UNIVERSIDAD SANTO TOMÁS

IMPLEMENTACIÓN DE PLANIFICADORES DE TRAYECTORIAS DIJKSTRA Y ASTRA PARA UN ROBOT MÓVIL DIFERENCIAL

Realizado por

Camilo Arturo Montaña Castro

Monografía enviada en cumplimiento del requisito parcial
para optar por el grado de Ingeniería Electrónica

División de Ingenierías

Septiembre de 2021

IMPLEMENTACIÓN DE PLANIFICADORES DE TRAYECTORIAS DIJKSTRA Y ASTRA PARA UN ROBOT MÓVIL DIFERENCIAL

Realizado por

Camilo Arturo Montaña Castro

Monografía enviada en cumplimiento del requisito parcial
para optar por el grado de Ingeniería Electrónica

Dirigido por

José Guillermo Guarnizo Marín

Co-Dirigido por

Sindy Paola Amaya

Facultad de Ingeniería Electrónica
División de Ingenierías

Septiembre de 2021

Índice general

Capítulo 1

Planteamiento del problema

La tecnología ha tenido un gran crecimiento en las últimas décadas y a partir de esta la robótica ha tomado fuerza a nivel mundial para diversas aplicaciones, destacando la robótica social. Un robot social es un robot autónomo que interactúa y se comunica con los seres humanos, u otros agentes físicos autónomos siguiendo los comportamientos sociales que conllevan las reglas asociadas a su función [1].

Dentro de los distintos problemas que se pueden solucionar con robots móviles que tienen como fin interactuar con personas en entornos cotidianos, se destaca el de seguimiento de trayectorias y evasión de obstáculos, resaltando que el ambiente no debe ser modificado sino que el robot debe adaptarse al mismo. Una reseña a la evolución de la robótica móvil a lo largo de los últimos años [2]. Por este motivo se busca que el robot sea autónomo y dependa de sí mismo para realizar este tipo de trabajos, adaptándose también al ambiente en el cual es expuesto.

La planificación de trayectorias es una sucesión de posiciones que toma el robot para llegar aun punto final establecido, esta planificación está sujeta a diversos parámetros dependiendo el método que se establezca para planificar dicha trayectoria.

A partir de lo planteado se formulará la siguiente pregunta de investigación:

¿Se pueden seleccionar las técnicas Dijkstra y Astra para la planificación de trayectorias aplicada a la navegación de un robot diferencial, posicionado en un entorno previamente diseñado?

Capítulo 2

Justificación

Un robot puede percibir y explorar diferentes entornos a partir de algoritmos que le brinden la capacidad suficiente para recorrer distancias, encontrar rutas y lograr un objetivo sin colisiones mediante el uso de mapas previamente creados por diferentes tipos de métodos. El problema de generación de trayectorias para un robot móvil esta dada por, un robot y un ambiente en los cuales se puede determinar una trayectoria entre dos puntos específicos si satisface ciertos criterios de desempeño [3]. Los algoritmos de búsqueda por grafos son algunas opciones de planificadores de rutas para un robot móvil, estos algoritmos permiten obtener la ruta mas corta entre un punto inicial y uno final por lo que resultan ser bastante útiles, algunos de los algoritmos mas conocidos son: Primero el algoritmo de caminos cortos o también conocido como algoritmo Dijkstra que define el camino de coste mínimo de un vértice u a otro v, como el camino donde la suma de los pesos de los arcos que lo forman es la más baja entre las de todos los caminos posibles de u a v [4]. Segundo el algoritmo Astra es un algoritmo de búsqueda inteligente o informada que rastrea el camino más corto desde un estado inicial al estado final a través de un espacio de problema, usando una variable de referencia heurística.[5].

Estos algoritmos permiten a un robot móvil navegar dentro del mapa, recorrer las distancias mas cortas con el fin de que el robot consuma menos recursos, tenga una mayor durabilidad de batería y un tiempo mas bajo en las traslaciones desde un punto inicial hasta un punto final.

Capítulo 3

Impacto Social

La robótica social ha tenido un impacto dentro de la sociedad en los últimos años bastante grande, puesto que el crecimiento de la población de robots para estos ámbitos a crecido de manera exponencial lo cual indica que la sensación de estos robots en los humanos es bastante positivo, ya que la mayoría de estos están diseñados para facilitar ciertas tareas y/o actividades a personas con alguna discapacidad.

Con ese crecimiento exponencial que ha tenido la robótica social en los últimos años, el desarrollo e investigación en esta área también se ha ampliado, buscando ser mas segura y precisa en las tareas que desarrolla el robot al igual un costo menor y de esta forma ser mas asequible, con el fin de que el ser humano se sienta mas cómodamente con el robot. Por esto un robot que tenga la capacidad de calcular las distancias mas cortas entre un punto de llegada y uno final, puede llevar diversos objetos con una mayor rapidez y con un costo de recursos óptimo puesto que se va consumir un voltaje mínimo para llegar a dicho punto final.

En la robótica social es necesario que el robot tenga claro el entorno sobre el cual se va a movilizar, puesto que de esto dependen muchas acciones como que el robot no choque contra objetos y pueda realizar las trayectorias de manera mas óptima como se vio anteriormente.

Capítulo 4

Estado del Arte

En el 2006 se propuso un método novedoso de planificación global de rutas óptimas para robots móviles basado en el algoritmo mejorado de Dijkstra. Este método tiene tres pasos principales en su desarrollo, para el primer paso se adoptó la teoría de grafos MAKLINK para establecer el modelo de espacio libre del robot móvil, en el segundo paso se adopta el algoritmo Dijkstra mejorado para encontrar una ruta sin colisiones y por ultimo se utiliza el algoritmo del sistema de hormigas para ajustar, y optimizar la ubicación de la ruta subóptima a fin de generar la ruta óptima global para el robot móvil [6]. Por otro lado mediante un sistema multi-agente típico Robot Soccer System se implementó el algoritmo de Dijkstra como el algoritmo de cortocircuito más típico que se utiliza para calcular la ruta más corta de un nodo a todos los demás nodos, una característica de este es tomar un punto de partida como el centro y se van expandiendo hacia afuera de las capas, hasta finalizar todas las extinciones. Este artículo lo que realiza es una planificación de la trayectoria del robot de fútbol mediante el algoritmo, esto con el fin del control de evasión de obstáculos. Después del reconocimiento del entorno, Robot Soccer System puede determinar el camino más corto oportunamente y el método se ha aplicado al control del robot real [7].

Otro tipo de métodos de búsqueda que son muy utilizados son por heurística como lo implementa el método Astra, este documento describe un enfoque para la planificación de rutas estratégicas que utiliza la búsqueda de gráficos ordenados como base para el planificador de rutas. Las heurísticas proporcionan una guía de búsqueda. La selección de heurísticas determina la calidad de la solución en el peor de los casos, que puede ser casi óptima o aceptable. La fuerza heurística también determina la cantidad de memoria, tiempo y otros recursos necesarios para realizar la búsqueda. El planificador realiza tres funciones principales que son

el procesamiento del área de juego, el desarrollo heurístico de la ruta de vuelo y la optimización de la ruta de vuelo [8]. Otro articulo que utiliza esta metodología de heurística consiste en estudiar el algoritmo de planificación de trayectorias de robots múltiples bajo la demanda de una corporación de logística. Se comparan los algoritmos de búsqueda de rutas clásicas con la planificación de rutas tradicionales y se propone un nuevo algoritmo mejorado basado en búsqueda heurística. Este algoritmo utiliza una función de estimación de valor correcta que se basa en la forma de la ruta local y un ajuste local de acuerdo con la ventana de tiempo para resolver el problema de la adaptabilidad de la ruta, y los conflictos tanto de tiempo como de espacio [9]. En el siguiente articulo plantean el uso de lógica difusa para la navegación de un robot móvil cuyo objetivo es localizar un punto final establecido [16]

Recientemente se planteo un artículo para implementar un sistema de planificación de trayectorias para un submarino utilizando otro método denominado como colonia de hormigas. Basado en el entorno hidrológico, el artículo propone un método de cálculo de costos de amenaza. La distancia desde el punto de ruta al centro de amenazas y la longitud de la pista en el área de amenaza se agregan a la función de costo. El algoritmo evolutivo diferencial tiene como objetivo resolver el problema que presenta el algoritmo de colonias de hormigas, ya que está sujeto a una optimización local y conduce a mejorar el rendimiento de la búsqueda global [11]. Vista esta aplicación en un entorno acuático se observa la versatilidad de estos métodos de planificación de trayectoria, otro medio en que se puede aplicar es el aéreo, el siguiente articuló ha propuesto un enfoque de solución del problema de la planificación de la trayectoria globalmente óptima del vehículo aéreo no tripulado (DRON) modelando la optimización de colonias de hormigas (ACO) con una estructura multiagente. La solución del problema de ruta del UAV en entornos que cuentan con una gran cantidad de puntos de control y obstáculos como radares y montaña es muy compleja y se puede resolver con la ayuda de los diferentes algoritmos de optimización. El método que plantean se basa en Netlogo, un entorno de programación basado en agentes [12]. Aplicando este método a robots móviles diferenciales en el año 2021 se presentó el algoritmo de colonia de hormigas mejorado para la planificación de trayectorias. Inicialmente se plantea un análisis en el proceso de movimiento de las hormigas, puesto que eligen el camino de acuerdo con la ruleta, el camino tiene redundancia, retroceso y progreso ondulatorio. Para resolver este problema el articulo plantea la corrección de la ruta. El método para modificar la ruta al punto objetivo puede mejorar efectivamente la convergencia del algoritmo de la colonia de hormigas, y la ruta óptima obtenida es más corta. Con el objetivo de la ruta óptima obtenida, se optimiza el nodo de la ruta para mejorar la suavidad de la ruta. Se compara y verifica mediante simulación que el algoritmo mejorado de colonias de hormigas tiene una mejor convergencia que el algoritmo tradicional de colonias de hormigas,

puesto que está reduciendo el número de nodos y está más en línea con las demandas reales del movimiento del robot [13].

Actualmente se siguen desarrollando diversos métodos para la planificación de trayectorias mucho mas rápidas y eficaces en términos de recursos y implementación. En la ultima década se ha venido desarrollando métodos basados en sistemas inmunes artificiales, un artículo propone este algoritmo evolutivo para resolver el problema de la evasión de obstáculos para un robot móvil en un entorno desconocido y aplicando tanto la teoría de la red idiotípica como la teoría de la selección clonal del sistema inmunológico. Mientras que el primero se utiliza para la navegación en general, el segundo se aplica durante situaciones de mínimos locales [14]. El siguiente articulo de 2012 esta situado en los inicios de este método y se plantea como objetivo planificar el camino para evitar obstáculos para los robots móviles basado en el algoritmo de inmunidad artificial (AIA) desarrollado a partir del principio inmunológico. Se propone un algoritmo de inmunidad que adapta las capacidades del sistema inmunológico y permite que el robot alcance el objetivo de manera segura y cumpla con éxito su tarea a través de una ruta óptima [15]. Otro algoritmo plantea la asignación de tareas en robots de rescate homogéneos este campo de investigación ha sido importante en los últimos años debido al avance tanto en robótica como en inteligencia artificial. Se propone un enfoque resolviendo la ecuación de la dinámica del replicador para reducir los efectos de la incertidumbre definen una métrica estándar basada en el progreso de las tareas y se calculan los elementos principales de la teoría de juegos, como la matriz de pagos y las proporciones de asignación, para obtener el número de robots asignados a cada tarea

.

Capítulo 5

Objetivos

5.1. Objetivo General

Implementar los algoritmos para planeación de trayectorias Dijkstra y Astra, estableciendo un punto A inicial y un punto B final dentro de un mapa previamente construido, con la finalidad de que el sistema encuentre la ruta de menor costo para llegar al punto B final.

5.2. Objetivos Específicos

- Realizar una revisión del estado del arte para algoritmos de planeación de trayectorias.
- Simular un entorno con obstáculos estáticos y nodos que representaran las posibles paradas del robot dentro del mapa.
- Implementar los algoritmos Dijkstra y Astra para la planificación de trayectorias.
- Comparar los algoritmos mediante pruebas de rendimiento.

Capítulo 6

Marco Teórico

6.1. MATLAB

MATLAB se define como un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño, con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente [17]. MATLAB tiene su propio lenguaje de programación denominado M, este entorno de desarrollo tiene aplicación en diversos ámbitos como la matemática, física, robótica entre muchas otras.

FIGURA 1: MATLAB/ Fuente: recluit.com

6.2. Método Dikjstra

El método Dijkstra o también conocido como camino más corto entre nodos es la manera más eficiente para encontrar la distancia mínima entre dos vértices de un grafo con pesos no negativos, el funcionamiento teórico del algoritmo consiste en construir un grafo, inicialmente formado únicamente por el nodo origen. En cada paso se observaran todos los nodos a los que se pueda llegar directamente desde el grafo, es decir aquellos que compartan arista con un nodo del grafo, y se irá agregando el nodo cuya distancia al origen sea mínima, y así sucesivamente sigue el algoritmo hasta que el vértice del grafo sea el nodo destino, para mostrar gráficamente esta teoría en la figura 2 se evidencian los pasos.

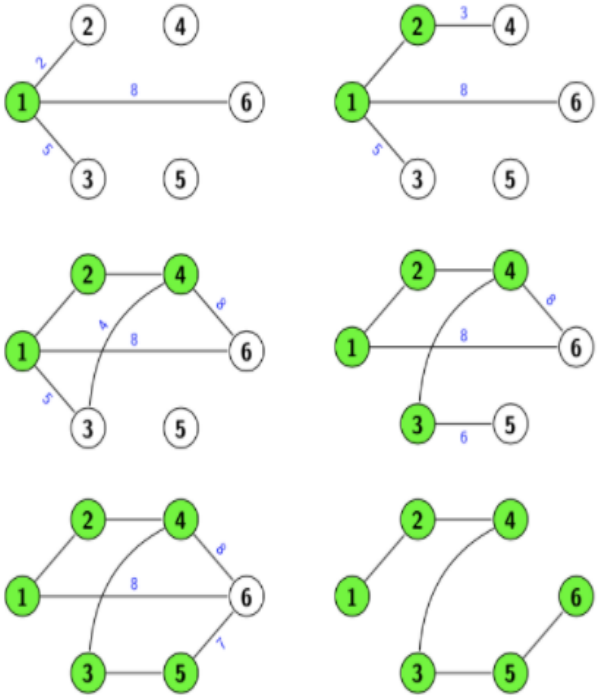

FIGURA 2: Método Dijkstra/ Fuente: aprende.olimpiada-informatica.org

En cada iteración, al incluir un nodo (cada numero en una circunferencia) al grafo se esta tomando el menor camino del origen al nodo. De existir otro camino, este debería pasar por alguno de los otros nodos a los que se pueda llegar desde el grafo, los cuales están a mayor o igual distancia del origen que el nodo, por construcción. Dado que los pesos son no negativos, cualquier camino que pase por alguno de estos nodos para llegar, no puede tener una distancia menor a la que se esta considerando inicialmente. De modo que al final del algoritmo se encontrará la distancia mínima entre el origen y el destino [18].

Para el caso presentado el grafo no era dirigido, pero este método también es aplicado para grafos dirigidos. La complejidad del algoritmo es $O(|E| + |V|log(|V|))$ para el presente ejemplo se presento un complejidad $O(|E|log(|V|)$, esta complejidad se puede calcular contando las operaciones realizadas por el algoritmo [19].

6.3. Método Astra* o búsqueda informada

Las estrategias de este método es que cuentan con información adicional del problema que busca resolver el agente, es por esto que puede encontrar las soluciones de una manera más eficiente, a diferencia de los métodos no informados que aunque implementen métodos de búsqueda bidireccionales o con limitación en los niveles, existe la probabilidad de que el algoritmo nunca encuentre el objetivo y entre en un ciclo infinito. Dentro este método se define el estado inicial en el que se puede encontrar el algoritmo, posteriormente una función define el conjunto de estados posibles que pueden darse a partir del estado inicial, el costo del camino y la función heurística que brinda la información adicional para llegar a la solución del problema [20].

Una vez establecido el concepto de búsqueda informada esta se divide en diversos métodos y para el presente informe se profundizara en el Astra que es el que se implemento inicialmente el método evalúa los nodos combinando $g(n)$ que es el coste para alcanzar el nodo y $h(n)$ que es el coste de ir al nodo objetivo, con la suma de estos obtenemos $f(n)$ coste más barato estimado de la solución a través de n.

$$f(n) = g(n) + h(n)$$

La capacidad de optimización del método Astra es más fácil de analizar si se usa con la grafos. En este caso Astra es óptima, si $h(n)$ es una heurística probable si nunca se supera el coste de alcanzar el objetivo. Las heurísticas probables son por naturaleza optimistas, porque piensan que el coste de resolver el problema es menor que el que es en realidad. Puesto que $g(n)$ es el coste exacto para alcanzar n, tenemos como consecuencia inmediata que la $f(n)$ nunca supera el coste verdadero de una solución a través de n [21].

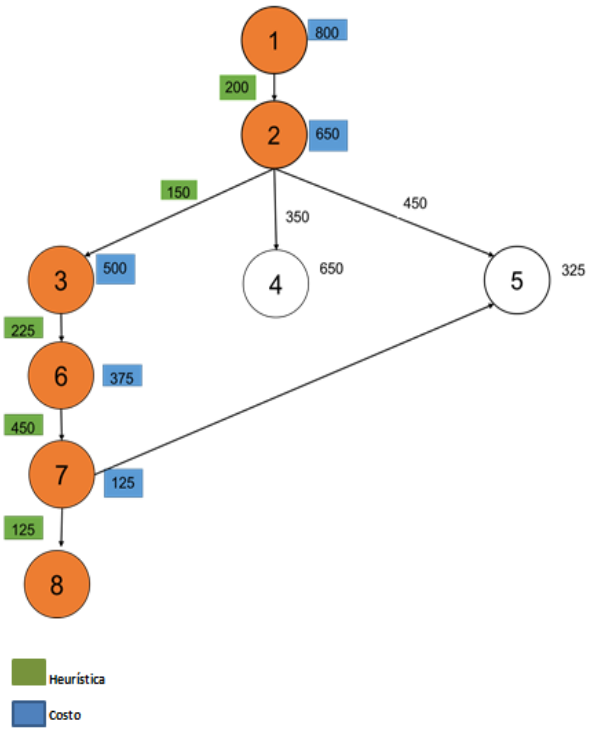

FIGURA 3: Método Astra / Fuente: inteligenciartificial2kc.wordpress.com

En la figura 3 se puede evidenciar la teoría del método Astra y como es el funcionamiento de la heurística, que son los cuadros de color verde sobre las aristas de cada nodo y adicional se observa que cada nodo tiene un peso o coste específico que es el cuadro azul, el método básicamente suma el coste o peso del nodo más el costo de la arista que vendría siendo la heurística, finalmente se obtendrán tanto los nodos de menor costo como el trayecto de menor costo, ya que mediante la suma de las aristas este obtendrá el valor mínimo de todo el grafo para este ejemplo dio un total de $1000 + 800 + 725 + 450 + 125 = 3600$.

Capítulo 7

Metodología

Basados en el estado del arte y mediante el entorno de desarrollo de MATLAB, se implementó un algoritmo de planificación de trayectorias para un robot móvil diferencial, apoyado por un repositorio de GitHub [22]. Esté se divide en diversas funciones, inicialmente se definen variables necesarias para lo largo del código que son: la cantidad de nodos que se desea crear, la posición inicial del robot y su destino final, estas dos últimas se definen como puntos [x,y] dentro del mapa, luego de esto se implementó la primera función para definir un mapa con sus respectivas dimensiones y algunos obstáculos estáticos, seguido a esto se define la siguiente función que tiene como fin generar las posiciones de los nodos que se definen al inicio del algoritmo, estas posiciones son aleatorias y se verifica que no tomen los valores de los obstáculos previamente establecidos. La siguiente función que se define luego de tener el mapa, y los posibles destinos del robot (los nodos y su posición inicial) es implementar el costo que llevará cada nodo esto también es generado de forma aleatoria además de esto se generar las conexiones entre los nodos que son bidireccionales, por último se implementan las dos funciones de los algoritmos de planeación de trayectorias para este caso Dijkstra y Astra. Finalmente se implementa la ultima función que toma los nodos seleccionados por cada algoritmo y dibuja la ruta final como se representa en la figura 4, siendo esta la salida final de todo el algoritmo.

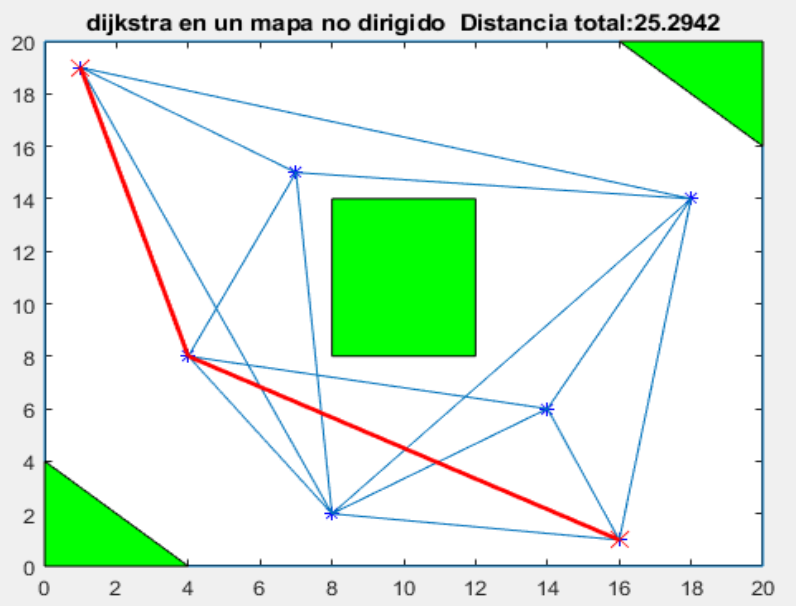

FIGURA 4: Salida final algoritmo de planificación de rutas Dijkstra / Fuente: Autor

Capítulo 8

Procedimiento

8.1. Implementación Algoritmo Parámetros Iniciales

El programa se desarrolló en el entorno de MATLAB, se establecieron dos parámetros iniciales como entrada del algoritmo que son la cantidad de nodos (Posibles posiciones del robot dentro del mapa), el punto inicial del robot (Donde el robot se localizará por primera vez dentro del mapa) y finalmente el punto de llegada (Nodo final donde el algoritmo finaliza), estos se pueden evidenciar en las siguientes lineás de código.

```
ns = 3;
startp = [1, 19];
endp = [16, 1];
```

Mediante los parámetros iniciales establecidos se procede a crear el mapa por el cual se desplazará él robot como se puede observar en la figura 5. Adicional dentro de este mapa se generarán obstáculos estáticos (figuras geométricas con fondo verde) esta para simular un entorno de la vida real donde se pueda desenvolver el robot, para el presente informe se realizarán tres mapas cada uno con cierta complejidad para poder someter el algoritmo a diversas pruebas de rendimiento.

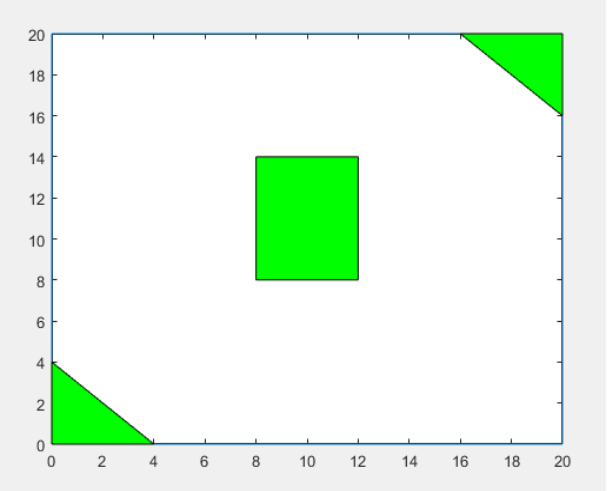

FIGURA 5: Mapa Inicial / Fuente: Autor

Paso siguiente a tener un mapa con sus respectivos obstáculos, es posicionar todos los nodos dentro del mapa, esto lo genera un ciclo *while* de forma aleatoria, ya que en cada ejecución la posición de los nodos varía, adicional a esto verifica que estas posiciones no coincidan con las de los obstáculos, este ciclo se observa en el siguiente segmento de código.

```
n=1;
while (n<=nnode)
    rx=rand* (map.xrange(2)-map.xrange(1)) + map.xrange(1);
    ry=rand* (map.yrange(2)-map.yrange(1)) + map.yrange(1);
    state=0;
    if ~inpolygon(rx,ry,obsx,obsy)
        nodelocation(n,1)=rx;
        nodelocation(n,2)=ry;
        n=n+1;
    end
end
```

8.2. Implementación Algoritmo Conexión Entre Nodos

Una de las funciones más importantes en la implementación de este algoritmo ya que genera un mapa no dirigido, y su conjunto de direcciones esto para establecer la conexión entre los nodos de forma bidireccional, posteriormente establece un costo aleatorio a cada nodo, lo cual

es importante a la hora de implementar cada algoritmo de planificación de trayectorias puesto que dependiendo el costo de un nodo se tomará cierta decisión, esto se verá a mayor detalle cuando se explique cada algoritmo y sus diferencias. El diagrama de flujo de esta función se puede evidenciar en la figura 6.

Inicialmente la función recibe dos parámetros que son el mapa y el arreglo de nodos, luego de esto genera un gráfico no dirigido que se define como un conjunto de V nodos y un conjunto E de aristas (arcos o lados) tales que a cada arísta queda asociada un par no ordenado de vértices. En este contexto (v, w) denota una arista de v a w en una gráfica no dirigida y no un par ordenado [23]. Este procedimiento se puede evidenciar en el siguiente segmento de código que contiene dos ciclos *for* anidados.

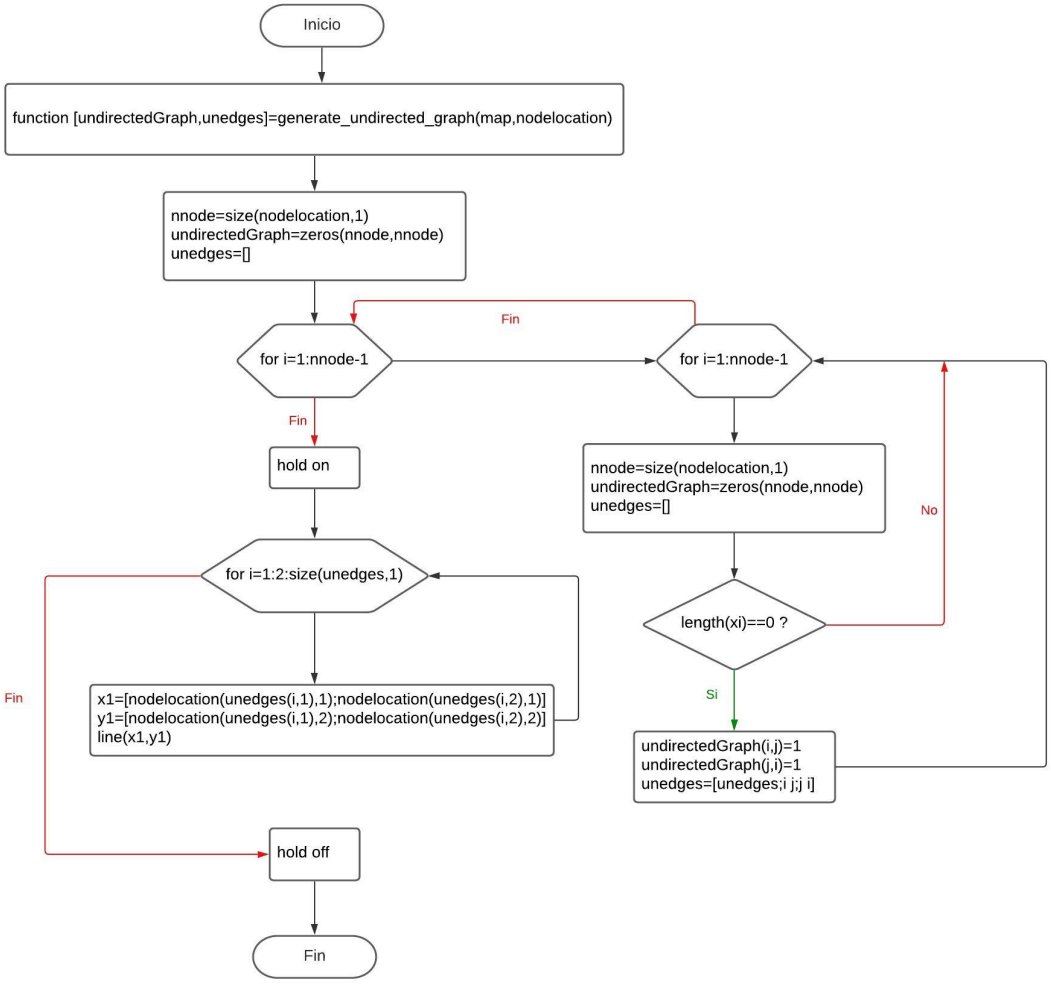

FIGURA 6: Diagrama de flujo de generación de conexión de nodos / Fuente: Autor

```
for i=1:nnode-1
    for j=i+1:nnode
        % tomar cada par de nodo ^ s ubicación x e y
        x1=[nodelocation(i,1);nodelocation(j,1)];
        y1=[nodelocation(i,2);nodelocation(j,2)];
        % compruebe que estos dos nodos se cruzan con cualquier
        % obstáculo o no
        [xi,yi] = polyxpoly(x1,y1,map.obsx,map.obsy);
        % si no hay ninguna interseccion
        if length(xi)==0
            % conecta estos dos nodos en el gráfico
            undirectedGraph(i,j)=1;
            undirectedGraph(j,i)=1;
            % agregar esta conexión al conjunto de bordes
            unedges=[unedges;i j;j i];
        end
    end
end
```

La conexión como bien se había mencionado es de forma bidireccional lo cual traduce que los nodos tienen dos direcciones tanto la entrada como la salida es valida, adicional a esta conexión cada nodo se conecta con sus posibles receptores es decir todos están interconectados, ningún nodo queda sin posibles conexiones, para ver gráficamente esta conexión se puede observar la figura 7.

FIGURA 7: Conexión Bidireccional / Fuente: redeweb.com

8.3. Implementación Algoritmo Agregar Nodos

Luego de obtener las conexiones entre nodos resta sumar los nodos iniciales, el punto de partida y punto final del robot dentro del mapa para posteriormente aplicar los algoritmos de planificación de rutas, la siguiente función agrega estos dos nuevos nodos en un gráfico no dirigido y verifica si dentro del arreglo de nodos hay algunos que no tengan una conexión, para así poder crearla ya que como se mencionó antes todos deben estar conectados. Esta función se puede ver el diagrama de flujo en la figura 8.

Como evidencia el diagrama de flujo de la función en su primer paso esta recibe seis parámetros de entrada que son: el mapa, el grafo no dirigido que se obtuvo de la función anterior, la localización de los nodos, las conexiones entre los nodos y finalmente las posiciones de él nodo inicial y él nodo final, cabe resaltar que el nodo inicial se le establece un costo igual a 0 y al nodo final de la misma manera.

En código el proceso es bastante similar al de la función anterior ya que se verifica la conexión de los nodos ya existentes pero adicional se verá la agregación de los dos nuevos nodos y su respectiva conexión con los nodos anteriores, esto se puede evidenciar en el siguiente segmento de código.

```
for i=1:nnode+1
    x1=[exnodelocation(i,1);endp(1)];
    y1=[exnodelocation(i,2);endp(2)];
    % si alguno de los nodos y los segundos nodos nuevos
    % tienen una conexión o no.
    [xi,yi] = polyxpoly(x1,y1,map.obsx,map.obsy);
    % si no hay intersección con obstáculo
    if length(xi)==0
        extungraph(nnode+2,i)=1;
        extungraph(i,nnode+2)=1;
        exunedges=[exunedges;i nnode+2;nnode+2 i];
    end
end
```

Finalmente la función retorna el grafo no dirigido actualizado al igual que las conexiones y el arreglo total de los nodos con el fin de tener de manera general el diseño total del grafo para el posterior análisis de los algoritmos de planificación de trayectorias.

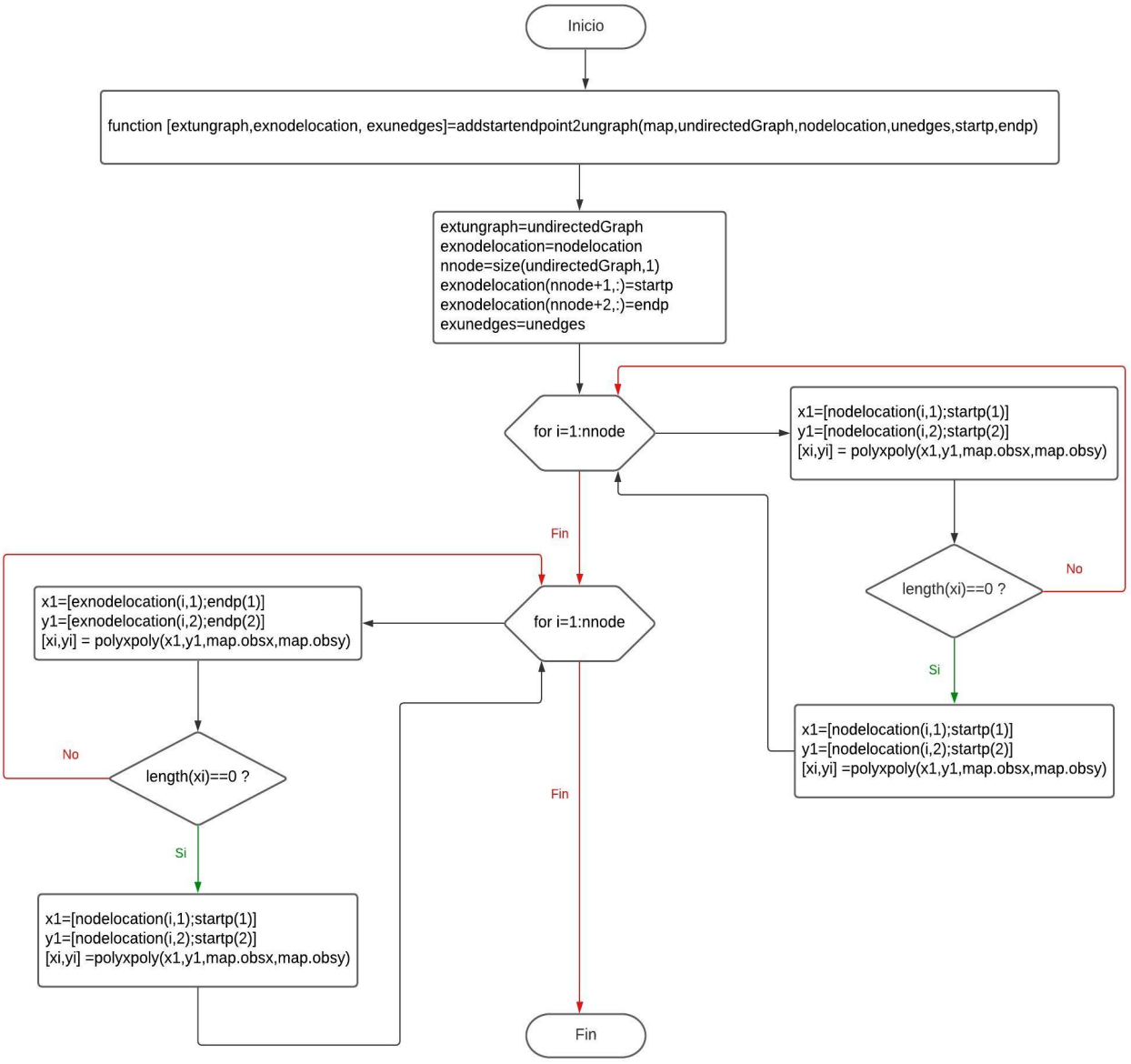

FIGURA 8: Diagrama de flujo de agregación de nodos punto inicial y punto final / Fuente: Autor

8.4. Implementación Algoritmo de planificación de trayectorias Dijkstra

Por ultimo en el flujo de todo el programa se realiza la implementación del algoritmo de búsqueda que para este primer caso se realizará con el método de Dijkstra, antes de explicar el diagrama de flujo de la función que realiza este procedimiento es necesario tener un concepto de la teoría del método, esto se puede evidenciar en el marco teórico del escrito donde se

explica a mayor detalle.

8.4.1. Algoritmo Dijkstra

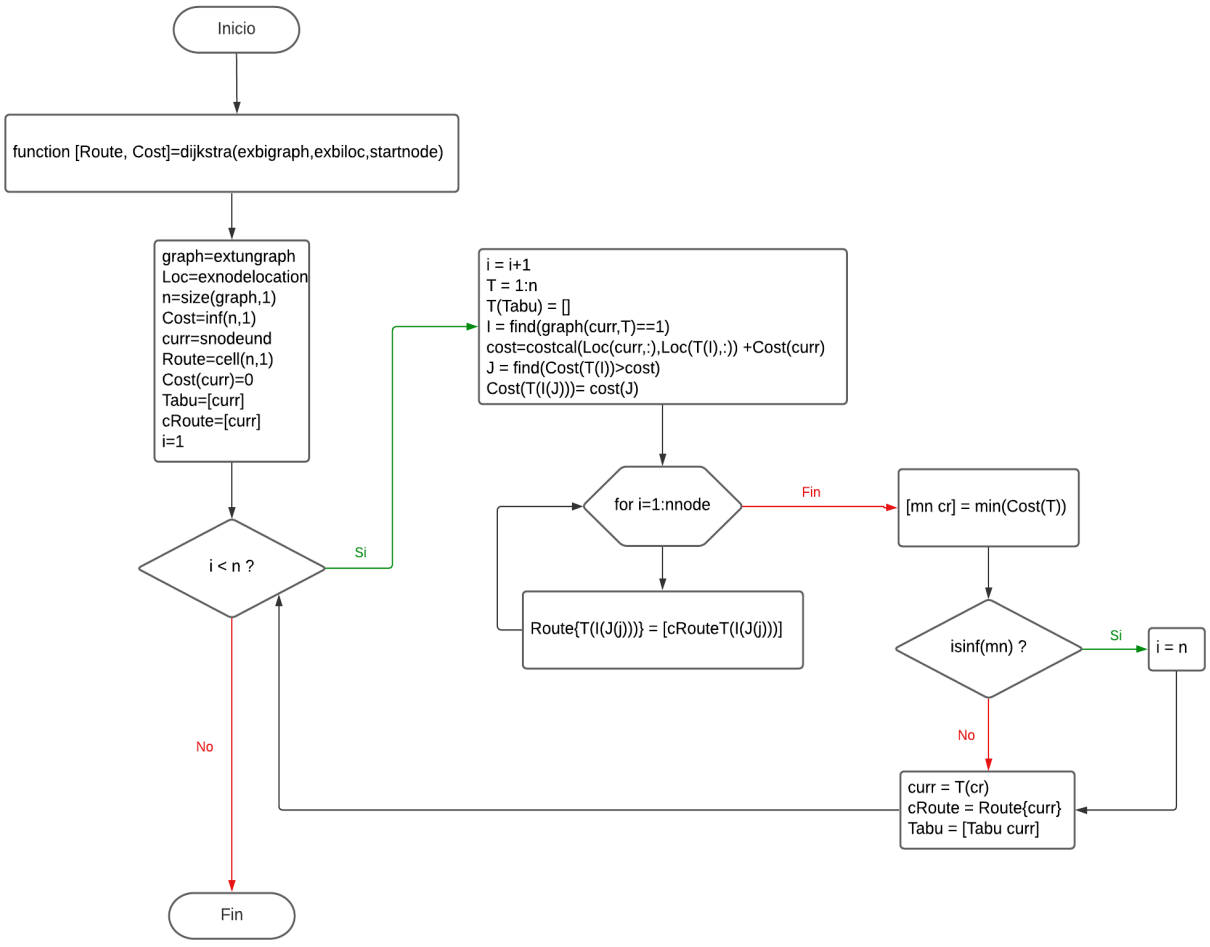

FIGURA 9: Diagrama de flujo de algoritmo de planeación de trayectorias Dijkstra/ Fuente: Autor

Se procede a implementar en el algoritmo de planificación de trayectorias, en la figura 9 se puede observar el diagrama de flujo de la función que se implementó para aplicar este método y lograr que el robot encuentre el camino de menor costo hasta su nodo de destino final.

Inicialmente la función recibe tres parámetros principales que son el grafo no direccionado, la localización de todos los nodos y el nodo inicial del robot, seguido a esto se definen diversas variable que se irán utilizando a lo largo del código, principalmente enfocaremos la explicación al ciclo *while* donde analizará cada nodo y realizará la búsqueda que se observo en la parte teórica, a continuación se observara el segmento de código.

```
while i < n
    i = i+1;
    T = 1:n;
    % eliminar nodo ya visitado
    T(Tabu) = [];
    % encuentra el nodo conectado del nodo actual que aún no se visitado
    I = find(graph(curr,T)==1);
    % calcula la distancia euclidiana de los nodos conectados
    % desde el nodo actual y agrega del costo actual de los nodos
    cost=costcal(Loc(curr,:),Loc(T(I),:)) + Cost(curr);
    % si la nueva clase lateral de búsqueda es menor que el
    % costo anterior, reemplace
    J = find(Cost(T(I))>cost);
    Cost(T(I(J)))= cost(J);
    % actualiza memoria del nuevo nodo si el nuevo costo es más eficiente
    for j = 1:length(J)
        Route{T(I(J(j)))} = [cRoute T(I(J(j)))];
    end
    % encuentra el nodo con el costo mínimo que aún no se
    % visita en esta iteración
    [mn cr] = min(Cost(T));
    if isinf(mn)
        i = n;
    end
    curr = T(cr);
    % seleccione la ruta seleccionada como ruta prohibida para no
    % seleccionar estos nodos de nuevo
    cRoute = Route{curr};
    Tabu = [Tabu curr];
end
```

Una vez establecidos dentro del ciclo *while* lo primero que se realiza es fijar el nodo inicial, luego buscar mediante el grafo no dirigido los nodos que se conectan y que pueden ser los posibles destinos del robot, ya generada esa conexión se miran los costos de cada nodo para tomar la desición final, antes de elegir el nodo hay que verificar que tanto el peso del nodo como el recorrido del robot sean el costo más bajo, para esto se implementa una pequeña función donde

se calcula la distancia entre los dos puntos, esto con fin de que el robot seleccione el nodo de menor peso, y también el nodo con la distancia más corta a recorrer. Esta selección del nodo, el robot lo puede reflejar en términos de consumo de batería y desplazamiento, ya que se asegura una ruta óptima en términos de consumo. Esta función la podemos ver en las siguientes lineas de código.

```
function sc=costcal(a,b)
tp=b-a;
sc=sqrt(tp(:,1).^2+tp(:,2).^2);
```

Finalmente el algoritmo entrega una variable llamada *Route* que es un arreglo con los nodos que fueron seleccionados por el algoritmo, esto posteriormente lo tomará una función que resalta gráficamente estos nodos seleccionados para obtener una salida final como en la figura 10 al igual el costo de distancia entre nodos se suma en su totalidad y este se refleja en la parte superior de la figura 10.

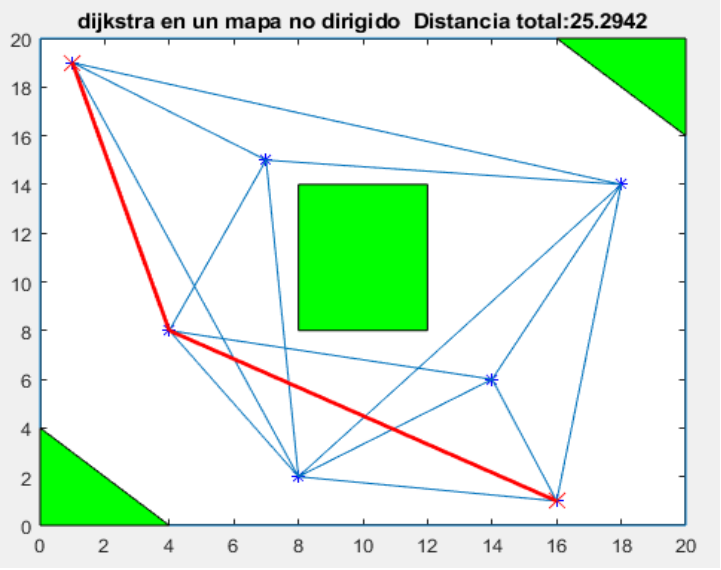

FIGURA 10: Salida final algoritmo de planificación de trayectorias Dijkstra / Fuente: Autor

8.5. Implementación Algoritmo de planificación de trayectorias Astra

Finalmente se implementa el ultimo método de los dos que se seleccionaron y para el presente caso es el método Astra o también conocido como algoritmo de búsqueda inteligente o informada, la teoría de este método se evidencia en el marco teórico y basados en esto se procede a la explicación del diagrama de flujo.

8.5.1. Algoritmo Astra

Se implementó el método Astra y se realizó el diagrama de flujo de la función en la figura 11, que representa cada uno de los pasos que se realizo para implementar este método.

La función recibe cuatro parámetros iniciales que son el grafo no direccionado, la localización de todos los nodos, el nodo inicial y el nodo final del robot, paso seguido se definen variables que serán utilizadas a lo largo del flujo del programa, posterior a esto se calcula la heurística de todos los nodos mediante un ciclo *for*, esto se observa en el siguiente segmento de código.

```
for i=1:n
    HCost(i,1)=costcal(Loc(endnode,:),Loc(i,:));
end
```

La función interna de este ciclo es la función de calculo de distancia que se presentó en el desarrollo del método Dijkstra ya que la heurística es la arista entre un nodo y otro para esta implementación, posteriormente se guarda en una variable para verificar y sumar tanto costo de nodo como costo de arista del nodo, para cumplir con la teoría del método. El resto del programa se ubica dentro de un ciclo *while* que contiene tanto la búsqueda como la operación de seleccionar el nodo que cumpla con los parámetros del método, este ciclo se observa en las siguientes lineas de código.

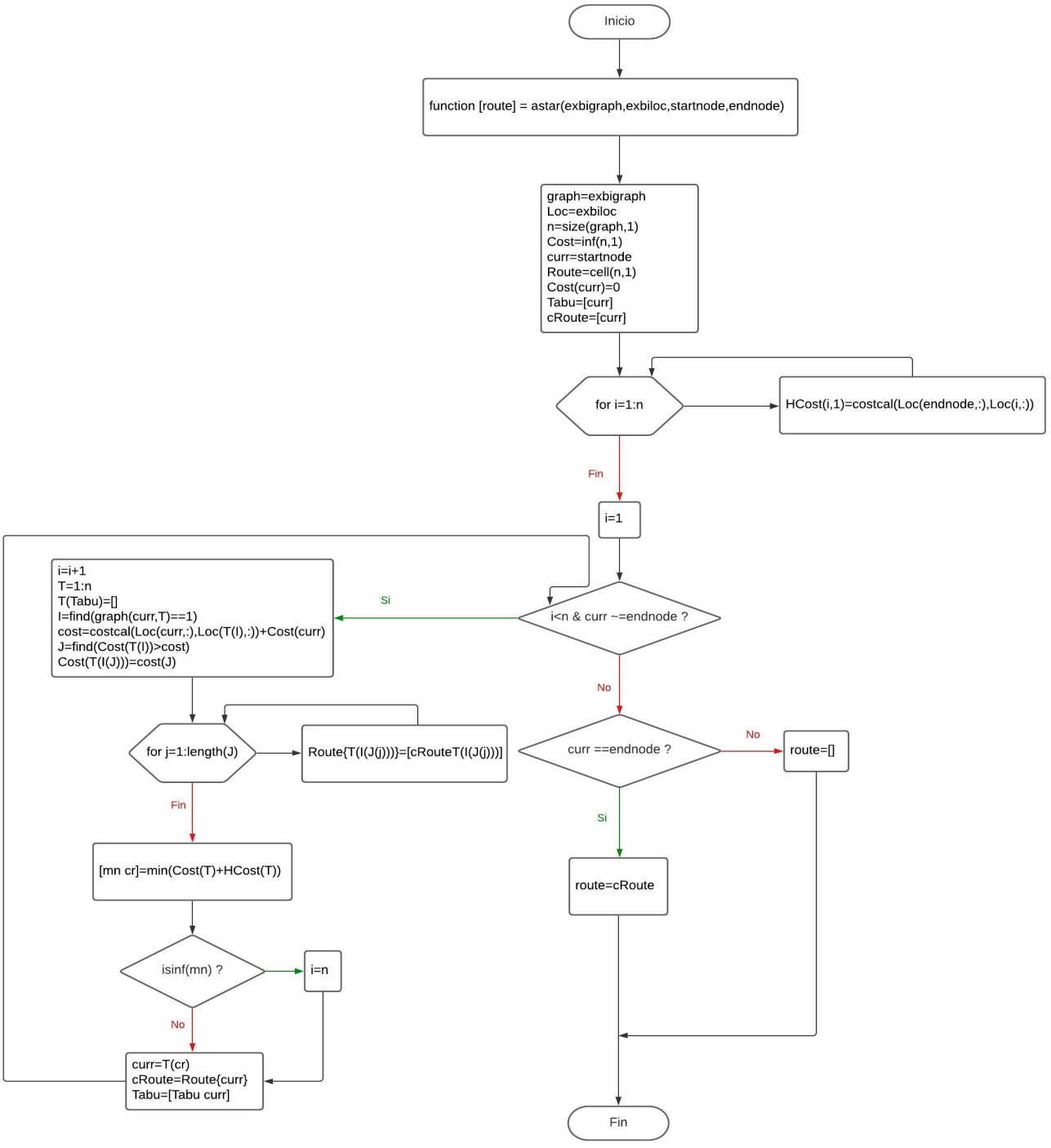

FIGURA 11: Diagrama de flujo de algoritmo de planeación de trayectorias Astra / Fuente: Autor

```
while i<n & curr~=endnode
    i=i+1;
    T=1:n;
    T(Tabu)=[];
    % encontrar todas las posibilidades del nodo actual
    I=find(graph(curr,T)==1);
    % encontrar el costo de todas las posibilidades desde el nodo
    % actual hasta el nodo en cuestión
    cost=costcal(Loc(curr,:),Loc(T(I),:)) + Cost(curr);
    % encontrar si el costo calculado es menor que la forma antes explorada
    J=find(Cost(T(I))>cost);
    % si el nuevo calculado es menor que antes de uno, establezca menos uno
    Cost(T(I(J)))=cost(J);
    % si la nueva ruta calculada es menor que la anterior que
    % establecer la nueva ruta también
    for j=1:length(J)
        Route{T(I(J(j)))}=[cRoute T(I(J(j)))];
    end
    % encontrar el mínimo del costo normal más el costo heurístico
    % del posible nodo
    [mn cr]=min(Cost(T)+HCost(T));
    if isinf(mn)
        i=n;
    end
    % establecer el nodo de costo total mínimo como nodo actual
    curr=T(cr);
    cRoute=Route{curr};
    % agregar el nodo actual a la lista tabú para evitar bucles
    Tabu=[Tabu curr];
end
```

Se puede evidenciar que el flujo es muy parecido al del método de Dijkstra, pero como bien se sabe el método Astra incluye esta variable de heurística que permite que el algoritmo sea óptimo y mas seguro que el Dijkstra. Esta diferencia se puede observar en la linea de código $[mncr] = min(Cost(T) + HCost(T))$ que evidencia la suma entre el costo del respectivo nodo y la heurística. Finalmente para salir del ciclo *while* se esta preguntando constantemente si

el nodo a examinar es el nodo final del robot, si esto se cumple retornara un arreglo con los nodos seleccionados y el costo total de la ruta seleccionada, para obtener una salida como se representa en la figura 12.

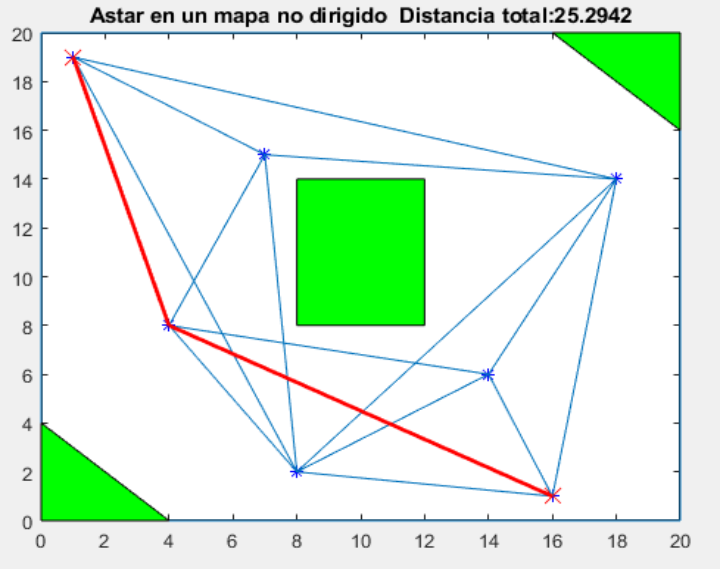

FIGURA 12: Salida final algoritmo de planificación de trayectorias Astra / Fuente: Autor

8.6. Resultados

Para las pruebas se examinaron puntos como, la cantidad de nodos fijados, el costo total de la ruta y finalmente el tiempo de ejecución de cada método implementado. Esto con el fin de comparar los métodos y poder obtener unas conclusiones a partir de las pruebas. De la misma manera se realizará en dos mapas diferentes con una cantidad diferente de nodos en cada uno para agregar complejidad al algoritmo y saber la respuesta del método.

Inicialmente se planteará un mapa con tres obstáculos estáticos y una cantidad de cinco nodos iniciales mas los nodos de inicio y fin del robot móvil, se realizaran cuatro ejecuciones por cada método iniciando con el Dijkstra y finalizando con el Astra.

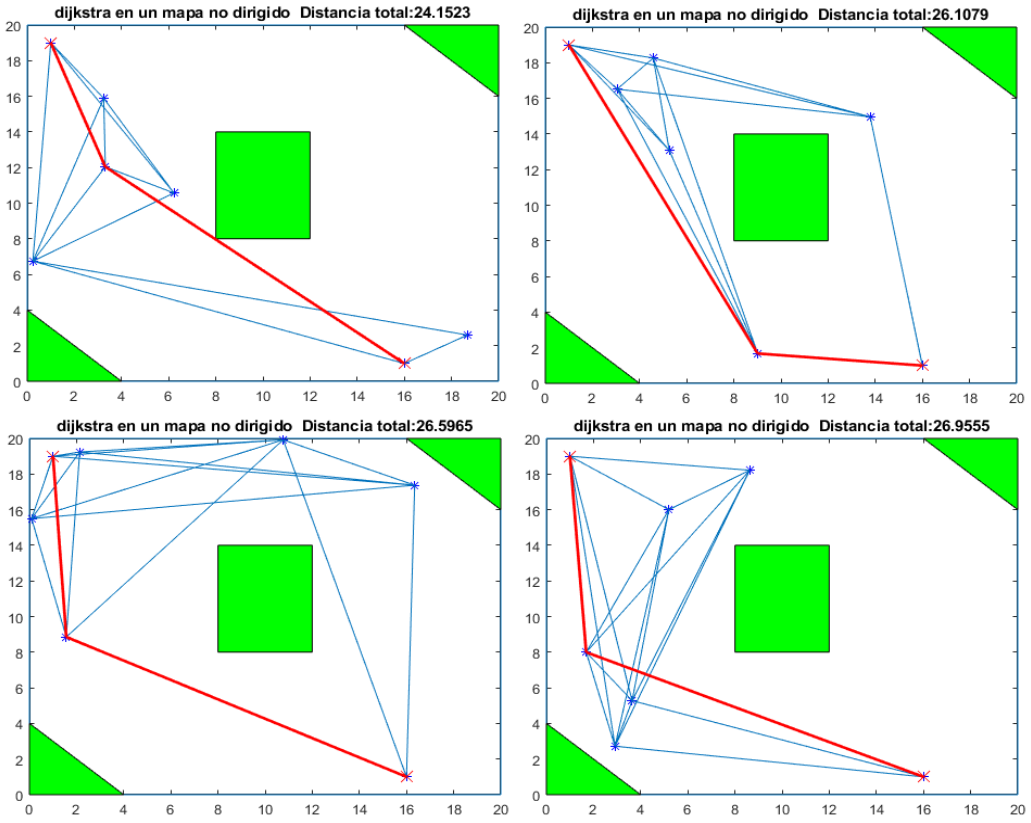

FIGURA 13: Pruebas del método Dijkstra en el mapa numero uno / Fuente: Autor

Ejecución	Cantidad Nodos	Costo Total	Tiempo Ejecución
Ejecución 1	7	24.1523	0.323519 seg
Ejecución 2	7	26.1079	0.419850 seg
Ejecución 3	7	26.5965	0.443270 seg
Ejecución 4	7	26.9555	0.317991 seg
Media	7	25.95305	0.3761575 seg

TABLA 1: Datos Obtenidos del método Dijkstra para el mapa uno/ Fuente: Autor

Seguido a estas pruebas se realizaron las cuatro ejecuciones para el método Astra con la misma cantidad de nodos y con los mismos obstáculos estáticos.

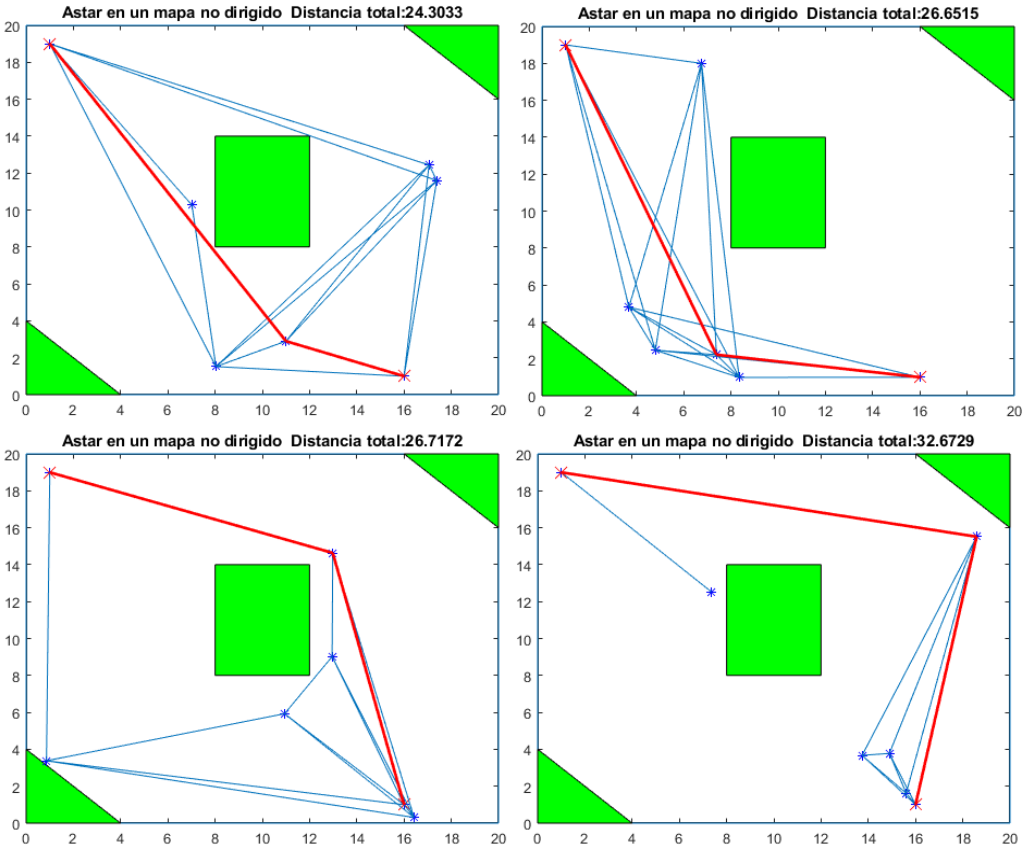

FIGURA 14: Pruebas del método Astra en el mapa numero uno / Fuente: Autor

Ejecución	Cantidad Nodos	Costo Total	Tiempo Ejecución
Ejecución 1	7	24.3033	0.332289 seg
Ejecución 2	7	26.6515	0.298521 seg
Ejecución 3	7	26.7172	0.289622 seg
Ejecución 4	7	32.6729	0.323211 seg
Media	7	27.586225	0.31091075 seg

TABLA 2: Datos Obtenidos del método Astra para el mapa uno/ Fuente: Autor

Con los resultados obtenidos por las tablas 1 y 2 se puede evidenciar que en términos de optimización el algoritmo Astra tiene una media de tiempo de ejecución menor a la del método Dijkstra lo cual avala la teoría del algoritmo que es más eficiente y rápido por el uso de la variable de heurística.

Observando la media de costo total se puede concluir que el Dijkstra por ser un método que busca los nodos con menor costo tiene una media mas baja que el del método Astra que si bien

también busca nodos con un valor menor por temas de la variable heurística se incremente en un pequeño porcentaje ese costo final.

El siguiente mapa tendrá 6 obstáculos estáticos y contará con un numero de veinte nodos mas el nodo inicial y el final.

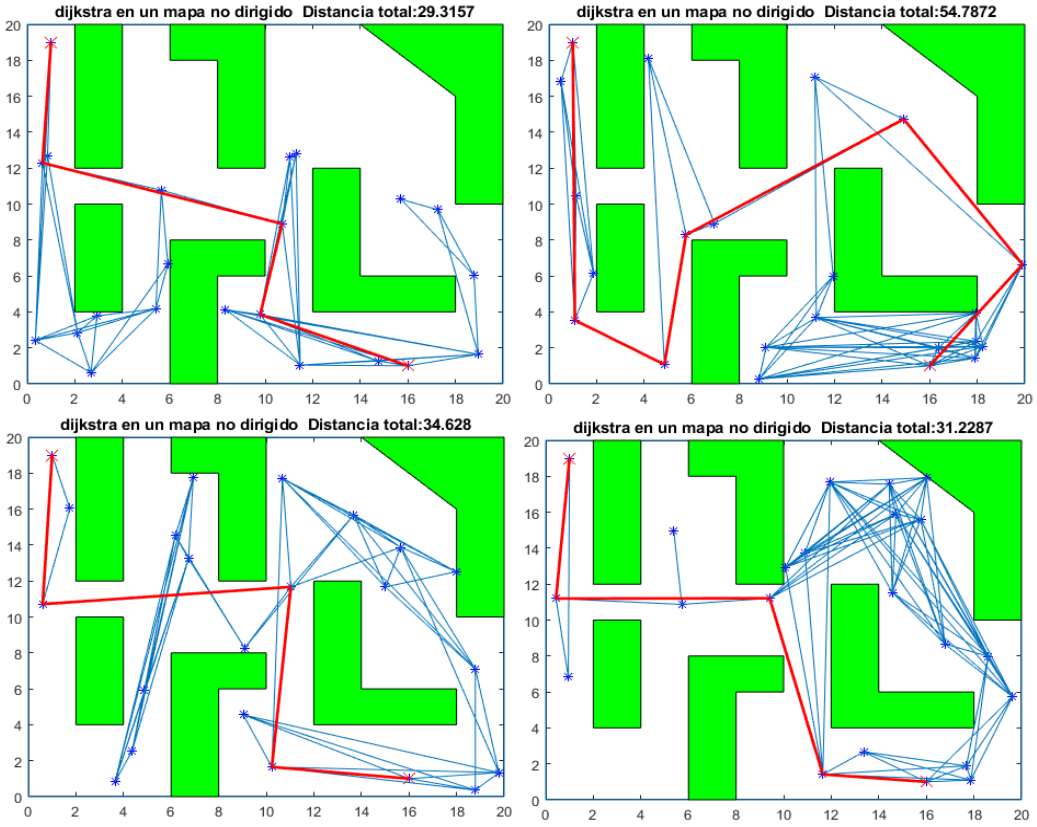

FIGURA 15: Pruebas del método Dijkstra en el mapa numero dos / Fuente: Autor

Ejecución	Cantidad Nodos	Costo Total	Tiempo Ejecución
Ejecución 1	22	29.3157	0.636408 seg
Ejecución 2	22	54.7872	0.682686 seg
Ejecución 3	22	34.628	0.556968 seg
Ejecución 4	22	31.2287	0.606262 seg
Media	22	37.4899	0.620581 seg

TABLA 3: Datos Obtenidos del método Dijkstra para el mapa dos/ Fuente: Autor

Finalizadas las pruebas al método Dijkstra se prosigue a realizarlas con el método Astra.

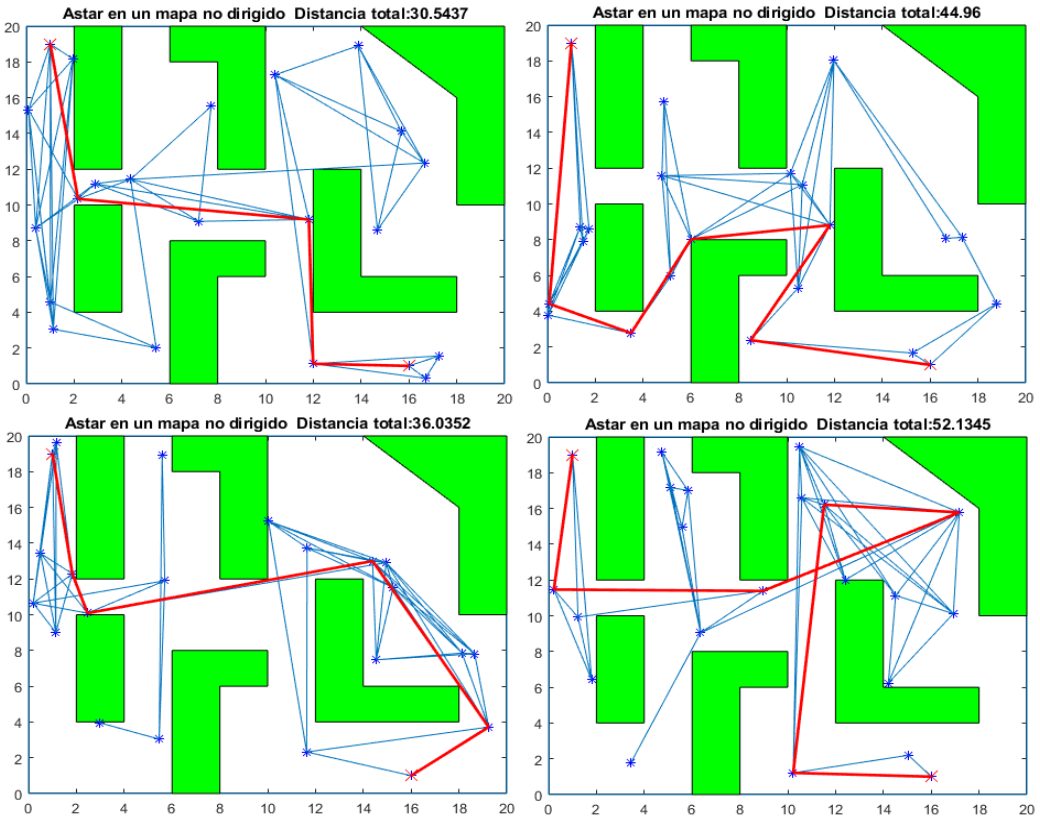

FIGURA 16: Pruebas del método Astra en el mapa numero dos / Fuente: Autor

Ejecución	Cantidad Nodos	Costo Total	Tiempo Ejecución
Ejecución 1	22	30.5437	0.487223 seg
Ejecución 2	22	44.96	0.526397 seg
Ejecución 3	22	36.0352	0.665602 seg
Ejecución 4	22	52.1345	0.547533 seg
Media	22	40.91835	0.55668875 seg

TABLA 4: Datos Obtenidos del método Astra para el mapa dos / Fuente: Autor

Con las tablas 3 y 4 se puede ver que la tendencia del tiempo se mantiene y debido a la cantidad de nodos el tiempo de ambos también crece, debido también a la cantidad de obstáculos el algoritmo necesita mas nodos para solventar y poder conectar desde el nodo inicial al final, por ende las distancias y el costo total también aumentan respecto a las primeras pruebas. Sin embargo la ruta que encuentra cada método siempre es la de menos costo.

Capítulo 9

Conclusiones y Trabajos Futuros

- La implementación de estos métodos de planificación de trayectorias suelen ser bastante comparados puesto que si bien su finalidad es la misma la implementación de uno u otro puede tener diversos beneficios como rapidez, complejidad, consumo de recursos entre otras.
- Presentadas las pruebas de comparación de los algoritmos de planificación de trayectorias y analizando sus respectivas salidas, basados en la teoría de cada método se validó de forma experimental y mediante simulaciones, ya que se conocía que el método Dijkstra hace una búsqueda del nodo conectado que tenga menor costo y con este patrón encuentra el nodo final, esto se evidencia en la media de los costos de las pruebas realizadas, ya que siempre fue menor que la media de las pruebas al método Astra.
- Con método Astra también se pudo comprobar su teoría mediante las pruebas implementadas, puesto que al realizar un análisis de los cuadros expuestos se evidencia que en términos de rendimiento es mucho mas veloz que el método Dijkstra, por la media de los tiempos que presenta el método Astra son menores respecto al otro, y esto se puede explicar desde la teoría ya que se sabe que el método utiliza una variable extra para la toma de decisiones, esto hace que el algoritmo encuentre de forma mas rápida la ruta de menor costo hasta el punto final, como se evidencia en la media de los tiempos de las simulaciones, por otro lado los costos que se obtuvieron son mas altos en comparación al método Dijkstra.
- Como parte de trabajos futuros se plantea realizar una evaluación mas profunda de cada algoritmo en términos de simulación para comprobar otros aspectos. Adicional e esto plantear mejoras para cada uno de los dos métodos.

- La utilización de estos métodos es una pequeña parte de la gran cantidad de métodos que existen para la planificación de trayectorias, actualmente se siguen desarrollando mas métodos. Un método novedoso que se investigo en el estado del arte denominado sistema inmunológico artificial se puede plantear como un trabajo futuro, implementando dicho algoritmo y comprarlo con estos dos métodos, para observar cuales pueden ser las ventajas de este nuevo método a demás de incluir obstáculos dinámicos.

Bibliografía

[1] Duffy, B. R. (2006). Fundamental issues in social robotics. International Review of Information Ethics, 6(12), 2006.

[2] Guarnizo Marin, J. G. Bautista Díaz, Daniela. Sierra Torres, Juan Sebastián (2021, 24 junio). Una revisión sobre la evolución de la robótica móvil. CRAI USTA. https://repository.usta.edu.co/handle/11634/34565

[3] Aníbal Ollero Baturone. Robótica, manipuladores y robots móviles. Marcombo, 2001.

[4] TORRUBIA, G.; TERRAZAS, V. Algoritmo de Dijkstra. Un tutorial interactivo. VII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2001), 2012.

[5] Instituto Tecnológico de Nuevo Laredo, Inteligencia Artificial. ALGORTIMO A*. 2005-II-A.

[6] Tan, Gz., He, H. & Aaron, S. Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm. J Cent. South Univ. Technol. 13, 80–86 (2006). https://doi-org.crai-ustadigital.usantotomas.edu.co/10.1007/s11771-006-0111-8

[7] Yi Y., Guan Y. (2012) A Path Planning Method to Robot Soccer Based on Dijkstra Algorithm. In: Jin D., Lin S. (eds) Advances in Electronic Commerce, Web Application and Communication. Advances in Intelligent and Soft Computing, vol 149. Springer, Berlin, Heidelberg.

[8] G. F. Wilber, "Strategic route planning using informed best-first search,"Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, 1988, pp. 1137-1144 vol.3, doi: 10.1109/NAECON.1988.195149.

[9] Z. Yongxiang and Z. Lei, " Improvement and application of heuristic search in multi-robot path planning," 2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS), 2017, pp. 1-4, doi: 10.1109/EIIS.2017.8298561.

[10] F. d. P. Gonzalez, J. G. Guarnizo and G. Benavides, ” Emulation System for a Distribution Center Using Mobile Robot, Controlled by Artificial Vision and Fuzzy Logic,” in IEEE Latin America Transactions, vol. 12, no. 4, pp. 557-563, June 2014, doi: 10.1109/TLA.2014.6868855.

[11] Y. Shan, "Study on Submarine Path Planning Based on Modified Ant Colony Optimization Algorithm,"2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018, pp. 288-292, doi: 10.1109/ICMA.2018.8484484.

[12] S. K. Çalık, ” UAV path planning with multiagent Ant Colony system approach, "2016 24th Signal Processing and Communication Application Conference (SIU), 2016, pp. 1409-1412, doi: 10.1109/SIU.2016.7496013.

[13] Q. Su, W. Yu and J. Liu, "Mobile Robot Path Planning Based on Improved Ant Colony Algorithm,"2021 Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), 2021, pp. 220-224, doi: 10.1109/ACCTCS52002.2021.00050.

[14] Frases K. Mohanty, AA Kodapurath and Rishi Kumar Singh, ” A Hybrid Artificial Immune System for Mobile Robot Navigation in Unknown Environments,"14 January 2020 Iranian Journal of Science and Technology, Transactions of Electrical Engineering.

[15] P.K. Das, S.K. Pradhan, Dr. S.N. Patro, and B.K. Balabantaray, ” Artificial Immune System Based Path Planning of Mobile Robot*", 2012 Soft Computing Techniques in Vision Sci.

[16] Amaya S., Mateus A. (2019) Tasks Allocation for Rescue Robotics: A Replicator Dynamics Approach. In: Rutkowski L., Scherer R., Korytkowski M., Pedrycz W., Tadeusiewicz R., Zurada J. (eds) Artificial Intelligence and Soft Computing. ICAISC 2019. Lecture Notes in Computer Science, vol 11509. Springer, Cham. https://doi.org/10.1007/978-3-030-20915-5_54

[17] MATLAB - El lenguaje del cálculo técnico. (2021). MATLAB & Simulink. https://es.mathworks.com/products/matlab.html

[18] M.B.A.L.S.E.L.L.S. (2019, 23 septiembre). Camino más corto entre nodos | Aprende Programación Competitiva. Aprende Programación Competitiva. https://aprende.olimpiada-informatica.org/algoritmia-dijkstra-bellman-ford-floyd-warshall

[19] Algoritmo de Dijkstra - Wikipedia Republished // WIKI 2. (2021, 17 febrero). Wikipedia. https://wiki2.org/es/Algoritmo_de_Dijkstra

[20] Cevallos, K. (2015, 30 mayo). Búsqueda Informada y explorada : Introducción. Inteligencia Artificial II. https://inteligenciartificial2kc.wordpress.com/2015/05/01/busqueda-informada-y-explorada-introduccion/

[21] Russell, S y Norvig, P. 2008. Inteligencia Artificial Un Enfoque Moderno. 2 ed. España. Pearson Education. p 1242

[22] Muhammet balcilar (2021). RobotPathPlanning (https://github.com/balcilar/RobotPathPlanning), GitHub. Retrieved June 28, 2021.

[23] dspace - Matematicas discretas - teoría de gráficas. DMD5_teoria_de_graficas.pdf. http://www.dspace.espol.edu.ec/retrieve/2767/DMD5_teoria_de_graficas.pdf