Búsqueda de la ruta óptima mediante los algoritmos: genético y dijkstra utilizando mapas de visibilidad

Finding path through the algorithms: genetic and dijkstra using maps of visibility.

Jaime Alberto Guzmán Luna, Rafael Esteban Arango Sánchez, Leidy Diana Jiménez Pinzón
Escuela de Ingeniería de Sistemas e Informática, Universidad Nacional de Colombia, Medellín, Colombia
jaguzman@unal.edu.co
raerangosa@unal.edu.co
ldjimenezp@unal.edu.co

***Resumen*— Este artículo presenta el estudio de la generación de trayectorias entre dos puntos y una cantidad cualquiera de obstáculos entre ellos mediante el mapa de visibilidad teniendo en cuenta la geometría del robot, de igual manera en éste se plantea la comparación entre el algoritmo genético y el algoritmo Dijkstra al encontrar la ruta óptima entre las trayectorias ya generadas con el mapa de visibilidad. El algoritmo es implementado en Java, en Java Lejos versión 0.9 se desarrolla un algoritmo que envía al robot los puntos de navegación mediante bluetooth corrigiendo el error de su trayectoria y sus giros mediante el uso de los sensores como el compás, el tacómetro y el concepto de la odometría; las pruebas para obtener los resultados son aplicadas sobre el robot LEGO NXT 2.0.**

***Palabras clave*— Mapas de Visibilidad, Algoritmo Genético, Algoritmo Dijkstra, Java LeJos.**

***Abstract*— This article presents the study of the generation of paths between two points and a quantity any of obstacles between them through the map of visibility using the geometry of the robot, equally in this can find the comparison between the genetic algorithm and the algorithm Dijkstra finding optimal path between the paths already generated with the map of visibility. The algorithm is implemented in Java, in Java LeJos version 0.9 develops an algorithm that sends to the robot the points of navigation through bluetooth correcting the mistake of his path and his drafts through the use of the sensors as the compass, the tachometer and the concept of the odometry; the tests to obtain the results are applied on the robot LEGO NXT 2.0.**

***Key Word*— Maps of Visibility, Genetic Algorithm, Algorithm Dijkstra, Java LeJos.**

I. INTRODUCCIÓN

La navegación se define como la metodología que permite guiar el curso de un robot móvil desde un punto inicial hasta un punto final a través de un entorno que contiene obstáculos mediante la generación de diferentes trayectorias, las cuales se construyen mediante la unión de tres caminos: uno que conecta el punto inicial con el primer obstáculo, otro que es el que recorre los obstáculos entre si y el último es el que se une desde allí hasta el punto final, el objetivo siempre es llevar a su destino de forma óptima el robot teniendo en cuenta sus dimensiones para dejar de considerarlo como un punto en el espacio y tomarlo como un objeto que tiene unas medidas específicas y ocupa un volumen en el espacio de navegación pero para llevar esto a cabo es importante el estudio de la percepción del mundo, la planificación de la ruta, la generación del camino y el seguimiento del mismo; aspectos que se deben tener en cuenta en el desarrollo del proyecto y aunque son analizados independientemente se relacionan entre sí ya que demuestran continuidad, de allí la importancia de implementar su solución en el orden respectivo[1].

En este artículo se presenta el estudio de la generación de trayectorias, existen diferentes métodos que ya han sido implementados y son eficientes en la planificación de caminos, como por ejemplo, el método de descomposición por celdas o el diagrama de voronoi [2] pero para este proyecto se hará uso del grafo de visibilidad. Para encontrar la ruta óptima a partir de las trayectorias ya existentes se estudian los algoritmos de búsqueda como el algoritmo dijkstra o el algoritmo genético, se presenta la descripción de los algoritmos implementados, se proponen las pruebas para la verificación de los algoritmos y finalmente se presentan las conclusiones del trabajo.

Actualmente existen investigaciones que implementan el algoritmo de búsqueda Dijkstra con el diagrama de Voronoi [3], con el mapa de visibilidad [4], con la descomposición por celdas [5] o el algoritmo genético con el mapa de voronoi [6].

II. CONTENIDO

A. Descripción del robot lego NXT. El robot móvil utilizado para realizar las pruebas de navegación es el robot Lego Mindstorms NXT 2.0, puede ser programado en diferentes lenguajes pero para este proyecto se usó Java Lejos. El robot

Fecha de Recepción: 26 de enero de 2012
Fecha de Aceptación: 28 de Agosto de 2012

recibe desde el ordenador y por medio de conexión bluetooth los ángulos y las distancias que debe navegar en el mundo real, estos datos son los generados en la aplicación de java con el nombre de "Plan" y son los resultados arrojados por los algoritmos de búsqueda propuestos para estudiar.

Tal como se muestra en la figura 1a, el robot tiene el sensor de compás que le ayuda a minimizar el error en sus giros; dos servomotores, uno para cada llanta haciendo sus movimientos totalmente independientes y el brick, donde se descarga el proyecto de Java Lejos para poner en funcionamiento el robot.

En la parte trasera tiene una rueda loca la cual fue construida con una esfera que pertenece al kit de Lego (ver figura 1b), con este mecanismo los movimientos del robot son más fluidos y precisos a comparación de la rueda de apoyo tradicional pequeña que viene en el mismo kit. Por esta razón, el robot fue construido de tal manera que la mayor parte de su peso se concentra en la parte delantera, con el fin de que la rueda loca ejerza menor fricción sobre la superficie.

De acuerdo a lo experimentado el diámetro de la ruedas del robot es de 5.4cm; la distancia entre las ruedas, es decir, su eje es de 10.4cm y el robot mide 140mm de ancho y 255 mm de largo.

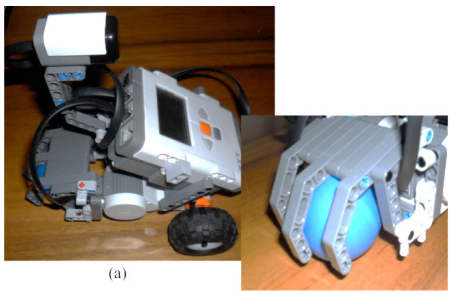

Figura 1. Estructura del robot Lego.

B. Grafo de visibilidad. El grafo de visibilidad proporciona un enfoque geométrico para generar la solución al problema de la planificación, está conformado por las líneas rectas que unen los vértices de los obstáculos tales que no intersectan con el interior de alguno de los otros obstáculos, se aplica principalmente a espacios de configuración de dos dimensiones con una región poligonal de N-obstáculos y comprende el punto inicial (Nodo 0), desde donde parten todas las trayectorias que sean factibles; el punto final (Nodo 1), que es la meta de las trayectorias y todos los vértices de los N-obstáculos.

Este método tiene una característica muy importante y es que dos vértices pueden estar conectados si y sólo si son **"*visibles*"**, es decir, se puede alcanzar el siguiente vértice desde el vértice anterior (o viceversa) al seguir la línea recta que los une, sin interceptar algún obstáculo del entorno. También se consideran ***visibles*** si la recta que une a los dos vértices coincide con una arista de los polígonos que se encuentran en el entorno como obstáculos [7].

En la figura 2 se pueden observar las diferentes trayectorias que genera el mapa de visibilidad antes de considerar la geometría del robot, es por esta razón que las rutas factibles consideran los vértices de los obstáculos como parte de su camino para llegar al punto final.

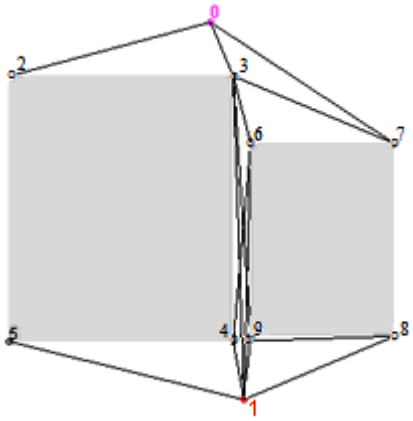

Figura 2. Generación de trayectorias mediante el grafo de visibilidad considerando al robot como punto en el espacio.

Para tratar la navegación del robot sin considerarlo como un punto en el espacio debemos aumentar el tamaño de cada obstáculo exactamente el radio del robot. De esta manera, se pueden introducir las técnicas comunes para la planificación de caminos del robot móvil [8]. Debido a esta nueva condición la cantidad de trayectorias generadas por el mapa de visibilidad son menores (Ver figura 3), ya que algunas rutas que antes eran factibles deben ser descartadas porque al aumentar el tamaño de los polígonos se pueden intersectar rectas entre los "nuevos" obstáculos, lo cual indicaría en un espacio real que el robot no podría navegar por alguna de esas trayectorias.

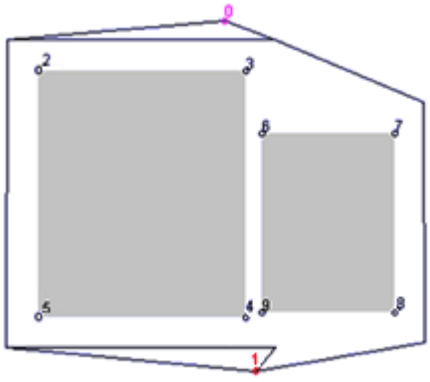

Figura 3. Generación de trayectorias mediante grafos de visibilidad considerando las dimensiones del robot.

Cuando se aplica el método de la expansión de los obstáculos los vértices con los cuales se halla la ruta óptima son aquellos que pertenecen a los nuevos polígonos, es por ello que las nuevas

trayectorias factibles no pasan por ningún punto perteneciente al polígono original o polígonos originales, de allí viene la importancia en la implementación del algoritmo para validar los casos especiales y saber con cual vector de vértices trabajar para que los algoritmos Dijkstra y genético encuentren la ruta óptima.

C. Algoritmo genético. Los algoritmos genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Éstos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas [9], a partir de una población inicial y mediante el uso de los operadores como la selección, el cruzamiento y la mutación se crean nuevas generaciones para encontrar la ruta óptima entre un punto inicial, un punto final y una cantidad cualquier de obstáculos entre ellos.

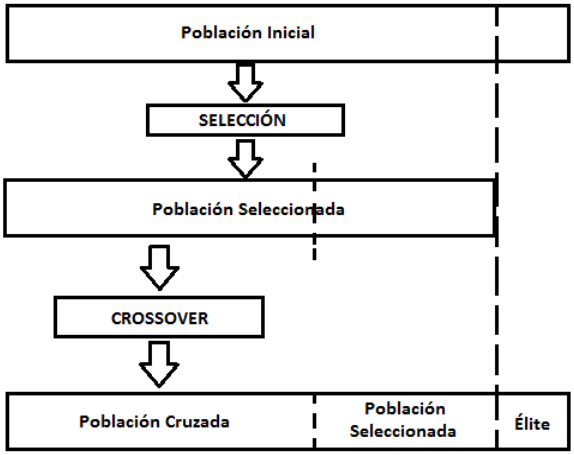

Figura 4. Arquitectura del algoritmo genético.

Como se muestra en la figura 4 el método inicia con una población inicial, esta población es generada aleatoriamente, es decir, mediante el uso del método de la heurística se generan rutas al azar desde el punto de inicio hasta el punto final. La heurística es un método que permite obtener soluciones subóptimas de manera rápida [10], lo cual ayuda a hallar una población que contenga las posibles soluciones del problema; a partir de estas soluciones y mediante el uso de los operadores del algoritmo genético se obtiene la ruta óptima.

Los Algoritmos Genéticos en general, no funcionan bien en problemas del tipo de particiones, estos problemas consisten en dividir el espacio en partes de forma de cubrirlo todo, sin repetición, optimizando una o más cantidades [11]. Es por ello que no todos los algoritmos genéticos usan todos los operadores, existen dos tipos de grupos de algoritmos genéticos que recomiendan no usar el operador mutación debido a que posiblemente pueda dañar algún individuo candidato a ser la solución óptima o en el peor de los casos la solución óptima del problema. Estos algoritmos son: ***Niching Genetic Algorithms*** y ***Grouping Genetic Algorithm*** [12].

Antes de alterar la población inicial o las siguientes generaciones, es decir, antes de aplicar el operador selección se toma el 10% de la población como un grupo élite, es decir, se toman los mejores individuos de la población que se encuentren dentro de ese 10% como soluciones para el problema.

Este algoritmo hace uso del operador selección, existen diferentes formas de seleccionar la nueva generación que contenga los mejores hijos de los padres de la población inicial o de las generaciones siguientes. Este algoritmo selecciona por torneo (Ver figura 5), método que selecciona dos individuos aleatoriamente y escoge el mejor entre los dos basándose en el resultado de su función fitness, que para este proyecto es la acumulación de la distancia que recorre el individuo desde el punto inicial hasta el punto final. Existe otro método muy conocido y utilizado en los algoritmo genéticos y es la selección por ruleta pero debido al gasto computacional que conlleva no se recomienda para este tipo de proyectos donde se generan cromosomas permutados sin repetición lo cual reduce la cantidad de soluciones posibles para el problema, de esta manera, los porcentajes que se asocian por el método de ruleta a los mejores individuos serían muy cercanos entre ellos y conociendo el funcionamiento de éste se convierte en un proceso innecesario que podría ahorrarse con la selección por torneo.

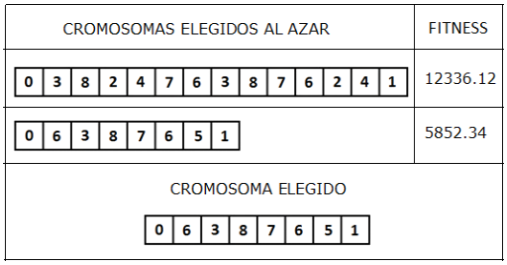

Figura 5. Selección por torneo.

Este algoritmo también hace uso del operador cruzamiento (Ver figura 6), este operador se ejecuta siempre luego de la selección, consiste en tomar individuos aleatoriamente y cruzar sus genes con un porcentaje, que en este caso es el 40% y de los dos padres escogidos resultan dos hijos, tal como se muestra a continuación:

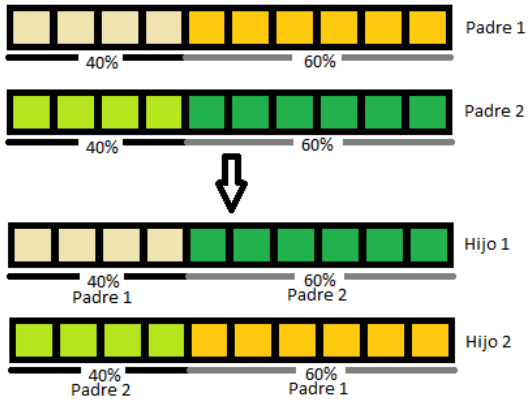

Figura 6. Operador cruzamiento.

Como se mencionó anteriormente, este algoritmo genera cromosomas permutados sin repetición, lo cual hace que la cantidad de posibles soluciones sea reducida y el orden de los genes en los cromosomas sea importante, por lo tanto no se aplica el operador mutación, debido a que éste lo que hace es cambiar aleatoriamente uno de los genes para crear un nuevo cromosoma que en teoría para este caso sería una trayectoria candidata a ser óptima, se dice en teoría porque hay que tener en cuenta que las posibles trayectorias se generaron por medio del mapa de visibilidad, en el algoritmo cada cromosoma es un camino factible que empieza desde el punto inicial formando la trayectoria vértice a vértice de los obstáculos que le sea posible hasta llegar al punto final, si se aplicara la mutación se cambiaría un vértice por otro que posiblemente no sería factible, pues aplicando el mapa de visibilidad no sería un trayectoria permitida dañando así una posible solución óptima al problema.

En la figura 7 y en la figura 8 se muestra la ruta óptima generada por medio del algoritmo genético sin tener en cuenta la geometría y tomándola en cuenta respectivamente. El algoritmo genético crea combinaciones de cromosomas con los vértices de los nuevos obstáculos, es decir, los vértices de los polígonos expandidos, los puntos inicial y final hasta encontrar la trayectoria más corta; existe un caso especial y es cuando la ruta óptima solo contiene los puntos inicial y final, por ello se hace la validación en el algoritmo genético para este tipo de casos ya que puede o no salir en esa combinación de cromosomas y es necesaria la existencia de éste pues es la ruta óptima.

D. Algoritmo dijkstra. Este algoritmo recorre todos los caminos más cortos que parten del punto inicial y que llevan a todos los vértices de los obstáculos. Cuando se obtiene el camino más corto desde el punto inicial al resto de los vértices el algoritmo se detiene y es porque se ha llegado hasta el punto final. Debido a que en este algoritmo se visitan todos los nodos, el cálculo de una solución para grafos extensos se hace lento. Pero a comparación de otros métodos el gasto computacional es menor y la solución óptima se obtiene en poco tiempo.

A continuación se presenta la ruta óptima sin geometría y con geometría obtenida mediante el algoritmo de búsqueda dijkstra partiendo del ejemplo del mapa de visibilidad propuesto en el capítulo 2. Como se puede ver en la figura 2 y la figura 3 debido a que al expandir el tamaño de los obstáculos se entrecruzan los nuevos polígonos las trayectorias que podían ser factibles dejaron de ser consideradas debido a que en un entorno real el robot no lograría navegar por ese espacio. Es por eso que la ruta óptima ya no es un camino directo desde el punto inicial hasta el punto final sino que al tener en cuenta las dimensiones del robot se encuentra la nueva ruta más corta y como se puede ver claramente es por la parte inferior.
La trayectoria óptima hallada por los dos algoritmos de búsqueda estudiados es:

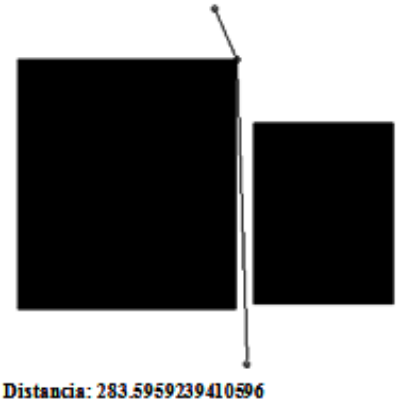

Figura 7. Ruta óptima considerando al robot un punto en el espacio.

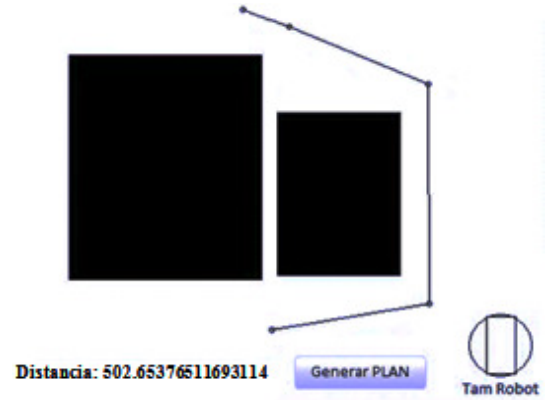

Figura 8. Ruta óptima considerando al robot como un objeto en el espacio.

Este ejemplo se realizó con el propósito de presentar la diferencia de la ruta óptima tomando en cuenta las dimensiones del robot en la navegación y considerándolo como un punto en el espacio. Si bien es cierto si se toma en cuenta la geometría del robot las rutas factibles entre las cajas desaparecen, tal como se ve en las figuras 2 y 3. Aun cuando la distancia recorrida en la figura 8 es mayor a comparación de la distancia que se recorre en la figura 7, dentro de las nuevas trayectorias factibles para este caso, esa es la mejor.

E. Arquitectura del algoritmo. El algoritmo implementado en java para este trabajo consta de dos partes fundamentales: la construcción del grafo y el algoritmo de búsqueda.

Para la implementación del mapa de visibilidad se usaron las herramientas de java, es decir, los graphics que trae incorporado este lenguaje en la librería java.awt.Graphics, éstos lo que permiten es hacer visible cada vértice y línea del grafo.

Dentro de los algoritmos de búsqueda, el código fuente del algoritmo dijkstra se descargó de la página algowiki.net [13] y el algoritmo genético fue completamente implementado. Para este último se creó un vector de rutas desde donde se creó

aleatoriamente la población inicial, luego se programó la selección por torneo y usando el random de java se tomaron los cuatro cromosomas padres y de acuerdo al resultado de su función fitness se escogieron los dos mejores. Después se usaba la función "Validar" que rectificaba si el cromosoma hijo era una ruta factible, los hijos válidos eran incorporados en la nueva generación. Este proceso se repite la cantidad de veces que proponga el usuario en la ventana principal. En java lejos se hace la implementación de un algoritmo que dirige la movilidad del robot en el espacio real que ha sido creado a partir del proyecto en java con el mapa de visibilidad.

Para asegurar un poco más de exactitud en los giros del robot móvil se hizo uso del CompassSensor, con éste se conoce el ángulo de giro real del robot y mediante una función denominada "Rotar", esta función ayuda a minimizar el error en los giros del robot y resultó ser mucho más preciso el DiferrencialPilot fusionado con la función "Rotar" que la clase CompassPilot de Lejos. Esto se debe a que la función rotar lo que hace es que cuando el robot gira compara los grados con lo que arroja el compás y se termina de girar los grados faltantes mientras que la clase CompassPilot navega en el espacio y por su misma imprecisión va acumulando el error excesivamente.

Debido a la ausencia de la clase TachoPilot en la versión que se está utilizando de Lejos (0.9) se debe utilizar DifferentialPilot, ambas cumplen la misma función. La clase DifferentialPilot está ubicada en el paquete lejos.robotics.navigation, con ésta se dirige el robot móvil mediante el control de la velocidad y la dirección de la rotación de los motores. Para el uso de esta clase es necesario saber el diámetro de las ruedas y la medida exacta que existe entre las mismas; el primer dato lo utiliza para calcular la distancia que ha recorrido y el segundo para calcular qué tanto ha girado el robot.

F. Pruebas. Prueba 1: Esta prueba se realizó con el propósito de mostrar la eficiencia del algoritmo genético para encontrar la ruta óptima.

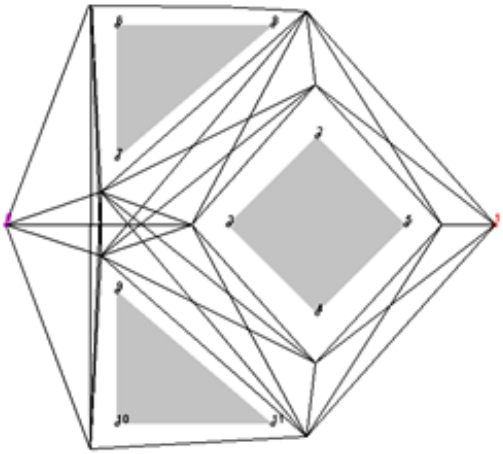

Figura 10. Mapa de visibilidad y generación de trayectorias mediante grafos de visibilidad considerando las dimensiones del robot.

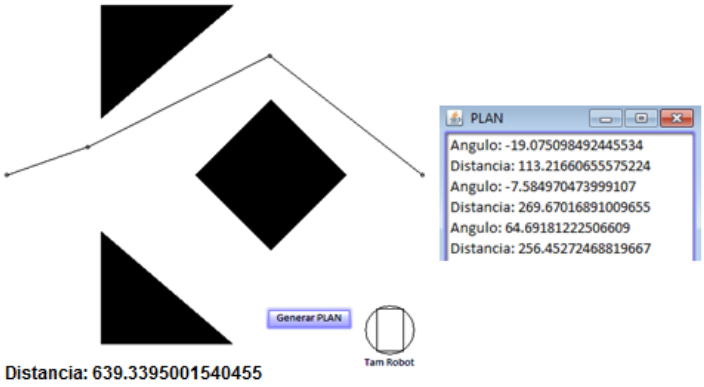

Figura 12. Ruta óptima y plan de navegación.

Prueba 2. Este ejemplo se realizó con el propósito de presentar que aunque la expansión de los obstáculos se hizo bajo la medida del radio de una circunferencia que encerraba al robot no es ésta la que define el tamaño del mismo en el momento de la navegación.

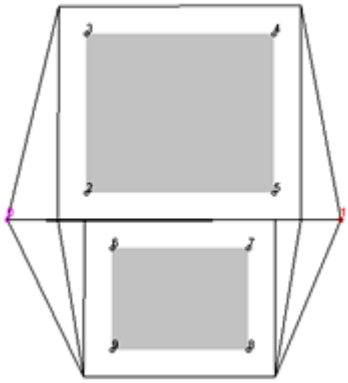

Figura 13. Mapa de visibilidad y generación de trayectorias mediante grafos de visibilidad considerando las dimensiones del robot.

Figura 15. Ruta óptima y plan de navegación.

III. CONCLUSIONES

A partir de las pruebas es necesario resaltar la importancia de la posición inicial del robot debido a que cualquier desvío adicional afecta la trayectoria considerablemente.

Los métodos de búsqueda estudiados arrojan la misma ruta óptima, lo cual era de esperarse si están implementados correctamente, la diferencia entre ellos es que el algoritmo dijkstra tuvo un gasto computacional mucho menor que el algoritmo genético utilizando como herramienta de medición el "Profile Project" de Netbeans.

El algoritmo genético no alcanza a demostrar su utilidad en su totalidad debido a que son pocas las trayectorias subóptimas que tiene para encontrar la mejor y de acuerdo a su funcionamiento crea cromosomas que son innecesarios para obtener la ruta óptima de manera más eficiente. El algoritmo genético puede arrojar mejores resultados cuando no se asocie con grafos, es decir, cuando tenga la disponibilidad de puntos posibles para encontrar la trayectoria óptima.

Este algoritmo genético funciona correctamente si se toman en cuanta dos aspectos importantes; el primero es que se le debe dar prioridad al número de población y el segundo es que debe estar muy bien configurado, pues si la población y el número de generaciones es muy grande genera un gasto computacional excesivo e innecesario pues la ruta óptima pudo haberse hallado en las primeras generaciones; por el contrario, si la población y el número de generaciones es muy pequeño no se garantiza encontrar la solución óptima.

El uso de la clase DiferrencialPilot y la función "Rotar" arroja resultados mucho más precisos que la clase CompassPilot de Lejos. Teóricamente para utilizar la clase DifferentialPilot se debe conocer el diámetro de las llantas del robot y de acuerdo con las medidas que trae el kit de Lego ese diámetro es 5.6 centímetros. La precisión en los parámetros enviados a esta clase es importante debido a que entre más exactos sean éstos, mejor es el resultado.

Este robot está diseñado para que el peso del mismo tenga su mayor concentración en la parte delantera, con el fin de que la rueda loca trasera no ejerza mucha fricción con el suelo. Al realizar las pruebas se propuso que el diámetro de las ruedas fuera 5.4 centímetros debido a que el peso del robot influía en dicha medida.

IV. RECOMENDACIONES

Este trabajo es apoyado por el proyecto "Programa de Fortalecimiento a Grupos de Investigación y de Creación artística de la Universidad Nacional de Colombia", mediante el cual se apoyan semilleros de investigación de la sede.

REFERENCIAS

[1] Muñoz Martínez Victor Fernando. *"Navegación en Robots Móviles"*. Capítulo 2. Páginas 21-23.

[2] Universidad de Salamanca. *"Cálculo de Caminos Óptimos para Manipuladores Articulados"*. Página 18. Año 2004.

[3] Hurtado Ferran, Palop Belén y Sacristán Vera. *"Diagramas de Voronoi con Funciones Temporales"*. Año 2004.

[4] Amaro Camargo Erika, Cruz Almanza Graciano. *"Sistema Didáctico Basado en Tres Algoritmos Fundamentales de Planificación de Trayectorias Desarrollado en Java"*. Año 2003.

[5] Restrepo C. Jorge Hernán, Sánchez C. John Jairo. *"Aplicación de la Teoría De Grafos y el Algoritmo Dijkstra para Determinar las Distancias y las Rutas más Cortas en una Ciudad"*. Año 2004.

[6] Benavides Facundo, Tejera Gonzalo, Pedemonte Martín, Casella Serrana. *"Real Path Planning based on Genetic Algorithm and Voronoi Diagrams"*. IEEE LARC Y CCAC. PDF 116.

[7] Amaro Camargo Erika, Cruz Almanza Graciano. *"Sistema Didáctico Basado en Tres Algoritmos Fundamentales de Planificación de Trayectorias Desarrollado en Java"* Año 2003.

[8] Siegwart Roland, Nourbakhsh Illah. *"Introduction to Autonomous Mobile Robots"*. Página 261.

[9] "*Algoritmos Genéticos*". Disponible en: http://eddyalfaro.galeon.com/geneticos.html

[10] Rey Juan Carlos. *"Aplicaciones de la visión artificial y la biometría informática"*. Madrid, España. Página 66.

[11] Wil Adrian Ph.D. *"Algoritmos Genéticos Y Optimización Heurística"*. Grupo de Aplicaciones de Inteligencia Artificial. Universidad Nacional de Tucumán. Clase Magistral 7. Diapositiva 27.

[12] Wil Adrian Ph.D. *"Algoritmos Genéticos Y Optimización Heurística"*. Grupo de Aplicaciones de Inteligencia Artificial. Universidad Nacional de Tucumán. Clase Magistral 7. Diapositivas 1-46.

[13] Implementación del Algoritmo Dikjstra. Disponible en:http://algowiki.net/wiki/index.php?title=Dijkstra%27s_algorithm.