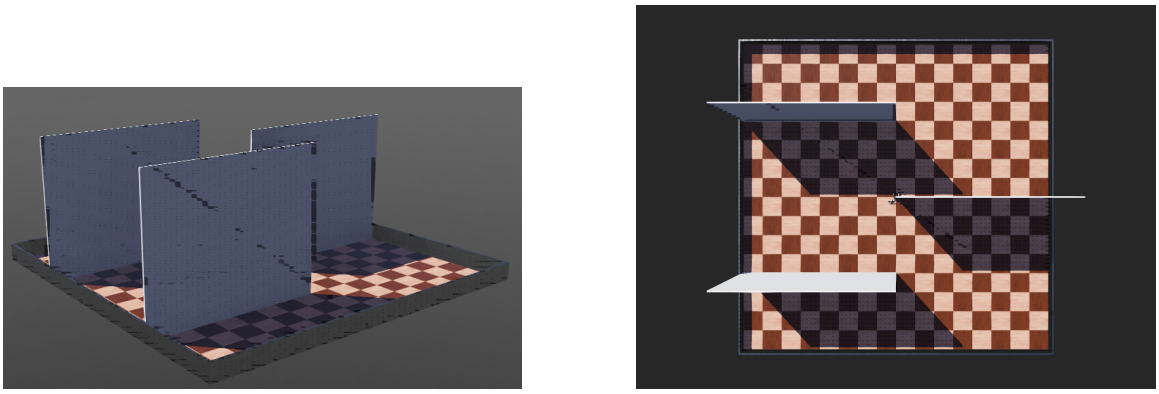
(f) Trayectoria total

Figura 90: Trayectoria resultante del AG para el mapa en Webots

En el escenario simulado de Webots, se logró cumplir el objetivo al implementar y validar el algoritmo genético en un entorno de simulación realista. El algoritmo demostró ser efectivo en la planificación de trayectorias y la evasión de obstáculos, utilizando un robot móvil que se desplazaba en un mapa con múltiples elementos estáticos. Los resultados de las simulaciones confirman que el algoritmo puede generar rutas seguras y eficientes, adecuadas para entornos complejos. Las iteraciones adicionales permitieron mejorar la calidad de las trayectorias, evidenciando cómo la correcta sintonización de los parámetros del AG (como la probabilidad de cruce y mutación) afecta la optimización de las rutas generadas. A continuación, se presentan diferentes escenarios con cambios en el mapa para validar si funcionamiento en diferentes condiciones.

10.2. Implementación de algoritmo genético en planificación de movimiento 3D con cuadricóptero

En esta validación se ha evaluado un algoritmo genético aplicado a la planificación de trayectorias tridimensionales en el entorno de simulación Webots, utilizando un dron Crazyflie como modelo de prueba. El escenario de simulación fue diseñado con paredes representadas como planos delimitantes, con el objetivo de emular un entorno controlado pero representativo de condiciones reales. La implementación del algoritmo buscó generar rutas óptimas que permitan al dron alcanzar un punto objetivo evitando colisiones con los obstáculos. El mapa de esta simulación se basó en un terreno de exploración, como el que se muestra en la Figura 91.



(a) Vista isométrica

(b) Vista desde la superficie

Figura 91: Mapa de simulación de movimiento en 3D con trayectoria generada por AG

La elección de un dron Crazyflie se justifica por su capacidad de maniobra en espacios 3D, así como por su relevancia en aplicaciones de navegación autónoma. Utilizando este modelo, Figura 92, también se identificó el tipo de controlador que se puede utilizar. Los controladores de este modelo consisten en un PID de orientación y de desplazamiento. Estos controladores están disponibles por defecto en la plataforma Webots, por lo que bastó con la unificación del código de control de movimiento y el AG generador de trayectoria.

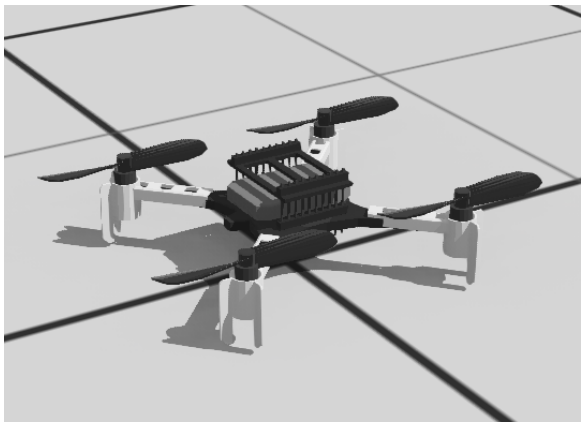


Figura 92: Robot Crazyflie utilizado para la simulación

10.2.1. Parámetros de simulación

Los parámetros del terreno de exploración se basaron en los buenos criterios desarrollados en las implementaciones anteriores. Al igual que con los parámetros de simulación, utilizando los mejores criterios demostrados, se optó por establecerlos como se muestran en el Cuadro 25. Estos parámetros se justifican de manera que al mantener un índice alto de población existe la posibilidad de ampliar la diversidad, y como se demostró, esta es necesaria en terrenos de alta complejidad. La longitud apropiada para esta implementación debe estar entre el rango estimado de 6 a 8, por ser un terreno con obstáculos considerables. Al igual que con la simulación anterior, los valores deben convertirse a dimensiones medibles, como se mostró en el Algoritmo 10.1.

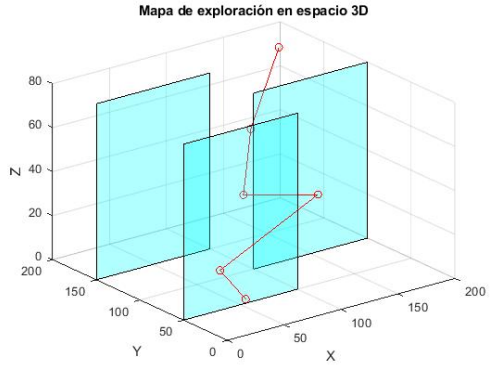
Parámetro	**Valor**
Dimensión del mapa	$2 \times 2 \times 1$ m
Punto de inicio	(-0.95, -0.95, 0.15)
Punto de meta	(0.95, 0.95, 0.7)
Población	100 ind.
Generaciones	50
Longitud del Cromosoma	6
Grosor de obstáculos	1 cm
Radio del Robot	6.5 cm
P_m	20 %

Cuadro 25: Parámetros de simulación de AG para búsqueda de trayectorias 3D

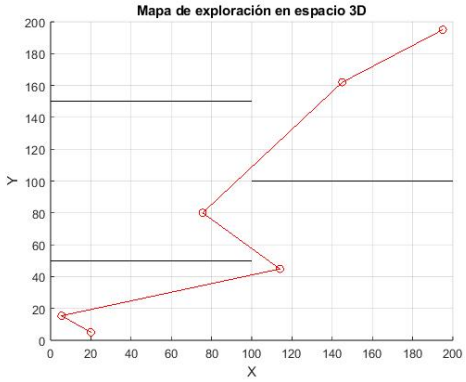
10.2.2. Resultados de la implementación del AG en simulación

En la simulación realizada en el entorno virtual de Webots, se verificaron los resultados correspondientes. Como primer punto, se determinó la mejor trayectoria libre de obstáculos por toda la arena considerado las restricciones de movimiento, como se muestra en la Figura 93. Luego esta solución se suavizó a varios puntos interpolables para que el controlador PID de posición pudiera tomar en secuencia estos puntos. De esta forma fue posible evaluar la

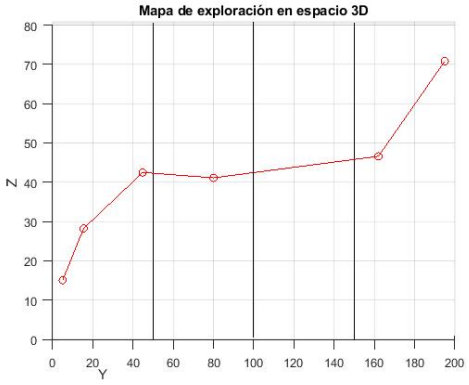
funcionalidad del algoritmo, con base en la calidad de la trayectoria. La verificación del seguimiento se realizó gracias los resultados gráficos mostrados en las Figuras 94 y 95 desde varias perspectivas. Los parámetros de rendimiento del AG se muestran el Cuadro 26.



(a) Vista isométrica



(b) Vista desde la superficie



(c) Vista horizontal

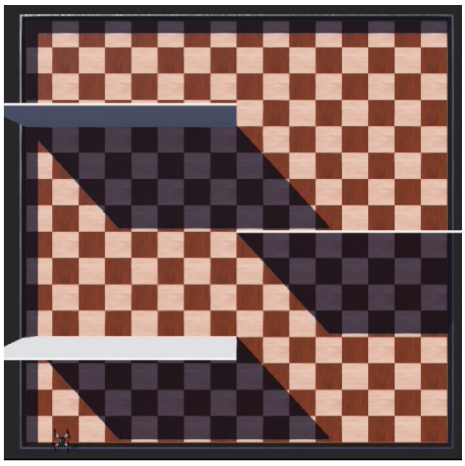
$$\mathbf{T} = \begin{bmatrix} -0.80 & -0.95 & 0.15 \\ -0.946 & -0.846 & 0.282 \\ 0.14 & -0.552 & 0.42 \\ -0.244 & -0.20 & 0.408 \\ 0.45 & 0.62 & 0.46 \\ 0.95 & 0.95 & 0.70 \end{bmatrix}$$

(d) Mejor cromosoma

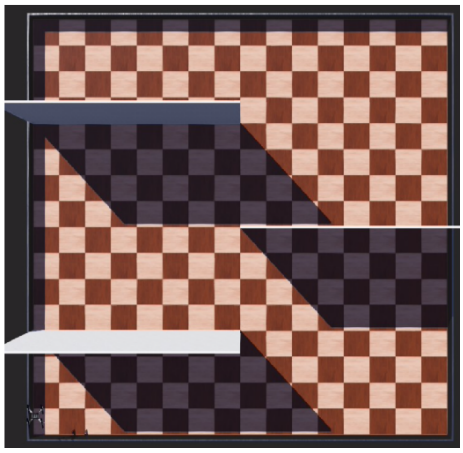
Figura 93: Trayectoria en 3D generada por el AG en Webots

Parámetro	**Valor**
Tiempo (s)	4.697
Aptitud	0.00076
Distancia (m)	13.16
Iteraciones	20

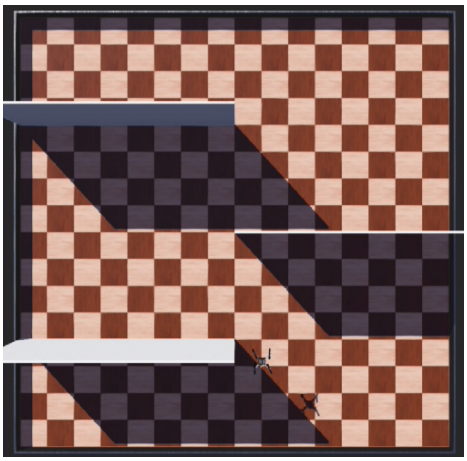
Cuadro 26: Parámetros de rendimiento del AG en 3D



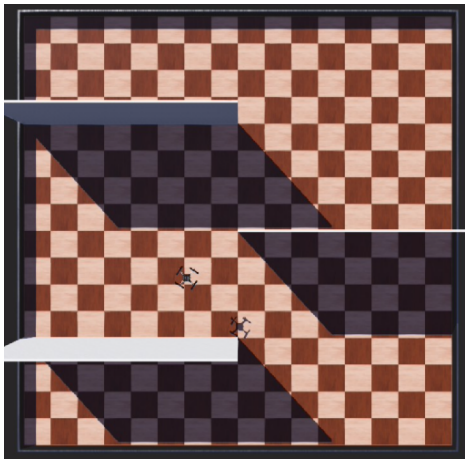
(a) Punto de inicio



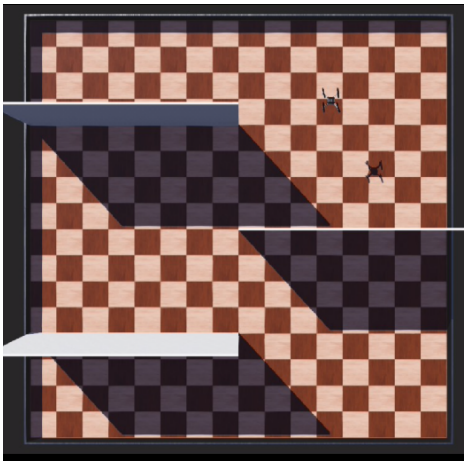
(b) Punto número 2



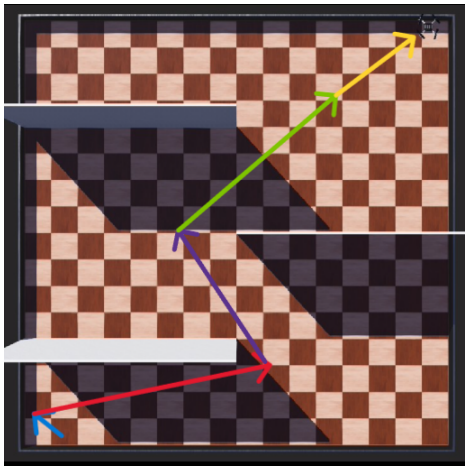
(c) Punto número 3



(d) Punto número 4

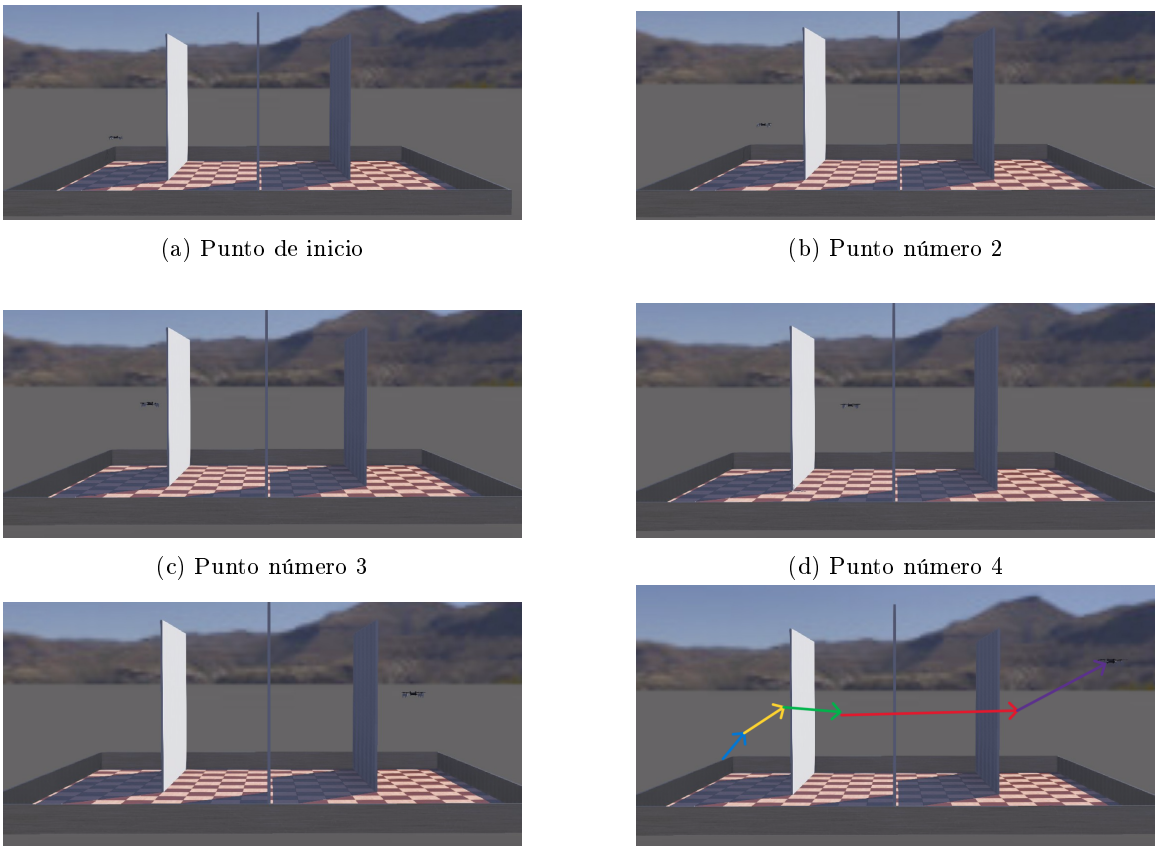


(e) Punto número 5



(f) Trayectoria al punto final

Figura 94: Trayectoria resultante del AG en 3D recorrida por Crazyflie



(a) Punto de inicio

(b) Punto número 2

(c) Punto número 3

(d) Punto número 4

(e) Punto número 5

(f) Trayectoria al punto final

Figura 95: Trayectoria resultante del AG en 3D recorrida por Crazyflie, vista en plano yx

La implementación del algoritmo genético (AG) en el simulador Webots permitió evaluar su desempeño en un entorno tridimensional realista, utilizando un dron Crazyflie para planificar y ejecutar una trayectoria libre de colisiones en presencia de tres paredes sólidas como restricciones. Los resultados (Tiempo: 4.696620 s, Aptitud: 0.0007581, Distancia: 13.19021, Generaciones: 20) reflejaron un balance adecuado entre eficiencia computacional y calidad de las soluciones generadas. La distancia recorrida validó la capacidad del algoritmo para optimizar trayectorias, mientras que el tiempo computacional evidencia su viabilidad para aplicaciones en tiempo real. Estas métricas y el éxito del dron al completar la trayectoria sin colisiones demuestran el cumplimiento del objetivo general, al implementar y evaluar un AG para resolver problemas complejos en robótica e Ingeniería Mecatrónica.

El uso de 20 generaciones y los parámetros optimizados definidos en escenarios previos resultaron suficientes para generar soluciones robustas, evitando redundancia computacional y asegurando una diversidad gráfica adecuada en las trayectorias generadas. La función de aptitud implementada, que penaliza colisiones y favorece trayectorias cortas, garantizó una convergencia eficiente hacia soluciones óptimas. Desde la perspectiva del objetivo planteado al inicio, esta simulación valida el uso de herramientas realistas para demostrar la eficacia de algoritmos genéticos en entornos tridimensionales. Los resultados confirman que los AG son herramientas adaptables y eficientes para resolver problemas de planificación de movimiento en robótica, consolidando su aplicabilidad en escenarios prácticos más allá del entorno

simulado.

10.3. Implementación de un algoritmo genético para la segmentación de objetos en un entorno simulado

En esta simulación se implementó el algoritmo genético desarrollado previamente, Sección 9.2, orientado a la segmentación y seguimiento de un objeto mediante un sensor de cámara integrado en un robot móvil. La simulación constó de un mapa de terreno plano, un robot Pololu Pi y una pelota de color naranja, utilizada como objeto de referencia. Este escenario se diseñó con el objetivo de validar la capacidad del AG para realizar segmentaciones precisas y extraer características relevantes del objeto, como su centroide, para facilitar el seguimiento por parte del robot.

La segmentación se realizó utilizando una imagen de referencia previamente cargada de la pelota, Figura 96, que permitió al AG identificarla como una sola región homogénea en las imágenes capturadas por la cámara del robot. Los parámetros del AG fueron configurados para segmentar en dos regiones: la pelota y el fondo del entorno. Esta decisión se justificó con base en la simplicidad del escenario, donde el objetivo principal era aislar el objeto de interés para su localización y seguimiento, como se muestra en la Figura 97. El AG procesó cada imagen capturada en tiempo real, determinando la posición del centroide de la pelota, segmentada mediante las coordenadas de su región.

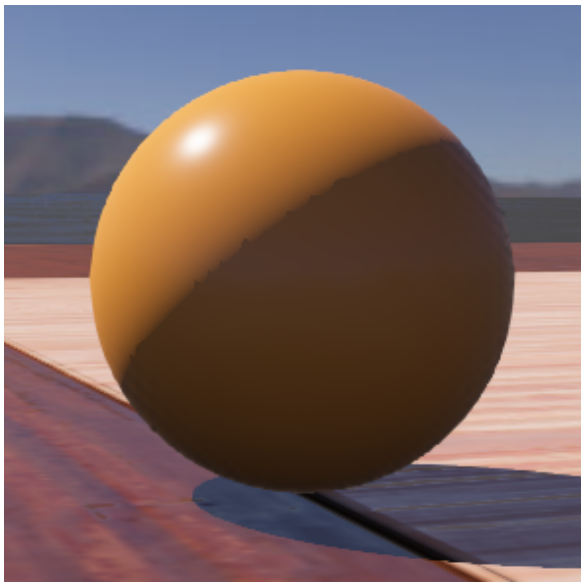


Figura 96: Objeto esférico a segmentar por el AG

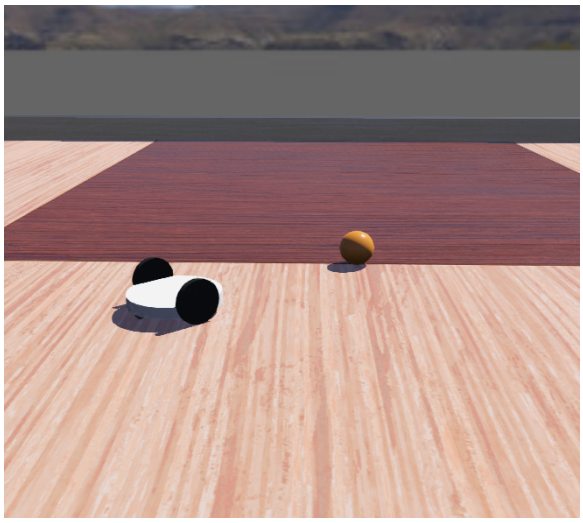


Figura 97: Conjunto de simulación

El seguimiento del objeto se basó en la posición del centroide calculado, que se transmitió al controlador del robot Pololu Pi para ajustar su movimiento mediante el controlador de pose no lineal. Este enfoque permitió al robot mantener la pelota dentro de su campo de visión y desplazarse hacia ella de manera continua. La utilización del AG en este escenario destacó por su capacidad para segmentar la pelota de manera precisa, incluso en condiciones de variaciones de iluminación o cambios en la orientación del objeto, validando su eficacia y robustez en tareas de procesamiento de imágenes integradas a sistemas robóticos.

10.3.1. Parámetros de simulación

Los parámetros elegidos cumplen con los criterios establecidos por la Sección 9.2, donde se evidenció la calidad en implementación. Con un número de individuos alto, baja cantidad de generaciones, detención al cumplimiento de las generaciones, y probabilidad de mutación del 20 %. El análisis se centra a dos regiones como subconjuntos de búsqueda. Los parámetros de simulación se describen a continuación, en el Cuadro 27.

Parámetro	**Valor**
Tamaño de la población	30 individuos
Número de generaciones	150
Tasa de mutación	15 %
Número de regiones	2 (pelota y fondo)
Imagen de referencia	Imagen de la pelota
Resolución de la cámara	640x480 píxeles
Frecuencia de captura de imágenes	30 fps

Cuadro 27: Parámetros de simulación de AG en escenario para segmentación de imágenes

10.3.2. Controlador de pose

La configuración del controlador de pose lineal, consistió en utilizar las coordenadas del centroide de la imagen para medir el error entre las referencias. De tal forma que el robot siga su orientación. Los parámetros del controlador utilizados se describen el Cuadro 28.

Parámetro	Valor
k_v	1.1
k_w	0.05

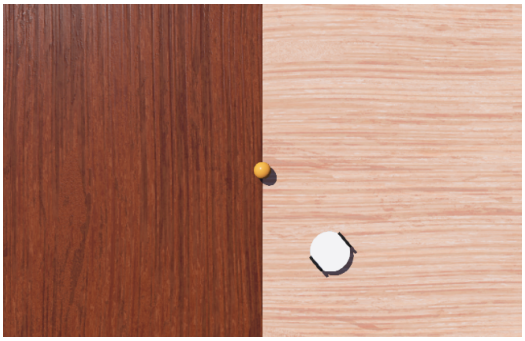
Cuadro 28: Valores de las constantes del controlador de pose lineal en segmentación de imágenes

10.3.3. Resultados de la implementación del AG en simulación

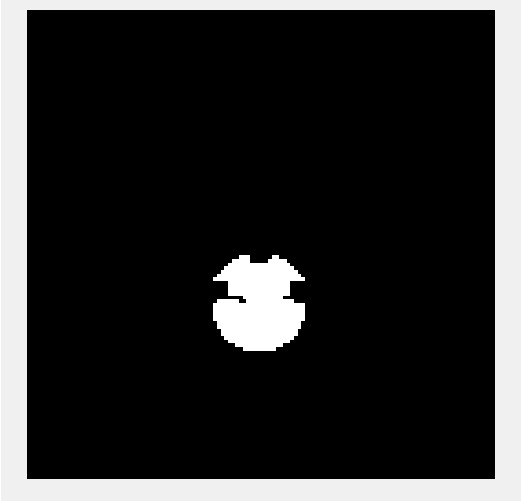
Los resultados obtenidos al implementar el algoritmo genético (AG) en la simulación de seguimiento de objetos mostraron que el método logró segmentar eficazmente la pelota naranja en las imágenes capturadas por la cámara del robot Pololu Pi. La mejor solución encontrada por el AG correspondió a los umbrales $t_1 = 0.1279$ y $t_2 = 0.9576$, con los cuales se segmentaron dos regiones: la pelota como una región homogénea y el fondo del entorno. Este resultado se tradujo en una entropía intrarregión de 11.9, lo que refleja la homogeneidad de la región segmentada, y una entropía interregión de 4.81, indicando una diferenciación efectiva entre la pelota y el fondo, como se muestra en la Figura 98.

Figura 98: Imagen segmentada a una sola región por AG

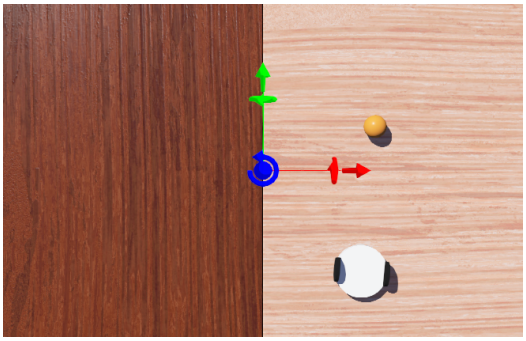
El tiempo total de ejecución del algoritmo para procesar las imágenes capturadas fue de 12.4555 segundos, lo que permitió al robot operar en tiempo real. La efectividad de la segmentación se verificó al observar que el robot logró identificar y seguir la pelota con precisión en cuatro posiciones diferentes dentro del entorno, manteniéndola dentro de su campo visual y ajustando su trayectoria en cada caso, como se muestra en la secuencia de imágenes generadas por el controlador 99.



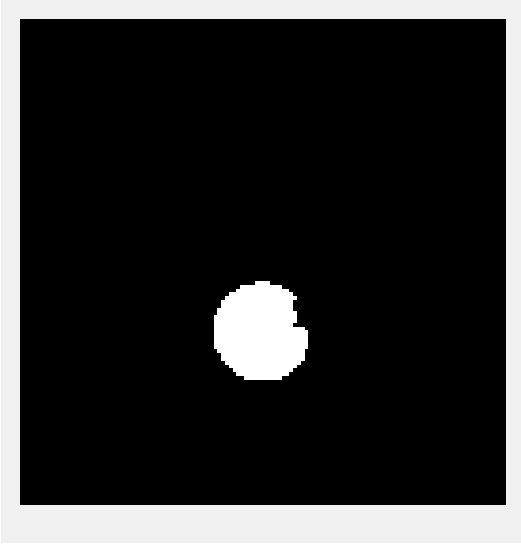
(a) Orientación del robot



(b) Fotograma del objeto segmentado



(c) Orientación del robot



(d) Fotograma del objeto segmentado

Figura 99: Imagen segmentada en la simulación conforma a las poses del robot

La implementación desarrollada y evaluada cumplió de manera satisfactoria con el objetivo planteado, que establece el desarrollo de escenarios prácticos y la validación de algoritmos genéticos mediante simulaciones realistas. Este experimento presentó un entorno dinámico y práctico, donde el AG no solo segmentó correctamente el objeto bajo condiciones variables, sino que también fue capaz de generar información clave para el sistema de control del robot.

Los resultados experimentales demostraron que el AG es una herramienta robusta para aplicaciones en tiempo real en sistemas robóticos, incluso en escenarios controlados. La precisión en la segmentación, medida a través de las métricas de entropía, y el éxito del seguimiento en diferentes posiciones reflejan la adaptabilidad del AG para integrar visión por computadora con control autónomo. Esto posiciona al algoritmo como una alternativa viable y efectiva para aplicaciones de procesamiento de imágenes en robótica móvil, alineándose con el objetivo general de la investigación y ampliando el alcance de los algoritmos genéticos

en la Ingeniería Mecatrónica.

CAPÍTULO 11

Conclusiones

- Los resultados de las simulaciones en la optimización de funciones de costo demostraron que los algoritmos genéticos son capaces de explorar eficazmente el espacio de búsqueda y converger hacia soluciones óptimas. Esto también resaltó la versatilidad del algoritmo para ajustarse a diferentes tipos de problemas de optimización, cumpliendo de manera satisfactoria con los requerimientos de eficiencia y precisión.
- Las pruebas realizadas en superficies como la hipérbola, el paraboloide y la campana de Gauss demostraron la adaptabilidad de los algoritmos genéticos para identificar máximos y mínimos en escenarios complejos. De esta forma, se logró evaluar el algoritmo investigado dentro del área de inteligencia computacional.
- Los resultados de la implementación del AG en un entorno 2D demostraron una mejora continua, en cuanto a distancia, en la planificación de trayectorias libres de obstáculos. En las simulaciones, se validaron los criterios y parámetros adecuados para implementar el algoritmo en esta área de aplicación. Con esto se evaluó y demostró que estos algoritmos pueden implementarse en el área de robótica.
- Se determinó que el AG es efectivo en la búsqueda de soluciones óptimas para entornos complejos, cumpliendo con el objetivo de generar trayectorias eficientes y seguras, mientras evita obstáculos en diferentes configuraciones de mapas.
- Se logró implementar el AG en otras áreas de la Ingeniería Mecatrónica, que demostraron ser eficientes para planificación de movimiento en 3D y para procesamiento de imágenes.
- Con base en los resultados, el AG implementado demostró su efectividad al segmentar con precisión las regiones en imágenes complejas. Los resultados obtenidos validaron su aplicación práctica en la Ingeniería Mecatrónica.
- La implementación del algoritmo genético en el entorno de simulación Webots demostró ser efectiva en la planificación de trayectorias y la evasión de obstáculos, tanto para 2D y 3D. Al igual que en la implementación de segmentación de imágenes para aplicaciones de visión por computadora.

- Las simulaciones confirmaron que la correcta sintonización de parámetros, como la probabilidad de cruce y mutación, juega un papel crucial en la calidad de las trayectorias generadas, lo que resalta mantener un criterio adecuado en el uso de los parámetros del AG.
- Los resultados en las simulaciones demostraron poder aplicar los AG en varios escenarios prácticos donde se validó su implementación. Estos resultados dieron a conocer los criterios apropiados que mejoran la eficiencia de los AGs en resolución de problemas de búsqueda y optimización.

CAPÍTULO 12

Recomendaciones

- En [5] se presenta que, para calcular la distancia y colisión con obstáculos, se debe usar la ecuación de la recta y un punto. Es mejor utilizar proyecciones vectoriales para medir la distancia entre la partícula y el obstáculo. Existen otros métodos además de este que pueden ser implementados.
- Para mejorar el rendimiento computacional se recomienda en próximas implementaciones generar computación paralela. Esto con el fin de reducir los tiempos de búsqueda.
- Hacer un análisis más profundo del número de individuos y parámetros en general, dependiendo de la aplicación.
- Debido a que el AG en la implementación de procesamiento de imágenes toma un alto tiempo de ejecución, es recomendable realizar la comparación con otros lenguajes de programación. Estos programas podrían ser Python, C+ u otros más eficientes.
- En este trabajo, el método de selección en la implementación de trayectorias 2D y 3D, se basó en selección por comparación. Se recomienda modificar este método para futuras implementaciones y usar una alternativa que sea por torneo. Este brindará más variabilidad en las generaciones.
- Para reducir el tiempo computacional aún más, se recomienda que la generar la población aleatoria, se pruebe acotar el rango del valor del gen dentro del cromosomas hasta demostrar mejores soluciones.

CAPÍTULO 13

Bibliografía

[1] A. P. Engelbrech, "Introduction to Computational Intelligence," en *Computational Intelligence: An introduction*. South Africa: John Wiley & Sons, Ltd, 2007, págs. 3-5.

[2] G. Iriarte, "Aprendizaje automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[3] D. M. Baldizón, "Aplicaciones prácticas para algoritmos de inteligencia y robótica de enjambre," Tesis de licenciatura, Universidad del Valle de Guatemala, 2022.

[4] J. Menéndez, "Validación de los algoritmos de robótica de enjambre *Particle Swarm Optimization* y *Ant Colony Optimization* con sistemas robóticos fisicos en ecosistema Robotat," Tesis de licenciatura, Universidad del Valle de Guatemala, 2023.

[5] N. S. Utami, A. Jazidie y R. E. A. Kadier, "Path Planning for Differential Drive Mobile Robot to Avoid Static Obstacles Collision using Modified Crossover Genetic Algorithm," en *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2019, págs. 282-287. DOI: 10.1109/ISITIA.2019.8937184.

[6] A. S. Akopov y M. A. Hevencev, "A Multi-agent Genetic Algorithm for Multi-objective Optimization," en *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 2013, págs. 1391-1395. DOI: 10.1109/SMC.2013.240.

[7] M. Lin, J. Xiaoming y Q. Fei, "A Robot Obstacle Avoidance Method Based on Improved Genetic Algorithm," en *2018 11th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2018, págs. 327-331. DOI: 10.1109/ICICTA.2018.00081.

[8] K. Rodríguez. "Cómputo evolutivo." Consultado el: 10 de marzo de 2024, Coursera. (2018), dirección: https://www.coursera.org/learn/computo-evolutivo.

[9] M. Gestal, *Introducción a los Algoritmos Genéticos*, LaTeX2HTML translator Version 2002-2 (1.70), Generado con LaTeX2HTML por Nikos Drakos y Ross Moore, 2006. dirección: http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/node22.html.

[10] B. Siciliano y L. Sciavicco, "Robotics modelling, Planning and Control," en London: Springer Verlag, 2010, págs. 140-190.

[11] T. W. Ian J Griffinths Qasim H Mehdi y N. Gough, "A Genethic algorithm For Path Planning," en *IFAC Intelligent Components and Instruments for Control Applications*, 1997.

[12] A. Sheta, M. S. Braik y S. Aljahdali, "Genetic Algorithms: A tool for image segmentation," en *2012 International Conference on Multimedia Computing and Systems*, 2012, págs. 84-90. DOI: 10.1109/ICMCS.2012.6320144.

[13] A. Singh y R. K. Gupta, "Entropy-based image thresholding: A new perspective," *Signal Processing*, vol. 104, págs. 64-67, 2014.

[14] MathWorks, *What Is Image Segmentation?* Último acceso: noviembre 18, 2024, 2023. dirección: https://www.mathworks.com/discovery/image-segmentation.html.

[15] S. Surjanovic y D. Bingham, *Optimization Test Functions and Datasets*, https://www.sfu.ca/~ssurjano/optimization.html, Simon Fraser University, 2013.

[16] W. Mora, *Cálculo de Varias Variables*, 1era. Escuela de Matemática, Instituto Tecnológico de Costa Rica, 2017, Revista Digital Matemática, Educación e Internet. dirección: http://tecdigital.tec.ac.cr/revistamatematica/.

[17] H. Choset, K. M. Lynch, S. Hutchinson et al., *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[18] S. I. Grossman, *Álgebra Lineal*, 8.ª ed. Ciudad de México: Cengage Learning, 2008, Traducción al español de la obra original en inglés.

[19] C.-H. Chang y C.-Y. Liu, "Image segmentation using region growing and entropy thresholding," *Pattern Recognition Letters*, vol. 32, n.º 7, págs. 797-803, 2011.

[20] C. Ltd., *Webots User Guide*, Webots R2023b Documentation, 2023. dirección: https://cyberbotics.com/doc/guide/tutorials.

[21] T. Dubail, *Brain tumors 256x256*, Accessed: YYYY-MM-DD, 2024. dirección: https://www.kaggle.com/datasets/thomasdubail/brain-tumors-256x256.

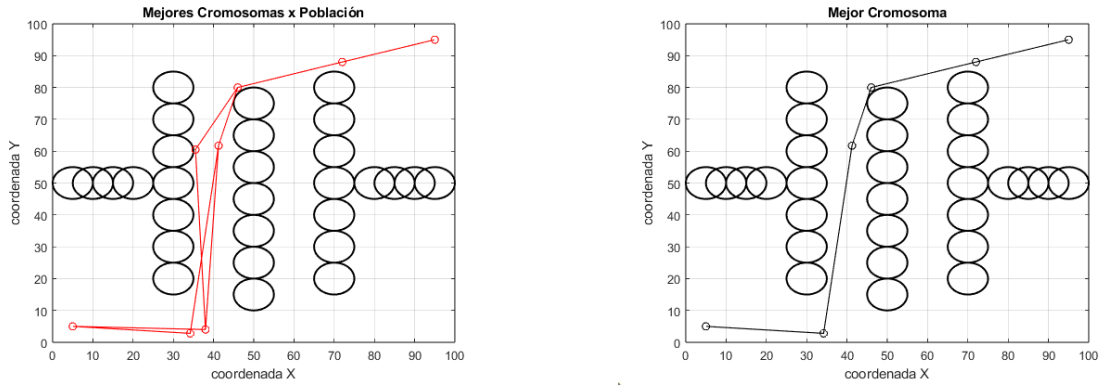
[22] I. Bombonato, *Body Parts X-Ray Images in PNG*, Accessed: YYYY-MM-DD, 2024. dirección: https://www.kaggle.com/datasets/ibombonato/xray-body-images-in-png-unifesp-competion.

CAPÍTULO 14

Anexos

14.1. Evaluaciones adicionales de AG en planificación de trayectorias 2D

En la siguiente Figura 100 se muestran los resultados de una implementación con los parámetros de simulación: tamaño 100×100, longitud del cromosoma de 6 puntos, población 100 individuos, inicio $p_0(5,5)$ y final $p_0(95,95)$. Los resultados se resumen a: tiempo de 5.89 seg., aptitud de 0.0061, distancia de 161 unidades y 9 generaciones.

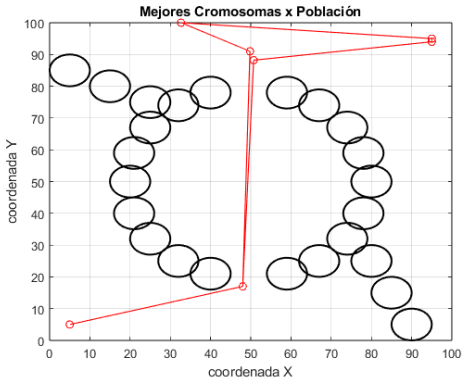


(a) Mapa con mejores cromosomas

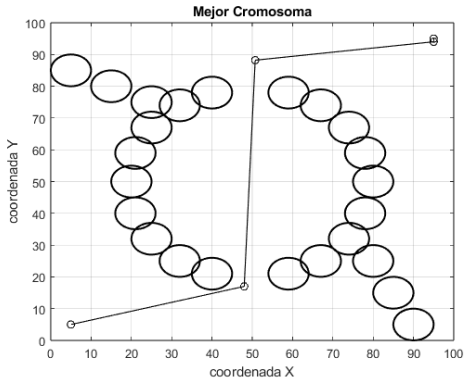
(b) Mapa con el mejor cromosoma seleccionado

Figura 100: Visualización de las trayectorias generadas por los algoritmos genéticos

En la Figura 101, se muestran los resultados de una implementación usando un terreno más complejo con los parámetros de simulación: tamaño 100×100, longitud del cromosoma de 5 puntos, población 125 individuos, inicio $p_i(95,95)$ y final $p_f(5,5)$. Los resultados se resumen a: tiempo de 5.15 seg., aptitud de 0.006087, distancia de 158.2 unidades y 6 generaciones.



(a) Mapa con mejores cromosomas



(b) Mapa con el mejor cromosoma seleccionado

Figura 101: Visualización de las trayectorias generadas por los algoritmos genéticos

14.2. Códigos adicionales

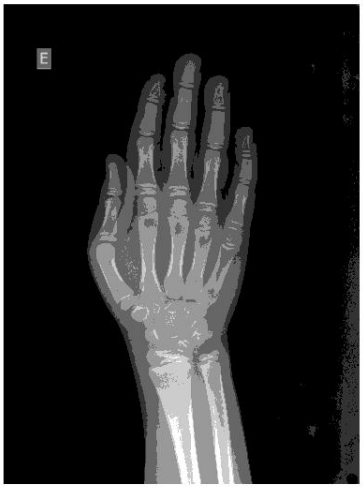
Algoritmo 14.1: Definición de los centros de los obstáculos

```
obstacleCenter = [10,50; 15,50; 20,50; 25,50; 30,50; 35,50;...
40,50; 45,50; 50,50; 55,50; 60,50; 65,50;...
40,50; 45,50; 50,50; 55,50; 60,50; 65,50;...
70,50; 75,50; 80,50; 85,50; 90,50; 95,50;...
100, 100; 105,100; 110,100; 115,100; 120,100;...
125, 100; 130,100; 135,100; 140,100; 145,100;...
150, 100; 155,100; 160,100; 165,100; 170,100;...
175, 100; 180,100; 185,100; 190,100; 195,100;...
10,170; 15,170; 20,170; 25,170; 30,170; 35,170;...
40,170; 45,170; 50,170; 55,170; 60,170; 65,170;...
40,170; 45,170; 50,170; 55,170; 60,170; 65,170;...
70,170; 75,170; 80,170; 85,170; 90,170; 95,170; 5,50;];
```

14.3. Evaluaciones adicionales de AG en segmentación de imágenes

En esta sección se encuentran pruebas adicionales realizadas con el AG para segmentar imágenes médicas en varias regiones de búsqueda, como rayos-x y resonancias magnéticas. Las imágenes se obtienen del banco generado por los autores en [21] y [22].

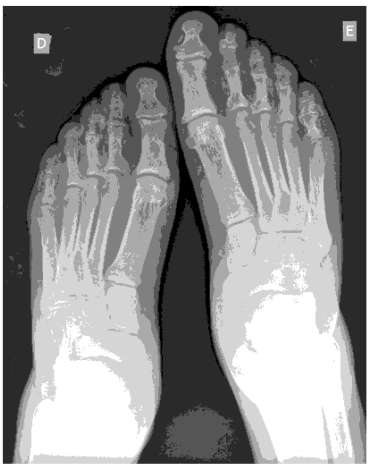
(a) Radiografía original



(b) Segmentación resultante de AG con 5 regiones de referencia

Figura 102: Segmentación de imagen usando población de 100 individuos y $P_m = 20\,\%$

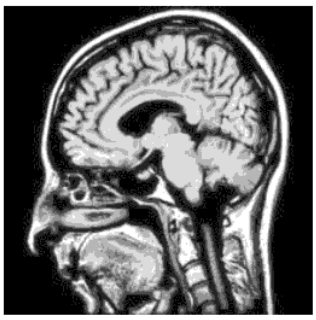
(a) Radiografía original



(b) Segmentación resultante del AG con 6 regiones de referencia

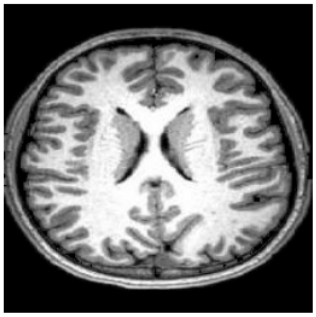
Figura 103: Segmentación de imagen usando población de 80 individuos y $P_m = 20\,\%$

(a) Resonancia magnética original



(b) Segmentación resultante del AG con 7 regiones de referencia

Figura 104: Segmentación de imagen usando población de 150 individuos y $P_m = 20\,\%$



(a) Resonancia magnética original

(b) Segmentación resultante del AG con 7 regiones de referencia

Figura 105: Segmentación de imagen usando población de 70 individuos y $P_m = 30\,\%$

14.4. Repositorio en Github

El código fuente y los archivos relacionados con este trabajo de graduación se encuentran disponibles en el siguiente repositorio público de GitHub:

- **Repositorio de GitHub:** `https://github.com/JosePetion/Trabajo-de-Graduacion`

Este repositorio incluye el código desarrollado, la documentación técnica y los ejemplos de implementación descritos en este documento.