Optimización multiobjetivo de trayectorias para robots móviles asistida por un gemelo digital

por

Néstor Andrés García Rojas

Tesis sometida como requisito parcial

para obtener el grado de

MAESTRÍA EN CIENCIAS EN EL ÁREA DE CIENCIAS COMPUTACIONALES

por el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Agosto, 2024

Santa María de Tonantzintla, Puebla

Dirigida por:

Dr. Saúl Zapotecas Martínez

Dra. Raquel Díaz Hernández

Índice general

Índice de figuras

Agradecimientos

Quiero agradecer a mis padres, dos personas que siempre me han brindado su apoyo de todas las formas posibles durante cada una de las etapas de mi vida.

Un agradecimiento muy especial a mi asesor, el Dr. Saúl Zapotecas Martínez, por la oportunidad brindada de trabajar con él, por todo su tiempo, motivación, guía, consejos y conocimiento otorgados a lo largo de la realización de este trabajo de tesis. Un agradecimiento especial a mi co-asesora, la Dra. Raquel Díaz Hernández, por su tiempo, guía y consejos otorgados a lo largo de este trabajo.

Extiendo mi agradecimiento a mis maestros y sinodales, el Dr. Leopoldo Altamirano Robles, el Dr. José Martínez Carranza y la Dra. Hayde Peregrina Barreto, por aceptar evaluar mi trabajo, por su tiempo y sus valiosos comentarios que ayudaron a mejorar este trabajo.

Gracias al Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), por la oportunidad brindada de ser parte de esta institución, por los servicios y por permitirme hacer uso de sus instalaciones y equipo para que fuera posible la realización de esta tesis.

También quiero agradecer a mis amigos y compañeros de maestría por los aportes y conocimientos que me compartieron ya sea en el ámbito personal como académico.

Por último, quisiera agradecer al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por la beca otorgada, la cual fue un importante apoyo durante mi estancia en este programa de maestría.

Resumen

La planificación de trayectorias enfrenta múltiples desafíos debido a la creciente complejidad de los entornos y las demandas de los usuarios. Entre los principales retos se encuentra la dificultad de los ambientes, la optimización multiobjetivo y la interacción con humanos y otros robots. Este trabajo presenta una metodología para la optimización multiobjetivo de trayectorias de un robot móvil, utilizando un gemelo digital que asiste en el proceso de optimización. La metodología propuesta aborda cinco objetivos principales: 1) minimizar la longitud de la trayectoria, 2) optimizar la suavidad de la trayectoria, 3) minimizar el consumo energético, 4) optimizar el torque de los motores y 5) minimizar el tiempo de viaje, todo ello cumpliendo con la necesidad de evasión de obstáculos. En este estudio, se comparan tres algoritmos evolutivos multiobjetivo basados en los principios de optimalidad de Pareto, indicadores y descomposición. Estos algoritmos se emplean para optimizar dos de los cinco objetivos: 1) minimizar la distancia de la trayectoria y 2) mejorar la suavidad de la trayectoria, con la restricción de evasión de obstáculos. El gemelo digital se utiliza para optimizar los objetivos restantes: consumo de energía, torque de los motores y tiempo de ejecución. Los resultados demuestran que la metodología propuesta es una herramienta eficaz para la planificación de trayectorias. La integración del gemelo digital permite encontrar soluciones que equilibran múltiples objetivos sin añadir complejidad a los algoritmos evolutivos multiobjetivo. Este enfoque mejora la eficiencia operativa y permite una planificación de trayectorias más precisa y adaptable a diferentes entornos, lo que lo hace beneficioso para una amplia gama de aplicaciones.

Abstract

Path planning faces multiple challenges due to the growing complexity of environments and user demands. Among the main challenges are dynamic and unknown environments, multi-objective optimization, and interaction with humans and other robots. This thesis presents a methodology for multi-objective optimization of trajectories of a mobile robot, using a digital twin to assist in the optimization process. The proposed methodology addresses five main objectives: 1) minimize trajectory length, 2) optimize trajectory smoothness, 3) minimize energy consumption, 4) optimize motor torque, and 5) minimize travel time, all while meeting the need for obstacle avoidance. In this study, three multi-objective evolutionary algorithms based on Pareto optimality, indicator, and decomposition principles are compared. These algorithms are used to optimize two of the five objectives: 1) minimize trajectory distance and 2) improve trajectory smoothness, with the constraint of obstacle avoidance. The digital twin is used to optimize the remaining objectives: energy consumption, motor torque, and execution time. The results show that the proposed methodology is an effective tool for path planning. The integration of the digital twin allows finding solutions that balance multiple objectives without adding complexity to the multi-objective evolutionary algorithms. This approach improves operational efficiency and enables more accurate and adaptable path planning in different environments, making it beneficial for a wide range of applications.

Capítulo 1

Introducción

En los últimos años, la planificación de trayectorias ha emergido como un área de investigación de gran relevancia. Este campo se centra en determinar la ruta más eficiente para desplazarse desde una posición inicial hasta una posición deseada, optimizando tanto el tiempo como los recursos utilizados. No obstante, el proceso de planificación de trayectorias enfrenta numerosos desafíos, siendo uno de los principales la necesidad de evitar colisiones con los diversos obstáculos presentes en el entorno. Para superar estos retos, se han desarrollado una variedad de algoritmos que permiten realizar esta tarea de manera autónoma, sin necesidad de intervención humana.

El diseño de estos algoritmos implica la creación de métodos eficientes y robustos para resolver problemas complejos, permitiendo su aplicación en entornos altamente desafiantes. La relevancia actual de la planificación de trayectorias se debe a su amplio abanico de aplicaciones, que abarca desde la industria y el sector automotriz hasta la manufactura y la medicina. Por ejemplo, en la industria automotriz, los vehículos autónomos utilizan algoritmos de planificación de trayectorias para navegar de manera segura y eficiente por las carreteras. En la manufactura, los robots emplean estos algoritmos para optimizar sus movimientos en las líneas de producción. En el ámbito médico, la planificación de

trayectorias es fundamental para el desarrollo de procedimientos quirúrgicos asistidos por robots, mejorando la precisión y reduciendo el riesgo de errores.

En la planificación de trayectorias, los entornos pueden clasificarse como estáticos o dinámicos. En los entornos estáticos, los obstáculos se mantienen en posiciones fijas, mientras que en los entornos dinámicos, los obstáculos pueden moverse de manera aleatoria. La presencia de esta característica dinámica en un entorno añade complejidad, aunque también es posible encontrar entornos estáticos que presenten un nivel de complejidad considerable. En el proceso de búsqueda de trayectorias, es importante considerar diversos objetivos que pueden variar según lo que se desee optimizar. Estos objetivos pueden incluir la velocidad máxima del robot, restricciones en los movimientos, minimización del tiempo de la trayectoria, reducción de la energía utilizada, entre otros. Es fundamental tener en cuenta estos diferentes objetivos para lograr una planificación de trayectorias efectiva y eficiente.

La incorporación de múltiples objetivos a optimizar plantea un desafío en la planificación de trayectorias, ya que el uso de algoritmos tradicionales se vuelve inviable. Esto se debe a que, al agregar estos objetivos, el tarea se convierte en un problema optimización multiobjetivo. La optimización multiobjetivo de trayectorias es un campo fascinante que se encuentra en la intersección de la ingeniería, las matemáticas y la inteligencia computacional. En la resolución de problemas complejos de planificación de trayectorias, es común enfrentarse a múltiples objetivos en conflicto, como minimizar el tiempo de viaje, reducir el consumo de energía y evitar obstáculos. Por lo tanto, este desafío multidimensional requiere enfoques innovadores para encontrar soluciones que equilibren eficientemente estos objetivos que comúnmente están en conflicto. En este contexto, la optimización multiobjetivo emerge como una herramienta esencial, permitiendo la exploración de soluciones que conforman el denominado “frente de Pareto”, donde ninguna mejora en un objetivo es posible sin afectar negativamente a otro.

En esta área del conocimiento se centra no solo en alcanzar soluciones eficientes, sino también en proporcionar a los diseñadores y planificadores una comprensión profunda de los compromisos inherentes a la toma de decisiones en el diseño de trayectorias. Por lo que en esta investigación, se explora los fundamentos, desafíos y aplicaciones de la optimización multiobjetivo de trayectorias, destacando su importancia en la resolución de diferentes problemas en diversos escenarios.

1.1. Antecedentes

La planificación de trayectorias en robótica ha experimentado una evolución significativa a lo largo de los años, con varios hitos importantes en su desarrollo. Estos hitos han surgido de diversas disciplinas y han llevado a la creación de algoritmos cada vez más sofisticados y eficientes. A continuación, se presentan algunos de los hitos más destacados de la planificación de trayectorias para robots.

Algoritmos de planificación de movimientos tempranos. Desde los primeros días de la robótica, se han desarrollado algoritmos para planificar los movimientos de los robots. Estos algoritmos abarcan desde métodos simples, como el control de trayectorias lineales, hasta algoritmos más complejos basados en la cinemática y dinámica del robot. Estos primeros algoritmos sentaron las bases para el desarrollo posterior de técnicas más avanzadas.

Planificación de trayectorias basada en campos de potencial. En la década de 1980, se popularizó un enfoque que utiliza campos de potencial para guiar a los robots hacia su objetivo mientras evitan obstáculos. Los robots se mueven siguiendo gradientes de potencial, lo que les permite sortear obstáculos de manera natural. Esta técnica ha demostrado ser efectiva en entornos estáticos y ha sentado las bases para enfoques más complejos.

Algoritmos de búsqueda de caminos en grafos. La planificación de trayectorias se puede modelar como un problema de búsqueda de caminos en un grafo, donde los nodos representan posiciones en el espacio y las aristas representan posibles movimientos entre posiciones. Algoritmos como A* [4] y Dijkstra [5] se utilizan comúnmente para encontrar la ruta óptima en este tipo de grafos. Estos algoritmos permiten la planificación de trayectorias en entornos más complejos y dinámicos.

Algoritmos de planificación de movimientos probabilísticos. Los algoritmos probabilísticos, como RRT (Rapidly-exploring Random Trees) [6] y PRM (Probabilistic Roadmaps), han ganado popularidad en la planificación de trayectorias en entornos complejos. Estos algoritmos generan trayectorias válidas de manera eficiente mediante el muestreo aleatorio del espacio de configuración del robot. Estos enfoques han demostrado ser efectivos para la planificación de trayectorias en entornos desconocidos y con obstáculos dinámicos.

Planificación de trayectorias en tiempo real. Con la creciente demanda de robots capaces de planificar y ejecutar movimientos en tiempo real, se han desarrollado algoritmos que pueden generar trayectorias seguras y eficientes en entornos dinámicos y desconocidos. Estos algoritmos utilizan técnicas como la predicción de movimientos de objetos y la adaptación en tiempo real para generar trayectorias óptimas y seguras. Esto ha permitido a los robots realizar tareas en entornos cambiantes de manera eficiente y segura, un ejemplo sería la integración de la tecnología ORB- SLAM [7] con algoritmos de planificación de trayectorias como lo puede ser A*, RRT o algoritmos de optimización multiobjetivo, esto se logra a partir de que ORB-SLAM proporciona los datos de localización y mapeo, los cuales pueden ser utilizados como entrada para los algoritmos mencionados.

Por lo tanto, la planificación de trayectorias en robótica ha evolucionado desde algoritmos simples hasta técnicas más sofisticadas y adaptativas. Estos hitos en el desarrollo de algoritmos de planificación de trayectorias han permitido a los robots moverse de manera

eficiente y segura en una variedad de entornos, desde entornos estáticos hasta entornos dinámicos y desconocidos.

1.2. Descripcion del problema

La navegación en entornos complejos presenta desafíos significativos, especialmente cuando se introducen restricciones adicionales como límites de recursos y ventanas de tiempo. Estas restricciones no solo aumentan la complejidad del problema, sino que también dificultan la aplicación efectiva de algoritmos de navegación tradicionales, que suelen estar diseñados para condiciones menos restrictivas y más predecibles.

En este trabajo, abordamos este desafío complejo mediante un enfoque evolutivo multiobjetivo. Este enfoque nos permite considerar y optimizar múltiples objetivos de manera simultánea, lo que es crucial en escenarios donde los objetivos pueden ser conflictivos o interdependientes. Por ejemplo, en un entorno de navegación, los objetivos pueden incluir minimizar el tiempo de viaje, reducir el consumo de recursos, y cumplir con restricciones de ventanas de tiempo, entre otros.

Para mejorar aún más el proceso de optimización y asegurar que las soluciones propuestas sean viables y efectivas en el mundo real, integramos un gemelo digital en el sistema. Un gemelo digital es una réplica virtual de un sistema físico que permite simular, analizar y optimizar el comportamiento del sistema real en diversas condiciones operativas [8]. La integración de un gemelo digital proporciona una plataforma robusta para realizar pruebas y ajustes antes de implementar soluciones en el entorno real, lo que resulta en resultados más precisos y confiables.

El uso de un enfoque evolutivo multiobjetivo junto con un gemelo digital no solo mejora la precisión y la confiabilidad de los resultados, sino que también permite una mayor flexibilidad y adaptabilidad en la solución de problemas complejos de navegación. Este método es especialmente útil en industrias donde las condiciones operativas pueden cambiar

rápidamente y donde la capacidad de adaptarse a nuevas restricciones y objetivos es crucial para el éxito.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo principal de esta investigación es diseñar y desarrollar una metodología para optimizar la navegación de robots móviles autónomos. Esta metodología empleará un enfoque multiobjetivo e integrará un gemelo digital para asistir en el proceso de optimización, mejorando así la eficiencia y precisión de la navegación en diversos entornos.

1.3.2. Objetivos específicos

- Diseñar una metodología que utilice optimización multiobjetivo para mejorar las trayectorias de un robot móvil autónomo.
- Desarrollar un gemelo digital basado en las características clave del robot móvil AutoNOMOS para facilitar el proceso de optimización.
- Validar la efectividad de los algoritmos evolutivos multiobjetivo empleados en la metodología propuesta mediante indicadores de rendimiento para identificar el más adecuado para resolver este tipo de problema.
- Realizar una comparación entre la metodología propuesta y las técnicas tradicionales de navegación, evaluando sus ventajas y desventajas.

1.4. Contribuciones

Este estudio presenta una metodología innovadora para optimizar trayectorias en entornos complejos mediante un enfoque multiobjetivo, respaldado por la implementación

de un gemelo digital. En contraste con la optimización convencional centrada en una única métrica, como la distancia o el tiempo, nuestro enfoque considera simultáneamente varios objetivos, como la seguridad de la ruta, la distancia, la suavidad del recorrido, el consumo energético, el torque y el tiempo de llegada. Para lograr esto, se desarrolló un gemelo digital que modela el robot y parte de sus capacidades, permitiendo simular diferentes trayectorias y evaluar su desempeño en tiempo real, lo que nos permite realizar la optimización de tres objetivos que serían el torque, el tiempo de llegada y el consumo de energía. El gemelo digital se actualiza con el tiempo que se necesite para realizar la trayectoria, lo que mejora la precisión de las predicciones y la efectividad de las decisiones de planificación. Además, implementamos tres algoritmos de optimización multiobjetivo que nos permiten optimizar tres de los objetivos que en este caso son la distancia de la trayectoria, la suavidad y la evasión de obstáculos. Este enfoque permite a los robots adaptarse a condiciones cambiantes en tiempo real, como la presencia de obstáculos o cambios en la disponibilidad de energía. Los experimentos demuestran que la metodología supera a los métodos tradicionales de optimización de trayectorias, logrando trayectorias más eficientes y seguras en entornos simulados. Consideramos que esta investigación tiene aplicaciones significativas en campos como la robótica móvil, la logística y la planificación de transporte, donde la optimización de trayectorias es crucial para el rendimiento y la seguridad de los sistemas autónomos.

En síntesis, las contribuciones principales de este trabajo de tesis son las siguientes:

- **Metodología multiobjetivo para optimizar las trayectorias de un robot móvil**: Se ha desarrollado una metodología que adopta un enfoque multiobjetivo para optimizar las trayectorias de un robot móvil. Esto implica considerar múltiples objetivos, como la eficiencia en la navegación, la seguridad y la minimización del consumo de energía. Esta metodología proporciona una forma más completa y eficiente de optimizar las trayectorias del robot.

- **Desarrollo de un gemelo digital basado en las características físicas del robot móvil AutoNOMOS**: Se ha creado un gemelo digital que se basa en las

características físicas del robot móvil AutoNOMOS. Este gemelo digital permite simular y probar diferentes escenarios y configuraciones antes de implementarlos en el robot real. Esto facilita el proceso de optimización, ya que se pueden realizar ajustes y mejoras virtuales antes de aplicarlos en el mundo real.

- **Estudio comparativo de algoritmos evolutivos multiobjetivo**: Se ha realizado un estudio comparativo de diferentes algoritmos evolutivos multiobjetivo para investigar cuál de ellos ofrece los mejores resultados en el problema planteado. Esto permite identificar los algoritmos más prometedores y brinda información valiosa para futuras investigaciones en el campo de la optimización de trayectorias de robots móviles.

- **Comparación de la metodología propuesta con técnicas tradicionales de navegación**: Se ha llevado a cabo una comparación entre la metodología propuesta y las técnicas tradicionales de navegación. Esto permite evaluar la efectividad y eficiencia de la metodología propuesta en comparación con las técnicas existentes. Los resultados de esta comparación pueden ayudar a determinar si la metodología propuesta representa una mejora significativa en la navegación de robots móviles.

Publicaciones. Durante el desarrollo de esta tesis, se redactó el artículo de investigación titulado "*Exploring Multi-objective Evolutionary Approaches for Path Planning of Autonomous Mobile Robots*" que fue aceptado para su publicación en el congreso "*IEEE Congress on Evolutionary Computation*" (IEEE CEC).

1.5. Limitaciones

En este trabajo, se presentan algunas limitaciones que es importante tener en cuenta. Una de ellas es la complejidad computacional. La optimización multiobjetivo implica encontrar un conjunto de soluciones óptimas en lugar de una única solución, lo que puede aumentar significativamente la complejidad computacional. Esto es especialmente cierto en entornos

con múltiples restricciones y objetivos conflictivos. Es importante tener en cuenta esta complejidad al implementar la metodología propuesta.

Otra limitación es la dificultad para encontrar el equilibrio entre los objetivos. En problemas con múltiples objetivos, puede ser complicado lograr un equilibrio óptimo entre ellos. Mejorar un objetivo puede implicar un empeoramiento en otro. Por lo tanto, es crucial encontrar un equilibrio adecuado y considerar cuidadosamente las implicaciones de cada decisión.

Además, se destaca la imposibilidad de aplicar la metodología en un robot real debido a diferentes dificultades como el tiempo y que el robot no estaba disponible para su uso. Sin embargo, se compensó esta limitación utilizando las dificultades de las trayectorias simuladas. Aunque no se implementó la metodología en un robot físico, se trabajó en encontrar alternativas para simular y evaluar los resultados.

Estas limitaciones resaltan los desafíos asociados con la implementación efectiva de la metodología propuesta. Es importante abordar estos desafíos de manera integral y considerar cuidadosamente las implicaciones de cada decisión. Solo así podremos lograr resultados óptimos y superar las dificultades que puedan surgir en el camino.

1.6. Estructura de la tesis

Esta tesis se encuentra dividida en seis capítulos. En el Capítulo 2 se presentan los conceptos básicos relacionados con la planeación de trayectorias, algoritmos tradicionales para la planeación de trayectorias, el cómputo evolutivo, la optimización y los gemelos digitales.

En el Capítulo 3 presentamos el trabajo relacionado, los trabajos que se presentan hablan de la planeación de trayectorias con enfoque multiobjetivo y sobre el uso de gemelos digitales en la planeación de trayectorias.

En el Capítulo 4 se describe cómo realizamos la metodología que se propone, hablamos de la realización del gemelo digital y de los algoritmo que se utilizaron para esta investigación

que son MOEA/D, SMS-EMOA Y NSGA-II, además de otros algoritmos que sirven para poder realizar una comparación.

Posteriormente en el Capítulo 5 se presentan los resultados que se obtuvieron en los experimentos que se realizaron, se evaluaron los algoritmos multiobjetivo, se presenta la optimización realizada por el gemelo digital y se realiza una comparación de los resultados obtenidos con diferentes algoritmos.

Finalmente, en el Capítulo 6 se presentan las conclusiones obtenidas durante el desarrollo de la tesis, asi como también el trabajo a futuro derivado de los resultados obtenidos.

Capítulo 2

Marco teórico

En este capítulo se ofrece una introducción a los temas relacionados con el problema abordado en este trabajo. Comenzaremos con una explicación sobre la planificación de trayectorias, seguida de los métodos convencionales utilizados en robótica para dicho propósito. Posteriormente, exploraremos el concepto de cómputo evolutivo y su relación con la optimización en general. También se abordarán los conceptos de optimización mono-objetivo y multiobjetivo, y finalizaremos con una descripción de los gemelos digitales.

2.1. Planificación de trayectorias

La planificación de trayectorias, conocida como "path planning" en inglés, es un concepto fundamental en el campo de la robótica y la ciencia de la computación. Se refiere al proceso mediante el cual un agente, como un robot móvil, determina la ruta más óptima o eficiente para llegar de un punto de inicio a un punto de destino, teniendo en cuenta la presencia de obstáculos y cumpliendo con ciertos criterios o restricciones definidos [9].

En otras palabras, la planificación de trayectorias implica la generación de una secuencia continua de movimientos que un agente, como un robot móvil, puede seguir para alcanzar un objetivo específico. Este proceso es especialmente relevante en entornos complejos donde

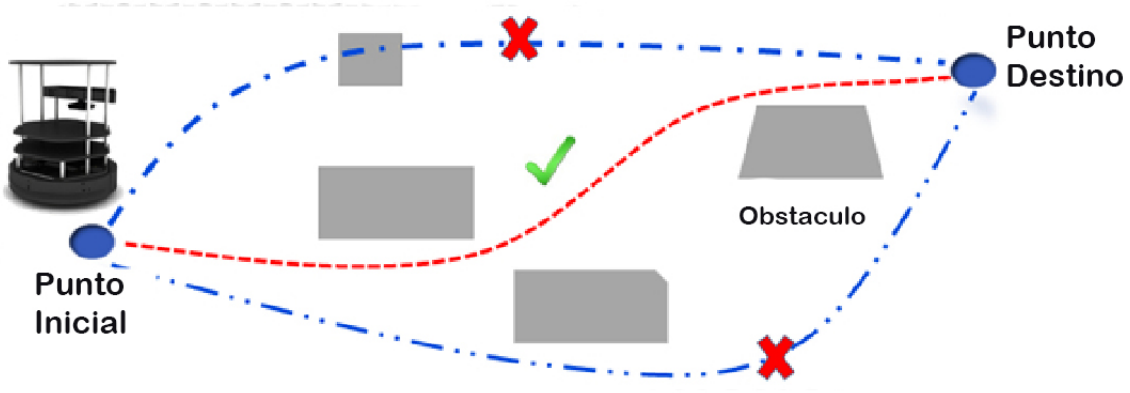

Figura 2.1: Ejemplo de planificación de trayectorias

existen obstáculos y se aplican limitaciones al movimiento del agente, como velocidad y capacidad de giro.

Existen diversos algoritmos y enfoques para la planificación de trayectorias, y la elección de uno de ellos depende del entorno y los requisitos del sistema en particular. Algunos métodos comunes incluyen el algoritmo de Dijkstra, el algoritmo A*, y RRT (Rapidly Exploring Random Trees). Estos algoritmos utilizan estrategias diferentes para explorar el espacio de búsqueda y encontrar la ruta más eficiente [10].

La planificación de trayectorias es fundamental en aplicaciones como la robótica móvil, vehículos autónomos, drones y otros sistemas automatizados que requieren navegar de forma autónoma en entornos dinámicos y desconocidos. A continuación, se mencionan algunos de los componentes clave del modelado de trayectorias.

- **Posición y Orientación**: El modelado de trayectorias comienza por describir la posición inicial del objeto y su orientación en un espacio tridimensional. La posición se refiere a la ubicación del objeto, mientras que la orientación indica la dirección en la que está orientado.

- **Parámetros Temporales**: Las trayectorias se modelan en función del tiempo. Esto implica definir cómo cambian la posición y la orientación del objeto a medida que

transcurre el tiempo. Pueden ser funciones matemáticas que describan la evolución continua o una serie discreta de puntos que representan la posición del objeto en momentos específicos.

- **Velocidad y Aceleración**: Además de la posición y la orientación, el modelado de trayectorias también puede incorporar la velocidad y la aceleración del objeto. Estos parámetros son cruciales para determinar cómo se mueve el objeto a lo largo de su trayectoria.
- **Curvas y Splines**: Las trayectorias pueden describirse mediante diversas curvas matemáticas, como líneas rectas, curvas cuadráticas, curvas cúbicas o splines. La elección de la curva dependerá de la aplicación específica y de los requisitos de suavidad o complejidad de la trayectoria.
- **Restricciones y Obstáculos**: En entornos del mundo real, el modelado de trayectorias debe tener en cuenta restricciones y obstáculos. Estas restricciones podrían incluir limitaciones de velocidad, capacidad de giro o la necesidad de evitar obstáculos en el entorno.
- **Planificación de Movimientos**: En aplicaciones como robótica y vehículos autónomos, la planificación de movimientos se basa en el modelado de trayectorias. Esto implica determinar la ruta óptima para que el objeto alcance su destino, evitando obstáculos y respetando restricciones específicas.

2.2. Métodos tradicionales para la planificación de trayectorias

Los algoritmos tradicionales para la planificación de trayectorias se pueden dividir en tres grupos: búsqueda de gráficos, basados en muestreo y algoritmos de curvas interpolantes. Estos se describen a continuacion.

2.2.1. Busqueda en grafos

Estos algoritmos exploran un grafo que representa el espacio de estados y las transiciones entre ellos. Utilizan estrategias como la búsqueda en anchura, la búsqueda en profundidad o la búsqueda mejorada para encontrar un camino desde el estado inicial al estado objetivo. Hay diversos algoritmos de búsqueda de gráficos. Entre los más populares se encuentran los algoritmos de Dijkstra [5] y A* [4].

Algoritmo de Dijkstra. El algoritmo de Dijkstra [5] es uno de los primeros algoritmos óptimos basados en la técnica de búsqueda mejorada para encontrar los caminos más cortos entre nodos en un grafo.

Los pasos del algoritmo de Dijkstra son los siguientes:

1. Convertir la red de caminos en un grafo, se espera encontrar las distancias entre nodos en el grafo mediante exploración.

2. Seleccionar el nodo no visitado con la distancia más baja desde el nodo fuente.

3. Calcular la distancia desde el nodo seleccionado a cada vecino no visitado y actualizar la distancia de todos los nodos vecinos si la distancia al nodo seleccionado es menor que la distancia anterior.

4. Marcar el nodo visitado cuando se haya realizado el cálculo de la distancia a todos los vecinos.

Los pasos anteriores se repiten hasta que se encuentre la distancia más corta entre el origen y el destino. En el Algoritmo 1 se describe el pseudocódigo del algoritmo de Dijkstra.

Algoritmo A*.

El algoritmo A* [4] se basa en una búsqueda mejorada que utiliza una función heurística para encontrar el camino más corto estimando el costo total. El algoritmo difiere del

Algorithm 1 Algoritmo de Dijkstra

Require: Un gráfico G
Require: Un nodo fuente s
Ensure: Los caminos más cortos de s a todos los demás nodos de G A

```
1: for all v ∈ V[G] do
2:     d[v] ← +∞
3:     Previo[v] ← indefinido
4: end for
5: d[s] ← 0
6: S ← conjunto vacio
7: Q ← V[G]
8: loop
9:     Q no es un conjunto vacío
10:    u ← Extract_Min(Q)
11:    S ← S ∪ {u}
12:    for all nodo (u, v) saliente de u do
13:        if d[u] + w(u, v) < d[v] then
14:            d[v] ← d[u] + w(u, v)
15:            previous[v] := u
16:        end if
17:    end for
18: end loop
```

algoritmo de Dijkstra en la estimación del costo del camino. La estimación del costo de un nodo i en un grafo dado por A* es la siguiente:

1. Estimar la distancia entre el nodo inicial y el nodo i.

2. Encontrar el vecino más cercano j del nodo i, y estimar la distancia entre los nodos i y j.

3. Estimar la distancia entre el nodo j y el nodo objetivo.

El costo total estimado es la suma de estos tres factores.

$$C_i = c_{start,i} + min_j \left(d_{i,j} + d_{j,goal}\right) \tag{2.1}$$

De la Ecuación 2.1 tenemos que C_i representa el costo total estimado del nodo i, $c_{start,i}$ el costo estimado desde el origen hasta el nodo i, $d_{i,j}$ la distancia estimada desde el nodo i hasta su nodo más cercano j, y $d_{j,goal}$ la distancia estimada desde el nodo j hasta el nodo objetivo. En el Algoritmo 2 tenemos el pseudocódigo del algoritmo.

2.2.2. Basados en muestreo

Estos algoritmos generan muestras aleatorias del espacio de estados y utilizan estas muestras para construir un plan. Ejemplos de estos algoritmos incluyen el muestreo de árboles de expansión (RRT) y el muestreo de crecimiento de la región (PRM).

El algoritmo RRT [6] es más popular y ampliamente utilizado para fines comerciales e industriales. Este algoritmo, construye un árbol que intenta explorar el espacio de búsqueda de manera rápida y uniforme mediante una búsqueda aleatoria. En cada iteración del algoritmo, se intenta extender el árbol agregando nuevos vértices en la dirección de una configuración *qrand* seleccionada aleatoriamente. El Algoritmo 3 muestra el seudocódigo

Algorithm 2 Algoritmo A*

Require: graph = el gráfico a buscar, startNode = el nodo inicial de la ruta, targetNode = el nodo objetivo de la ruta

```
1: for nodo en graph do
2:     node.score ← infinito
3:     node.heuristicScore ← infinito
4:     node.visited ← false
5: end for
6: startNode.score ← 0
7: startNode.heuristicScore ← 0
8: while true do
9:     currentNode ← nodeWithLowestScore(graph)
10:    currentNode.visited ← true
11:    for nextNode en currentNode.neighbors do
12:        if not nextNode.visited then
13:            newScore ← calculateScore(currentNode, nextNode)
14:            if newScore < nextNode.score then
15:                nextNode.score ← newScore
16:                nextNode.heuristicScore ← newScore+calculateHeuristicScore(nextNode,targetNode)
17:                nextNode.routeToNode ← currentNode
18:            end if
19:        end if
20:    end for
21:    if currentNode = targetNode then return trayectoria
22:    end if
23: end while
```

del algoritmo RRT. Toma como entrada una configuración inicial q_0 y hace crecer un árbol $\mathcal{T}$ con raíz en q_0

Algorithm 3 RRT(q_0)

$\mathcal{T}$.Init(q_0)
for $i = 1$ to K **do**
  $q_{rand} \leftarrow$ Rand($\mathcal{C}$)
  $q_{near} \leftarrow$ Nearest($q_{rand}, \mathcal{T}$)
  Extend($\mathcal{T}, q_{near}, q_{rand}$)
end for

Aunque en este trabajo no se utiliza el algoritmo PRM este se utiliza normalmente en un escenario estático. Se divide en dos fases: la fase de aprendizaje y la fase de consulta. En la fase de aprendizaje, se construye y almacena como un grafo una hoja de ruta probabilística libre de colisiones. En la fase de consulta, se busca un camino que conecte los nodos originales y objetivo a partir de la hoja de ruta probabilística.

2.2.3. Algoritmos basados en interpolación de curvas

Se define como un proceso que construye o inserta un conjunto de reglas matemáticas para dibujar trayectorias. El algoritmo de curva interpolante se basa en técnicas (por ejemplo, diseño geométrico asistido por computadora (CAGD)) para dibujar un camino suave. Se utilizan reglas matemáticas para el suavizado de caminos y la generación de curvas. Estos algoritmos construyen una curva suave que conecta el estado inicial con el estado objetivo. Un ejemplo común es el método de spline cúbico, que encuentra una trayectoria suave a través de puntos de control dados.

2.3. Computo evolutivo

El cómputo evolutivo [11] es una rama de la inteligencia artificial que se inspira en la teoría de la evolución biológica para resolver problemas complejos y encontrar soluciones óptimas

en diversos dominios. Este enfoque se basa en algoritmos evolutivos que simulan procesos evolutivos, como la selección natural, la reproducción y la mutación, para evolucionar una población de posibles soluciones hacia soluciones cada vez mejores. A continuación, una descripción detallada de los elementos clave del cómputo evolutivo:

1. Representación de Individuos:

    - En el cómputo evolutivo, las posibles soluciones a un problema se representan como individuos en una población.
    - La forma de representación puede variar según el problema y puede ser binaria, real, entera, o una combinación de estas, dependiendo de la naturaleza del espacio de búsqueda.

2. Función de Aptitud (Fitness):

    - Cada individuo en la población tiene asociado un valor de aptitud que indica qué tan buena es su solución en relación con el problema que se está abordando.
    - La función de aptitud evalúa qué tan bien se ajusta una solución a los criterios deseados y proporciona la base para la selección natural.

3. Selección Natural:

    - Los individuos de la población son seleccionados para reproducirse con una probabilidad proporcional a su aptitud. Los individuos más aptos tienen más posibilidades de ser seleccionados.
    - Este proceso se asemeja a la selección natural en la evolución biológica, donde los organismos más aptos tienen una mayor probabilidad de transmitir sus genes a las generaciones futuras.

4. Reproducción y Cruce (Crossover):

- Los individuos seleccionados se reproducen para formar nuevas soluciones. Este proceso puede incluir operadores de cruces que combinan partes de las soluciones de los padres para producir descendencia.
- El cruce simula la recombinación genética en la biología, donde los genes de los padres se combinan para formar la descendencia.

5. Mutación:

   - Ocasionalmente, se aplican cambios aleatorios a los individuos, simulando mutaciones genéticas. La mutación introduce variabilidad en la población y permite explorar nuevas regiones del espacio de búsqueda.
   - Al igual que en la biología, la mutación puede introducir cambios que podrían ser beneficiosos, perjudiciales o neutrales.

6. Generaciones:

   - El proceso de selección, reproducción y mutación se repite a lo largo de varias generaciones. A medida que progresa, se espera que la población evolucione hacia soluciones cada vez más óptimas según la función de aptitud.

7. Convergencia:

   - Con el tiempo, la población tiende a converger hacia soluciones que son buenas aproximaciones a la solución óptima del problema.

Aplicaciones del Cómputo Evolutivo: El cómputo evolutivo se aplica en una variedad de áreas, como la optimización de funciones, el diseño de algoritmos, la Planificación de trayectorias, el aprendizaje automático evolutivo y la ingeniería evolutiva.

Por lo tanto, el cómputo evolutivo proporciona un enfoque poderoso y versátil para resolver problemas complejos, inspirado en los principios de la evolución biológica. Este

enfoque ha demostrado ser eficaz en la búsqueda de soluciones en espacios de búsqueda extensos y difíciles.

2.4. Optimizacion

La optimización [12] es un proceso fundamental que implica encontrar la mejor solución posible para un problema, teniendo en cuenta ciertos criterios o restricciones. En términos más simples, se trata de hacer que algo sea lo mejor posible o de maximizar o minimizar una función objetivo, sujeto a ciertas condiciones. A continuación una descripción detallada de los conceptos clave asociados con la optimización:

1. Problema de Optimización:
    - En términos generales, un problema de optimización se define mediante una función objetivo que se busca maximizar o minimizar. Esta función representa la medida de qué tan "buena.º "mala.es una solución en relación con los objetivos del problema.
    - Además de la función objetivo, puede haber restricciones que limitan las posibles soluciones. Estas restricciones deben tenerse en cuenta durante el proceso de optimización.
2. Soluciones y Espacio de Búsqueda:
    - Las soluciones posibles al problema de optimización se encuentran en un espacio de búsqueda. Este espacio contiene todas las combinaciones factibles de variables que definen una solución.
    - Cada punto en este espacio se asocia con un valor de la función objetivo y puede estar sujeto a restricciones específicas.
3. Máximo y Mínimo:

- Dependiendo del problema, se busca un máximo o mínimo para la función objetivo. En algunos casos, se busca maximizar la eficiencia o el rendimiento, mientras que en otros casos se busca minimizar costos o errores.

4. Algoritmos de Optimización:

   - Los algoritmos de optimización son procedimientos computacionales que buscan encontrar la mejor solución en el espacio de búsqueda. Algunos métodos comunes incluyen el descenso de gradiente, algoritmos evolutivos, optimización convexa, entre otros.
   - Estos algoritmos iterativamente ajustan las soluciones en función de la información proporcionada por la función objetivo y las restricciones.

5. Optimización Continua y Discreta:

   - La optimización continua se refiere a problemas donde las variables pueden tomar cualquier valor dentro de un rango continuo. La optimización discreta se aplica cuando las variables solo pueden tomar valores discretos, como números enteros.
   - Ejemplos de problemas continuos incluyen la optimización de funciones matemáticas, mientras que problemas de asignación de recursos o planificación pueden ser ejemplos de problemas discretos.

6. Optimización Multiobjetivo:

   - En algunos casos, hay múltiples objetivos a optimizar simultáneamente. Estos problemas se conocen como problemas de optimización multiobjetivo y pueden requerir encontrar soluciones que representen un equilibrio entre diferentes metas.

7. Aplicaciones de la Optimización:

- La optimización tiene aplicaciones en una amplia gama de campos, como la ingeniería, la economía, la logística, la ciencia de datos, el diseño de algoritmos y muchos otros.
- Ejemplos específicos incluyen la optimización de rutas de entrega, la programación de la producción, la asignación de recursos, el diseño de estructuras eficientes, entre otros.

Con lo que podemos concluir que la optimización es un proceso esencial para encontrar soluciones eficientes y efectivas en una variedad de contextos. Implica la búsqueda de la mejor configuración posible, considerando las restricciones y los objetivos definidos por el problema en cuestión.

2.5. Optimizacion mono-objetivo

La optimización mono-objetivo [13] se refiere a la resolución de problemas de optimización en los cuales hay un solo objetivo o criterio que se busca maximizar o minimizar. En este tipo de problemas, la tarea principal consiste en encontrar la mejor solución posible que optimice el valor de una función objetivo específica. A continuación una descripción detallada de los elementos clave relacionados con la optimización mono-objetivo:

1. Definición del Problema:
   - En un problema de optimización mono-objetivo, se establece un objetivo claro y específico que se busca maximizar o minimizar. Este objetivo se formula como una función matemática llamada función objetivo.
2. Función Objetivo:
   - La función objetivo es la expresión matemática que cuantifica el rendimiento, la eficiencia o cualquier otra métrica relevante del sistema que se está optimizando.

Puede ser una función que se busca maximizar (en problemas de maximización) o minimizar (en problemas de minimización).

3. Variables de Decisión:

   - Las variables de decisión son los parámetros o variables que afectan el valor de la función objetivo. Estas son las incógnitas que se ajustan durante el proceso de optimización para encontrar la mejor solución.

4. Espacio de Búsqueda:

   - El espacio de búsqueda consiste en todas las combinaciones posibles de valores para las variables de decisión. Cada punto en este espacio representa una posible solución al problema.

5. Restricciones:

   - Además de la función objetivo, puede haber restricciones que limitan las soluciones factibles. Estas restricciones deben cumplirse para que una solución sea considerada válida.

6. Optimización Continua:

   - La optimización mono-objetivo puede involucrar variables continuas, lo que significa que las soluciones pueden tomar cualquier valor dentro de un rango específico. Este tipo de optimización se enfrenta a problemas donde se busca una configuración óptima en un espacio continuo.

7. Algoritmos de Optimización:

   - Diversos algoritmos de optimización se utilizan para encontrar la mejor solución en el espacio de búsqueda. Estos algoritmos pueden incluir métodos analíticos, métodos heurísticos o algoritmos evolutivos, según la naturaleza del problema.

8. Convergencia:

    - En el proceso de optimización, se busca alcanzar la convergencia, donde la solución converge hacia el óptimo global o local. La convergencia implica que no hay cambios significativos en la solución después de cierto número de iteraciones.

9. Resultados y Evaluación:

    - Una vez que se ha encontrado una solución que cumple con los requisitos del problema, se evalúa y verifica su validez en función de la función objetivo y las restricciones establecidas.

2.5.1. Evolución Diferencial (ED)

Evolución diferencial es un algoritmo de búsqueda basado en poblaciones propuesto por Storn y Price en 1997 [14]. En este sentido, el algoritmo de evolución diferencial minimiza funciones de costo multimodales (con más de una solución).

En este tipo de algoritmos, la función de costo suele ser una combinación lineal de restricciones u objetivos ponderados. Como cualquier algoritmo evolutivo, la implementación de la evolución diferencial se basa en un procedimiento iterativo que emplea algunos operadores: cruce y mutación para modificar la población en cada generación. Además, un operador de selección se encarga de elegir a los individuos de la nueva población, según la comparación de la función de costo; de esa manera, solo los individuos mejor adaptados alcanzan la siguiente generación.

En el algoritmo de evolución diferencial, la población inicial se selecciona aleatoriamente y los valores de los genes se distribuyen en el espacio de búsqueda. Posteriormente, la población se modifica mediante los operadores genéticos en un proceso iterativo hasta que se alcanza el criterio de parada. El criterio de parada puede ser un umbral mínimo para la

función de costo o un número máximo de iteraciones.

Algorithm 4 Evolución Diferencial

```
procedure ED(N, g, f_k(x_k))
Evalúe la población inicial P de individuos aleatorios.
while no se cumple el criterio de parada do
    for cada individuo en Población do
        Cree un candidato c a partir del padre p elegido al azar.
        Evaluar al candidato.
        if el candidato domina al padre t then
            El candidato reemplaza al padre.
        else
            El candidato es descartado
        end if
    end for
end while
End procedure
```

El pseudocódigo anterior esboza el procedimiento del nalgoritmo de evolución diferencial, como una función del tamaño de la población N, el número de generaciones g y la función de costo que a su vez depende de los parámetros del circuito x.

El principio de funcionamiento es la generación de un cromosoma mutante para cada individuo mediante la adición de la diferencia ponderada entre dos individuos seleccionados aleatoriamente de la población a un tercero. Es decir, $y = x_1 + F(x_2 - x_3)$, donde F es un factor de escala y x_1, x_2, x_3 se seleccionan aleatoriamente de la población actual con la restricción de que sean diferentes entre sí.

En este algoritmo evolutivo, la operación de diferencia permite una exploración gradual del espacio de búsqueda. Después de la mutación, tiene lugar el operador de cruce. Mediante este operador, partes del individuo actual y el cromosoma creado por la mutación se combinan en uno nuevo; como consecuencia, se construye un vector de prueba de acuerdo

con la probabilidad de cruce (CR). La probabilidad de cruce es un parámetro de control que puede tomar un valor de 0 a 1. Después de eso, se realiza una comparación entre la función de costo de ambos individuos, el generado por la prueba y el actual; solo el individuo con el mejor valor de costo sobrevive a la siguiente iteración.

2.5.2. Algoritmo Genético (AG)

Los Algoritmos Genéticos (AG) han ganado una gran popularidad en el campo de la optimización y búsqueda, y su éxito se debe a varias razones. Fueron desarrollados por John Holland, un reconocido científico en el campo de la computación evolutiva [15].

Una de las principales ventajas de los Algoritmos Genéticos es que no requieren información adicional sobre el espacio de búsqueda, como el gradiente. A diferencia de otras técnicas de optimización, los AG utilizan un enfoque basado en la evolución biológica para encontrar soluciones óptimas. Esto significa que los AG simulan el proceso de selección natural, reproducción y mutación para mejorar gradualmente las soluciones a lo largo de múltiples generaciones.

Otra razón por la cual los Algoritmos Genéticos son exitosos en aplicaciones de optimización y búsqueda es su capacidad para explorar ampliamente el espacio de búsqueda. Debido a su naturaleza basada en la evolución, los AG pueden examinar una amplia variedad de soluciones potenciales y encontrar soluciones óptimas en problemas complejos con múltiples variables y restricciones.

Además, los Algoritmos Genéticos son altamente adaptables y pueden manejar problemas con soluciones subóptimas o no lineales. A través de la selección y la reproducción, los AG pueden mejorar gradualmente las soluciones y adaptarse a cambios en el entorno o en

los requisitos del problema.

Los AG son algoritmos de búsqueda altamente no lineales, lo que significa que su comportamiento puede ser muy difícil de predecir cuando se varían sus parámetros. Esto se debe a que los AG utilizan un enfoque basado en la evolución biológica, donde las soluciones se mejoran gradualmente a través de procesos de selección, reproducción y mutación.

Cuando se modifican los parámetros de un AG, como el tamaño de la población, la tasa de mutación o el número de generaciones, el comportamiento del algoritmo puede cambiar drásticamente. Pequeños ajustes en los parámetros pueden tener un impacto significativo en la exploración del espacio de búsqueda y en la calidad de las soluciones encontradas. En el Algoritmo 5 se muestra el pseudocódigo para los AG

Algorithm 5 Algoritmo Genético

```
Generar población inicial de individuos de forma aleatoria
Evaluar la aptitud de cada individuo
while no se cumple el criterio de parada do
    Realizar la selección de los padres
    Realizar cruza
    Realizar mutación
    Evaluar la aptitud de cada hijo creado
    Seleccionar los individuos que formarán la siguiente generación
end while
```

Aplicaciones de la Optimización mono-objetivo: La optimización mono-objetivo tiene aplicaciones en una amplia variedad de campos, como la ingeniería, la economía, la logística, la planificación de proyectos, la toma de decisiones y muchos otros.

Para este punto podemos concluir que la optimización mono-objetivo es la búsqueda de la mejor solución en términos de un solo criterio objetivo, lo que la hace adecuada para problemas donde se puede definir claramente un objetivo específico a optimizar.

2.6. Optimizacion multiobjetivo

La optimización multiobjetivo [13], también conocida como optimización multicriterio, se refiere a la resolución de problemas de optimización en los cuales hay múltiples objetivos que deben ser considerados simultáneamente. A diferencia de la optimización mono-objetivo, donde se busca optimizar un solo criterio, la optimización multiobjetivo implica la búsqueda de soluciones que representen un equilibrio entre varios objetivos que pueden estar en conflicto. A continuación una descripción detallada de los elementos clave relacionados con la optimización multiobjetivo:

1. Definición del Problema:
    - En un problema de optimización multiobjetivo, se establecen dos o más objetivos que se buscan simultáneamente optimizar. Estos objetivos pueden ser contradictorios o competitivos, y no siempre es posible mejorar uno sin afectar negativamente a los demás.
2. Funciones Objetivo:
    - En lugar de tener una única función objetivo como en la optimización mono-objetivo, en la optimización multiobjetivo hay varias funciones objetivo. Cada función objetivo representa un aspecto específico del rendimiento del sistema que se busca mejorar.
3. Variables de Decisión:
    - Al igual que en la optimización mono-objetivo, hay variables de decisión que afectan los valores de las funciones objetivo. Sin embargo, en este caso, estas variables deben ajustarse para encontrar soluciones que equilibren los diferentes objetivos.
4. Espacio de Búsqueda:

- El espacio de búsqueda sigue existiendo, pero en este caso, se busca encontrar soluciones que no solo cumplan con las restricciones del problema, sino que también optimicen múltiples funciones objetivo.

5. Restricciones:

   - Pueden existir restricciones adicionales, además de las funciones objetivo, que limitan las soluciones factibles.

6. Frente de Pareto:

   - Una solución es considerada óptima en el contexto de la optimización multiobjetivo si no existe otra solución que sea mejor en todos los objetivos. El conjunto de soluciones no dominadas forma el "Frente de Pareto", que representa todas las soluciones no superadas por ninguna otra en todos los objetivos.

7. Algoritmos de Optimización Multiobjetivo:

   - Se utilizan algoritmos específicos de optimización multiobjetivo para explorar y encontrar soluciones en el espacio de búsqueda. Estos algoritmos pueden incluir el algoritmo genético multiobjetivo (MOGA), el algoritmo de evolución diferencial multiobjetivo (MODE), entre otros.

8. Trade-offs:

   - Dado que los objetivos pueden estar en conflicto, las soluciones suelen representar trade-offs entre los diferentes criterios. Mejorar un objetivo puede implicar comprometer otro, y encontrar soluciones que equilibren estos trade-offs es un aspecto clave de la optimización multiobjetivo.

9. Resultados y Evaluación:

- La calidad de una solución en la optimización multiobjetivo se evalúa considerando su posición en el frente de Pareto y su capacidad para ofrecer un equilibrio entre los objetivos definidos. Esto puede implicar la selección de una solución final basada en criterios adicionales o en la preferencia del tomador de decisiones.

2.6.1. Conceptos básicos

Problema de Optimización. Los algoritmos evolutivos son comúnmente aplicados en la solución de problemas de optimización. Asumiendo problemas de minimización, el problema de optimización puede ser expresado de forma matemática mediante la siguiente ecuación:

$$\begin{aligned} \underset{\mathbf{x} \in \Omega}{\textbf{minimizar:}} \quad & \mathbf{F}(\mathbf{x}) \\ \textbf{subjeto a:} \quad & g_j(\mathbf{x}) \leq 0, \ \forall \ j \in \{1, \ldots, p\} \\ & h_k(\mathbf{x}) = 0, \ \forall \ k \in \{1, \ldots, q\} \end{aligned} \tag{2.2}$$

donde $\mathbf{x}$ es un vector multi-dimensional en $\mathbb{R}^n$ de n parámetros de decisión, $F(\mathbf{x})$ representa un vector de m funciones objetivo $(f_1(\mathbf{x}), ..., f_m(\mathbf{x}))$. $\Omega \subset \mathbb{R}^n$ es el conjunto de soluciones factibles definido por las p restricciones de desigualdad $g_j \ \forall j \in \{1, \ldots, p\}$ y las q restricciones de igualdad $h_j \ \forall j \in \{1, \ldots, q\}$. Cuando m es igual a uno, se dice que el problema de optimización es mono-objetivo, mientras que para $m > 1$ el problema de optimización se vuelve multiobjetivo.

Óptimo de Pareto. Cuando se comparan dos soluciones $\mathbf{x}_1$ y $\mathbf{x}_2$ surge la necesidad de definir un criterio de dominancia. En la optimización multiobjetivo el criterio de Pareto es el mas utilizado. Este criterio establece lo siguiente:

1. Se dice que un vector objetivo $\mathbf{x}_1$ domina a otro vector objetivo $\mathbf{x}_2$ (por ejemplo, $\mathbf{x}_1 < \mathbf{x}_2$) si ningún componente de $\mathbf{x}_1$ es mayor que el elemento correspondiente de $\mathbf{x}_1$ y al menos uno de sus componentes es mayor.

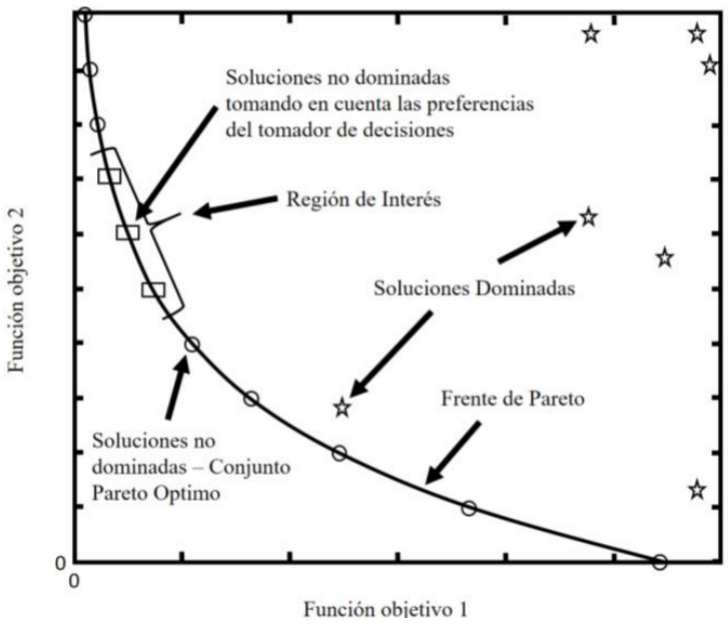

Figura 2.2: Frente de Pareto [1]

2. La solución $\mathbf{x}_1$ domina a $\mathbf{x}_2$, si $f(\mathbf{x}_1)$ domina a $f(\mathbf{x}_2)$

3. Todas las soluciones no dominadas son soluciones óptimas del problema, es decir, soluciones no dominadas por ninguna otra. El conjunto de estas soluciones es nombrado conjunto de Pareto.

Frente de Pareto. Para la obtención del frente de Pareto hay diferentes obstáculos que pueden convertir esta tarea en un problema complejo: espacio de búsqueda discontinuo, soluciones agrupadas en una misma región, o incluso la alta dimensión de las mismas.

Existen dos tipos de frentes de Pareto muy comunes que surgen al momento de resolver problemas multiobjetivo: frente convexo (ver Figura 2.2) y frente cóncavo. La forma del frente de Pareto indica la naturaleza de los compromisos existentes entre las distintas funciones objetivo.

Algorithm 6 Algoritmo NSGA-II

Require: $(\mathcal{N}', g, f_k(\mathbf{x}_k))$ ▷ $\mathcal{N}'$ los miembros evolucionan g generaciones para resolver $f_k(\mathbf{x})$

Población inicial $\mathbb{P}'$
Generar población aleatoria de tamaño $\mathcal{N}'$
Evaluar en la función objetivo
Ordenamiento basado en la dominancia de Pareto
Generar la descendencia de la población
  Selección de torneo binario
  Cruza y mutación
for $i = 1$ hasta g **do**
  for para cada padre e hijo de la población **do**
    Ordenamiento basado en la dominancia de Pareto
    Generar conjuntos de soluciones no-dominadas
    Determinar distancia de amontonamiento
    Bucle para agregar soluciones a la próxima generación desde el primer frente hasta encontrar los individuos $\mathcal{N}'$
  end for
  Elegir los puntos en el frente inferior con una distancia de amontonamiento alta
  Crear la siguiente generación
    Selección de torneo binario
    Cruza y mutación
end for

2.6.2. NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) es un algoritmo genético evolutivo utilizado para la optimización multiobjetivo. Fue desarrollado por Kalyanmoy Deb en 2002 [16] como una extensión del algoritmo NSGA original. NSGA-II aborda problemas de optimización que implican múltiples objetivos que deben ser optimizados simultáneamente, sin que uno se pueda mejorar a expensas de otro. Esto se conoce como un problema de optimización multiobjetivo. NSGA-II utiliza técnicas como el ordenamiento no dominado y el elitismo para evolucionar una población de soluciones hacia el conjunto de soluciones no dominadas, conocido como la "frente de Pareto", que representa las mejores soluciones posibles para los objetivos dados. En el Algoritmo 6 muestra el pseudocódigo del algoritmo NSGA-II.

2.6.3. MOEA/D

Algorithm 7 Algoritmo MOEA/D

Require: $\wedge = \{\lambda^{(1)}, ..., \lambda^{(\mu)}\}$ ▷ Vector de pesos

$t = 0$
Inicializa la población aleatoria P_0
Inicializa los vecinos $B(i)$ al recolectar k vectores de peso cercanos en $\wedge$ para cada $\lambda^{(i)}$
while cumpla criterio de parada **do**
    for $i \in \{1, ..., \mu\}$ **do**
        Seleccionar aleatoriamente dos soluciones $x^{(1)}, x^{(2)}$ en el vecindario de $B(i)$
        $y =$ Cruza de $x^{(1)}, x^{(2)}$ usando operadores genéticos
        $y' =$ Problema especifico local, se aplica una heurística de mejora o reparación
        if $g(y'|\lambda^{(i)}, z) \leq g(x^{(i)}|\lambda^{(i)}, z)$ **then**
            $x^{(i)} = y'$
        end if
        Remueve de P_t todos los vectores dominados por y'
        Agrega y' a P_t si ningún vector en P_t domina y'
    end for
    $t = t + 1$
end while

MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition) es un algoritmo evolutivo utilizado para resolver problemas de optimización multiobjetivo. Fue propuesto por Qingfu Zhang en 2007 [17] como una técnica eficaz para abordar problemas con múltiples objetivos conflictivos. La idea principal detrás de MOEA/D es descomponer el problema multiobjetivo en una serie de subproblemas más simples, cada uno de los cuales se optimiza de forma independiente. Estos subproblemas se resuelven utilizando un algoritmo evolutivo básico, como un algoritmo genético o un algoritmo de evolución diferencial. Luego, las soluciones obtenidas de cada subproblema se combinan para formar el conjunto final de soluciones no dominadas, que representan las mejores soluciones posibles para el problema multiobjetivo. El Algoritmo 7 muestra el pseudocódigo del algoritmo MOEA/D.

MOEA/D ha demostrado ser efectivo para abordar problemas complejos de optimización multiobjetivo y ha sido ampliamente utilizado en diversas áreas, como la ingeniería, la planificación urbana y la gestión de recursos.

2.6.4. SMS-EMOA

Algorithm 8 Algoritmo SMS-EMOA

1: $t = 0$
2: $P_0 = inicializa()$ ▷ Inicializa la población aleatoria de μ individuos
3: **while** $i = 1$ cumpla criterio de parada g **do**
4: $q_{t+1} = genera(P_t)$ ▷ Genera descendencia por variación
5: $Q_t = P_t \cup q_{t+1}$ ▷ Selecciona μ mejores individuos
6: $\{\mathcal{R}_1, ..., \mathcal{R}_v\} = ordedamientoRapidoNo\text{-}dominado(Q_t)$ ▷ Todos los v frentes de Q_t
7: $r = \arg\min s \in \mathcal{R}_v[\Delta_l(s, \mathcal{R}_v)]$
8: $P_{t+1} = (Q_t \setminus \{r\})$
9: $t = t + 1$
10: **end while**

SMS-EMOA (S-Metric Selection Evolutionary Multi-Objective Algorithm) es un algoritmo evolutivo utilizado para la optimización de problemas multiobjetivo. Fue propuesto por Eckart Zitzler en 2007 [18] como una mejora del algoritmo EMOA (Evolutionary Multiobjective Optimization Algorithm).

La principal característica de SMS-EMOA es su capacidad para explorar y explotar múltiples soluciones óptimas de manera secuencial. Esto se logra mediante la estimación de la mejora esperada en cada paso de la optimización. SMS-EMOA utiliza un enfoque de selección basado en la mejora esperada para guiar la búsqueda hacia regiones prometedoras del espacio de búsqueda. El Algoritmo 8 muestra el pseudocódigo del algoritmo SMS-EMOA.

SMS-EMOA ha demostrado ser eficaz en la optimización de problemas multimodales difíciles y ha sido aplicado con éxito en una variedad de campos, como la ingeniería, la biología computacional y la planificación logística.

Aplicaciones de la Optimización Multiobjetivo: La optimización multiobjetivo se aplica en áreas como la planificación urbana, el diseño de ingeniería, la gestión de la cadena de suministro, la toma de decisiones empresariales y en problemas donde la optimización de un solo objetivo no es suficiente.

Por lo tanto, la optimización multiobjetivo busca encontrar soluciones que equilibren múltiples objetivos, ofreciendo un conjunto de soluciones no dominadas que representan compromisos efectivos entre los diferentes criterios definidos.

2.7. Gemelos digitales

Un gemelo digital [19] es una réplica virtual de un objeto, proceso o sistema del mundo real. Se utiliza principalmente en la industria para simular, predecir y optimizar el rendimiento de productos y operaciones. Consiste en un modelo digital en tiempo real que refleja el estado y el comportamiento de su contraparte física, lo que permite a los usuarios interactuar con él y probar diferentes escenarios sin tener que modificar el objeto real.

Los gemelos digitales se crean utilizando una variedad de tecnologías, como modelado 3D, Internet de las cosas (IoT), sensores, análisis de datos, inteligencia artificial (IA) y realidad aumentada (AR). Estas tecnologías permiten recopilar datos del mundo real, procesarlos en el gemelo digital y luego visualizarlos de una manera que sea fácil de entender y utilizar para la toma de decisiones.

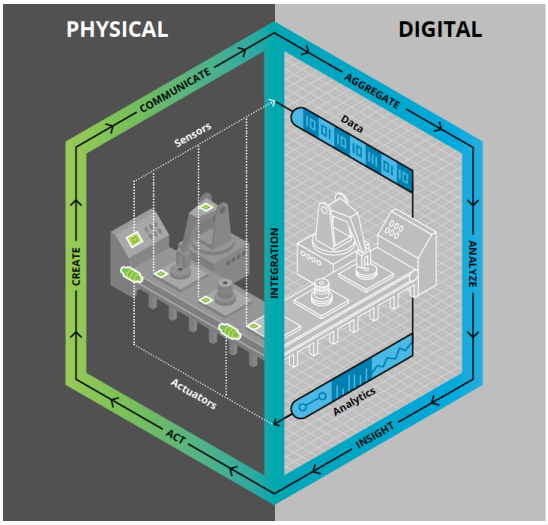

Figura 2.3: Modelo de un gemelo digital (Deloitte University Press)

Los gemelos digitales se utilizan en una amplia gama de industrias, incluyendo la manufactura, la energía, la salud, la construcción y la logística, entre otras. Algunos ejemplos de su aplicación incluyen la optimización de la producción, el mantenimiento predictivo, la planificación de la cadena de suministro, la simulación de procesos y la formación de empleados.

A continuación se mencionan algunos elementos clave que caracterizan a los gemelos digitales:

- **Modelado Preciso**: Los gemelos digitales se construyen mediante modelos digitales altamente precisos que replican tanto la geometría como el comportamiento de los objetos o sistemas que representan. Estos modelos suelen incorporar datos en tiempo real y permiten la simulación de situaciones y escenarios específicos.

- **Conexión con el Mundo Real**: Los gemelos digitales no son estáticos; están vinculados al mundo real a través de sensores y otros dispositivos de recopilación de datos. Estos sensores recopilan información en tiempo real y la alimentan de vuelta al gemelo digital para mantener su representación actualizada y precisa.

- **Interconexión con Tecnologías Emergentes**: Los gemelos digitales a menudo se utilizan en conjunción con tecnologías como el Internet de las Cosas (IoT), la inteligencia artificial (IA), el aprendizaje automático (Machine Learning) y la realidad aumentada (AR). Esta interconexión mejora la capacidad del gemelo digital para interactuar, analizar datos y realizar simulaciones más complejas.

- **Aplicaciones Diversas**: Los gemelos digitales encuentran aplicación en una amplia variedad de campos, desde la manufactura y la gestión de la cadena de suministro hasta la gestión de la energía, la simulación de ciudades inteligentes, la medicina, la exploración espacial y mucho más. Estas representaciones digitales pueden ser herramientas valiosas para la toma de decisiones, la optimización de procesos y la resolución de problemas complejos.

- **Colaboración y Toma de Decisiones**: Los gemelos digitales también facilitan la colaboración entre equipos y la toma de decisiones informada. Permiten a los expertos de diferentes disciplinas trabajar en un entorno compartido y experimentar con diversas soluciones antes de implementar cambios en el mundo real.

2.7.1. Gemelos digitales vs. simulaciones

Aunque las simulaciones y los gemelos digitales utilizan modelos digitales para replicar los diversos procesos de un sistema, un gemelo digital se caracteriza por ser un entorno virtual, lo que lo convierte en una herramienta considerablemente más enriquecedora para el estudio.

La diferencia entre un gemelo digital y una simulación radica principalmente en la escala y la capacidad de estudiar múltiples procesos. Mientras que una simulación normalmente se enfoca en un proceso específico, un gemelo digital tiene la capacidad de ejecutar múltiples simulaciones, lo que lo convierte en una herramienta más versátil y poderosa para el estudio.

En términos de escala, una simulación se limita a analizar un único proceso esto es debido a sus limitaciones en términos de escalabilidad y rendimiento. Por ejemplo, podría simular el comportamiento de un sistema mecánico en particular. Sin embargo, un gemelo digital puede abarcar un sistema completo y replicar todos sus procesos interconectados. Esto significa que puede simular no solo el comportamiento de una sola máquina, sino también cómo interactúa con otras máquinas y componentes en un entorno más amplio.

Las diferencias entre los gemelos digitales y las simulaciones no se limitan solo a la escala y la capacidad de estudio de múltiples procesos. También existen diferencias significativas en términos de flujo de información y uso de datos en tiempo real.

En primer lugar, las simulaciones generalmente no se benefician de tener datos en tiempo real. Las simulaciones se basan en modelos predefinidos y no requieren una interacción constante con el entorno real. Por lo tanto, los datos utilizados en las simulaciones suelen ser datos estáticos o históricos que se ingresan antes de ejecutar la simulación. Esto significa que las simulaciones no están diseñadas para adaptarse a cambios en tiempo real o para recibir datos actualizados durante la ejecución.

Por otro lado, los gemelos digitales están diseñados específicamente en torno a un flujo de información bidireccional en tiempo real. Los gemelos digitales se conectan a sensores y otros dispositivos en el entorno real para recopilar datos en tiempo real. Estos datos se utilizan para actualizar y alimentar continuamente el gemelo digital, lo que permite una representación precisa y actualizada del sistema en todo momento. A su vez, el gemelo digital puede generar conocimientos y análisis en tiempo real que se comparten con el objeto de origen original, lo que permite una retroalimentación y una toma de decisiones más eficientes.

Este flujo de información bidireccional en tiempo real es fundamental para el funcionamiento de los gemelos digitales. Permite una mayor precisión y capacidad de respuesta, ya que los datos se actualizan constantemente y se utilizan para tomar decisiones en tiempo real. Además, el uso de datos en tiempo real en los gemelos digitales permite una mayor

integración con otros sistemas y procesos en el entorno real, lo que facilita la colaboración y la optimización de todo el sistema.

Los gemelos digitales tienen un potencial mayor para mejorar productos y procesos en comparación con las simulaciones estándar. Su capacidad para acceder a datos actualizados, realizar cálculos complejos de manera eficiente y simular interacciones en tiempo real les permite estudiar problemas desde múltiples perspectivas y encontrar soluciones más efectivas. Con su enfoque basado en datos en tiempo real y su capacidad de adaptación, los gemelos digitales representan una herramienta poderosa para la mejora continua y la optimización de productos y procesos.

Capítulo 3

Trabajo relacionado

En este capítulo se presentan los trabajos que están relacionados con el trabajo propuesto, se detallan trabajos que hablan sobre la optimización de trayectorias con enfoque multiobjetivo, asi como tambien de los trabajos que hablan de la planificación de trayectorias con gemelos digitales y del uso de los gemelos digitales.

El diseño de trayectorias consiste en la búsqueda de una ruta libre de colisiones entres dos posiciones de acuerdo a ciertos criterios de evaluación. Por el momento, los métodos disponibles para el diseño de trayectorias se pueden agrupar en cinco categorías: 1) métodos basados en modelos matemáticos [20], 2) métodos basados en búsquedas geométricas [21], 3) métodos basados en algoritmos heurísticos [5], 4) métodos basados en el campo de la inteligencia artificial [22], 5) métodos basados en algoritmos evolutivos [23].

3.1. Optimización de trayectorias con algoritmos evolutivos multiobjetivo

En los últimos años, el diseño de trayectorias ha tenido mucha atención. Jinshuai et al. [24] estudiaron el problema de la Planificación de trayectorias de un robot de detección

en caso de desastre en una mina de carbón. El objetivo en este trabajo es minimizar simultáneamente la longitud del camino y el grado de peligro que hay en las secciones de la mina dividiendo la planificación de la ruta en dos subproblemas. El primer subproblema es encontrar la ruta entre dos puntos por lo que utilizan el algoritmo A* para obtener la matriz de la ruta de acuerdo a los puntos objetivo. El segundo subproblema es la optimización de la ruta en la que el robot considera todos los puntos objetivo y encuentra una ruta óptima para esto utilizan un algoritmo MOEA/D mejorado.

Nianbo et al. [25] plantearon mejorar la eficiencia de un robot que hace entregas en un entorno de un campus, usan un algoritmo MMOEA/D que combina el algoritmo MOEA/D con una estrategia de búsqueda local. Inicialmente establecen el modelo de entorno del campus utilizan A* para planificar la matriz de la ruta entre dos puntos objetivo, luego teniendo como objetivos encontrar la ruta mas corta y el tiempo de retardo mas corto utilizan MMOEA/D para obtener la mejor ruta.

Chaoda et al. [26] plantearon que los problemas de planificación de trayectorias de vehículos aéreos no tripulados (UAV) generalmente tienen restricciones complicadas. Para solucionar este problema los autores proponen un algoritmo evolutivo multiobjetivo restringido basado en descomposición (M2M-DW), en principio, se presentan tres conjuntos de problemas en la planificación de trayectorias de UAVs con dos objetivos: la minimización de la distancia del viaje y el riesgo que tiene un UAV al momento de volar. El problema plantea cuatro restricciones, la altitud mínima de vuelo , la altitud máxima de vuelo, el ángulo mínimo de vuelo y el alcance mínimo/máximo de vuelo.

Yougang et al. [27] propusieron un algoritmo evolutivo multiobjetivo mejorado basado en descomposición (MOEA/D) con una estrategia de ajuste adaptativo del peso del área (AAWA) para lograr un equilibrio entre la longitud total de la trayectoria de vuelo y la dificultad del terreno. El funcionamiento de AAWA es que primero elimina un individuo hacinado y su vector de peso de la población actual y luego agrega un individuo disperso en una población externa que mantiene todas soluciones no dominadas encontradas en la

población actual, para permitir que el individuo recién agregado evolucione hacia el área más dispersa de la población en el espacio objetivo, lo que de acuerdo con los autores logra mejores trayectorias.

Xin et al. [28] propusieron la planificación de trayectoria de un robot de soldadura, propone un enfoque con planificación y optimización de trayectorias para resolver el problema de la soldadura de arco. En este trabajo se aplica el algoritmo RRT* para generar el conjunto de rutas entre dos puntos a soldar, y se utiliza MOEA/D para mejorar la eficiencia de la soldadura, optimizando los siguientes objetivos, longitud de la trayectoria, suavidad de la trayectoria y el consumo de energía.

Sathiya et al. [29] presentaron en su estudio de investigación, la optimización en la planificación de la trayectoria de un robot móvil, esto mediante el uso de dos técnicas, una variante de evolución diferencial multiobjetivo (evolución diferencial multiobjetivo heterogénea) y un algoritmo genético de clasificación elitista no dominado (NSGA-II). En esta investigación se considera un robot móvil con ruedas y accionamiento diferencial, a través de los algoritmos propuestos se determina una trayectoria entre dos puntos en un entorno con obstáculos, se obtiene una ruta mas segura donde se optimizan dos objetivos que es el tiempo de recorrido y el esfuerzo de los actuadores donde se tienen en cuenta las limitaciones del robot.

Xinhao et al. [30] presentaron un artículo donde se abarca un problema en una tarea agrícola, en este caso es la planificación de trayectorias en robots móviles terrestres agrícolas donde se pretende optimizar la distancia de navegación y minimizar el ángulo de giro total. Inicialmente, se escanea el mapa la ruta entre todos los puntos objetivo se calcula utilizando la hoja de ruta probabilística (PRM), y la planificación de ruta del robot se lleva a cabo de acuerdo con la suma de la longitud de la ruta y el ángulo de la ruta. Para determinar la mejor ruta para el robot, se comparan cuatro algoritmos: algoritmo de estimación de hipervolumen (HypE), algoritmo evolutivo basado en cuadrícula (GrEA),

algoritmo evolutivo impulsado por el punto de rodilla (KnEA) y algoritmo genético de clasificación no dominado (NSGA-III).

3.2. Optimización mediante gemelos digitales

Por otro lado, se tiene que los gemelos digitales también han tenido atención en diferentes áreas de investigación. Yongkui et. al [31] discutió que la construcción de gemelos digitales para robots industriales es de gran importancia, por lo que, para permitir una gestión transparente y aplicaciones inteligentes de robots industriales, este artículo propone una arquitectura y un método para construir gemelos digitales de alta fidelidad para robots. La arquitectura funcional consta de cinco capas, incluida la capa de equipo, la capa de transmisión, la capa de gemelo digital, la capa de aplicación y la capa industrial. Finalmente, a partir del modelo y arquitectura, y tomando como ejemplo el robot industrial SD3/500, se desarrolla un sistema de gemelo digital, que implementa funciones de sincronización virtual-real, monitorización en tiempo real, simulación cinemática, visualización de datos y visualización real.

Pauwels et al. [32] discutieron que los robots a menudo dependen de escáneres láser planos y 3D para ubicarse y navegar de la manera más autónoma posible, y los modelos de información de construcción (BIM) rara vez se utilizan. Los autores investigaron específicamente en qué medida y cómo se pueden utilizar estos datos de construcción para dicha navegación robótica. Los flujos de datos se construyen desde el modelo BIM hasta el repositorio local y luego hasta el robot, utilizando modelos de datos gráficos (RDF) y formatos de datos JSON. Por lo tanto, el repositorio local puede considerarse un gemelo digital del edificio del mundo real. La navegación basada en un modelo BIM se prueba en un entorno del mundo real. Los autores concluyen que es posible confiar en los datos BIM y describen diferentes flujos de datos desde el modelo BIM hasta el gemelo digital y el robot.

Liu et al. [33] indagaron que la precisión de la trayectoria de movimiento del robot se ha convertido en una cuestión clave que afecta a su eficiencia en el trabajo. Por lo tanto,

los autores presentan un método para optimizar la trayectoria del robot móvil basado en el gemelo digital del robot. Mediante Unity se crea el gemelo digital del robot móvil, y la trayectoria del robot móvil se entrena en el entorno virtual y se aplica al espacio físico. El entrenamiento de simulación en el entorno virtual proporciona esquemas para el movimiento real del robot. En función de los datos de movimiento reales devueltos por el robot físico, la trayectoria preestablecida del robot virtual se ajusta dinámicamente, lo que a su vez permite la corrección de la trayectoria de movimiento del robot físico.

Chancharoen et al. [34] desarrollaron un robot de pintura colaborativo que puede utilizarse como alternativa a los trabajadores utilizando un marco de gemelo digital. El gemelo digital del robot de pintura automático simula todo el proceso y estima el resultado de la pintura antes de la ejecución real. Un operador puede ver el proceso simulado y el resultado con la opción de confirmar o cancelar la tarea. Si se acepta la tarea, el gemelo digital genera todos los parámetros, incluida la trayectoria del efector final del robot, el flujo de material hacia el robot colaborativo y un mecanismo de pulverización.

Li et al. [35] discutieron que la teleoperación y la coordinación de múltiples robots industriales desempeñan un papel importante en los sistemas industriales. Por lo que ellos en esta investigación proponen un novedoso sistema de fabricación colaborativa de múltiples robots con control humano integrado aprovechando las técnicas de realidad aumentada y gemelo digital. En el sistema propuesto, los gemelos digitales de los robots industriales se asignan primero a robots físicos y se visualizan en las gafas de realidad aumentada, con el fin de que el operario pueda operar el robot a distancia.

Denk et al. [36] realizaron un trabajo donde se utiliza la denominada reducción homotópica para generar un gemelo digital, que se puede utilizar para extraer todas las propuestas de caminos posibles, incluidos sus anchos de paso para entornos 2D y 3D y múltiples tareas y robots. La erosión del medio ambiente está controlada por limitaciones tales que se consideran las estaciones de trabajo, las posiciones del robot o dron y la topología del medio ambiente, esto para la Planificación de trayectorias de robots y drones autónomos.

Alves et al. [37] Proponen un trabajo en el cual presentan el desarrollo de un simulador VR para navegación robótica y detección de caídas con gemelos digitales como una solución para probar el robot virtual sin tener acceso a la ubicación física real, ni a personas reales. El proceso de desarrollo requirió el desarrollo de sensores virtuales que sean capaces de crear datos LIDAR para que el robot virtual navegue y detecte obstáculos.

Capítulo 4

Metodologia propuesta

En este capítulo se describe el procedimiento del desarrollo e implementación de la metodología propuesta. Para esto se desarrolla el gemelo digital, se hace uso de un entorno virtual simulado en Gazebo mediante ROS y se describe el uso de los algoritmos de optimización utilizados. Es importante mencionar que se deben buscar alternativas para optimizar la navegación en robots móviles con un enfoque multiobjetivo. En este trabajo, se propone una metodología que utiliza algoritmos multiobjetivo, como NSGA-II, MOEA/D y SMS-EMOA, para optimizar las trayectorias. Los objetivos considerados en este proceso de optimización son la distancia de la trayectoria y la suavidad de la misma considerando la evasión de obstáculos.

Además, se realiza la integración de un gemelo digital, lo que nos permite optimizar objetivos adicionales, como el consumo de batería y el torque de los motores. De esta manera, se obtiene una trayectoria final que ha sido optimizada con cinco objetivos, lo que resulta en mejores resultados durante la ejecución o simulación. Recordemos que la optimización de la navegación en robots móviles es un tema de gran importancia, por lo que el uso de algoritmos multiobjetivo y la integración de un gemelo digital son enfoques prometedores para lograr trayectorias más eficientes y efectivas.

Para su mejor entendimiento, el presente capítulo se organiza como sigue.

Desarrollo del gemelo digital. En esta sección se describe detalladamente el proceso de desarrollo del enfoque propuesto. Comenzamos con la creación del gemelo digital, el cual es una réplica virtual del robot móvil que se está usando y que nos permite realizar simulaciones y optimizaciones.

Algoritmos bioinspirados. En esta sección se explica cómo se implementaron los algoritmos bioinspirados. Estos algoritmos se basan en principios y comportamientos observados en la naturaleza, como el comportamiento de las colonias de hormigas o el vuelo de los pájaros. Su aplicación nos permite encontrar soluciones eficientes y efectivas para la navegación del robot.

Algoritmos tradicionales para la planeación de trayectorias. En esta sección se detalla la implementación de los algoritmos tradicionales utilizados para el cálculo de trayectorias. Estos algoritmos se basan en métodos clásicos de Planificación de trayectorias y nos brindan una base sólida para comparar y evaluar el rendimiento de los algoritmos multiobjetivo.

Algoritmos de optimización multi-objetivo. En esta sección se aborada la implementación de los algoritmos multiobjetivo. Estos algoritmos nos permiten considerar múltiples objetivos de optimización, como la evasión de obstáculos, la distancia de la trayectoria y la suavidad del movimiento. Su aplicación nos ayuda a encontrar soluciones que equilibren estos objetivos de manera eficiente.

Indicadores de rendimiento multiobjetivo. En esta sección se introducen los indicadores de desempeño para evaluar los algoritmos multiobjetivo adoptados en este trabajo. En esta investigación indagamos el tipo de algoritmos que benefician de mejor manera a la optimización de trayectorias. Por tal motivo, estos indicadores son de relevancia para distinguir que algoritmo ofrece un mejor desempeño.

Integración del gemelo digital en el proceso de optimización. En esta sección se describe el proceso de integración del gemelo digital en el marco de trabajo para la optimización del torque, consumo de batería y el tiempo de ejecución.

4.1. Desarrollo del gemelo digital

Para el desarrollo del gemelo digital nos basamos en el prototipo de carro autónomo a escala "AutoNOMOS" desarrollado por la Universidad Libre de Berlín con fines educativos ver Figura 4.1 . Este carro a escala está conformado por:

- Odroid XU4
- RPLIDAR A2
- Arduino Nano
- IMU
- Batería de 14.8V (5000mAh)
- Cámara Realsense
- Servomotor para dirección
- Motor sin escobillas para la tracción

De los componentes mencionados, se han creado modelos dinámicos (representación matemática) tanto para la batería como para el motor de tracción. Estos modelos nos permitirán analizar el comportamiento de dichos componentes y llevar a cabo el proceso de optimización del consumo de batería y el torque del motor de tracción. Por otro lado, los componentes restantes no cuentan con un modelo dinámico, pero se representan como clases que les permiten tener valores para sus características.

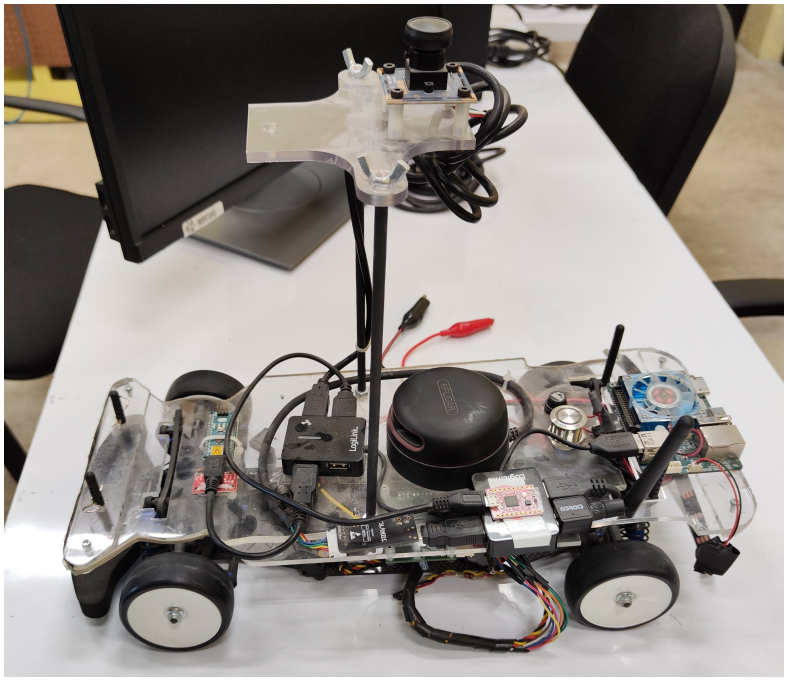

Figura 4.1: Carrito a escala AutoNOMOS

Es importante destacar que existen varios tipos de gemelos digitales, dependiendo del nivel de mejora del sistema. Es común que un sistema o proceso integre diferentes tipos de gemelos digitales para abordar diversas necesidades y objetivos. En nuestro caso, hemos implementado un gemelo activo [8], el cual se caracteriza por la colaboración de dos o más componentes interconectados. Específicamente, estamos trabajando con el motor de tracción y la batería, los cuales juntos forman un gemelo activo. Esta configuración nos permite estudiar detalladamente la interacción entre los componentes, generando datos precisos sobre su rendimiento.

Además, es importante mencionar que un gemelo digital puede operar sin recibir información en tiempo real [38] [39]. En nuestro trabajo, hemos optado por implementar un gemelo digital estático, ya que se adapta perfectamente a nuestras necesidades de simulaciones y estudios que no requieren actualizaciones en tiempo real. Esta elección es adecuada para nuestros objetivos, ya que no estamos trabajando con entornos dinámicos.

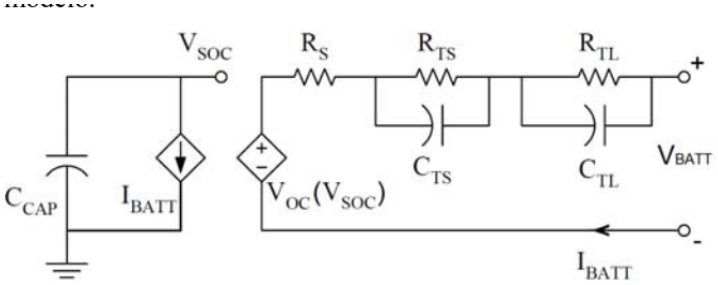

Figura 4.2: Esquema de batería

Utilizamos el gemelo digital para optimizar la trayectoria y evaluarla exhaustivamente antes de su implementación física. Dado que nuestro enfoque no requiere actualizaciones continuas, un gemelo digital estático nos permite lograr una optimización precisa y efectiva sin la complejidad adicional de gestionar datos en tiempo real.

4.1.1. Modelo dinámico batería

El modelo que se uso para el modelo de la batería se puede ver en la Figura 4.2 [40], este consiste en dos circuitos separados, que están relacionados entre si por una fuente de tensión y una fuente de corriente. El primer circuito (izquierda) modela la capacidad de almacenamiento de energía de la batería y la carga almacenada durante los procesos de carga o descarga. El segundo (derecha) describe la resistencia interna de la batería y el comportamiento transitorio ante distintas cargas.

La fuente de tensión controlada por tensión representa la dependencia no lineal entre el estado de carga (SOC) y V_{OC}. La tensión V_{SOC} se normaliza de forma que $V_{SOC} = 1$ V equivale al SOC (100 %).

Dado que la tensión se normaliza a 1V y que $C = \frac{Q}{V}$, siendo C capadidad en faradios, Q carga en culombios y V tensión en voltios, el valor del condensador C_{CAP}, en faradios, resulta:

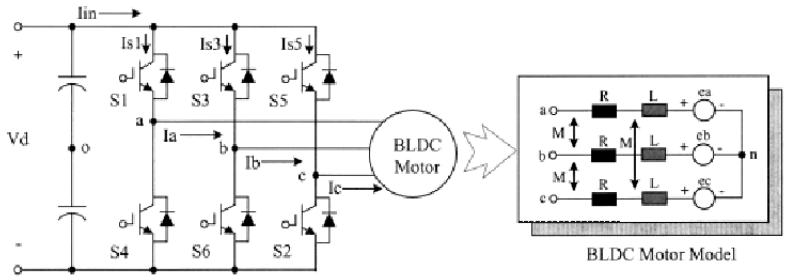

Figura 4.3: Esquema de motor eléctrico

$$C_{CAP}[F] = 3600[\frac{C}{Ah}] \cdot Capacidad[Ah] \cdot 1[V^{-1}] \cdot f_1(t) \cdot f_2(T) \tag{4.1}$$

Donde *Capacidad* es la capacidad nominal de la batería, f_1 es un factor corrector que modela el envejecimiento y f_2 es un factor corrector dependiente de la temperatura.

4.1.2. Modelo dinámico motor de tracción

Para el modelo dinámico del motor nos basamos en la Figura 4.3 [41], este modelo incluye un controlador eléctrico del motor, este se une al motor el cual tiene 3 terminales, los cuales tienen: fuerza resistiva, fuerza contraelectromotriz inducida y efecto inductivo, por lo que el modelo para cada fase se define de la siguiente manera:

$$(L - M)\frac{di_a}{dt} + Ri_a + e_a = v_a \tag{4.2}$$

$$(L - M)\frac{di_b}{dt} + Ri_b + e_b = v_b \tag{4.3}$$

$$(L - M)\frac{di_c}{dt} + Ri_c + e_c = v_c \tag{4.4}$$

Donde: L: Es la inductancia de cada fase, M: Representa las inductancias mutuas, R: Es la resistencia de casa fase y e_a,e_b,e_c: Son las fuerzas contraelectromotrices en cada devanado. Además debemos de tener en cuenta que la fuerza contraelectromotriz depende de la velocidad angular y la posición del rotor (θ), por lo que:

$$e_a = \begin{cases} \frac{6E}{\pi}\theta_r, & \left(0 < \theta_r < \frac{\pi}{6}\right) \\ E, \left(\frac{\pi}{6} < \theta_r < \frac{5\pi}{6}\right) \\ -\left(\frac{6E}{\pi}\right)\theta_r + 6E, \left(\frac{5\pi}{6} < \theta_r < \frac{7\pi}{6}\right) \\ -E, \left(\frac{7\pi}{6} < \theta_r < \frac{11\pi}{6}\right) \\ (\left(\frac{6E}{\pi}\right)\theta_r 6 - 12E, \left(\frac{11\pi}{6} < \theta_r < 2\pi\right) \end{cases} \tag{4.5}$$

$$e_b = \begin{cases} -E, & \left(0 < \theta_r < \frac{\pi}{6}\right) \\ \left(\frac{6E}{\pi}\right)\theta_r - 4E, & \left(\frac{\pi}{6} < \theta_r < \frac{5\pi}{6}\right) \\ E, \left(\frac{5\pi}{6} < \theta_r < \frac{7\pi}{6}\right) \\ -\left(\frac{6E}{\pi}\right)\theta_r + 10E, & \left(\frac{7\pi}{6} < \theta_r < \frac{11\pi}{6}\right) \\ E, \left(\frac{11\pi}{6} < \theta_r < 2\pi\right) \end{cases} \tag{4.6}$$

$$e_c = \begin{cases} E, & \left(0 < \theta_r < \frac{\pi}{6}\right) \\ \left(\frac{6E}{\pi}\right)\theta_r + 2E, \left(\frac{\pi}{6} < \theta_r < \frac{5\pi}{6}\right) \\ -E, \left(\frac{5\pi}{6} < \theta_r < \frac{7\pi}{6}\right) \\ -\left(\frac{6E}{\pi}\right)\theta_r - 8E, \left(\frac{7\pi}{6} < \theta_r < \frac{11\pi}{6}\right) \\ E, \left(\frac{11\pi}{6} < \theta_r < 2\pi\right) \end{cases} \tag{4.7}$$

Donde:

$$E = w * K_e \tag{4.8}$$

Siendo K_e la constante de la fuerza electromotriz.

También tenemos la ecuación que modela los torques que vemos en la Ecuacion 4.9:

$$T_e = T_l + J_{motor}\frac{dw}{dt} + Bw \tag{4.9}$$

Donde: T_e: Es el torque electromagnético, T_l: Torque de carga, J_{motor}: Inercia, w: Velocidad de angular, B: Amortiguación y Bw: Torque por fricción.

La implementación del gemelo digital se realizó mediante Python, en el cual se crearon los gemelos digitales y se realiza el proceso de optimización del consumo de batería y el torque de los motores, de acuerdo a los modelos dinámicos implementados.

El proceso de optimización se llevó a cabo mediante la simulación de la ruta previamente calculada por el algoritmo de optimización multiobjetivo. En este proceso, se tuvo en cuenta el tiempo que le tomaría al robot completar dicha trayectoria con diferentes valores de torque en el motor de tracción, con el objetivo de calcular el consumo de batería que se requeriría en el robot real al realizar la misma trayectoria.

Para lograr esto, el gemelo digital nos permitió simular y evaluar en el escenario elegido el consumo de batería al variar los valores de torque en el motor de tracción. Esta simulación se repitió 30 veces, cada vez con un valor de torque diferente, con el fin de encontrar el torque adecuado que permitiera realizar la trayectoria con el menor consumo de batería posible.

Al finalizar el proceso de optimización realizado por el gemelo digital, se obtuvo la trayectoria óptima a seguir y los valores de torque recomendados para el motor de tracción, los cuales minimizaban el consumo de batería. Para finalizar la trayectoria final calculada y los valores obtenidos son simulados en Gazebo.

4.2. Algoritmos bioinspirados

Para la implementación de los algoritmos bioinspirados que se usaron en este trabajo hacemos uso de la librería Pymoo [42] que contiene diferentes algoritmos de optimización. Para este trabajo se hace uso del Algoritmo Genético (AG) y de Evolución Diferencial (ED).

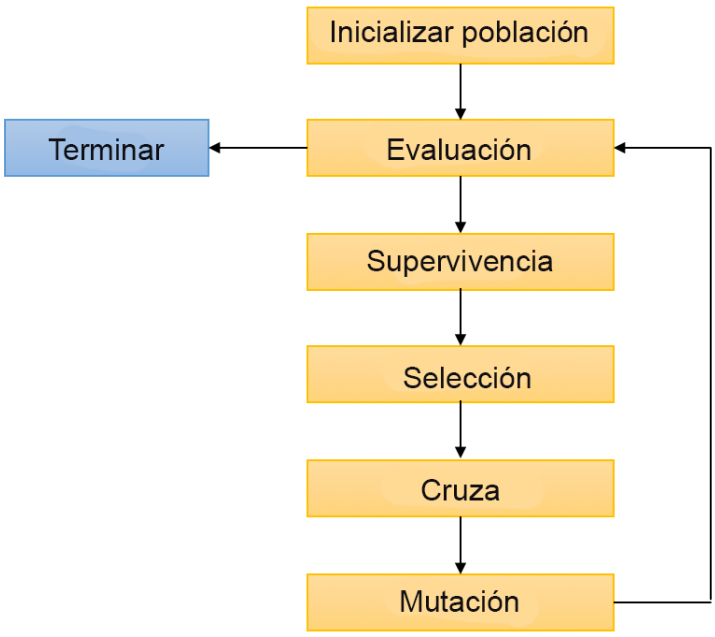

Figura 4.4: Diagrama algoritmo genético

4.2.1. Algoritmo Genético (AG)

El algoritmo utilizado representa un algoritmo genético básico $(\mu + \lambda)$ para problemas de un solo objetivo. La Figura 4.4 muestra el flujo del algoritmo genético.

Los valores de los parámetros que se utilizaron para los experimentos de este algoritmo son los siguientes:

- Población inicial = 300
- Probabilidad de cruce = 0.9
- Probabilidad de mutación = 0.2

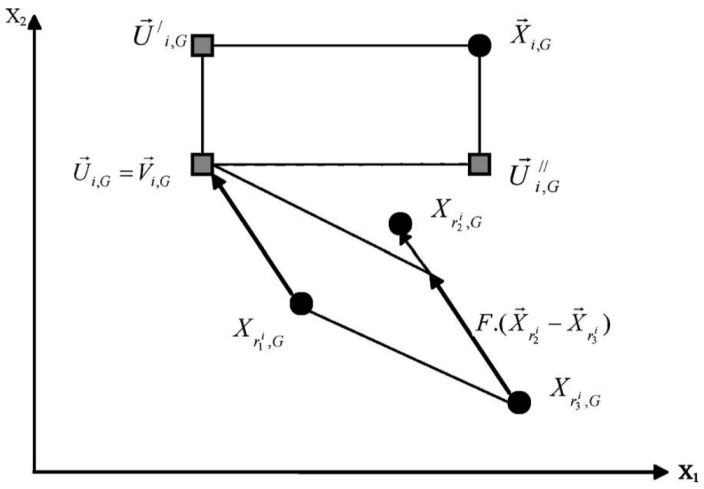

Figura 4.5: Cruce AG

4.2.2. Evolución Diferencial (ED)

El algoritmo utilizado es el clásico algoritmo de objetivo único, que es conocido por dar buenos resultados para la optimización. El cruce del algoritmo está definido por:

$$v = x_{\pi_1} + F \cdot (x_{\pi_2} - x_{\pi_3}) \tag{4.10}$$

Donde π es una permutación aleatoria de 3 entradas. La diferencia se toma entre el individuo 2 y 3 y se suma al primero, la Figura 4.5 [43] demuestra lo anterior. Luego, un segundo cruce entre un individuo y el llamado vector donante v es interpretado. El segundo cruce puede ser simplemente binomial/uniforme o exponencial.

Los valores de los parámetros que se utilizaron para los experimentos de este algoritmo son los siguientes:

- Población inicial = 200
- Factor de ponderación = 0.8
- Constante de cruce = 0.9

4.3. Algoritmos tradicionales para la planeación de trayectorias

Para los algoritmos tradicionales, se utilizaron tres métodos de búsqueda adaptados para resolver el problema planteado. Estos algoritmos son ampliamente utilizados en diferentes áreas de la inteligencia artificial, robótica y planificación de trayectorias.

Algoritmo A* [4]. Este método se utiliza para encontrar el camino más corto entre dos puntos en un grafo. Es especialmente popular en aplicaciones de planificación de trayectorias y juegos. En el contexto del problema, se adaptó el algoritmo A* para encontrar la ruta óptima en un entorno específico.

Algoritmo Dijkstra [5]. Este método se utiliza para encontrar el camino más corto desde un nodo de origen a todos los demás nodos en un grafo con aristas no negativas. Es ampliamente utilizado en redes de computadoras, sistemas de transporte y planificación de trayectorias. En este caso, el algoritmo de Dijkstra se adaptó para encontrar la ruta más eficiente en el contexto del problema planteado.

Algoritmo RRT [6]. Este método se utiliza en robótica y planificación de movimientos para encontrar caminos viables en entornos complejos y de alta dimensión. RRT construye un árbol de rutas posibles en el espacio de búsqueda mediante la expansión aleatoria desde un nodo inicial. Esto permite cubrir rápidamente grandes áreas del espacio de búsqueda. En el problema en cuestión, el algoritmo RRT se adaptó para encontrar caminos óptimos en un entorno específico.

Es importante mencionar que las implementaciones de los algoritmos A*, Dijkstra y RRT fueron tomados del repositorio de Github [44]. Estas implementaciones fueron tomadas y adaptas específicamente para resolver el problema planteado en este trabajo de tesis.

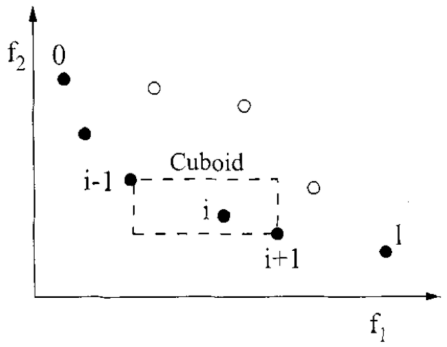

Figura 4.6: Cálculo de la distancia de aglomeración [2]

4.4. Algoritmos de optimizacion multi-objetivo

Para la implementación de los algoritmos multiobjetivo que se usaron en este trabajo al igual que los algoritmos bioinspirados hacemos uso de la librería Pymoo [42]. Para este trabajo hacemos uso de NSGA-II, MOEA/D y SMS-EMOA.

4.4.1. NSGA-II

El algoritmo se implementa en base a [16]. El algoritmo sigue el esquema general de un algoritmo genético con una selección de apareamiento y supervivencia modificada. En NSGA-II, primero, los individuos se seleccionan de frente. Al hacerlo, se producirá una situación en la que será necesario dividir un frente porque no a todos los individuos se les permite sobrevivir. En este frente de división, las soluciones se seleccionan en función de la distancia de aglomeración.

La distancia de aglomeración es la Distancia de Manhattan en el espacio objetivo. Sin embargo, se desea que los puntos extremos se mantengan en cada generación y, por lo tanto, se les asigna una distancia de aglomeración infinita. Además, para aumentar cierta presión de selección, NSGA-II utiliza una selección de apareamiento de torneo binario. Cada individuo se compara primero por rango y luego por distancia de hacinamiento.

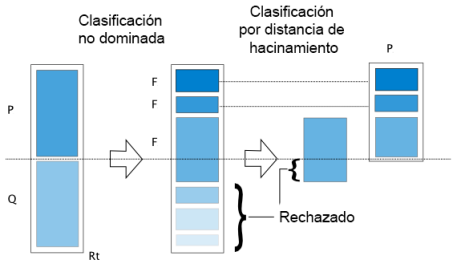

Figura 4.7: Distancia de aglomeración [2]

4.4.2. MOEA/D

Este algoritmo se implementa en base a [17]. El algoritmo se basa en direcciones de referencia que deben proporcionarse al inicializar el objeto del algoritmo. La mayoría de los algoritmos evolutivos de optimización de muchos objetivos (EMaO), como lo es MOEAD, comienzan con una descripción de un número de conjuntos predefinidos de direcciones de referencia en una unidad simplex. Direcciones de referencia en un M el espacio dimensional está sujeto a:

$$\sum_{i=1}^{M} w_i = 1 \tag{4.11}$$

$$w_i \geq 0 \tag{4.12}$$

$$\forall i \epsilon \, (1, ..., M) \tag{4.13}$$

Una dirección de referencia se construye mediante un vector que parte del origen y está conectado a cada uno de ellos. El número de puntos en la unidad simplex está determinado por un parámetro p, que indica el número de espacios entre dos puntos consecutivos a lo largo de un eje objetivo. Resulta que el número total de puntos (n) en la unidad simplex es:

$$n = C_P^{M+p-1} \tag{4.14}$$

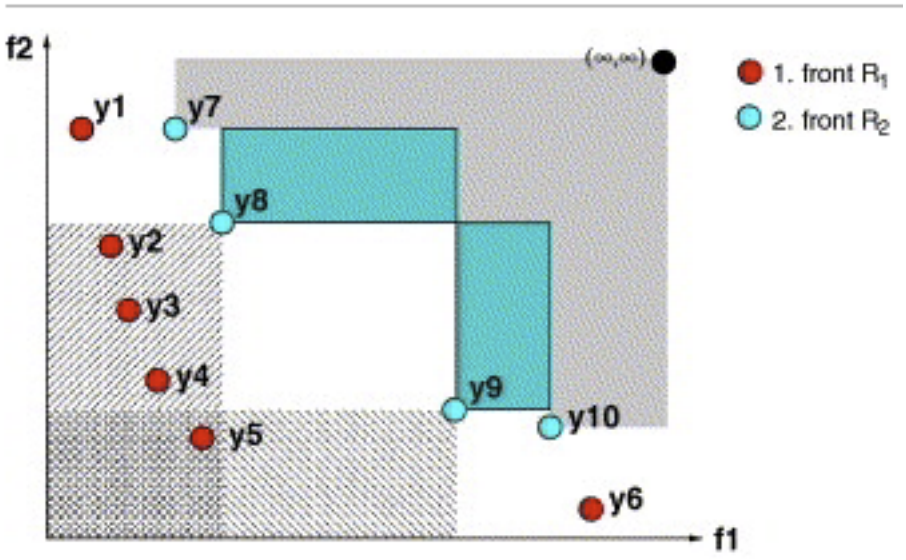

Figura 4.8: Hipervolumen dominado por el frente R2 [3]

4.4.3. SMS-EMOA

El algoritmo se implementa en base a [18]. La medida de hipervolumen (o s-metric) es una medida de calidad que se aplica con frecuencia para comparar los resultados de algoritmos evolutivos de optimización multiobjetivo (EMOA).

SMS-EMOA tiene como objetivo maximizar el hipervolumen dominado dentro del proceso de optimización. Cuenta con un operador de selección basado en la medida de hipervolumen combinado con el concepto de clasificación no dominada. Como resultado, la población del algoritmo evoluciona hacia un conjunto de soluciones bien distribuidas, centrándose en regiones interesantes del frente de Pareto.

4.5. Indicadores de rendimiento multiobjetivo

Debemos de realizar una comparación de rendimiento entre los algoritmos multiobjetivo usados en esta investigación. Por lo tanto, evaluamos los algoritmos multiobjetivo en base a

tres indicadores de rendimiento tomados de la literatura de optimización evolutiva multiobjetivo. A continuación, se proporcionan las definiciones de los indicadores de rendimiento utilizados.

4.5.1. Indicador de hipervolumen

El indicador de hipervolumen, propuesto por primera vez por Zitzler [45] sirve como una medida confiable para evaluar el rendimiento de los algoritmos multiobjetivo. Una de sus cualidades atractivas es su compatibilidad con la optimalidad de Pareto, lo que le permite evaluar eficazmente tanto la convergencia como la distribución de soluciones en el frente de Pareto ($\mathcal{PF}$) de un problema dado. El indicador de hipervolumen normalizado para S mide la región dominada por S y limitada por los vectores $\mathbf{u}$ y $\mathbf{r}$. Que es expresada de la siguiente manera:

$$\mathcal{H}_n(S, \mathbf{u}, \mathbf{r}) = \frac{\mathcal{H}(S, \mathbf{r})}{\Pi_{i=1}^{M} |r_i - u_i|} \tag{4.15}$$

Donde $\mathcal{H}(S, \mathbf{r}) = \mathcal{L}\left(\mathbf{q} \in \mathbb{R}^M | \exists \mathbf{p} \in S : \mathbf{p} \preceq \mathbf{q} \text{ y } \mathbf{q} \preceq \mathbf{r}\right)$, con $\mathcal{L}(\cdot)$ denotando la medida de Lebesgue, y M representando el número de funciones objetivo de un MOP dado.

4.5.2. Distancia generacional invertida más ($\mathcal{IGD}^+$)

La distancia generacional invertida plus ($\mathcal{IGD}^+$) propuesta por Ishibushi et. al. [46] extiende el indicador $\mathcal{IGD}$ propuesto por [47] y se utiliza para medir qué tan cerca está un conjunto de soluciones aproximadas del frente de Pareto verdadero (las soluciones óptimas no dominadas). La $\mathcal{IGD}^+$ es una extensión de la métrica IGD y está diseñada para penalizar las soluciones que están demasiado dispersas o lejos del frente de Pareto verdadero, lo que la hace útil para evaluar tanto la convergencia como la diversidad de las soluciones obtenidas por un algoritmo multiobjetivo. Este indicador se representa de la siguiente manera:

$$IGD^{+}(A) = \frac{1}{|Z|}\left(\sum_{i=1}^{|Z|} {d_i^{+}}^2\right)^{1/2} \tag{4.16}$$

Donde para minimizar $d_i^{+} = max\{a_i - z_i, 0\}$ representa la distancia modificada desde z_i hasta su punto de referencia más cercano en A con el valor correspondiente a_i.

4.5.3. Indicador de espaciado ($\mathcal{S}$)

El indicador de rendimiento de espaciado, propuesto por Schott [48], cuantifica la dispersión de las soluciones a lo largo del frente no dominado obtenido. El indicador $\mathcal{S}$ se calcula de la siguiente manera. Sea P el conjunto de soluciones no dominadas producidas por un algoritmo. El indicador $\mathcal{S}$ está dado por:

$$\mathcal{S} = \sqrt{\frac{1}{|P|-1}\sum_{i=1}^{|P|}(\overline{d} - d_i)^2} \tag{4.17}$$

Donde $d_i = \text{mín}_{i,j\neq i}\left\{\sum_{k=1}^{M}|f_k^i - f_k^j|\right\}$ y $\overline{d} = \frac{\sum_{i=1}^{|P|} d_i}{|P|}$, con M representando el número de funciones objetivo del problema de optimización multiobjetivo dado. Un valor de cero (el mejor valor posible) para este indicador indica que todas las soluciones están uniformemente distribuidas en todo el $\mathcal{PF}$.

4.6. Integración del gemelo digital

El gemelo digital se emplea en el proceso de optimización tras haber perfeccionado los objetivos de minimizar la distancia de la trayectoria y suavizarla mediante el uso de algoritmos evolutivos multiobjetivo.

Una vez que se ha determinado la mejor trayectoria generada, se lleva a cabo el proceso de optimización del gemelo digital mediante la simulación de esta trayectoria. Durante la simulación, se tiene en cuenta el tiempo en el que el usuario desea que se realice la

trayectoria. De esta manera, el gemelo digital prueba diferentes valores de torque en el motor de tracción para optimizar el torque, el consumo de batería y el tiempo de ejecución.

Al concluir el proceso de optimización realizado por el gemelo digital, se obtiene la trayectoria optimizada y los valores de torque necesarios para el motor de tracción. Estos valores son transferidos al robot para su ejecución, permitiendo que el robot siga la trayectoria optimizada. A medida que el robot se mueve siguiendo la trayectoria, proporciona retroalimentación al gemelo digital. La retroalimentación del robot es utilizada para futuras optimizaciones. El gemelo digital analiza esta retroalimentación y ajusta los parámetros de la trayectoria y los valores de torque en busca de mejoras adicionales.

El proceso de optimización completo, que abarca todos los módulos mencionados anteriormente, se muestra de manera detallada en el diagrama presentado en la Figura 4.9. Este diagrama representa la integración y la interacción de cada uno de los componentes del proceso de optimización, permitiendo una visión clara y completa de cómo se lleva a cabo el proceso de optimización en su totalidad.

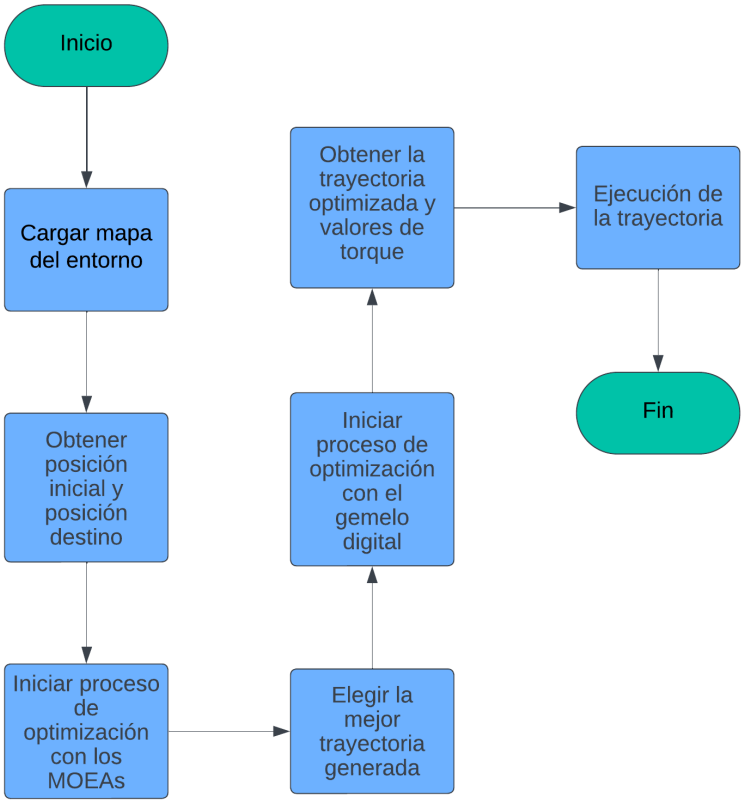

Figura 4.9: Diagrama del proceso de optimización

Capítulo 5

Estudio experimental

En este capítulo, presentaremos los experimentos realizados para validar la efectividad de la metodología propuesta en la optimización de trayectorias. Además, abordaremos los elementos relevantes del entorno y las trayectorias estudiadas. También se incluirá una comparación de los algoritmos utilizados.

Para contextualizar este capítulo, es importante recordar el objetivo de investigación de esta tesis. Nuestro objetivo principal es diseñar y desarrollar una metodología que permita optimizar la navegación de un robot móvil en un entorno simulado, utilizando un enfoque multiobjetivo. Esta metodología busca mejorar tanto la eficiencia como la seguridad de la navegación del robot, tomando en cuenta múltiples criterios u objetivos simultáneamente. A continuación, detallaremos cómo se planificaron y llevaron a cabo los experimentos, así como los resultados obtenidos.

5.1. Entorno

En este trabajo, se utilizó el entorno de una casa (ver Figura 5.1) para llevar a cabo los experimentos. Este entorno forma parte del paquete de simulación del robot Turtlebot. Para poder utilizar este entorno en nuestros experimentos, lo primero que se hizo fue mapear el

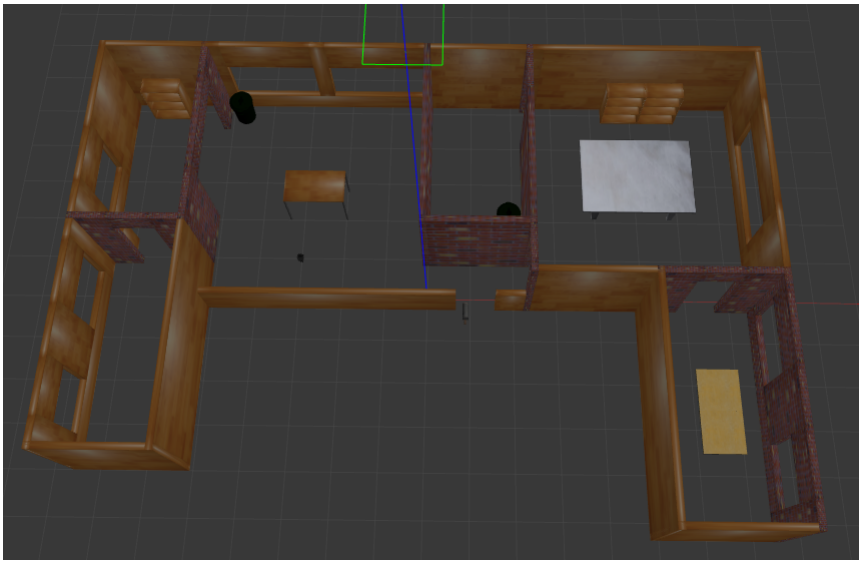

Figura 5.1: Entorno de la casa

entorno para generar su mapa en 2D (ver Figura 5.2). Este mapa se generó utilizando la simulación en Gazebo mediante ROS.

El proceso de mapeo consistió en transformar el entorno en un mapa de grid, es decir, en una matriz compuesta por ceros y unos. En esta representación, el valor 0 indica que el punto está libre y el valor 1 indica que el punto es un obstáculo. En la Figura 5.3 se puede observar una parte del mapa y cómo está conformado.

Este mapa generado es utilizado por todos los algoritmos que se emplean en este trabajo. Los algoritmos pueden utilizar tanto la representación de matriz de ceros y unos como la imagen del mapa 2D. El mapa es fundamental para que los algoritmos puedan planificar las trayectorias del robot de manera eficiente y segura dentro del entorno simulado.

Debemos aclarar que la representación en matriz y el mapa 2D se emplean exclusivamente para el proceso de optimización de la trayectoria del robot. Estas representaciones permiten al algoritmo comprender mejor el entorno en el que se encuentra el robot, lo que facilita un análisis y procesamiento más eficiente de la información ambiental para optimizar la trayectoria.

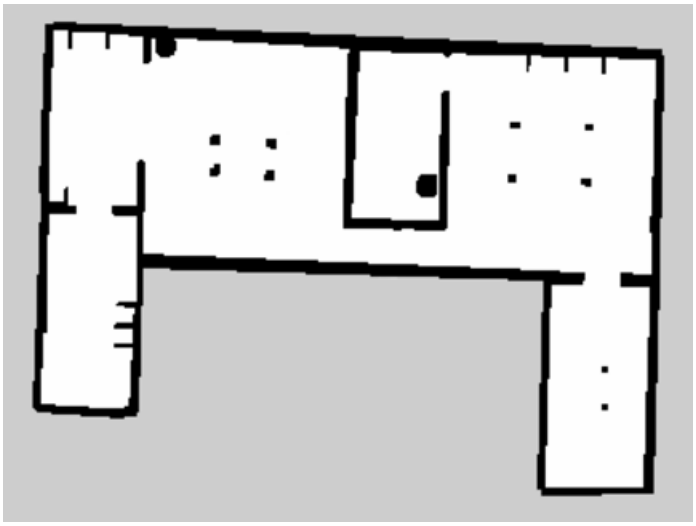

Figura 5.2: Mapa 2D de la casa

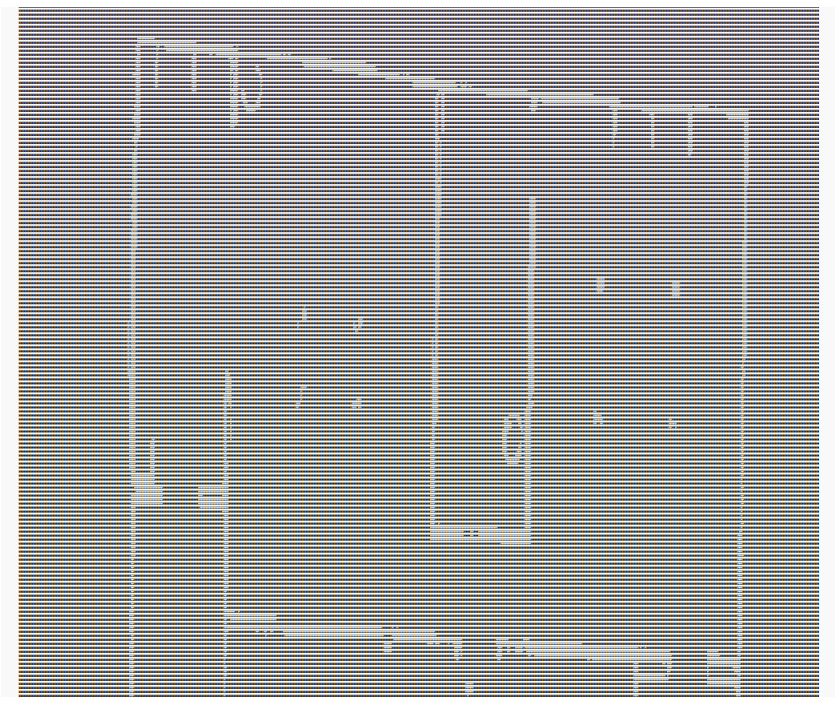

Figura 5.3: Representación mapa en matriz

Además, es importante destacar que las trayectorias generadas son simuladas en Gazebo al final del proceso de optimización. Durante esta simulación, se utilizan los parámetros obtenidos para garantizar un ajuste preciso de la trayectoria al entorno simulado.

5.2. Representación de la trayectoria

Basándose en la representación del mapa, una trayectoria está compuesta por m nodos (x_i, y_i), que representan posiciones en la matriz M y están ordenados en una tupla de la forma:

$$\begin{aligned} \mathbf{x} \;=\; & [(x_1, y_1), (x_2, y_2), ..., (x_i, y_i), ..., \\ & (x_{m-1}, y_{m-1}), (x_m, y_m)], \end{aligned} \tag{5.1}$$

Cada nodo en la tupla contiene información importante sobre la trayectoria. El valor x_1 representa la coordenada x del nodo en la matriz M, mientras que el valor y_1 representa la coordenada y del nodo. Estas coordenadas nos permiten ubicar cada nodo en el mapa y trazar la trayectoria de manera precisa.

La representación de la trayectoria en forma de tupla es útil porque nos permite almacenar y manipular fácilmente la información de cada nodo. Podemos acceder a las coordenadas x e y de cada nodo de forma individual, lo que facilita el procesamiento y análisis de la trayectoria.

Además, el orden en el que se encuentran los nodos en la tupla es crucial, ya que determina el recorrido de la trayectoria. Siguiendo el orden de los nodos en la tupla, podemos trazar la ruta desde el nodo inicial hasta el nodo final de la trayectoria.

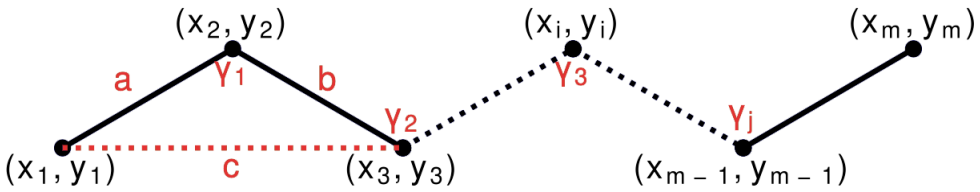

Figura 5.4: Representación de la trayectoria

5.3. Objetivos de optimización

En esta sección presentamos explícitamente los objetivos considerados en la optimización de trayectorias. Además se introduce la restricción a la cual está sujeta la búsqueda. La definición des estos objetivos junto con la restricción del problema, son tratados con los algoritmos de optimización multiobjetivo adoptados en este trabajo de tesis. Sin embargo, es importante señalar, que cualquier otro algoritmo multiobjetivo con manejo de restricciones puede ser adoptado a esta metodología.

5.3.1. Objetivo de longitud de la trayectoria

La longitud de la trayectoria guarda una estrecha relación con el consumo de energía de un robot móvil. A medida que la trayectoria se vuelve más larga, el consumo de energía experimentado por el robot aumenta. Por ende, obtener las trayectorias más cortas posibles se convierte en un objetivo de suma importancia. Por consiguiente, la longitud de la trayectoria se define como la suma de las distancias euclidianas entre todos los segmentos de línea que conectan dos nodos de trayectoria consecutivos, por lo tanto, la longitud de la trayectoria se calcula mediante la siguiente ecuación:

$$\text{minimize: } f_1(\mathbf{x}) = \sum_{i=1}^{m} d_i \tag{5.2}$$

Donde $d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$ es la distancia euclidiana entre dos tuplas consecutivas en el array x.

5.3.2. Objetivo de suavidad de la trayectoria

La suavidad de la trayectoria es una característica importante que también es relacionada con el consumo de energía de un robot móvil. Esta suavidad se refiere al número de curvas que tiene la trayectoria. Si una trayectoria tiene muchas curvas cerradas, el consumo de energía aumentará, ya que el robot móvil deberá realizar ajustes continuos.

Por lo tanto, el segundo objetivo es lograr una trayectoria ideal que tenga la menor cantidad posible de curvas cerradas. Para medir la suavidad de la trayectoria, se utiliza la suma de las diferencias de los ángulos internos γ_i con respecto a π entre dos segmentos de trayectoria consecutivos. El objetivo es minimizar esta suma, como se muestra en la Ecuación 5.3.

$$\textbf{minimizar: } f_2(\mathbf{x}) = \sum_{i=3}^{m}(\pi - \gamma_i) \tag{5.3}$$

donde γ_i es definida como:

$$\gamma_i = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right) \tag{5.4}$$

La Ecuación 5.4 denota el ángulo interno entre dos segmentos de trayectoria consecutivos, formado por tres nodos como se muestra en la Figura 5.4. Que se puede explicar de la siguiente forma:

a es la distancia entre (x_{i-2}, y_{i-2}) y (x_{i-1}, y_{i-1}),

b es la distancia entre (x_{i-1}, y_{i-1}) y (x_i, y_i),

c es la distancia entre (x_{i-2}, y_{i-2}) y (x_i, y_i)

5.3.3. Restricciones del problema

El enfoque propuesto para garantizar la seguridad del robot móvil en términos de encontrar trayectorias libres de colisiones debe considerar una restricción de igualdad

$h(\mathbf{x}) = 0$. Esta restricción es crucial para asegurar que el robot móvil evite cualquier colisión durante su trayectoria.

La restricción de igualdad $h(\mathbf{x}) = 0$ se obtiene al evaluar si un segmento de trayectoria cae dentro de un obstáculo. Si esto ocurre, se considera una instancia de colisión y se suma a la restricción. Para penalizar toda la trayectoria en caso de colisión, se multiplica esta suma por un factor β. La ecuación que representa esta restricción es la siguiente:

$$h(\mathbf{x}) = \left(\sum_{k=1}^{m} c(\mathbf{x}_k) \right) \cdot \beta \tag{5.5}$$

$$c(\mathbf{x}_k) = \begin{cases} 0 & \text{if collision-free} \\ n_c & \text{number of collisions in segment} \end{cases} \tag{5.6}$$

Donde x es la tupla de la trayectoria, $c(\mathbf{x}_k)$ es la función que determina el número de colisiones en el segmento de trayectoria entre dos nodos consecutivos, y $\beta \in \mathbb{R}^+$ es un número grande que penaliza toda la trayectoria, que es este caso es 1000.

5.3.4. Formulación del problema multiobjetivo

A partir de las condiciones previamente establecidas, la función multiobjetivo puede expresarse de la siguiente manera:

$$\begin{aligned} \textbf{minimizar: } & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))^T \\ \textbf{sujeto a: } & h(\mathbf{x}) = 0 \\ & \mathbf{x} \in \Omega \end{aligned} \tag{5.7}$$

donde Ω define el espacio de variables de decisión, $\mathbf{F} : \mathbf{x} \to \mathbb{R}^{2 \times m}$ consiste en dos funciones objetivo de valores reales, donde $f_1(\mathbf{x})$ es la longitud de la trayectoria (Ecuación 5.2), $f_2(\mathbf{x})$ es la suavidad de la trayectoria (Ecuación 5.3), y $h(\mathbf{x})$ es la restricción del problema (Ecuación 5.5) que define el espacio de búsqueda factible Ω.

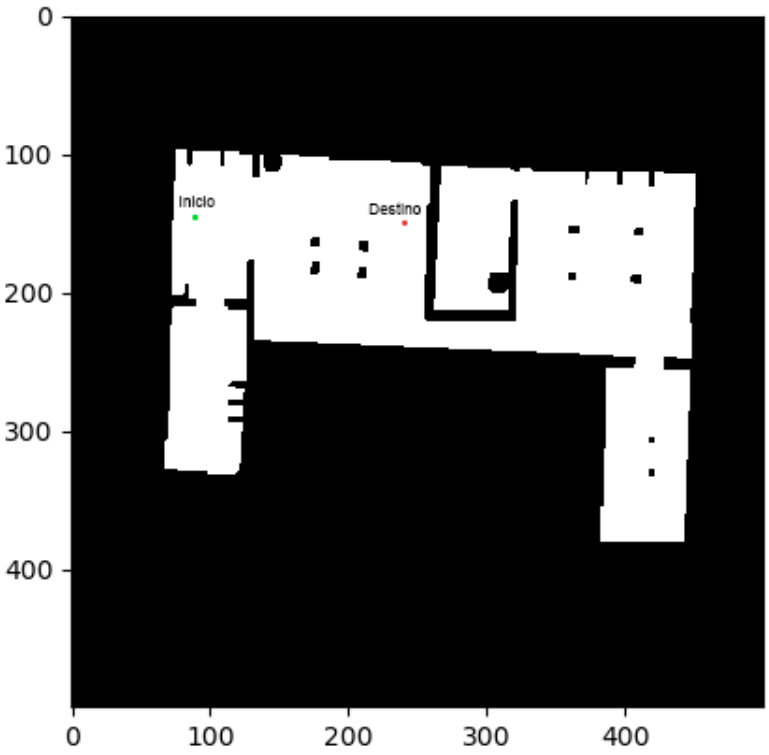

Figura 5.5: Trayectoria dificultad facil

5.4. Definición de las trayectorias

Como ya se mencionó, en los experimentos realizados se presentan tres dificultades para calcular las trayectorias. Estas dificultades varían en nivel de dificultad, desde fácil hasta difícil. A continuación, se definen cada una de las trayectorias que se usaron.

5.4.1. Trayectoria fácil

Considerando que esta trayectoria es muy fácil de calcular para los algoritmos que se se utilizaron, como se puede observar en la Figura 5.5, donde los puntos inicial y destino forman una línea recta, ¿no supondría esto un reto para los algoritmos?

Con base en la literatura, se puede afirmar que efectivamente, una trayectoria en la que los puntos inicial y destino forman una línea recta no supondría un gran reto para

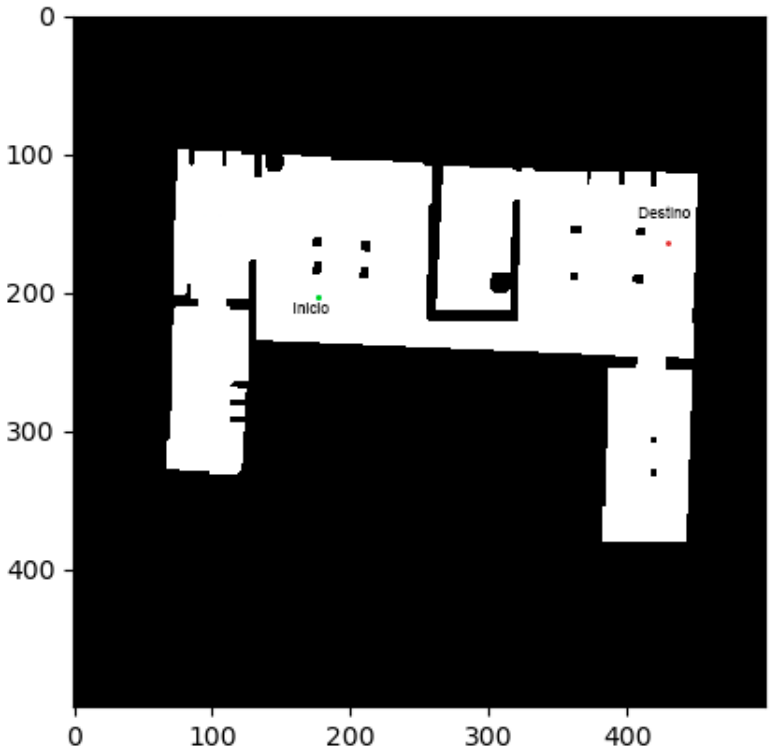

Figura 5.6: Trayectoria dificultad intermedia

los algoritmos utilizados. Este tipo de trayectorias se consideran relativamente simples de calcular, ya que la ruta es directa y no involucra cambios de dirección u obstáculos adicionales.

5.4.2. Trayectoria intermedia

Considerando el nivel de dificultad intermedia, en el que los puntos inicial y destino están más separados y tienen diferentes alturas, y además están situados de forma que el cuarto intermedio interfiere entre ellos como se ve en la Figura 5.6, genera una complicación al momento de calcular la trayectoria.

En situaciones donde los puntos están más distantes y presentan diferencias de altura, los algoritmos de cálculo de trayectorias deben considerar múltiples variables, como la distancia horizontal, la diferencia de alturas y la presencia de obstáculos o interferencias en el camino.

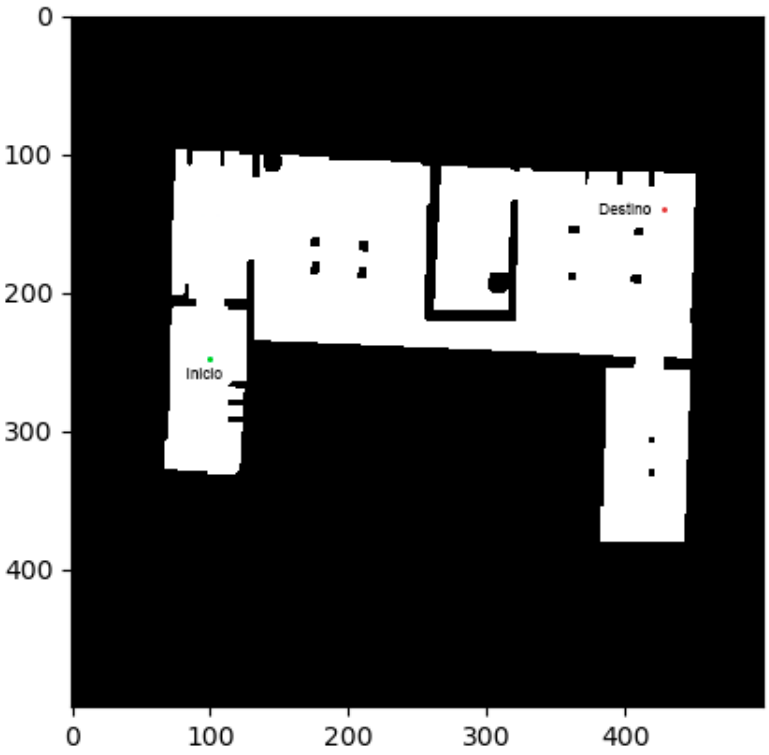

Figura 5.7: Trayectoria dificultad difícil

Estos factores pueden influir en la elección de la ruta óptima y en la determinación de los ángulos y ajustes necesarios para alcanzar el punto destino.

5.4.3. Trayectoria Difícil

Para el nivel difícil, se deben considerar puntos con una separación y diferencia de alturas más notable, además de un mayor número de obstáculos entre ellos. Esto genera una mayor complejidad al calcular la trayectoria, como se observa en la Figura 5.7.

Como se mencionó anteriormente, los factores que contribuyen a esta mayor dificultad implican que los algoritmos deben tomar decisiones sobre la suavidad y longitud de la trayectoria.

5.5. Resultados de los indicadores de rendimiento

Después de haber realizado las ejecuciones de cada uno de los algoritmos utilizados y teniendo los resultados, empezamos con la evaluación de los algoritmos evolutivos multiobjetivo que en este estudio son NSGA-II, MOEA/D y SMS-EMOA.

Para evaluar el desempeño de estos algoritmos en el escenario elegido y en las tres dificultades, se aplicaron tres indicadores de rendimiento: hypervolumen, $\mathcal{IGD}^+$ y $\mathcal{S}$. Estos indicadores se aplicaron a las soluciones no dominadas factibles obtenidas por cada algoritmo. La Tabla 5.1 muestra los valores promedio del indicador de Hipervolumen sobre 40 ejecuciones para cada dificultad en la ruta. La Tabla 5.3 muestra los valores promedio del indicador $\mathcal{S}$ sobre 40 ejecuciones para cada dificultad en la ruta. Por último la Tabla 5.2 muestra los valores promedio del indicador $\mathcal{IGD}^+$ sobre 40 ejecuciones para cada dificultad en la ruta. Cada valor promedio tiene su correspondiente desviación estándar (entre paréntesis) que está entre paréntesis, los mejores resultados se resaltan en negrita.

Tabla 5.1: Resultados Hipervolumen

	Hipervolumen		
Complejidad	NSGA-II	SMS-EMOA	MOEA/D
Fácil	**0.957** (0.057)	0.921 (0.098)	0.923 (0.089)
Intermedia	0.916 (0.143)	0.892 (0.164)	**0.92** (0.141)
Difícil	0.822 (0.107)	0.755 (0.144)	**0.827** (0.105)

También, se utilizó la prueba estadística no paramétrica de Mann-Wilcoxon [49] con un nivel de significancia de 0.05 y la corrección de Bonferroni [50] para determinar cualquier diferencia estadísticamente significativa entre los valores de los promedios de los indicadores usados para cada una de las dificultades. Por lo que si un algoritmo supera estadísticamente a los otros dos para un indicador de rendimiento y dificultad de ruta, se considera el mejor (subrayado).

Tabla 5.2: Resultados $\mathcal{IGD}^+$

	$\mathcal{IGD}^+$		
Complejidad	NSGA-II	SMS-EMOA	MOEA/D
Fácil	**13.108** (2.781)	13.734 (2.397)	13.624 (2.623)
Intermedia	23.373 (5.192)	24.464 (8.623)	**22.105** (5.391)
Difícil	34.041 (11.003)	35.735 (10.944)	**33.116** (12.248)

Tabla 5.3: Resultados $\mathcal{S}$

	$\mathcal{S}$		
Complejidad	NSGA-II	SMS-EMOA	MOEA/D
Fácil	**0.049** (0.044)	0.077 (0.060)	0.067 (0.058)
Intermedia	0.041 (0.052)	**0.036** (0.028)	0.039 (0.051)
Difícil	0.016 (0.051)	0.036 (0.027)	**0.015** (0.034)

Como se puede ver en la Tabla 5.1 en términos del indicador de Hipervolumen, en la ruta de dificultad fácil el algoritmo NSGA-II es el que tiene mejor rendimiento a los otros dos algoritmos, por otro lado tanto en la dificultad Intermedia y difícil MOEA/D a sido el algoritmo que tiene mejor rendimiento.

En la Tabla 5.3 en términos del indicador $\mathcal{S}$, podemos observar que similar a la tabla anterior en la trayectoria fácil el algoritmo NSGA-II es el que tiene el mejor rendimiento, pero en este caso en la trayectoria con dificultad Intermedia el algoritmo SMS-EMOA es el que tiene mejor rendimiento a comparación de los otros dos, por otra parte en la trayectoria de dificultad difícil el algoritmo que tiene mejor rendimiento es MOEA/D. Nota sin embargo, que el indicador $\mathcal{S}$ estima la buena distribución de soluciones al lo largo del frente de Pareto. No obstante, una buena distribución de soluciones sólo es relevante cuando se tiene una buena aproximación al frente de Pareto.

Por último en la Tabla 5.2 en términos del indicador $\mathcal{IGD}^+$, podemos observar que similar a las tablas anteriores el algoritmo que tiene mejor rendimiento en la trayectoria de dificultad fácil es NSGA-II. Por otra parte en la trayectoria de dificultad intermedia el algoritmo con mejor rendimiento es MOEA/D al igual que en la trayectoria de dificultad difícil.

5.6. Comparación de trayectorias

Ahora analicemos las trayectorias obtenidas a través de los algoritmos evolutivos multiobjetivo. Se eligió la ejecución número 20 de estos algoritmos y se representaron gráficamente todas las trayectorias generadas, con el objetivo de examinar su generación y las diferencias entre ellas.

En las Figuras 5.9, 5.15 y 5.21, podemos observar las trayectorias generadas por el algoritmo NSGA-II en las tres dificultades, en las Figuras 5.10, 5.16 y 5.22 podemos ver las trayectorias generadas por el algoritmo SMS-EMOA en las tres dificultades y en las Figuras 5.8, 5.14 y 5.20 se pueden observar las trayectorias generadas por el algoritmo MOEA/D en las tres dificultades. Así mismo en las Figuras 5.12, 5.12 y 5.12 podemos ver las aproximaciones de los frentes de Pareto obtenidos por NSGA-II en el calculo de las trayectorias en cada dificultad, en las Figuras 5.13, 5.19 y 5.25 podemos ver las aproximaciones de los frentes de Pareto obtenidos por SMS-EMOA y en las Figuras 5.11, 5.17 y 5.23 podemos ver las aproximaciones de los frentes de Pareto obtenidos por MOEA/D.

5.7. Optimización asistida con el gemelo digital

Después de haber realizado la evaluación de los algoritmos evolutivos multiobjetivo, obtuvimos al mejor de los tres que en este caso es MOEA/D, por lo que ahora lo que se hizo fue realizar la optimización del torque de los motores y del consumo de batería, para esto el gemelo digital realizó 30 simulaciones de la mejor trayectoria generada por MOEA/D

Figura 5.8: Trayectorias MOEA/D (Dificultad Fácil)

Figura 5.9: Trayectorias NSGA-II (Dificultad Fácil)

Figura 5.10: Trayectorias SMS-EMOA (Dificultad Fácil)

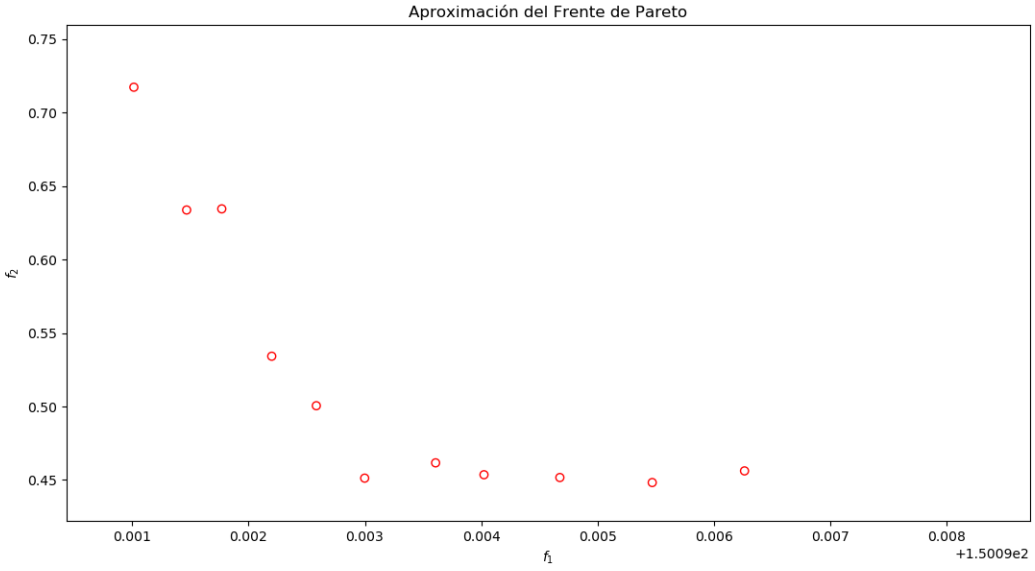

Figura 5.11: Aproximación del frente de Pareto obtenido por MOEA/D (Dificultad Fácil)

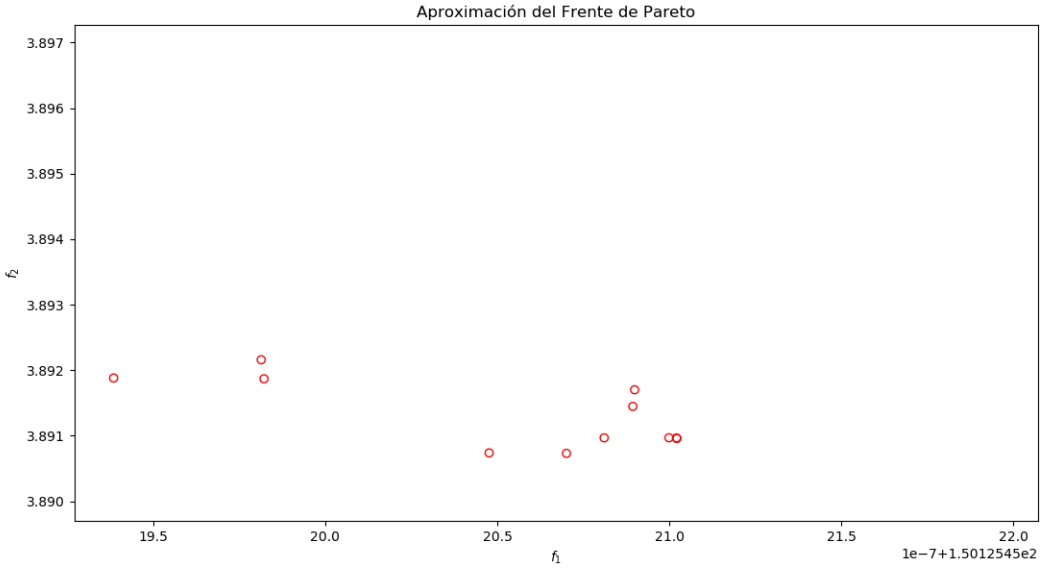

Figura 5.12: Aproximación del frente de Pareto obtenido por NSGA-II (Dificultad Fácil)

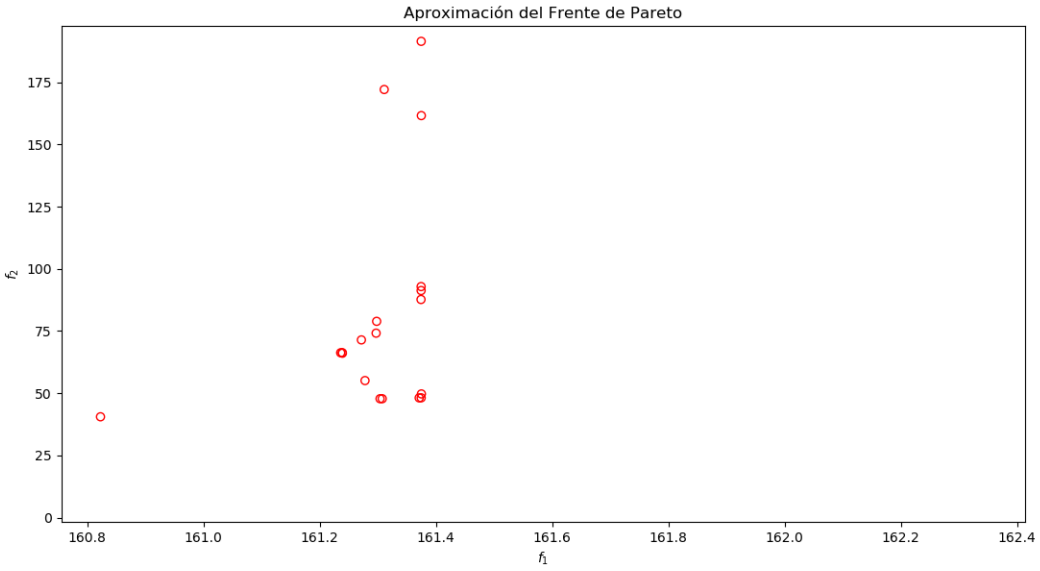

Figura 5.13: Frente de pareto SMS-EMOA (Dificultad Fácil)

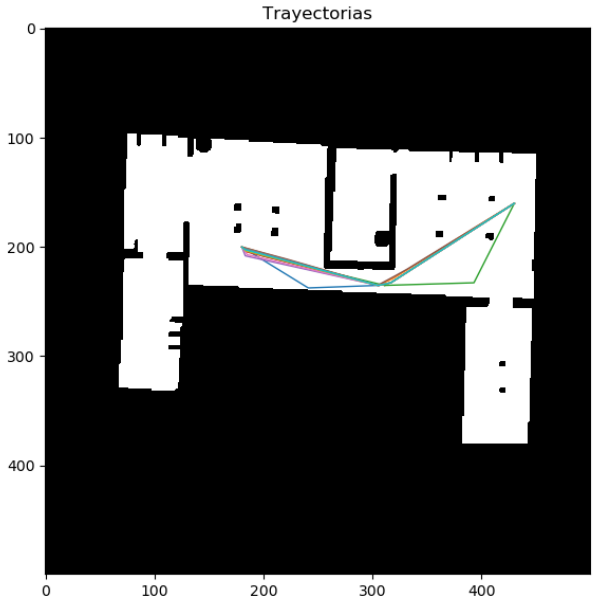

Figura 5.14: Trayectorias MOEA/D (Dificultad Intermedia)

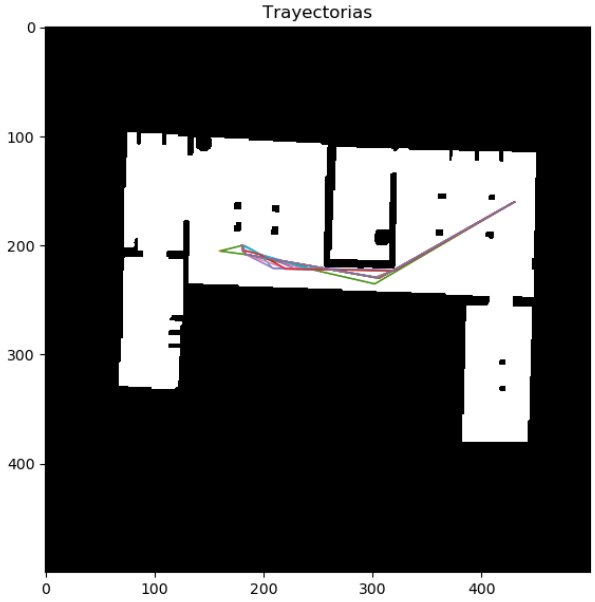

Figura 5.15: Trayectorias NSGA-II (Dificultad Intermedia)

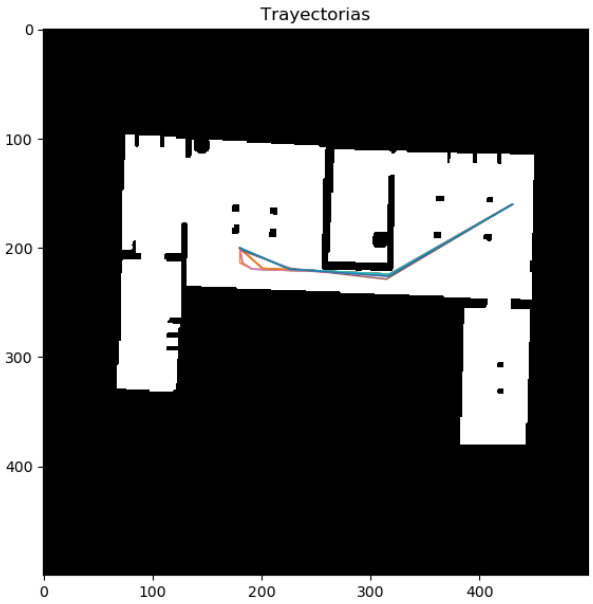

Figura 5.16: Trayectorias SMS-EMOA (Dificultad Intermedia)

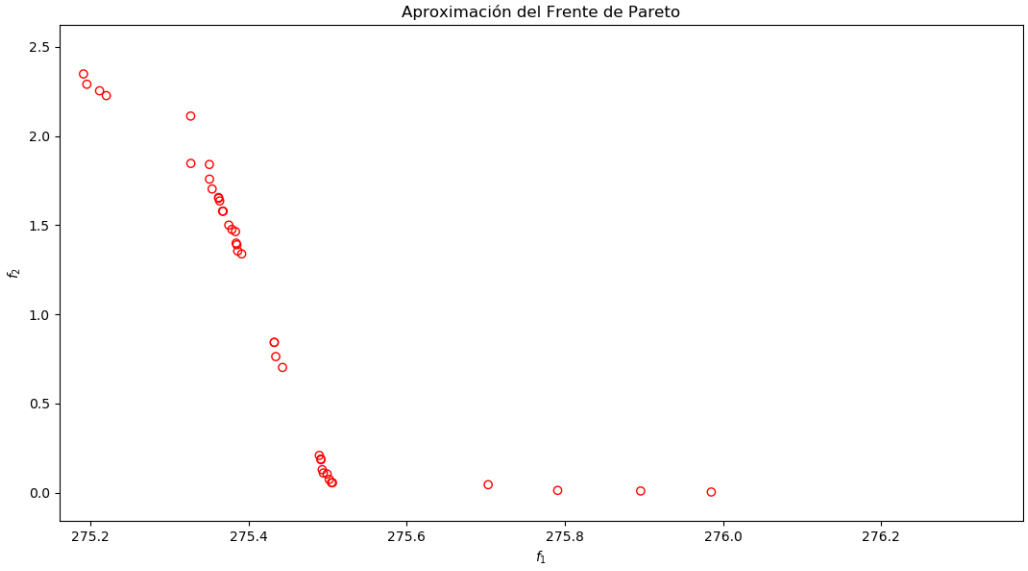

Figura 5.17: Aproximación del frente de Pareto obtenido por MOEA/D (Dificultad Intermedia)

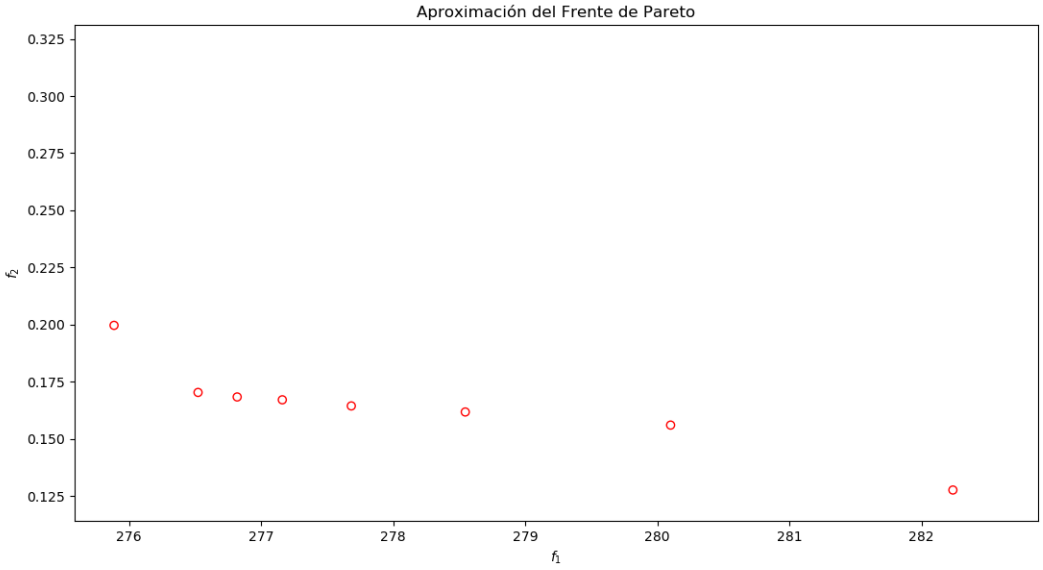

Figura 5.18: Aproximación del frente de Pareto obtenido por NSGA-II (Dificultad Intermedia)

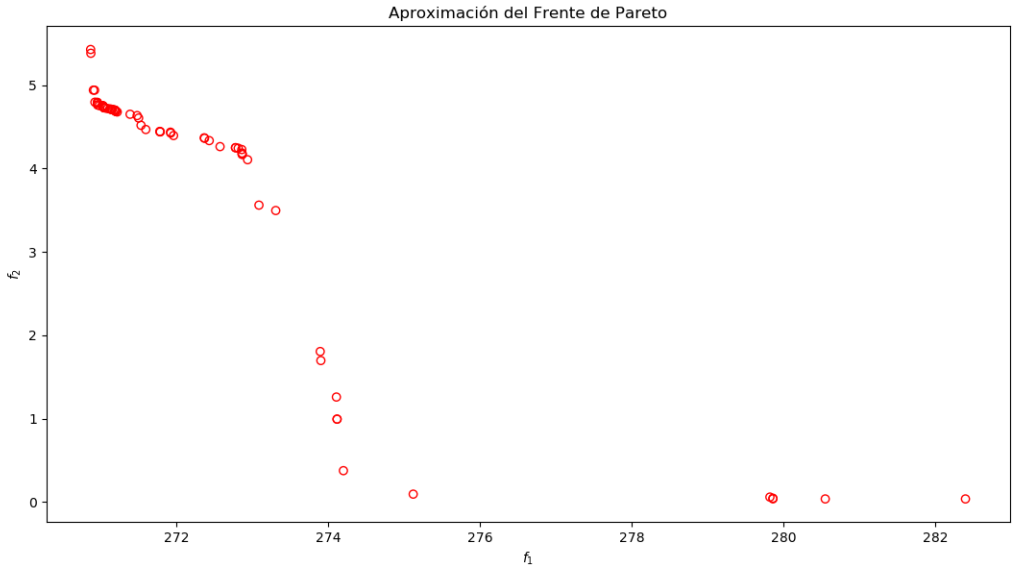

Figura 5.19: Aproximación del frente de Pareto obtenido por SMS-EMOA (Dificultad Intermedia)

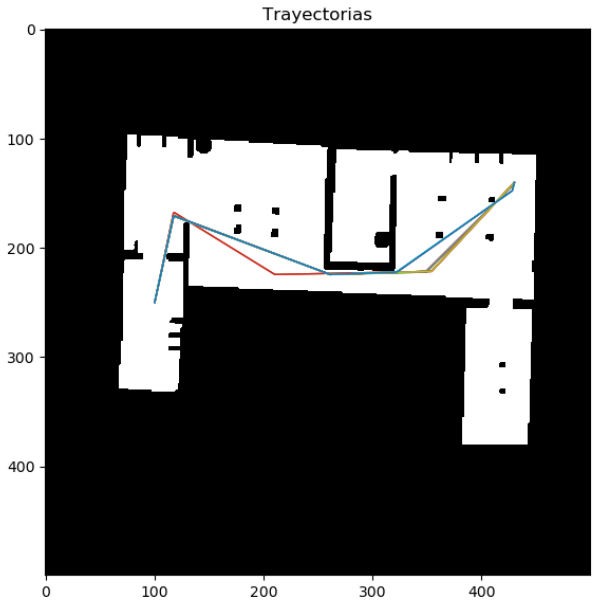

Figura 5.20: Trayectorias MOEA/D (Dificultad Difícil)

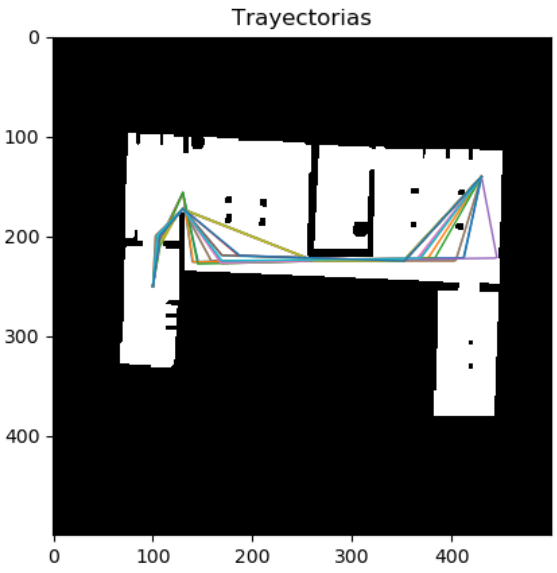

Figura 5.21: Trayectorias NSGA-II (Dificultad Difícil)

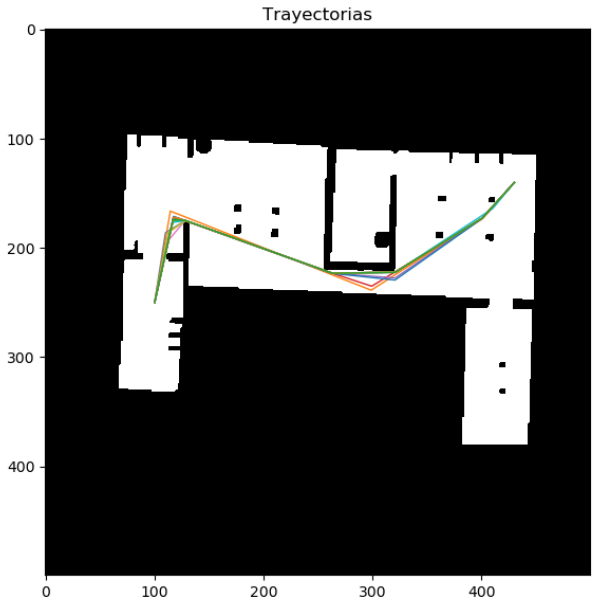

Figura 5.22: Trayectorias SMS-EMOA (Dificultad Difícil)

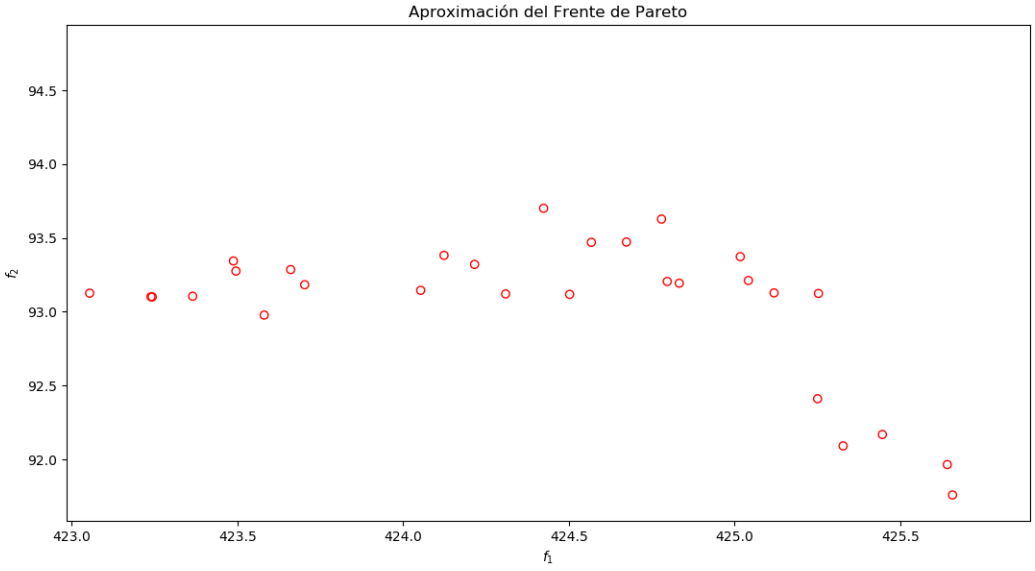

Figura 5.23: Aproximación del frente de Pareto obtenido por MOEA/D (Dificultad Difícil)

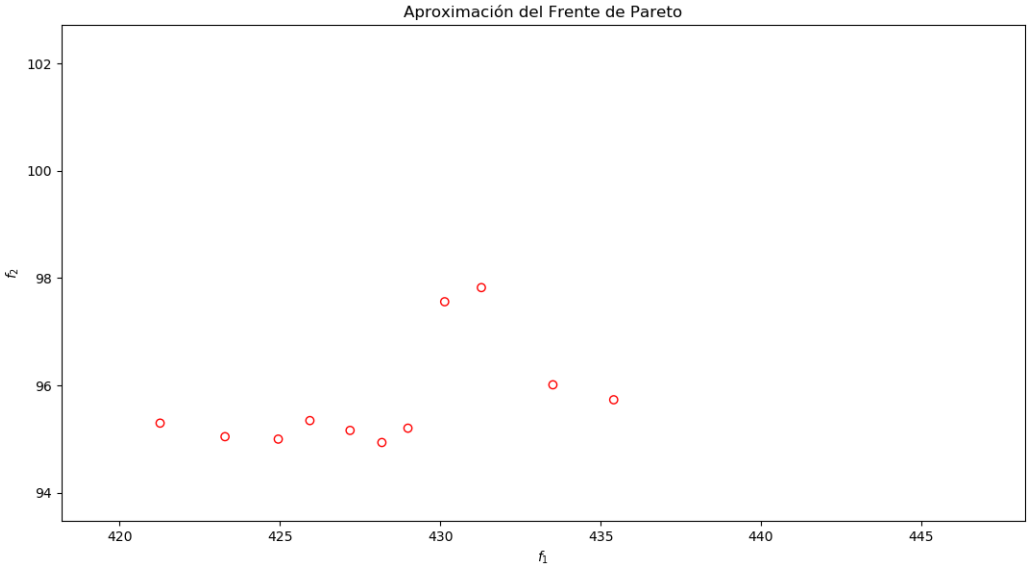

Figura 5.24: Aproximación del frente de Pareto obtenido por NSGA-II (Dificultad Difícil)

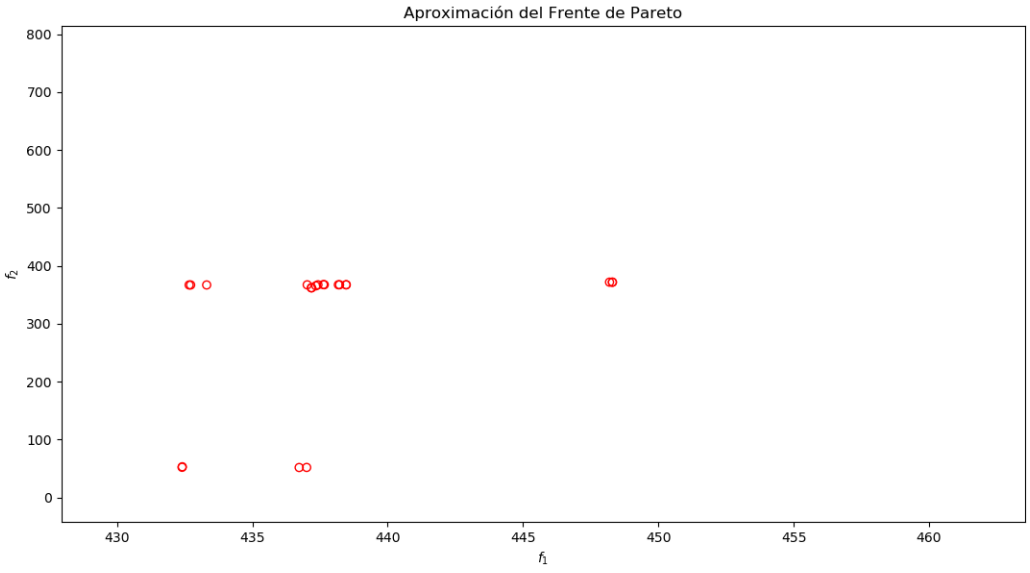

Figura 5.25: Aproximación del frente de Pareto obtenido por SMS-EMOA (Dificultad Difícil)

en cada una de las dificultades, en cada una de estas simulaciones se prueban diferentes valores de torque, al final de las simulaciones se evalúan los resultados en cada una de las simulaciones tomando en cuenta el tiempo que tomó recorrer la trayectoria y el porcentaje de batería consumido, después se elige el valor de torque que satisfaga las necesidades de acuerdo al tiempo que se requiera, que en esta investigación se eligieron los valores de 1 min para la trayectoria facil, 3 min para la trayectoria de dificultad intermedia y 6 min para la difícil.

Los resultados que dio el gemelo digital para cada una de las trayectorias los podemos ver en la Tabla 5.4, donde tenemos el torque requerido para ejecutar la trayectoria en el tiempo que se eligió y el porcentaje de batería que se requiere para la ejecución de la trayectoria.

Tabla 5.4: Resultados Gemelo Digital

Complejidad	Torque (ft/lbs)	Bateria (%)	Tiempo (min.)
Fácil	1.80	2.35	1
Intermedia	2.20	7.41	3
Difícil	2.85	13.97	6

5.8. Comparación metodología propuesta con tecnologías tradicionales

Para esta sección después de haber analizado los resultados de los algoritmos evolutivos multiobjetivo y del gemelo digital que son el objetivo de este trabajo y como comparación adicional, presentamos una comparación de todos los algoritmos que se usaron en este trabajo que son los algoritmos clásicos confirmados por A*, RRT y Dijkstra, tenemos los bioinspirados que son el Algoritmo Genético e Evolución Diferencial y por último los

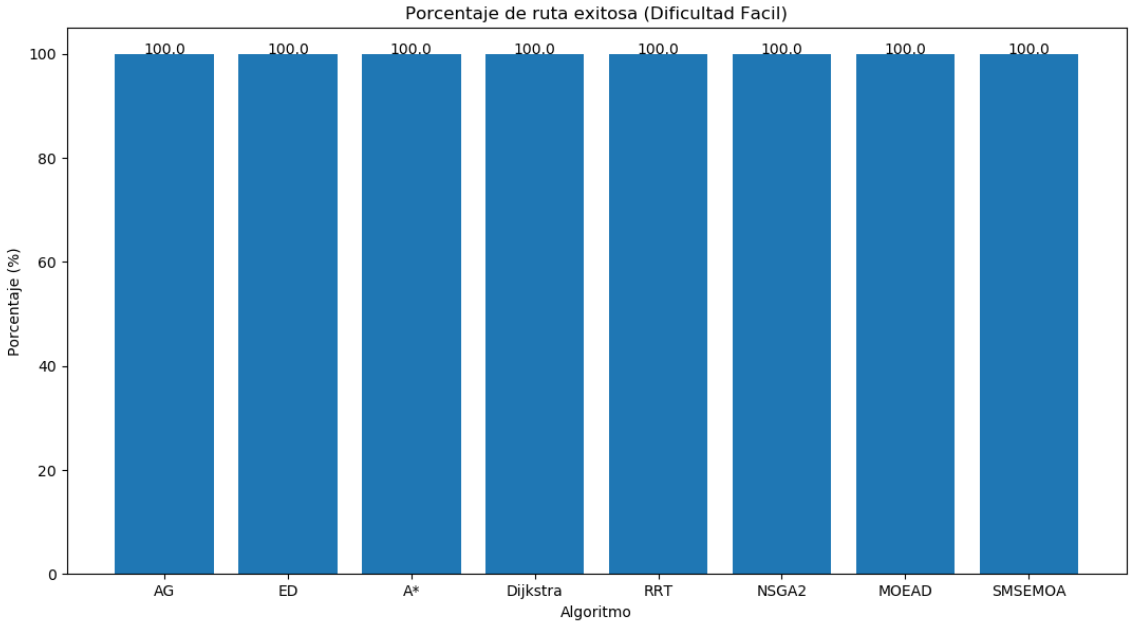

Figura 5.26: Porcentaje de ruta exitosa (Dificultad fácil)

algoritmos evolutivos multiobjetivo que son NSGA-II, MOEA/D y SMS-EMOA.

La comparación la realizamos de acuerdo a los tres objetivos que utilizamos en este trabajo que son: la distancia de la trayectoria, la suavidad y la evasión de obstáculos. Empezamos evaluando a los algoritmos de acuerdo a la evasión de obstáculos en las Figuras 5.26, 5.27 y 5.28, podemos observar el porcentaje de trayectoria correcta, es decir el porcentaje de las trayectorias calculadas que no chocan con algún obstáculo, como podemos observar todos los algoritmos tiene un porcentaje por arriba del 90 % lo que nos indica que los algoritmos si realizan su tarea de buena manera.

Por otro lado para poder evaluar los otros dos objetivos, hacemos uso del evaluador de objetivos que tenemos en los algoritmos evolutivos multiobjetivo, lo que se hace es tomar las trayectorias y evaluarlas con los objetivos de los algoritmos multiobjetivo. Los resultados

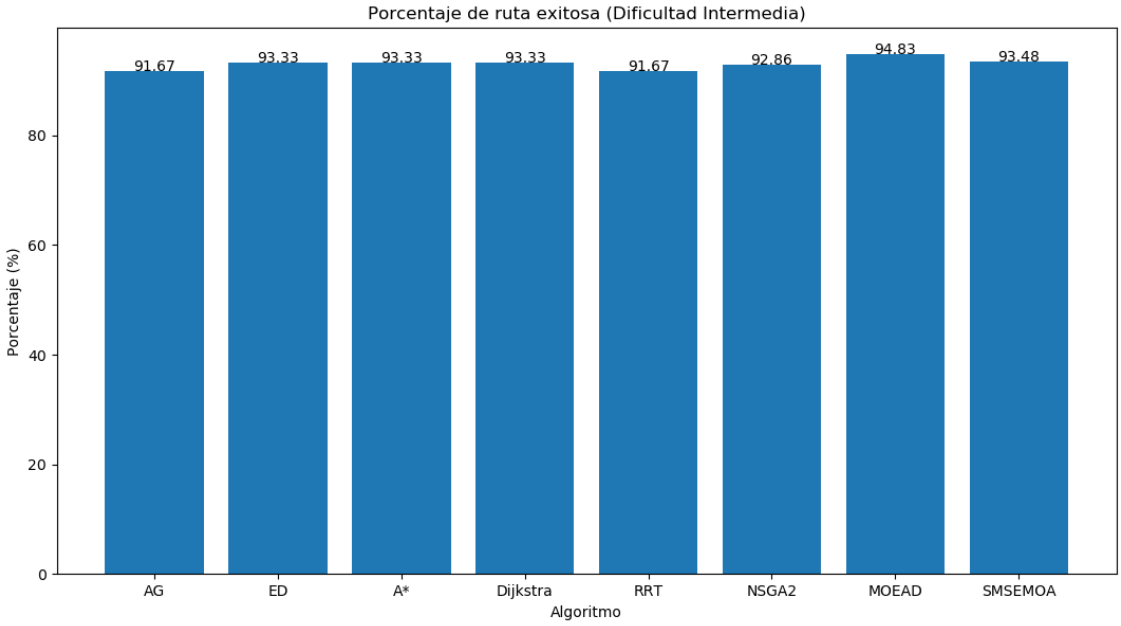

Figura 5.27: Porcentaje de ruta exitosa (Dificultad intermedia)

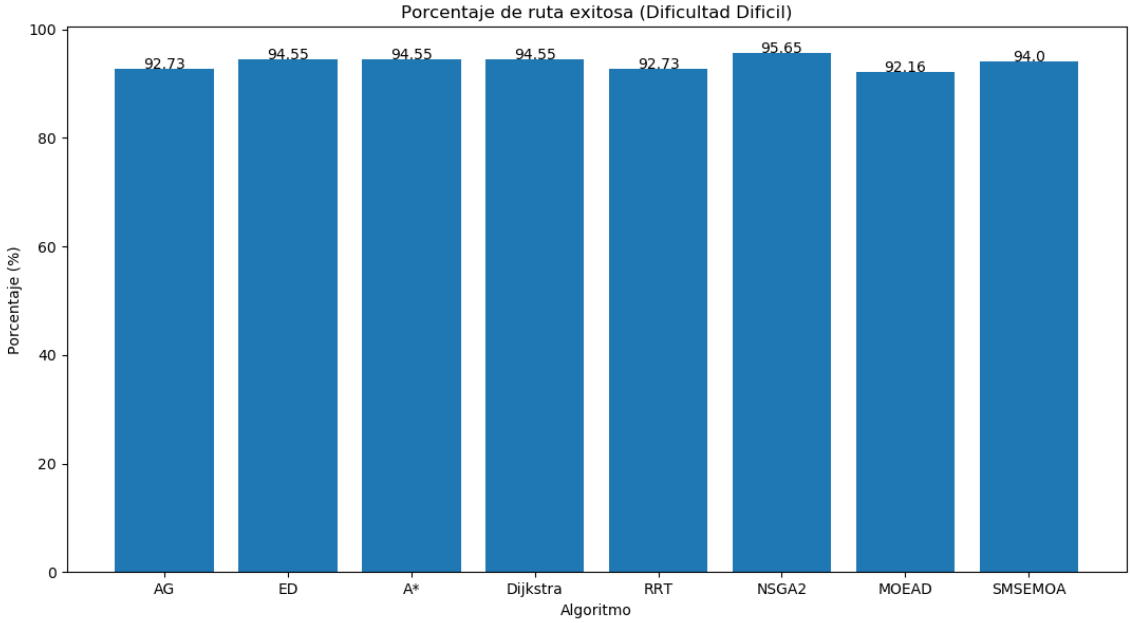

Figura 5.28: Porcentaje de ruta exitosa (Dificultad difícil)

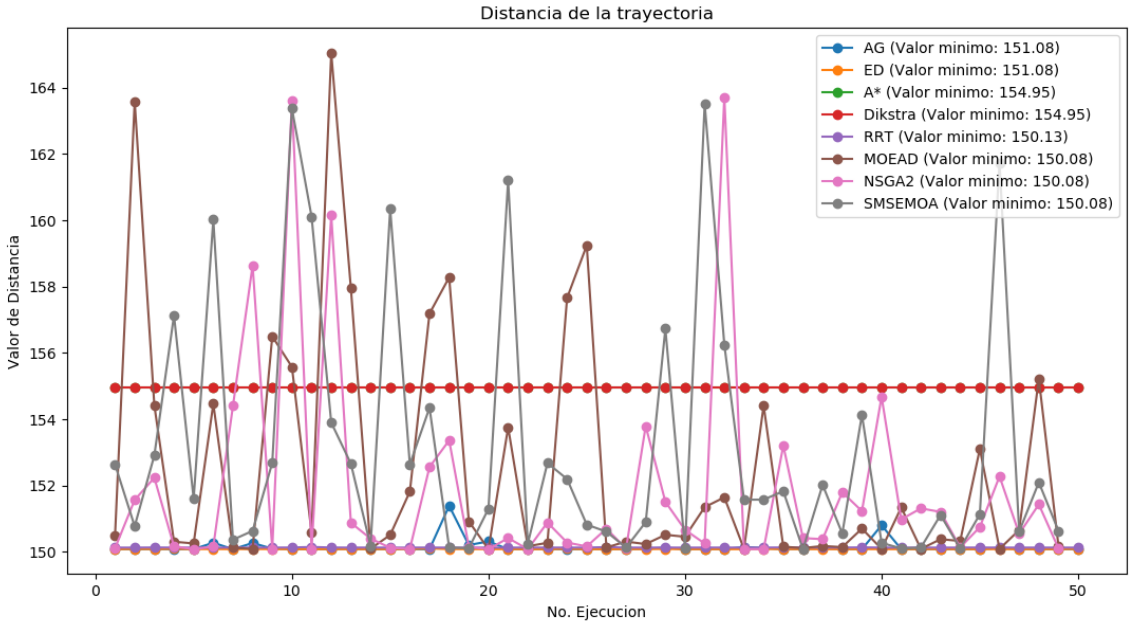

Figura 5.29: Valores de la distancia (Dificultad Fácil)

obtenidos para la distancia de la trayectoria pueden ser observados en las Figuras 5.29, 5.30 y 5.31. Para el objetivo de la suavidad, los resultados pueden ser observados en las Figuras 5.32, 5.33 y 5.34. Como se puede ver los mejores resultados en estos objetivos los tienen los algoritmos evolutivos multiobjetivo.

Al analizar esta comparación, podemos concluir que los algoritmos tradicionales demuestran una buena capacidad para realizar su tarea. Además, podemos utilizar algoritmos evolutivos de un solo objetivo para calcular trayectorias, y estos también logran un buen desempeño en esta tarea. Sin embargo, cuando el problema se vuelve multiobjetivo, podemos observar que, aunque todos los algoritmos pueden calcular la trayectoria entre un punto A y un punto B, solo los algoritmos multiobjetivo producen los resultados esperados en términos de optimización multiobjetivo. En resumen, los algoritmos multiobjetivo son más adecuados para resolver eficientemente problemas de optimización con múltiples objetivos.

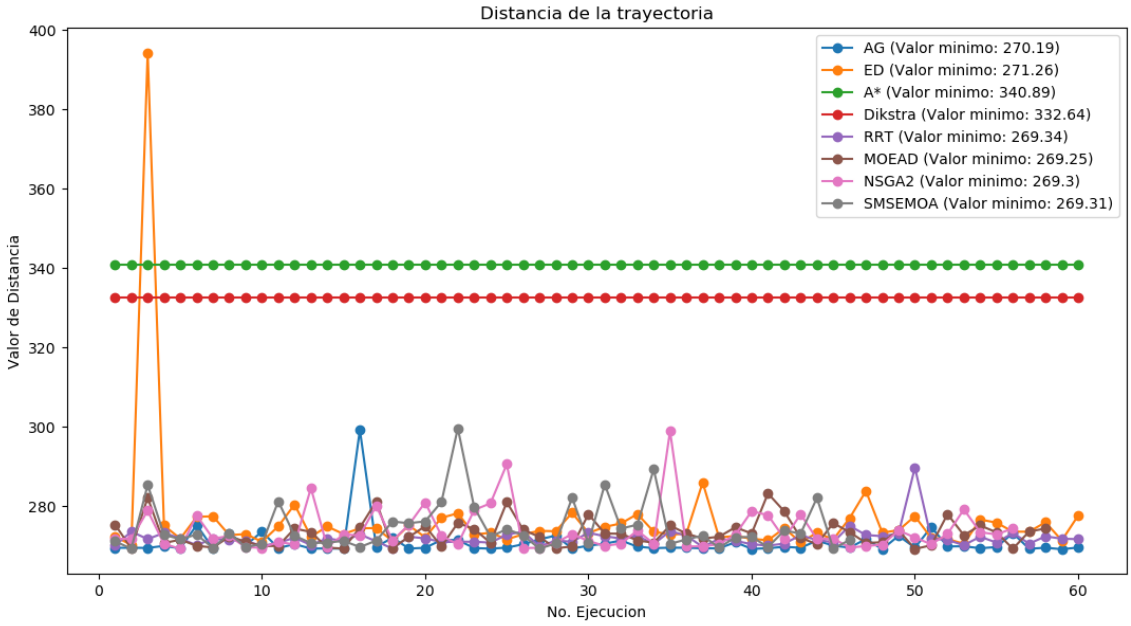

Figura 5.30: Valores de la distancia (Dificultad Intermedia)

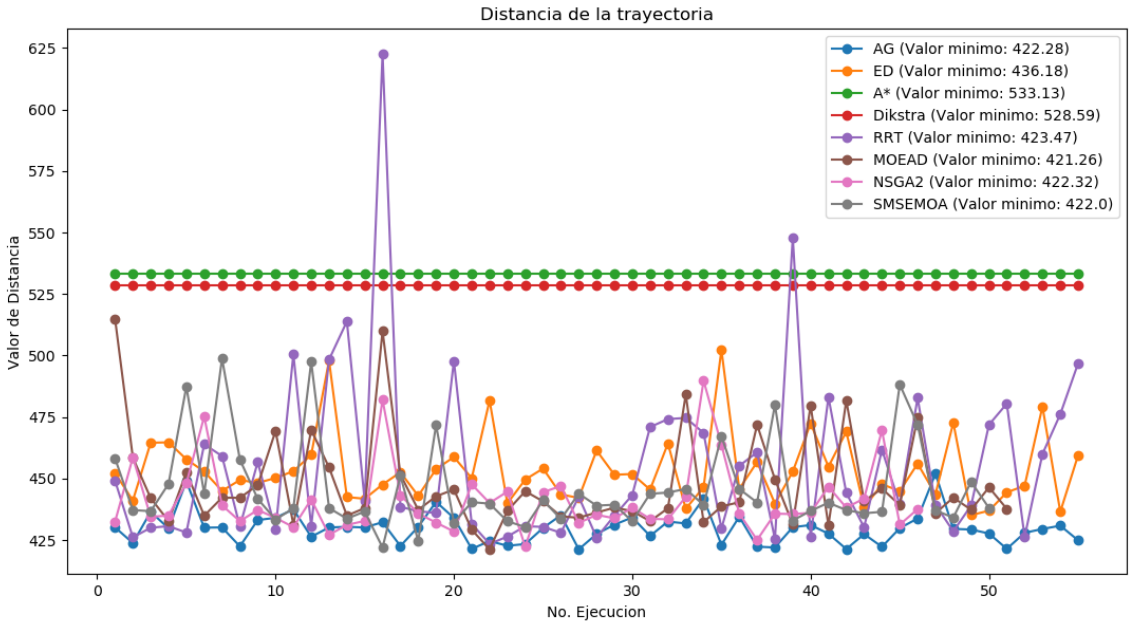

Figura 5.31: Valores de la distancia (Dificultad Difícil)

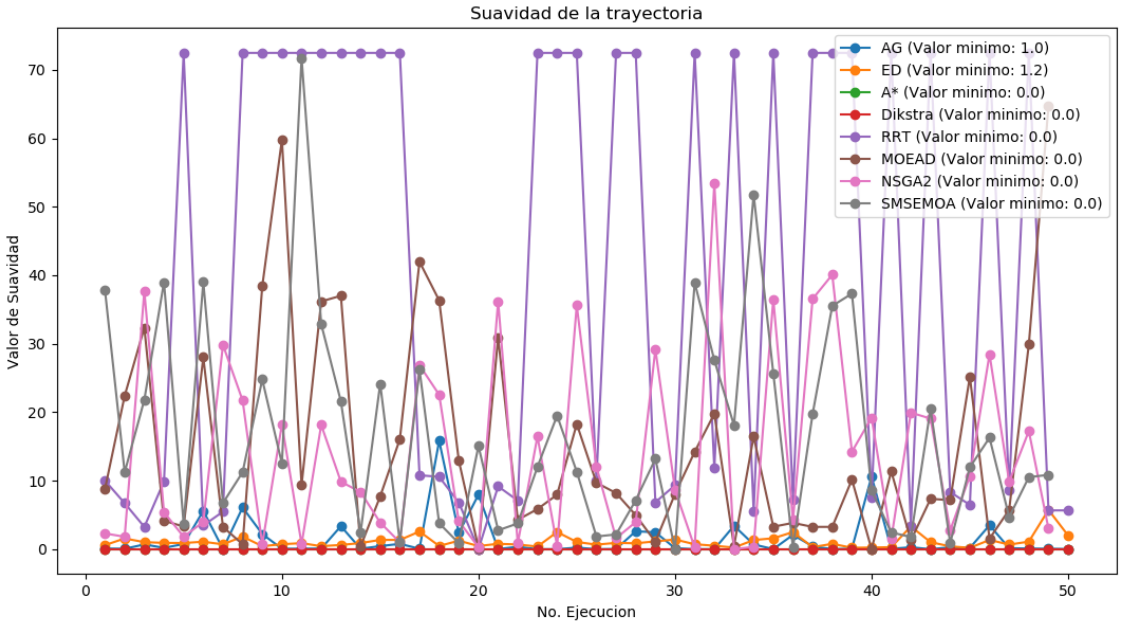

Figura 5.32: Valores de la suavidad (Dificultad Fácil)

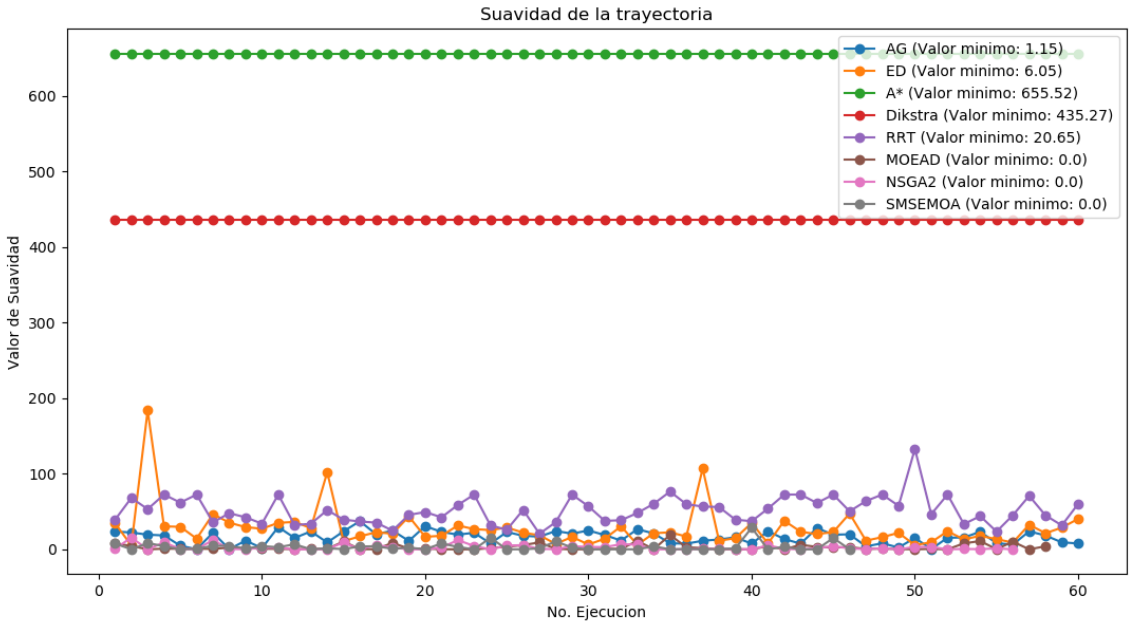

Figura 5.33: Valores de la suavidad (Dificultad Intermedia)

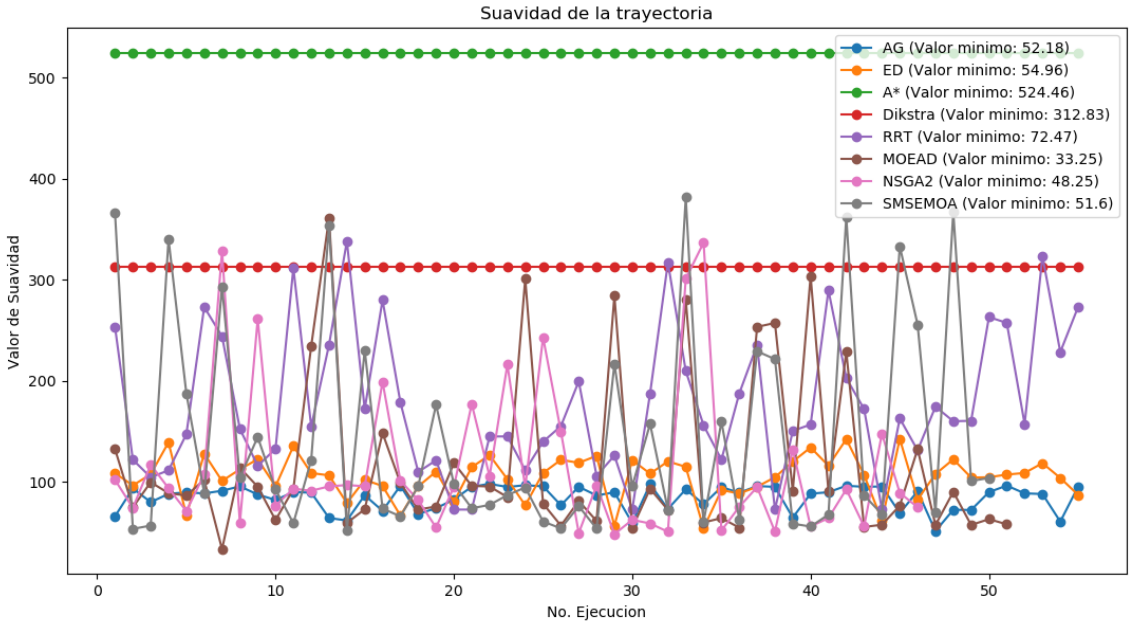

Figura 5.34: Valores de la suavidad (Dificultad Difícil)

5.9. Pruebas de la metodología en otros entornos

Después de haber probado la metodología propuesta con un entorno complejo mediante el cálculo de tres trayectorias de diferente dificultad, ahora la probaremos la metodología en dos entornos diferentes y con una trayectoria a calcular que podemos definir cómo de dificultad Intermedia.

5.9.1. Entorno de un hospital

El primer entorno adicional en el que vamos a probar la metodología propuesta es de un hospital como se puede ver en la Figura 5.35, este entorno fue diseñado por Amazon y es de uso libre, en este entorno se probó a calcular una trayectoria de dificultad Intermedia y se realizaron 30 ejecuciones para cada algoritmo, los resultados los vemos a continuación.

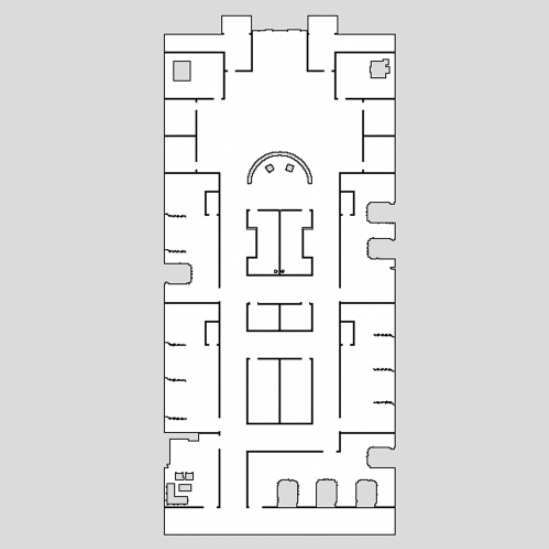

Figura 5.35: Entorno de un hospital

Indicadores de rendimiento

Después de realizar las ejecuciones de cada algoritmo tenemos los resultados del análisis mediante los indicadores de rendimiento de los algoritmos evolutivos multiobjetivo. Cómo en el entorno principal se utilizaron los indicadores de hipervolumen, IGD+ y S.

La Tabla 5.5 muestra los valores promedio de los indicadores sobre 30 ejecuciones, cada valor tiene su desviación estándar (entre paréntesis) y los mejores resultados se resaltan en negrita. Igualmente se usó la prueba estadística no paramétrica de Mann-Wilcoxon con un nivel de significancia de 0.05 y la corrección de Bonferroni para identificar cualquier diferencia estadísticamente significativa entre los valores el cual se identifica con un subrayado.

Cómo se puede ver en la Tabla 5.5 en términos de los tres indicadores el algoritmo NSGA-II es el que tiene mejor rendimiento, además de que es significativamente diferente.

Comparación de trayectorias

Ahora veremos las trayectorias generadas por los algoritmos evolutivos multiobjetivo, se eligió la ejecución número 15 de estos algoritmos y se representaron gráficamente las

Tabla 5.5: Resultados indicadores de rendimiento

	Resultados		
Indicador	NSGA-II	SMS-EMOA	MOEA/D
Hv	**0.849** (0.110)	0.798 (0.170)	0.823 (0.107)
$\mathcal{IGD}^+$	**12.085** (5.870)	13.866 (6.717)	12.145 (6.011)
$\mathcal{S}$	**0.016** (0.014)	0.034 (0.047)	0.021 (0.019)

trayectorias generadas y las aproximaciones del frente de Pareto, con el objetivo de examinar cómo se generaron y las diferencias que pueda haber.

En la Figura 5.36 podemos ver las trayectorias generadas por el algoritmo NSGA-II, en la Figura 5.37 podemos ver las trayectorias del algoritmo MOEA/D y en la Figura 5.38 podemos ver las trayectorias del algoritmo SMS-EMOA. Así mismo en la Figura 5.39 podemos ver la aproximación del frente de Pareto obtenida por NSGA-II, en la Figura 5.40 podemos ver la aproximación del frente de Pareto de MOEA/D y en la Figura 5.41 vemos la aproximación del frente de Pareto de SMS-EMOA.

Optimización con el gemelo digital

Después de haber realizado la evaluación de los algoritmos evolutivos multiobjetivo, obtuvimos que el mejor es NSGA-II, con esto en cuenta realizamos la optimización del torque y el consumo de batería con el gemelo digital, mediante la ejecución de 30 simulaciones de la mejor trayectoria generada por NSGA-II y con un tiempo de ejecución establecido de 4 minutos.

Tabla 5.6: Resultados Gemelo Digital (Hospital)

	Torque (ft/lbs)	Bateria (%)	Tiempo (min.)
	2.28	8.16	4

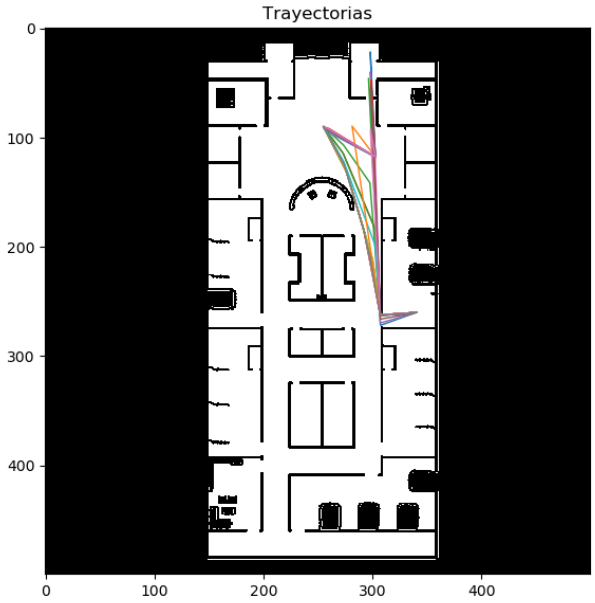

Figura 5.36: Trayectorias NSGA-II (Hospital)

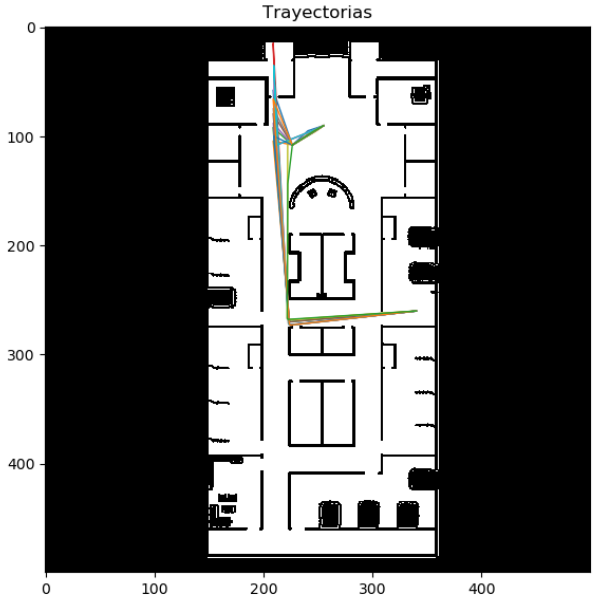

Figura 5.37: Trayectorias MOEA/D (Hospital)

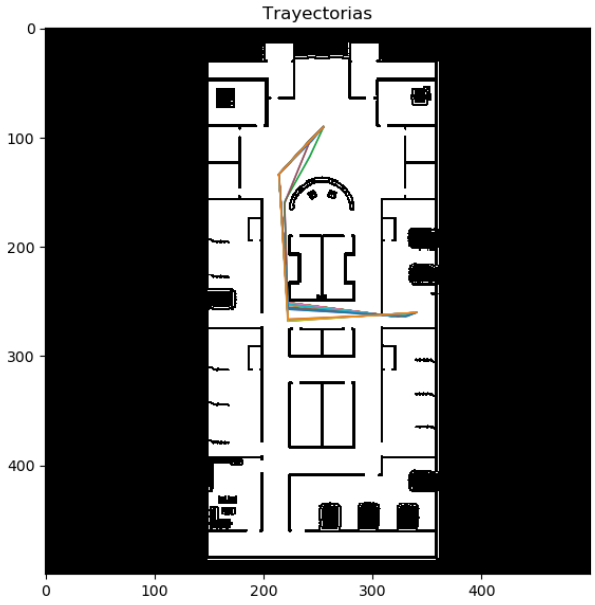

Figura 5.38: Trayectorias SMS-EMOA (Hospital)

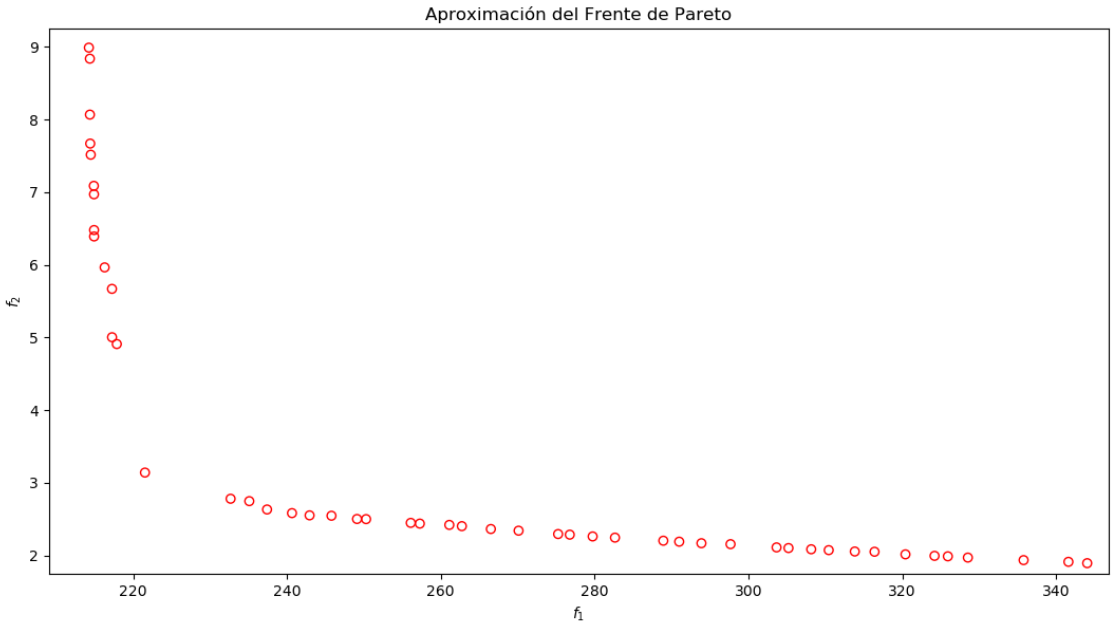

Figura 5.39: Aproximación del frente de Pareto NSGA-II (Hospital)

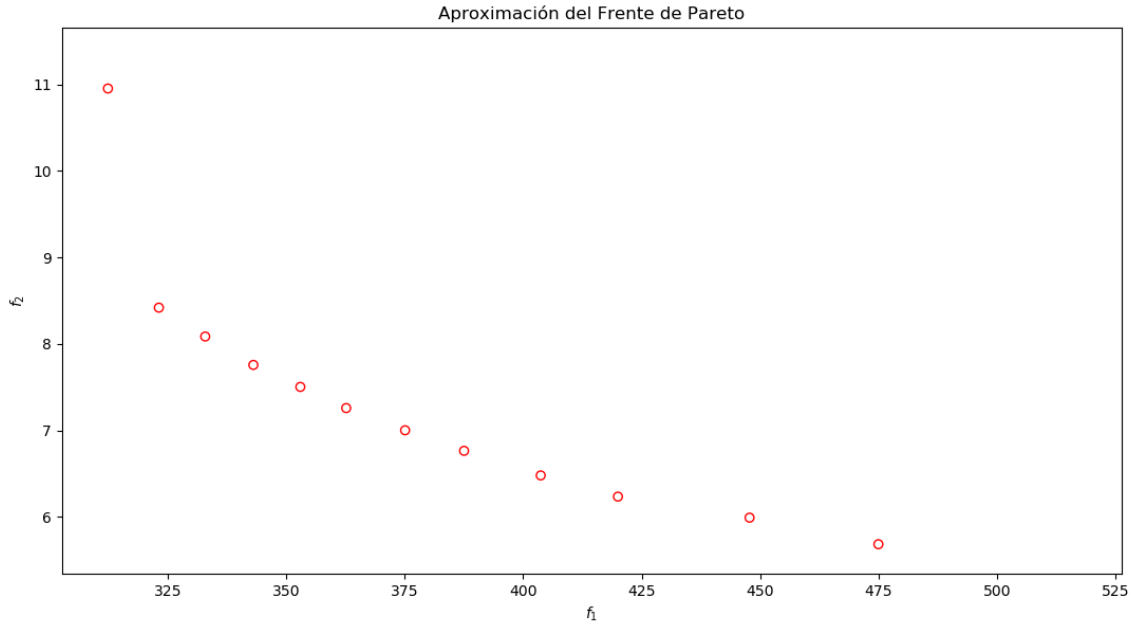

Figura 5.40: Aproximación del frente de Pareto MOEA/D (Hospital)

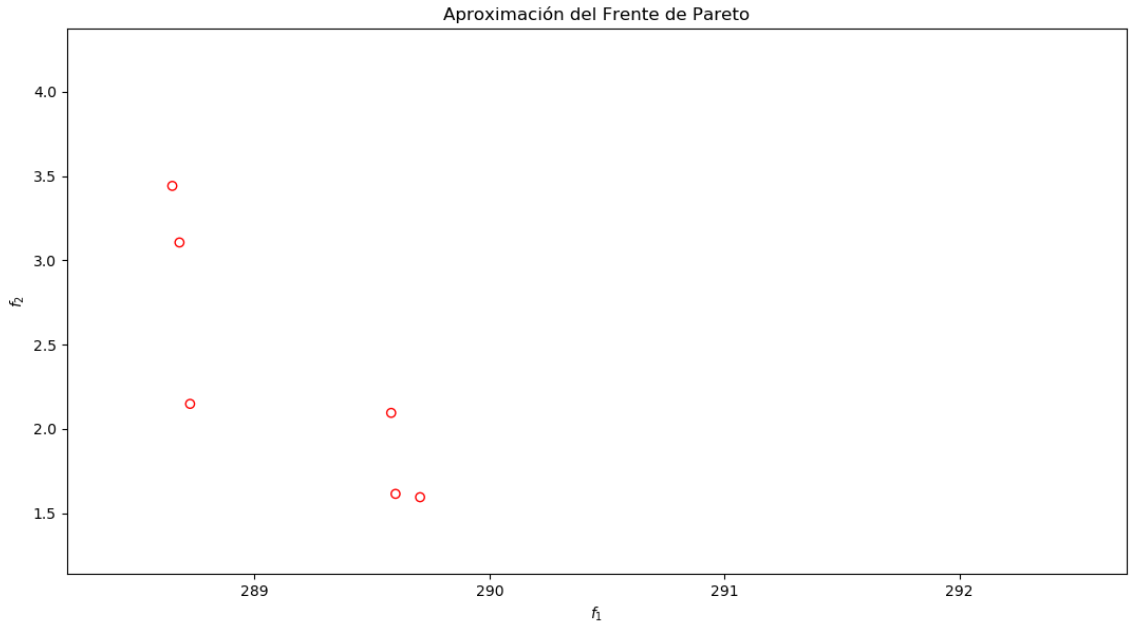

Figura 5.41: Aproximación del frente de Pareto SMS-EMOA (Hospital)

Los resultados que dió el gemelo digital lo podemos ver en la Tabla 5.6, dónde vemos el torque requerido para ejecutar la trayectoria en el tiempo establecido y el consumo de batería que se requiere para su ejecución.

Comparación con tecnologías tradicionales

Cómo comparación adicional, presentamos la comparación de todos los algoritmos utilizados, la comparación la realizamos de acuerdo a tres objetivos utilizados en este trabajo que son: la distancia de la trayectoria, la suavidad y la evasión de obstáculos.

Empezamos con la evaluación de la evasión de obstáculos, en la Figura 5.42 podemos observar el porcentaje de trayectoria correcta, es decir el porcentaje de las trayectorias calculadas que no chocan con algún obstáculo. Podemos observar que el porcentaje es arriba del 90 % lo que indica que los algoritmos hacen bien su tarea.

Para los otros dos objetivos como en la sección anterior hacemos uso del evaluador de objetivos que tenemos en los algoritmos evolutivos multiobjetivo. Los resultados de la distancia de la trayectoria los podemos ver en la Figura 5.43 y los resultados de la suavidad los podemos ver en la Figura 5.44. Cómo se puede ver los mejores resultados en estos objetivos los tienen los algoritmos evolutivos multiobjetivo.

5.9.2. Entorno de una librería

El segundo entorno adicional en el que vamos a probar la metodología propuesta es de una librería como se puede ver en la Figura 5.45, este entorno también fue diseñado por Amazon y es de uso libre, igual que en el entorno anterior, en este entorno se probó a calcular una trayectoria de dificultad Intermedia y se realizaron 30 ejecuciones para cada algoritmo, los resultados los vemos a continuación.

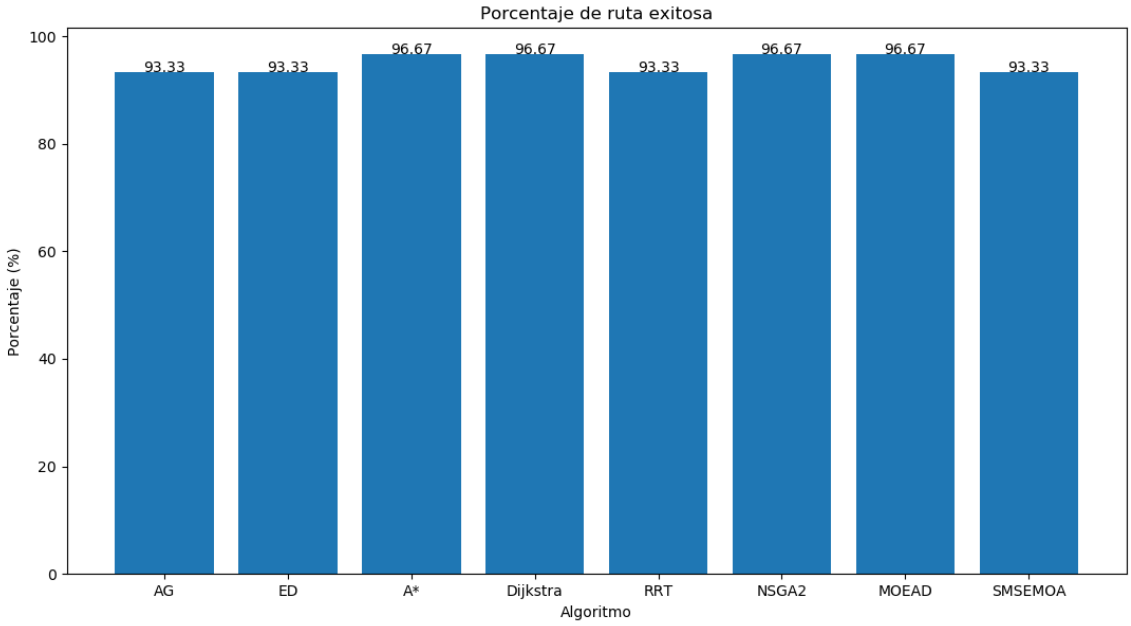

Figura 5.42: Porcentaje de trayectoria exitosa (Hospital)

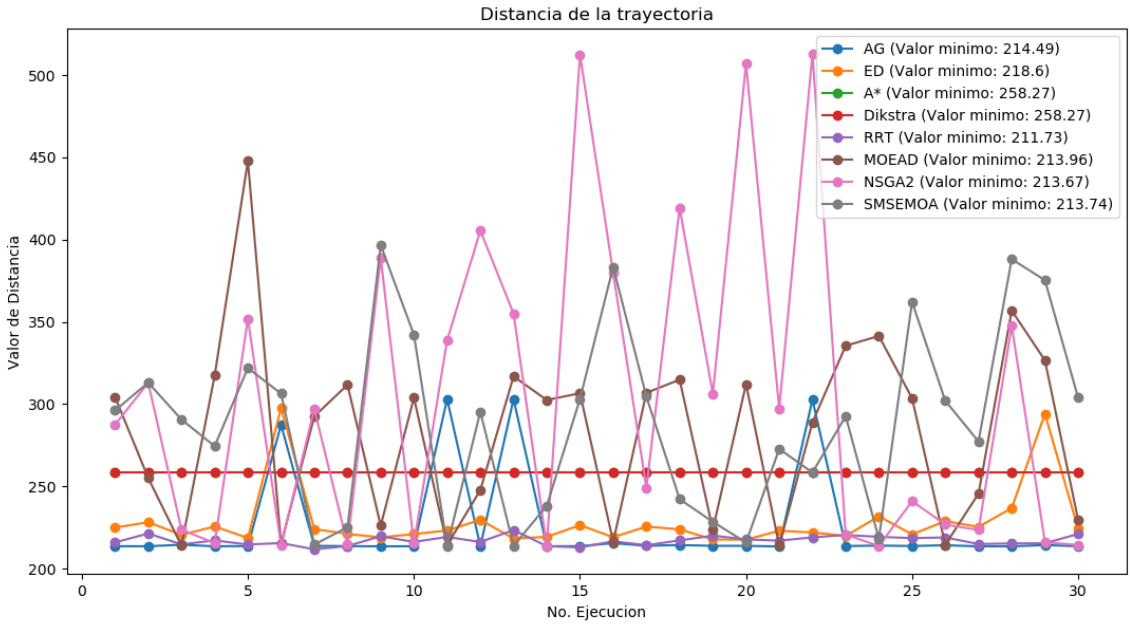

Figura 5.43: Distancia de la trayectoria (Hospital)

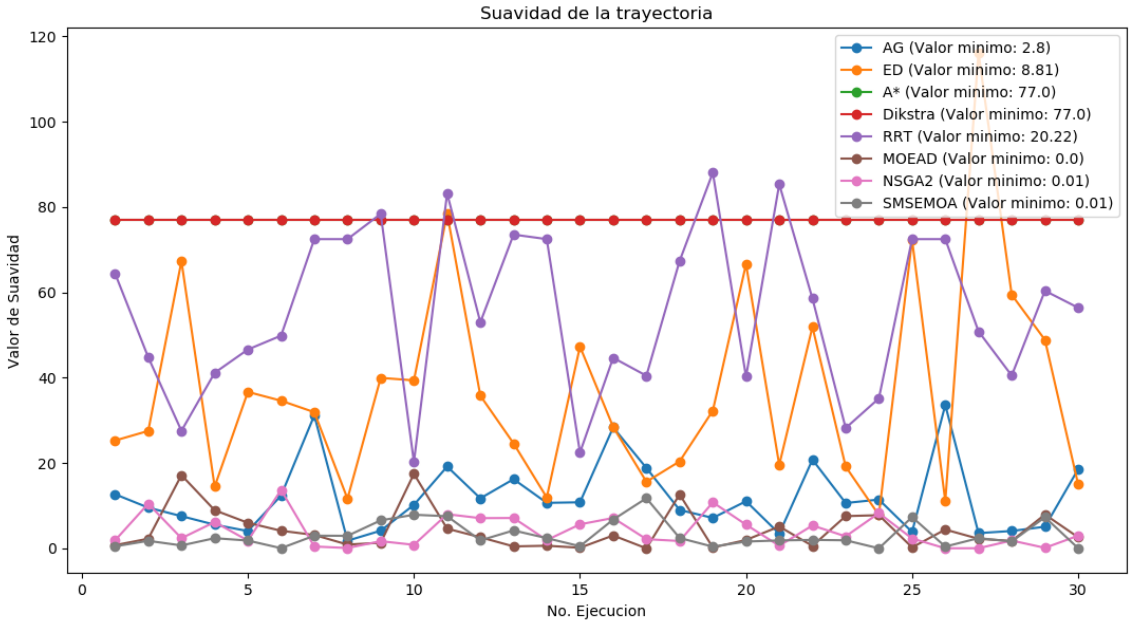

Figura 5.44: Suavidad de la trayectoria (Hospital)

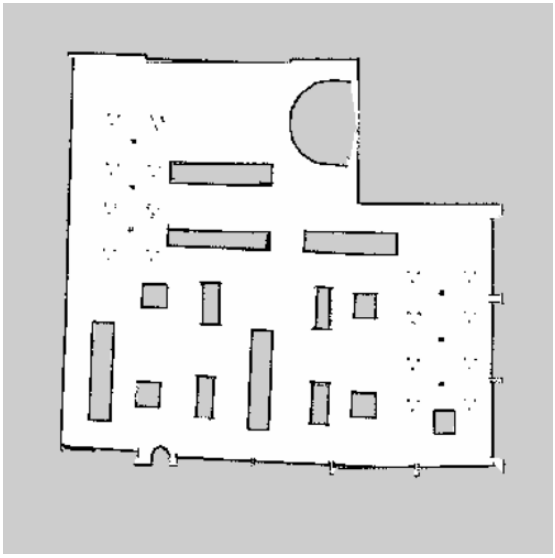

Figura 5.45: Entorno de una librería

Indicadores de rendimiento

Después de realizar las ejecuciones de cada algoritmo tenemos los resultados del análisis mediante los indicadores de rendimiento de los algoritmos evolutivos multiobjetivo.

Cómo en el entorno principal se utilizaron los indicadores de hipervolumen, IGD+ y S. La Tabla 6 muestra los valores promedio de los indicadores sobre 30 ejecuciones, cada valor tiene su desviación estándar (entre paréntesis) y los mejores resultados se resaltan en negrita. Igualmente se usó la prueba estadística no paramétrica de Mann-Wilcoxon con un nivel de significancia de 0.05 y la corrección de Bonferroni para identificar cualquier diferencia estadísticamente significativa entre los valores el cual se identifica con un subrayado.

Tabla 5.7: Resultados indicadores de rendimiento

	Resultados		
Indicador	NSGA-II	SMS-EMOA	MOEA/D
Hv	0.782 (0.110)	0.774 (0.170)	**0.789** (0.107)
$\mathcal{IGD}^+$	**12.085** (5.870)	13.866 (6.717)	12.145 (6.011)
$\mathcal{S}$	**<u>0.016</u>** (0.014)	0.034 (0.047)	0.021 (0.019)

Cómo se puede ver en la Tabla 5.8 en términos de los tres indicadores el algoritmo NSGA-II es el que tiene mejor rendimiento, además de que es significativamente diferente.

Comparación de trayectorias

Ahora veremos las trayectorias generadas por los algoritmos evolutivos multiobjetivo, se eligió la ejecución número 15 de estos algoritmos y se representaron gráficamente las trayectorias generadas y las aproximaciones del frente de Pareto, con el objetivo de examinar cómo se generaron y las diferencias que pueda haber.

En la Figura 5.46 podemos ver las trayectorias generadas por el algoritmo NSGA-II, en la Figura 5.48 podemos ver las trayectorias del algoritmo MOEA/D y en la Figura

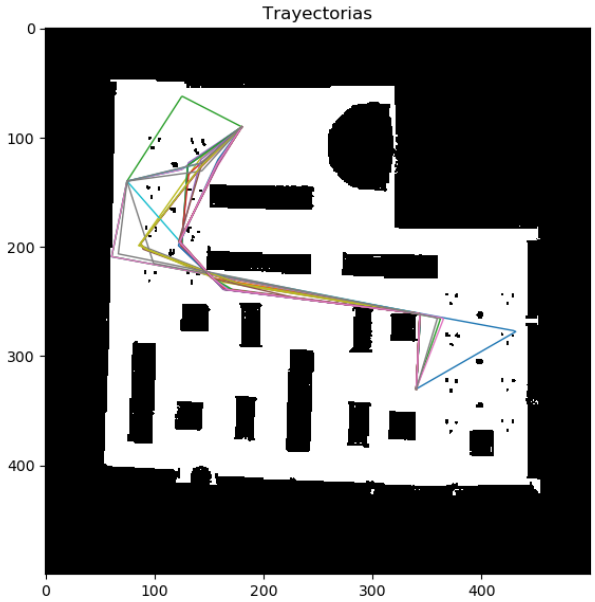

Figura 5.46: Trayectorias NSGA-II (Librería)

los

5.48 podemos ver las trayectorias del algoritmo SMS-EMOA. Así mismo en la Figura 5.49 podemos ver la aproximación del frente de Pareto obtenida por NSGA-II, en la Figura 5.50 podemos ver la aproximación del frente de Pareto de MOEA/D y en la Figura 5.51 vemos la aproximación del frente de Pareto de SMS-EMOA.

Optimización con el gemelo digital

Después de haber realizado la evaluación de los algoritmos evolutivos multiobjetivo, obtuvimos que el mejor es MOEA/D con esto en cuenta realizamos la optimización del torque y el consumo de batería con el gemelo digital, mediante la ejecución de 30 simulaciones de la mejor trayectoria generada por MOEA/D y con un tiempo de ejecución establecido de 7 minutos.

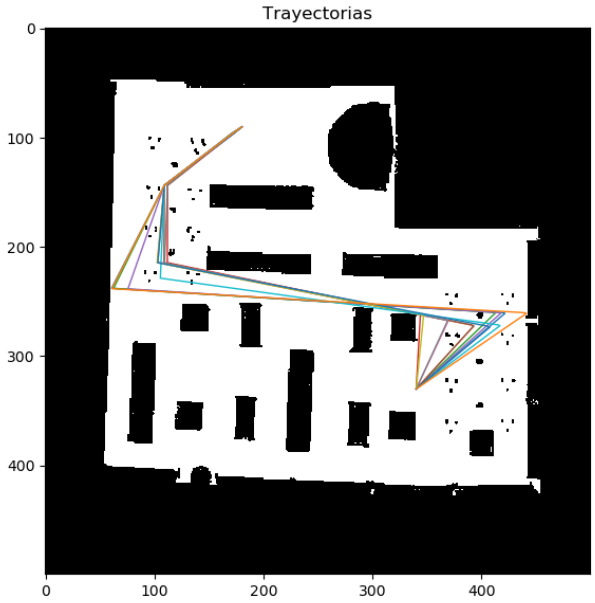

Figura 5.47: Trayectorias MOEA/D (Librería)

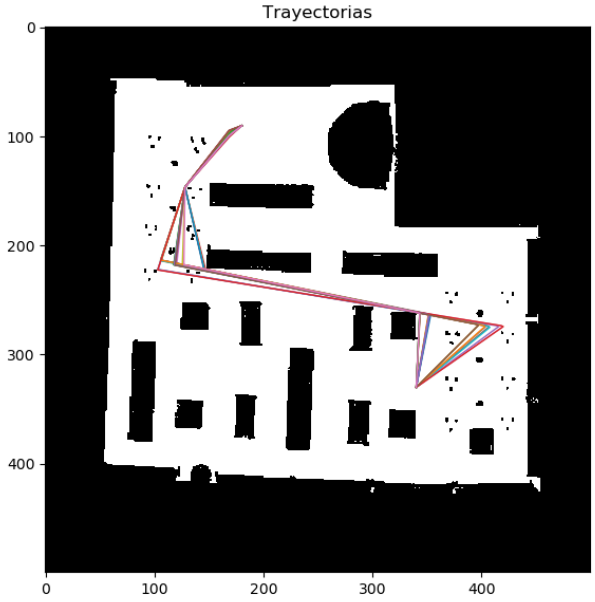

Figura 5.48: Trayectorias SMS-EMOA (Librería)

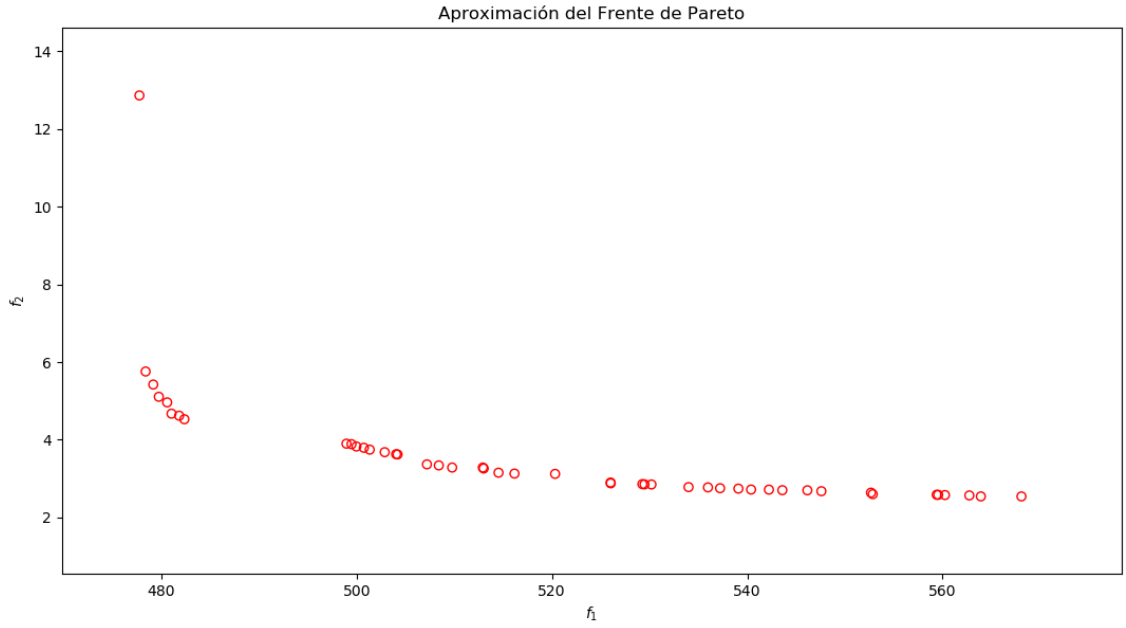

Figura 5.49: Aproximación del frente de Pareto NSGA-II (Librería)

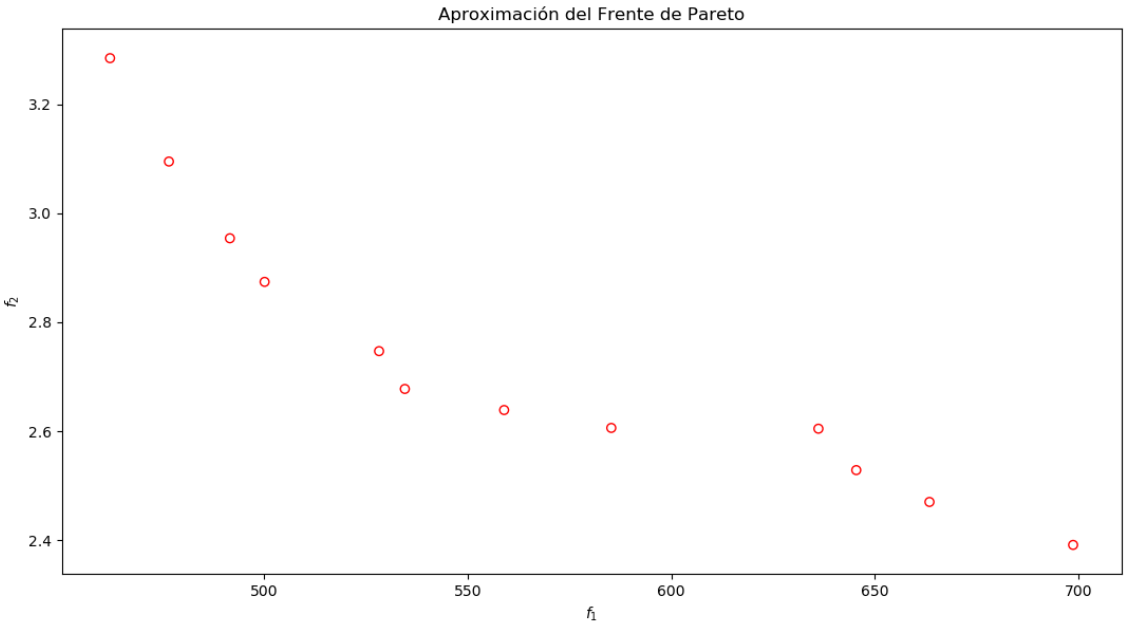

Figura 5.50: Aproximación del frente de Pareto MOEA/D (Librería)

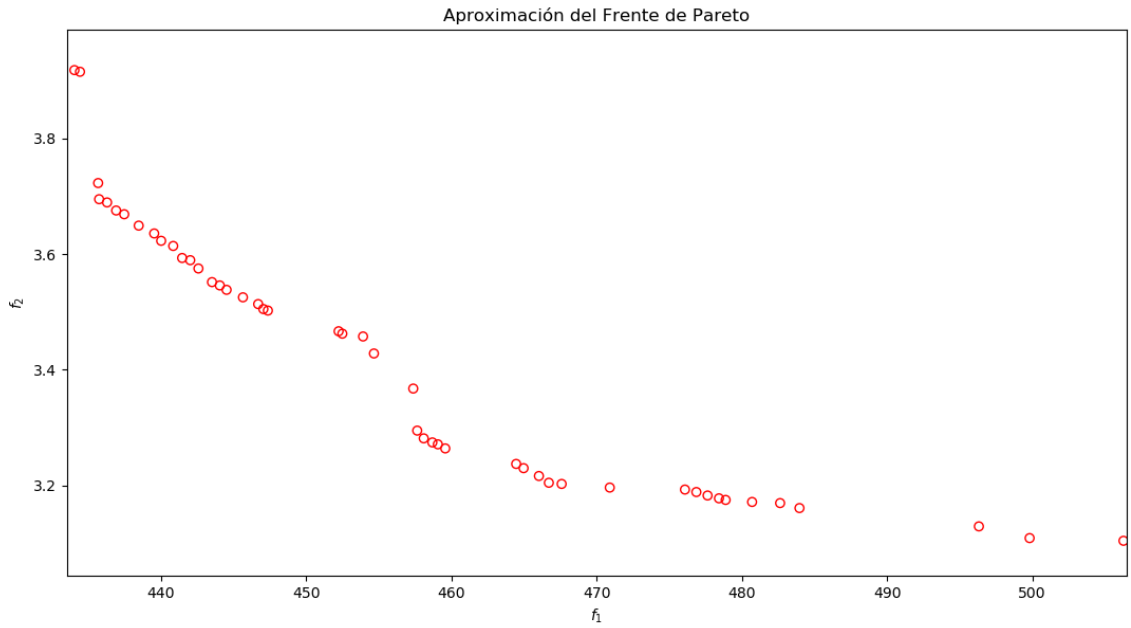

Figura 5.51: Aproximación del frente de Pareto SMS-EMOA (Librería)

Tabla 5.8: Resultados Gemelo Digital (Librería)

	Torque (ft/lbs)	Bateria (%)	Tiempo (min.)
	3.28	13.46	7

Los resultados que dió el gemelo digital lo podemos ver en la Tabla 5.8, dónde vemos el torque requerido para ejecutar la trayectoria en el tiempo establecido y el consumo de batería que se requiere para su ejecución.

Comparación con tecnologías tradicionales

Cómo comparación adicional, presentamos la comparación de todos los algoritmos utilizados, la comparación la realizamos de acuerdo a tres objetivos utilizados en este trabajo que son: la distancia de la trayectoria, la suavidad y la evasión de obstáculos.

Empezamos con la evaluación de la evasión de obstáculos, en la Figura 5.52 podemos observar el porcentaje de trayectoria correcta, es decir el porcentaje de las trayectorias

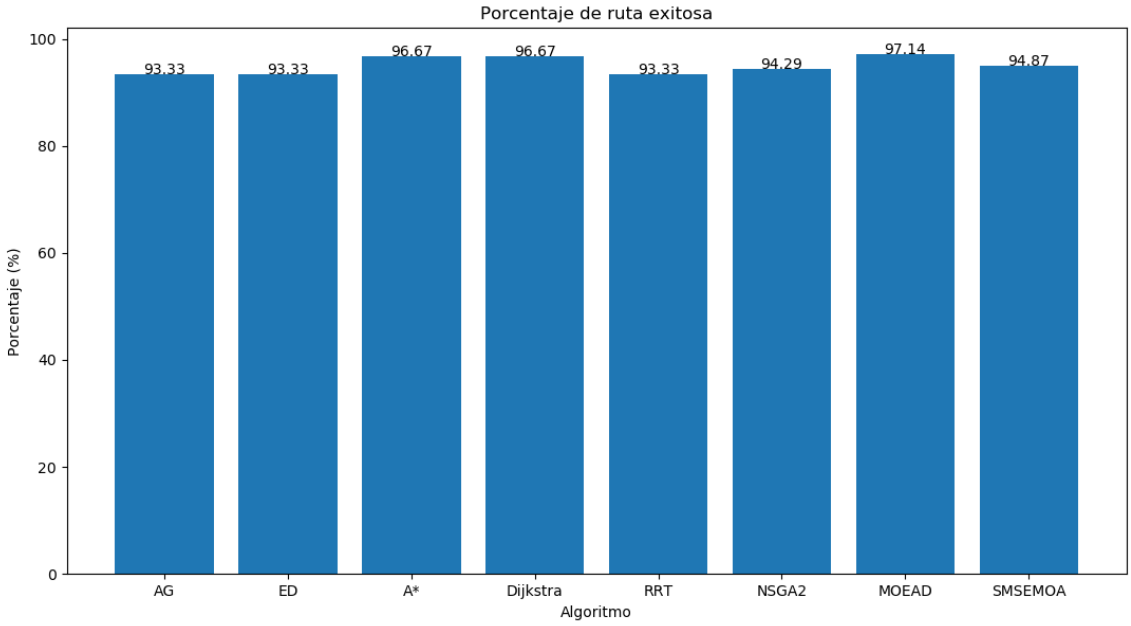

Figura 5.52: Porcentaje de trayectoria exitosa (Librería)

calculadas que no chocan con algún obstáculo. Podemos observar que el porcentaje es arriba del 90 % lo que indica que los algoritmos hacen bien su tarea.

Para los otros dos objetivos como en la sección anterior hacemos uso del evaluador de objetivos que tenemos en los algoritmos evolutivos multiobjetivo. Los resultados de la distancia de la trayectoria los podemos ver en la Figura 5.53 y los resultados de la suavidad los podemos ver en la Figura 5.54. Cómo se puede ver los mejores resultados en estos objetivos los tienen los algoritmos evolutivos multiobjetivo.

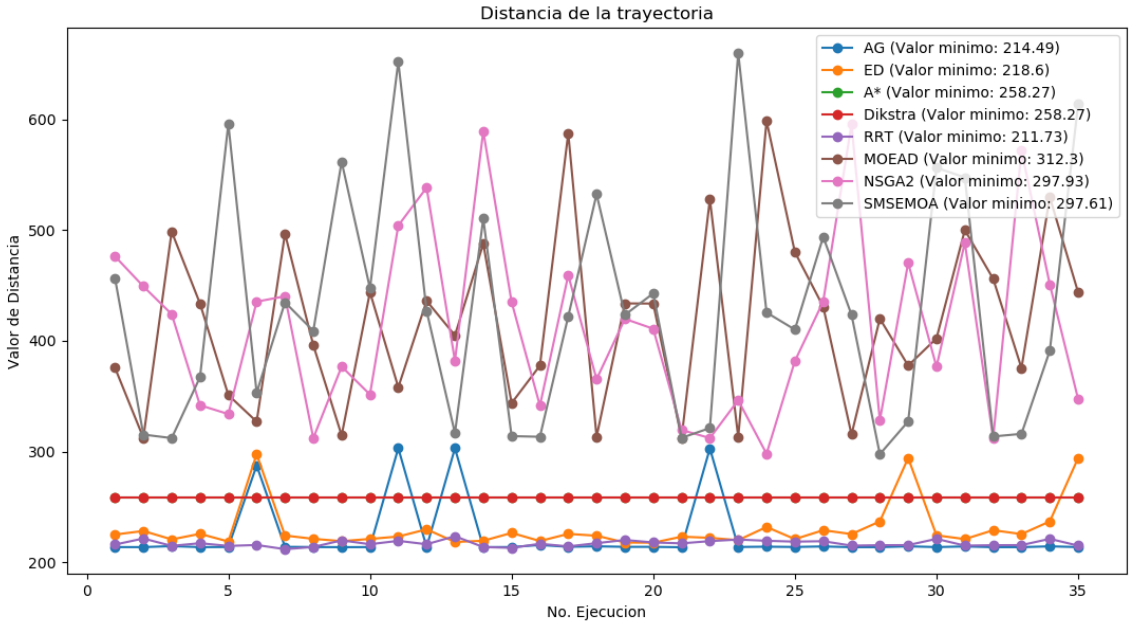

Figura 5.53: Distancia de la trayectoria (Librería)

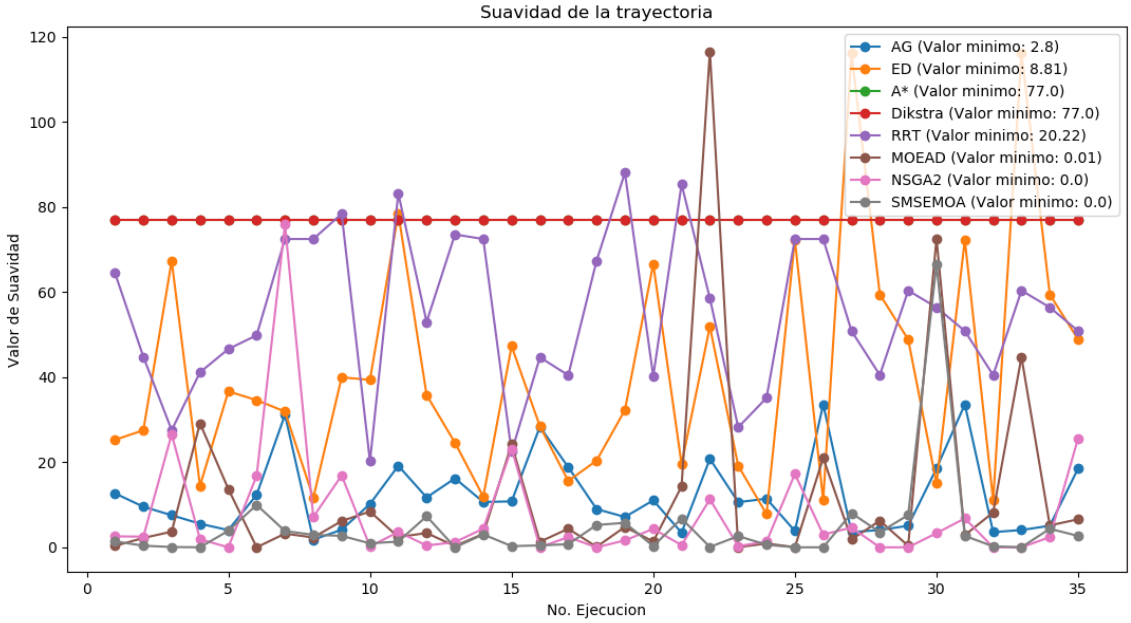

Figura 5.54: Suavidad de la trayectoria (Librería)

Capítulo 6

Conclusiones y trabajo a futuro

6.1. Conclusiones

La planificación de trayectorias ha evolucionado considerablemente en los últimos tiempos, adaptándose a las necesidades cambiantes de los usuarios y haciendo uso de diversas tecnologías. Estos factores pueden incluir el entorno de trabajo, el tipo de robot utilizado y la presencia de múltiples objetivos en la planificación de trayectorias.

En este estudio de tesis, hemos desarrollado una metodología que permite la optimización de trayectorias en diferentes entornos utilizando un enfoque de optimización multiobjetivo. Además, hemos creado un gemelo digital que nos ha permitido llevar a cabo la optimización de tres objetivos. Utilizando la metodología propuesta, hemos llevado a cabo un estudio comparativo de algoritmos evolutivos multiobjetivo para investigar cuál de ellos se comporta mejor en el problema de la optimización de trayectorias.

La metodología propuesta se presenta como una herramienta excelente para abordar la complejidad del problema multiobjetivo. Al incorporar el gemelo digital en el proceso de optimización, somos capaces de encontrar soluciones que equilibran eficientemente múltiples objetivos. Esto no añade complejidad adicional a los algoritmos evolutivos multiobjetivo, ya que sabemos que cuanto más objetivos se optimicen, más complejo se vuelve el problema. Este

enfoque no solo mejora la eficiencia operativa, sino que también permite una planificación más precisa y adaptable en diferentes entornos. Estos beneficios tienen un impacto significativo en una amplia gama de aplicaciones, desde la robótica hasta la logística y la movilidad autónoma.

En conclusión, basándonos en los resultados obtenidos en este trabajo, podemos afirmar que el uso de gemelos digitales en la optimización de trayectorias es una estrategia efectiva que aporta numerosos beneficios al desarrollo e implementación de sistemas robóticos. Los gemelos digitales representan fielmente el entorno, el robot y su comportamiento, lo que permite probar eficientemente las trayectorias y realizar ajustes antes de desplegar el robot en entornos del mundo real.

Además, el uso de algoritmos evolutivos multiobjetivo en el cálculo de trayectorias ha demostrado ser una estrategia efectiva, versátil y muy útil para abordar problemas de navegación. Los algoritmos utilizados han demostrado una capacidad destacada para encontrar soluciones de alta calidad. Según nuestra evaluación, el mejor algoritmo es MOEA/D, seguido de NSGA-II y, por último, SMS-EMOA. Sin embargo, esto no implica que SMS-EMOA sea un mal algoritmo para abordar el problema de la planificación de trayectorias multiobjetivo. Cada algoritmo tiene sus propias fortalezas y destaca de diferentes maneras.

6.2. Trabajo a futuro

Como investigaciones futuras, se sugieren diversas mejoras para ampliar y enriquecer el trabajo realizado. Una de las propuestas consiste en explorar otros algoritmos evolutivos multiobjetivo distintos a los utilizados en este estudio, a fin de comprender su comportamiento y evaluar su efectividad en la optimización de trayectorias. Esto permitirá una comparación más completa y detallada de los diferentes enfoques en esta área.

Además, se plantea la realización de experimentos con robots reales en entornos reales. Para lograr esto, será necesario establecer una conexión continua entre el gemelo digital y el

robot físico, con el fin de facilitar la transferencia de datos y la comunicación entre ambos. Esta integración permitirá evaluar el rendimiento y la adaptabilidad de la metodología propuesta en situaciones prácticas, brindando una validación más sólida de su eficacia.

Otra mejora sugerida es la modificación y expansión del gemelo digital para que pueda adaptarse a diferentes tipos de robots. Esto implicará desarrollar un marco flexible y versátil que permita utilizar la metodología propuesta en una amplia variedad de aplicaciones y contextos. De esta manera, se podrán aprovechar las ventajas de la planificación de trayectorias en diversos escenarios, lo que incrementará la utilidad y la aplicabilidad de la metodología.

Bibliografía

[1] César Alejandro Aguilar Rodríguez. Desarrollo de arquitectura de soporte a la toma de decisiones para agentes en hardware, Mayo 2021.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[3] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[4] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[5] E.W. DIJKSTRA. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[6] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[7] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[8] Zeki Murat Cinar, Abubakar Abdussalam Nuhu, Qasim Zeeshan, and Orhan Korhan. Digital twins for industry 4.0: A review. In Fethi Calisir and Orhan Korhan, editors, *Industrial Engineering in the Digital Disruption Era*, pages 193–203, 2020.

[9] Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, and Pengfei Wang. Path planning techniques for mobile robots: Review and prospect. *Expert Systems with Applications*, 227:120254, 2023.

[10] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. *Path Planning and Trajectory Planning Algorithms: A General Overview*, pages 3–27. Springer International Publishing, Cham, 2015.

[11] Enrique Carmona and Severino Fernández. *Fundamentos de la Computación Evolutiva*. 01 2020.

[12] Joaquim Martins and Andrew Ning. *Engineering Design Optimization*. 10 2021.

[13] E. CUANT DURÓN, J.A. PAMANES GARCÍA, and S. ZEGHLOUL. Optimización mono-objetivo y multi-objetivo del emplazamiento de trayectorias de robots manipuladores redundantes. *Ingeniería Investigación y Tecnología*, 9(003), 2009.

[14] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 01 1997.

[15] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–73, 1992.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[17] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[18] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[19] Michael Grieves. *Virtually Intelligent Product Systems: Digital and Physical Twins*, pages 175–200. 07 2019.

[20] Manish Kumar Mohammadreza Radmanesh. Flight formation of uavs in presence of moving obstacles using fast-dynamic mixed integer linear programming. *Aerospace Science and Technology*, 50(4):149–160, 2016.

[21] Xia Chen and Xiangmin Chen. The uav dynamic path planning algorithm research based on voronoi diagram. *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 1069–1071, 2014.

[22] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

[23] Xiaobing Yu, Chenliang Li, and JiaFang Zhou. A constrained differential evolution algorithm to solve uav path planning in disaster scenarios. *Knowledge-Based Systems*, 204:106209, 2020.

[24] Jinshuai Dong, Hongguo Wang, and Quan-Ke Pan. An improved multi-objective evolutionary algorithm for a robot detect path planning problem. In *2022 41st Chinese Control Conference (CCC)*, pages 2071–2076, 2022.

[25] Nianbo Kang, Hongguo Wang, and Quan-ke Pan. Multi-objective path planning of single delivery robot in epidemic control campus. In *2022 41st Chinese Control Conference (CCC)*, pages 2077–2081, 2022.

[26] Chaoda Peng and Shaojian Qiu. A decomposition-based constrained multi-objective evolutionary algorithm with a local infeasibility utilization mechanism for uav path planning. *Applied Soft Computing*, 118:108495, 2022.

[27] Yougang Xiao, Hao Yang, Huan Liu, Keyu Wu, and Guohua Wu. Uav 3-d path planning based on moea/d with adaptive areal weight adjustment, 2023.

[28] Xuewu Wang Xin Zhou and Xingsheng Gu. An approach for solving the three-objective arc welding robot path planning problem. *Engineering Optimization*, 55(4):650–667, 2023.

[29] V. Sathiya and M. Chinnadurai. Evolutionary algorithms-based multi-objective optimal mobile robot trajectory planning. *Robotica*, 37(8):1363–1382, 2019.

[30] Xinhao Zhang, Yu Guo, Jinqi Yang, Daoliang Li, Yang Wang, and Ran Zhao. Many-objective evolutionary algorithm based agricultural mobile robot route planning. *Computers and Electronics in Agriculture*, 200:107274, 2022.

[31] Yongkui Liu, Xinyu Wang, Kang Yang, Yaduo Pan, and Qianji Wang. Architecture and implementation of high-fidelity digital twins for industrial robots. In *2023 International Conference on Frontiers of Robotics and Software Engineering (FRSE)*, pages 207–214, 2023.

[32] Pieter Pauwels, Rens de Koning, Bob Hendrikx, and Elena Torta. Live semantic data from building digital twins for robot navigation: Overview of data transfer methods. *Advanced Engineering Informatics*, 56:101959, 2023.

[33] Xin Liu, Du Jiang, Bo Tao, Guozhang Jiang, Ying Sun, Jianyi Kong, Xiliang Tong, Guojun Zhao, and Baojia Chen. Genetic algorithm-based trajectory optimization for digital twin robots. *Frontiers in Bioengineering and Biotechnology*, 9, 2022.

[34] Ratchatin Chancharoen, Kantawatchr Chaiprabha, Lunchakorn Wuttisittikulkij, Widhyakorn Asdornwised, Muhammad Saadi, and Gridsada Phanomchoeng. Digital twin for a collaborative painting robot. *Sensors*, 23(1), 2023.

[35] Chengxi Li, Pai Zheng, Shufei Li, Yatming Pang, and Carman K.M. Lee. Ar-assisted digital twin-enabled robot collaborative manufacturing system with human-in-the-loop. *Robotics and Computer-Integrated Manufacturing*, 76:102321, 2022.

[36] Martin Denk, Sebastian Bickel, Patrick Steck, Stefan Götz, Harald Völkl, and Sandro Wartzack. Generating digital twins for path-planning of autonomous robots and drones using constrained homotopic shrinking for 2d and 3d environment modeling. *Applied Sciences*, 13(1), 2023.

[37] Silas F. R. Alves, Alvaro Uribe-Quevedo, Delun Chen, Jon Morris, and Sina Radmard. Developing a vr simulator for robotics navigation and human robot interactions employing digital twins. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 121–125, 2022.

[38] Sebastian Richard Newrzella, David W. Franklin, and Sultan Haider. 5-dimension cross-industry digital twin applications model and analysis of digital twin classification terms and models. *IEEE Access*, 2021.

[39] Wenjie Jia, Wei Wang, and Zhenzu Zhang. From simple digital twin to complex digital twin part i: A novel modeling method for multi-scale and multi-scenario digital twin. *Advanced Engineering Informatics*, 53, 2022.

[40] BYOUNG-KUK LEE and MEHRDAD EHSANI. Advanced simulation model for brushless dc motor drives. *Electric Power Components and Systems*, 31(9):841–868, 2003.

[41] Min Chen and G.A. Rincon-Mora. Accurate electrical battery model capable of predicting runtime and i-v performance. *IEEE Transactions on Energy Conversion*, 21(2):504–511, 2006.

[42] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[43] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.*, 15(1):4–31, 2011.

[44] Adheesh. Motion-planning-algorithms. https://github.com/adheeshc/Motion-Planning-Algorithms, 2020.

[45] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

[46] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In *International conference on evolutionary multi-criterion optimization*, pages 110–125. Springer, 2015.

[47] Carlos A. Coello Coello and Margarita Reyes Sierra. A Study of the Parallelization of a Coevolutionary Multi-Objective Evolutionary Algorithm. In Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Humberto Sossa, editors, *Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI'2004)*, volume LNCS 2972, pages 688–697. Springer Verlag, 2004.

[48] Jason Ramon Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. PhD thesis, Massachusetts Institute of Technology, 1995.

[49] Frank Wilcoxon. Individual comparisons by ranking methods. *Biom. Bull.*, 1(6):80–83, 1945.

[50] C. E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.