Instituto Superior Politécnico

"José Antonio Echeverría"

cujae

MAESTRÍA EN SISTEMAS DIGITALES

7MA EDICIÓN

Universidad de las Ciencias Informáticas

PLANIFICACIÓN CON ROBOTS MÓVILES BASADA EN AGENTES EN ENTORNOS ESTRUCTURADOS

Autor: Lic. Orlando A. Valenzuela Aguilera

Tutores: Dr. Valery Moreno Vega.
Msc. Maikel O. Torres Piñeiro

La Habana, Cuba

2014

DEDICATORIA

A dios por darnos la existencia, a la memoria de mis padres, a mi familia.

AGRADECIMIENTOS

A mis hermanos, sobrinos y a mi cuñado Alexey que es un hermano más para nosotros.

A mis compañeros de trabajo por su ayuda y estímulo en el logro de este objetivo.

A todas las personas que han contribuido en mi formación, en especial al claustro de profesores de la maestría.

A mis tutores por su sabia orientación, por iniciarme en este campo, por su ayuda en este empeño, que no es una meta, es un comienzo.

A todos, muchas gracias.

RESUMEN

El desarrollo de la robótica móvil ha ido creciendo junto al desarrollo de la tecnología. En los últimos años nuevos campos de aplicación se han establecido, desde la esfera de los servicios, los juegos, hasta aplicaciones militares, de mapeo de superficies, y exploración en diferentes medios. Este desarrollo ha abierto nuevas líneas de investigación orientadas a la planificación, generación, y control de trayectorias de robots. Los nuevos algoritmos de planificación y generación necesitan ser simulados en entornos similares a los reales para poder probar la efectividad de los mismos, y las mejoras que pudieran introducir con relación a los existentes. En este sentido, este trabajo propone un entorno de planificación de trayectorias para robots móviles en entornos estructurados basado en agentes. En el trabajo se explica el diseño del planificador, la justificación del uso de agentes y las métricas principales para comparar los algoritmos de planificación (costo total, tiempo, número de iteraciones). Como resultado del trabajo se obtuvo una aplicación que haciendo uso de agentes permite crear un entorno para el movimiento del robot, seleccionar un algoritmo de planificación y simular la ejecución del mismo, mostrándose el rendimiento del algoritmo para ese entorno en términos de tiempo, costo total y número de iteraciones.

INDICE

INTRODUCCIÓN

La robótica móvil y sus aplicaciones constituyen un campo de investigación de gran interés a nivel mundial. Este hecho ha sido motivado en gran medida por la creciente necesidad de emplear robots que puedan sustituir a los seres humanos en actividades riesgosas como la manipulación y transportación de material peligroso, el rescate y salvamento, las excavaciones mineras, la limpieza industrial, entre disímiles tareas, logrando mayor eficiencia en la producción (Guzmán, 2008).

Para que las aplicaciones mencionadas anteriormente u otras se cumplan satisfactoriamente, el robot debe tener cierto grado de autonomía, de manera que bajo determinadas circunstancias él pueda tomar sus propias decisiones. Desde el punto de vista de la navegación, teniendo en cuenta la autonomía con respecto al hombre, los robots pueden ser teledirigidos o autónomos. En los teledirigidos un operador humano es quien guía al vehículo a través de un enlace, ya sea inalámbrico o por cables. Los autónomos deben ser capaces de recorrer el espacio de trabajo sin la guía de un operador humano. Para esto emplean técnicas de navegación donde se tienen en cuenta la percepción del entorno a través de sensores, la planeación, la generación y el control de la trayectoria.

La planificación es donde se obtiene una representación del espacio de trabajo y una ruta que conduzca de un estado inicial a uno final (Muñoz, 1995), la generación suaviza la ruta y le impregna un perfil cinemático convirtiéndola en un camino y el control tiene en cuenta el camino calculado, las velocidades requeridas y calcula la señal que deben ser enviadas a los accionamientos del robot.

En estos robots el éxito en el cumplimiento de una misión depende de varios factores entre los que se pueden mencionar los algoritmos tanto de control como de planificación y generación que se hayan utilizado. Estos algoritmos varían en función de la caracterización del espacio de trabajo del robot que pueden ser estructurados y no estructurados. El estructurado es aquel donde los elementos que lo componen no varían su posición en el tiempo, mientras que en el no estructurado pueden existir elementos u objetos con movilidad. A pesar de la existencia de una gran variedad de métodos de planeación y generación de trayectorias, este aún sigue siendo un campo de investigación activo.

Dentro de las características deseables de los algoritmos de planificación más importantes se encuentran: *Plenitud*, es la garantía de encontrar la solución si existe; *Optimización*, es la garantía de encontrar la solución de menor costo; *Complejidad tiempo/espacio*, es el tiempo y la memoria usada por el algoritmo, visto como la cantidad de estados explorados dentro de la estructura que describe el espacio de trabajo.

Para obtener criterio de las características deseables de los algoritmos de planificación y generación estos necesitan ser simulados en entornos similares a los reales, para poder probar la efectividad de los mismos, y las mejoras que pudieran introducir con relación a los existentes. En el Departamento de Automática del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE), como parte del proyecto de investigación en la robótica móvil, se necesita una herramienta de experimentación que permita poner a prueba diferentes tipos de algoritmos relacionados con la planificación para obtener elementos de comparación basados en ciertas métricas que faciliten la selección del más adecuado a utilizar. A su vez, se requiere verificar una arquitectura basada en agentes que permita posteriormente crear a nivel de laboratorio diferentes escenarios donde uno o más robots se coordinen con relativa autonomía.

Teniendo en cuenta la necesidad de evaluar los distintos algoritmos para la planificación de caminos se define en la presente investigación el siguiente **problema científico:** *Necesidad de desarrollo de un planificador basado en agentes que permita evaluar el desempeño de diferentes algoritmos en entornos estructurados.*

Para darle solución al problema planteado se define como:

Objetivo general

- Desarrollar una plataforma basada en agentes para la prueba de los algoritmos de planificación de trayectorias de robots móviles en espacio de trabajo estructurado.

Objetivos específicos

1. Crear el espacio de trabajo estructurado donde se represente la interacción de los robots y los obstáculos.

2. Desarrollar el planificador encargado de obtener una ruta a partir del entorno descrito.
3. Desarrollar el generador de caminos a partir de la ruta obtenida por el planificador.
4. Brindar la visualización de los criterios de optimización, tiempo/espacio, que permitan determinar su factibilidad en situaciones reales.

Tareas

1. Estudio de las principales tendencias actuales de los métodos de navegación para robots móviles.
2. Implementación de la descomposición en celdas cuadradas del entorno.
3. Implementación de los algoritmos de obtención de ruta A*, Bellman-Ford, SACO.
4. Implementación del algoritmo de generación ß-Spline.
5. Implementación de la plataforma mediante agentes.
6. Validación de la plataforma.

Hipótesis de investigación:

Si se desarrolla una plataforma basada en agentes para robots móviles en espacios de trabajo estructurados se puede obtener elementos de comparación entre los algoritmos de planificación que permitan determinar su factibilidad en situaciones reales.

Se utilizaron los siguientes métodos de investigación:

Teóricos

***Histórico-lógico*:** Este método científico facilitó realizar un estudio del estado del arte de la robótica móvil, centrado en la navegación autónoma en entornos estructurados, permitiendo conocer en profundidad las etapas por la que está compuesta o en las que se puede dividir (planificación, generación, control), así como las principales soluciones existentes en la literatura especializada. Esto permitió la selección e implementación de la descomposición en celdas cuadradas y los algoritmos A*, Bellman-Ford, SACO para obtención de la ruta; así como la determinación de las principales herramientas y estándares existentes para el trabajo con agentes.

Analítico-sintético: Facilitó el análisis documental de la bibliografía que recoge los elementos del estado del arte de la navegación para robots móviles.

Inducción-deducción: Al aplicar este método se estudiaron a profundidad los algoritmos de planificación más utilizados actualmente con el fin de definir sus características particulares, beneficios y cualidades, determinando que la configuración que adquiere la ruta se define por la distribución de los obstáculos a lo largo de todo el ambiente de trabajo y por supuesto, de la geometría del robot, su posición referenciada a un marco absoluto generalmente expresada por la combinación de las coordenadas cartesianas del centro del robot y la posición angular del eje principal de este, y de sus capacidades de movimiento.

Hipotético-deductivo: A partir de la interpretación de la realidad se establecen posibles situaciones o resultados para llegar a conclusiones.

Modelación: Usado en el proceso de conformación del espacio de trabajo donde se desempeñan los robots y en la creación de la plataforma.

Empíricos

Entrevista: Se empleó para comprobar la necesidad y la funcionalidad práctica de los algoritmos de generación y planificación de trayectorias evaluados en la investigación, realizando las mismas a especialistas en el tema.

Los aportes esperados de esta investigación son:

Aporte Teórico

1- Arquitectura de *software* basada en agentes para la puesta a punto de los algoritmos de planificación.

Aporte Práctico

1. Plataforma para la evaluación de los diferentes algoritmos de planificación de trayectoria.

La ***novedad científica*** radica en obtener una plataforma para la prueba de los algoritmos de planificación en entornos estructurados utilizando agentes, que permita hacer valoraciones de desempeño de los mismos.

La tesis se encuentra estructurada en tres capítulos. En el primer capítulo se realiza un análisis bibliográfico acerca de los métodos de planificación y generación de trayectorias, concepto de agente, estándares de agentes, marcos de desarrollo con agentes así como la metodología seleccionada para guiar el desarrollo con agentes. El segundo capítulo trata los algoritmos empleados en la planificación y generación de trayectorias. El tercero se destinó a la propuesta de la plataforma y la aplicación de los resultados obtenidos sobre un caso de estudio.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

La robótica móvil es un área de investigación desde hace varias décadas. Desde sus inicios la tarea de dotar con habilidades de navegación a un robot móvil fue compleja. Es por esto que dicho problema, generalmente, se ha dividido en varias etapas: planificación, generación y control de la trayectoria.

Para resolver cada una de las etapas mencionadas anteriormente existe una gran variedad de métodos, siendo de interés en esta tesis aquellos que se enfocan en la planificación y generación de trayectorias.

Estos métodos tienen en cuenta dos tipos de espacios de trabajo: estructurados y no estructurados. La investigación se centró en soluciones para espacios de trabajo estructurados. En este capítulo se realizó un estudio de las principales herramientas de simulación existentes vinculadas a la robótica móvil y las principales tendencias en la planificación y generación de trayectorias. Además se definió y justificó el empleo de agentes, se detallaron los estándares para su implementación, así como las principales herramientas para el desarrollo basada en agentes. Además se seleccionó la metodología de desarrollo para agentes que guió la investigación.

1.1 Soluciones existentes

En el estudio realizado se identificaron varias herramientas vinculadas a la robótica móvil. Uno de los pioneros fue SRIsim (SRIsim, 2008), (K. Konolige, 1997) que permitió simular un único robot en un mundo bidimensional estático. Actualmente se dispone por ejemplo de simuladores tridimensionales, como el que proporciona Microsoft en su Robotics Studio (MicroSoft Robotics Studio, 2009). Webots (Webots, 2009). Este simulador se comenzó a desarrollar en el Laboratory of Micro-Computers (LAMI) del Swiss Federal Institute of Technology. La plataforma PSG (Player-Stage-Gazebo, 2009), (Gerkey, 2003) está formada por los simuladores Stage y Gazebo, y por el servidor Player de acceso al hardware real o simulado (EPFL). La mayoría de estas herramientas están enfocadas a emular el hardware del robot, todas acarrean una política de licencias que dificulta el acceso o son propietarias, además no están orientadas a obtener parámetros que permitan emitir un criterio de evaluación de los algoritmos de planificación.

En trabajos como (Porta, 2007), (Alfonso, 2004) se proponen herramientas enfocadas a la evaluación de algoritmos de planificación. La principales similitudes entre estas herramientas es que son monorobot y solo soportan un algoritmo de planificación. El estudio evidencia la necesidad de herramientas escalables, con soporte multirobot que implementen varios algoritmos.

1.2 Planificación de trayectorias

La mayoría de los métodos de planificación tienen determinados pasos o características en común, a pesar de las grandes diferencias que pueden existir entre ellos. Estos pasos consisten en:

- Transformar el espacio de trabajo donde se encuentra el robot en un nuevo espacio llamado espacio de configuración.
- Construir un grafo que represente la conectividad del espacio de configuración.
- Buscar en el grafo la trayectoria que guíe al robot hasta el objetivo.

1.2.1 Espacio de configuración

El espacio de configuración C se define como el conjunto de todas las posibles configuraciones en que puede posicionarse un robot. Cada punto de C es una tupla de una cierta dimensión, donde se especifican los valores para los parámetros que se corresponden con los grados de libertad del robot, que en el caso de la robótica móvil viene dado por la posición y la orientación, $q = (p; \mu) = (x; y; \mu)$ (Blanco Rodríguez, 2003), (Lozano-Pérez, 1983).

$$C_l = \left\{ q \in C / R(q) \cap \left(\cup_{i=1}^{q} O_i(q) \right) = \emptyset \right\}$$

donde, $R(q)$ es el subconjunto de C ocupado por el robot, que en el caso de un robot puntual $R(q) = \{q\}$, $O_i(q)$ es el conjunto de configuraciones del espacio C ocupadas por un obstáculo y C_l es el espacio de configuraciones libre de obstáculos.

1.2.2 Descomposición del espacio de trabajo

Para la descomposición del espacio de trabajo existen tres formas principales de obtener el espacio de configuraciones: Descomposición en celdas, Grafos de visibilidad y Diagramas de Voronoi.

1.2.2.1 Descomposición en celdas

Este tipo de método se fundamenta en una descomposición en celdas del espacio libre (Thorphe ,1984). Así, la búsqueda de una ruta desde una postura inicial *qi* hasta otra final *qf*, consiste en encontrar una sucesión de celdas que no presenten discontinuidades, tal que la primera de ellas contenga a *qi* y la última a *qf.* Estas celdas pueden adoptar una determinada forma geométrica.

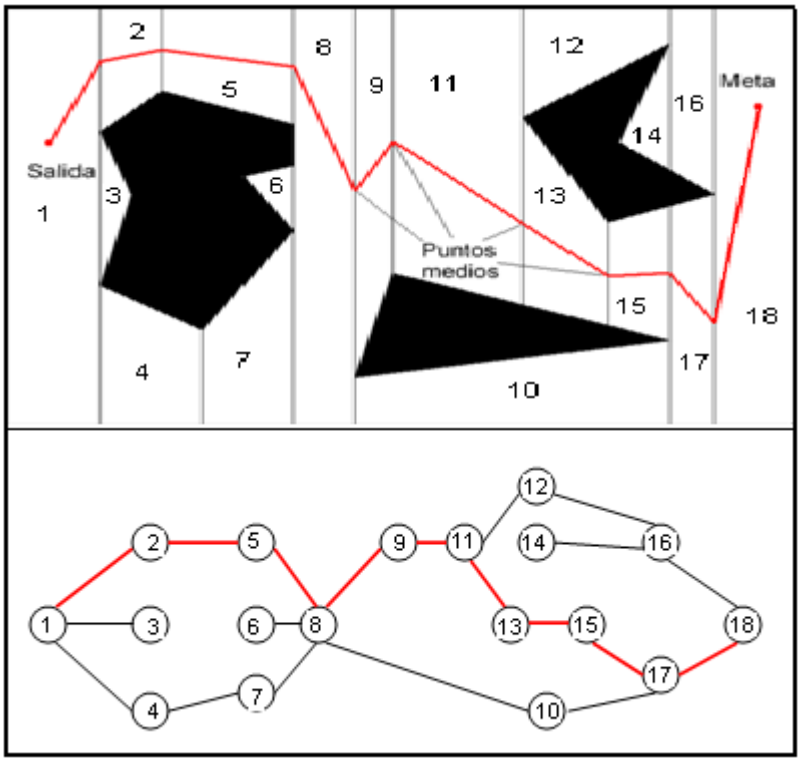


Figura 1.1 Descomposición en celdas trapezoidal.

La comprobación para averiguar si dos celdas son adyacentes debe ser lo más simple posible. La descomposición global del espacio libre implica que no deben existir solapamientos entre celdas y que la unión de todas ellas corresponda exactamente al espacio libre. A partir de esta descomposición en celdas se construye un grafo de conectividad no dirigido, de tal forma que los nodos van a ser cada una de las celdas, existiendo un arco entre dos celdas sí y solo sí son adyacentes. Una vez especificado el grafo de conectividad, sólo queda emplear un algoritmo de búsqueda en grafos que

determine una ruta de la celda que contiene la postura inicial hasta la que se desea llegar.

Existen varias formas de llevar a cabo la descomposición en celdas, dentro de las cuales se pueden encontrar la descomposición aproximada (Blanco Rodríguez, 2003), la adaptativa, (Nagabhushan and Pai, 2001), celdas cuadradas (Torres, 2010), y la exacta (Sleumer and Tschichold-Gurman, 1999). Según la bibliografía consultada la descomposición en celdas es uno de los métodos más utilizados.

1.2.2.2 Grafos de visibilidad

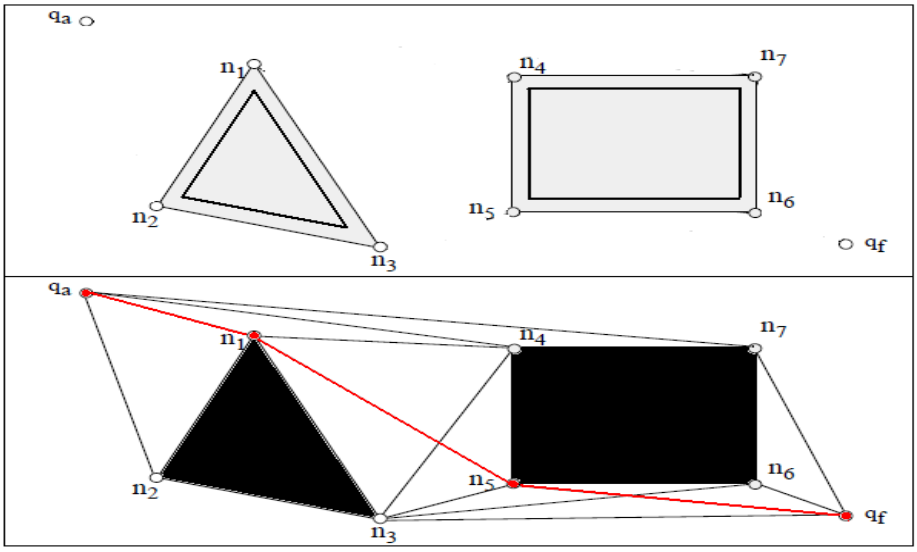
El término de visibilidad alude a que en este método se enlazan los vértices entre los que existe visión directa, es decir, sin obstáculos que lo impidan. El método de grafos de visibilidad como otros planificadores, precisa de una descripción poligonal de los obstáculos. Se determinan las coordenadas de los vértices de los obstáculos y se calcula el polígono convexo envolvente teniendo en cuenta las dimensiones del robot, si existe solapamiento entre los obstáculos tras el ensanchamiento de los polígonos se unen como si fuesen un único obstáculo y se aplica de nuevo el cálculo envolvente.

Así, el entorno queda definido por los límites del plano, el robot que se puede tratar de manera puntual y un conjunto de polígonos que identifican los obstáculos. Dentro del espacio de configuraciones libres de colisión se sitúan las posiciones de partida y meta, quedando completamente definido el problema.

El proceso comienza generando una lista de los vértices de los polígonos a los que se añaden las posiciones inicial y final (Muñoz, 1995). Con estos puntos se genera una matriz cuadrada con tantas filas como puntos. A continuación se evalúa para cada par de puntos su posibilidad de enlace directo, es decir, si el robot ubicado en uno de los puntos puede alcanzar el otro desplazándose en línea recta. Si un obstáculo no lo impide, esto será posible, y la longitud del tramo a recorrer se almacena en la matriz indicando el coste de realizar dicho enlace. Si por el contrario existe un obstáculo, en la matriz se anotará un valor que indique tal eventualidad. Esta matriz no es más que la descripción de un grafo que une los puntos entre los que es posible el enlace directo.

Solo queda emplear un algoritmo de búsqueda en grafos que determine una ruta de la postura inicial hasta la que se desea llegar. La principal desventaja de este método es

que los vértices de los obstáculos forman parte de la ruta, por lo que pueden presentarse colisiones.



¡Error! Utilice la pestaña Inicio para aplicar 0 al texto que desea que aparezca aquí.1.2 Grafo de visibilidad.

1.2.2.3 Diagramas de Voronoi

Al contrario de los métodos basados en grafos de visibilidad, la planificación basada en diagramas de Voronoi sitúa la ruta lo más alejada posible de los obstáculos. Al igual que en el método anterior, la aplicación de éste a la planificación exige el modelado de los obstáculos como polígonos, lo cual permite adoptar la analogía del robot puntual, es decir, reducir la búsqueda de una solución al identificar el camino seguido por un punto en el plano (Wilmarth, 1998). El diagrama de Voronoi resulta el lugar geométrico de las configuraciones que se encuentran a igual distancia de los dos obstáculos más próximos del entorno. El diagrama estará formado por dos tipos de segmentos: rectilíneos y parabólicos. La elección de la modalidad de segmento corresponde con la clase de elementos de los obstáculos más cercanos que se encuentren enfrentados entre sí. De esta forma, el lugar geométrico de las configuraciones que se hallan a igual distancia de dos aristas de dos obstáculos diferentes es una línea recta, mientras que en el caso de tratarse de un vértice y una arista resulta una parábola.

La ventaja de este método radica en su capacidad para alejarse de los obstáculos, eligiendo siempre el camino más despejado posible. Esto puede ser muy interesante en

escenarios con abundantes obstáculos y pasos estrechos, sin embargo resulta un inconveniente en entornos con escasos obstáculos. En este caso, dependiendo de la ubicación de los mismos, el diagrama arrojará trayectorias que pueden dar rodeos largos e innecesarios para llegar a la meta (Muñoz, 1995).

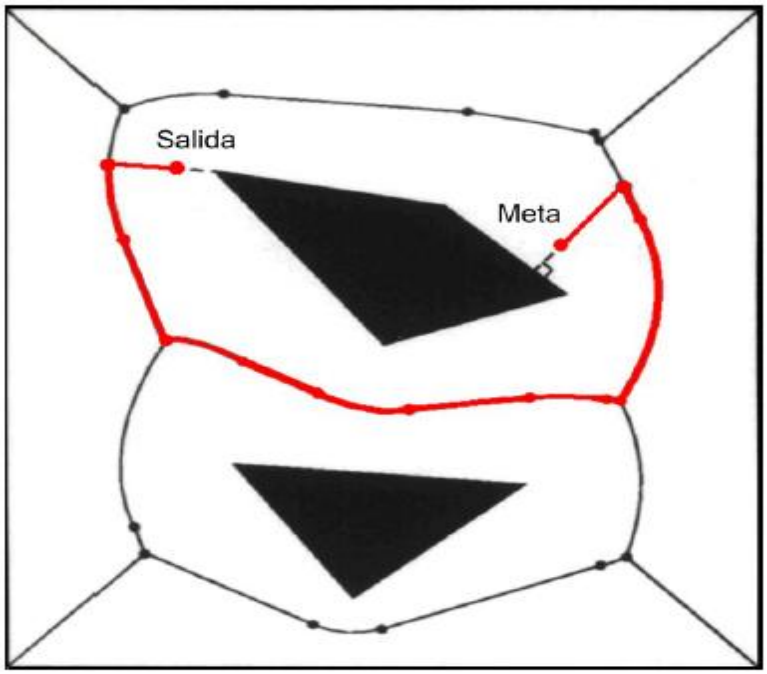


Figura 1.3 Diagrama de Voronoi

El grafo estará constituido por los vértices de los segmentos rectilíneos y parabólicos, en el que empleando un algoritmo de búsqueda en grafos se determina una ruta de la postura inicial hasta la que se desea llegar.

Del estudio realizado se puede concluir que la descomposición en celdas es uno de los métodos más utilizados. Maximiza la distancia entre la ruta y los obstáculos. Es el método utilizado en la investigación, específicamente la descomposición en celdas cuadradas propuesto por (Torres, 2010).

1.2.3 Algoritmos de búsquedas

En el proceso de planificación, una vez que se ha determinado la representación del espacio, es necesario buscar la mejor trayectoria a través de dicha representación.

Para ello existen diversos algoritmos de búsqueda en grafos que pueden ser usados en conjunto con las diferentes formas de representar el espacio (C).

Una descripción completa acerca de los diferentes algoritmos de búsquedas se puede encontrar en (Burker and Kendall, 2005). La utilización de muchos de ellos en la robótica móvil, específicamente en la planificación de trayectorias, se implementan en trabajos como (Liu and Ramakrishnan, 2001), (Stentz, 1994), (Nilsson, 1980), (Murphy, 2000), (Mataric, 1992). Uno de los más populares es el llamado A estrella (*A**) y Bellman-Ford para entornos estructurados siendo estos analizados en la investigación.

1.3 Generación de trayectorias

La etapa de generación de trayectorias es la encargada de obtener un camino que sea factible para el robot, teniendo en cuenta sus características cinemáticas. Este proceso comienza, generalmente, una vez obtenida la secuencia de puntos o localizaciones por la que debe transitar el robot.

Una de las formas más populares de obtener una trayectoria suave es interpolando la secuencia de puntos. Para ello se pueden usar métodos de interpolación tales como Lagrange o Splines. No obstante, existen una series de curvas polinomiales que se utilizan con este fin. Entre ellas se encuentran las llamadas espirales cúbicas (Kanayamaand Hartman, 1989), las ß -spline (Gómez-Bravo *et al.*, 2007), las CC-Steer (Fraichard and Scheuer, 2004) y las clotoides (McCrae and Singh, 2008; Kanayama and Hart-man, 1989; Meidenbauer, 2007) entre otras. Muchas de estas curvas utilizan la creación de una lista de segmentos de rectas y arcos de circunferencia entre las que se encuentra las ß-spline, escogidas para este trabajo, pues se ajustan con gran facilidad a las restricciones cinemáticas de los robots móviles.

1.4 Agente

El proceso de navegación, como se dijo anteriormente, se puede dividir en tres partes fundamentales: planificación, generación y control. Como se puede observar las diferentes acciones que se suelen realizar en cada una de estas etapas se pueden estructurar de manera natural mediante componentes de *software* independientes. Así, si el diseño se basa en la modularidad y ejecución independiente de sus componentes,

y se busca cierto grado de autonomía en el desempeño de los robots, entonces el diseño e implementación de esta solución puede basarse en agentes.

Con el término de agente, se han desarrollado una gran variedad de programas. Se puede decir que ha existido un uso excesivo del mismo sin lograrse un consenso de su significado. Entre las definiciones de agente que se encuentran frecuentemente en la literatura están las siguientes:

Weiss (Weiss, 1999) define a un agente como una entidad computacional, como un programa de software o un robot, que puede ser visto como un ente dotado de perceptores y actuadores en un ambiente y que es autónomo en su comportamiento.

En el caso de Stuart Russel (Russel S, 1995), lo refiere como una entidad que percibe su entorno a través de sensores y actúa sobre ese entorno a través de efectores o permanece como una entidad que presenta características de la inteligencia humana, trabajando de una forma autónoma y continua en su ambiente. Un agente es racional cuando realiza la mejor acción posible considerando sus objetivos y metas.

Según Walter Brenner, un agente inteligente es un programa de software que puede realizar tareas específicas para un usuario y posee un grado de inteligencia suficiente para ejecutar parte de sus tareas de forma autónoma y para interactuar con su entorno de forma útil (Brenner, 1998).

Cada autor tiene un punto de vista de acuerdo a su línea investigativa, por lo que es difícil escoger una definición. Este trabajo se ajusta a la definición planteada por Weiss, en mayor relación con el objetivo planteado y la función desempeñada por los agentes en la investigación.

Los agentes poseen características como:

- **Autonomía:** Es la capacidad por la cual los agentes operan por sí mismos sin la intervención del hombre u otros agentes.
- **Habilidad social:** Facilidad de interactuar con otros agentes (o humanos) por medio de un lenguaje de comunicación preestablecido.
- **Reactividad:** Los agentes perciben el ambiente y responden de manera oportuna.

- **Proactividad:** No sólo actúan en respuesta a un evento, deben estar habilitados para tomar la iniciativa, con el fin de conseguir la meta propuesta.

Además pueden presentar características del comportamiento humano como la racionalidad y la colaboración. Se han propuesto varios esquemas y taxonomías según sus características por diferentes investigadores: (Wooldridge, 2007), (Bradshaw, 1997), (Moulin, 1996), (Nwana, 1996), existiendo otros tipos de agentes en dependencia de las posibles combinaciones de estos atributos.

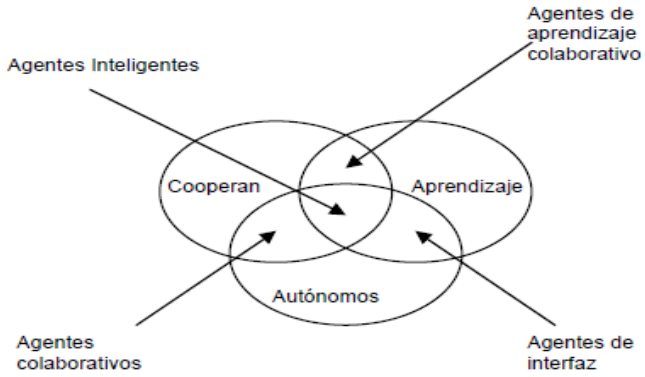


Figura 1.4 Tipología de agentes de acuerdo a combinaciones de sus propiedades.

- **Inteligentes:** Son agentes que pueden llevar a cabo conjuntos de operaciones en representación del usuario o de otro programa, siempre con un alto grado de independencia, cuyas acciones y resultados no difieren de lo que el usuario obtendría si las realizara él mismo.
- **Cooperativos:** Poseen la habilidad de socializar, que les facilita interactuar con otros agentes, incluso con seres humanos a través de algún lenguaje de comunicación. Dando la posibilidad de dar soluciones a determinados problemas por varios agentes e incluso de forma coordinada.
- **Colaborativos:** Estos agentes acentúan las características de autonomía y de cooperación con otros agentes con el objetivo de realizar tareas efectivas para sus propietarios. Los atributos más evolucionados en estos agentes son la autonomía, sociabilidad y auto-actividad.
- **Interfaz:** Los agentes de interfaz son asistentes personales que colaboran con el usuario en su mismo entorno. El agente observa e interpreta las acciones

realizadas por el usuario en la interfaz, aprende de ello, y sugiere mejores caminos para realizar la tarea.

Esta tipología que tiene un mínimo de atributos y es ampliamente aceptada. La potencialidad de los agentes no se expresa como un ente individual, sino cuando se establece algún tipo de relación entre más de uno de estos elementos para dar solución a determinados problemas de manera conjunta.

Con el objetivo de estandarizar las especificaciones de agente se crea la Fundación de Agentes Físicos Inteligentes (FIPA) (Huget, 2004), siendo un consorcio industrial fundado en 1996 por varias decenas de compañías de telecomunicaciones e informática para promover el desarrollo de tecnologías de agentes mediante la producción de especificaciones acordadas internacionalmente como son: la gestión de los agentes, el lenguaje de comunicación, interacción agente-humano, movilidad, seguridad, representación de los mensajes, etc.

Para llevar a cabo la comunicación, los agentes deben tener correspondencia en varios aspectos, definidos en las especificaciones FIPA:

- La terminología que se utilizará en el contenido de los mensajes.
- El lenguaje de codificación (FIPA-SL).
- La estructura de los mensajes (FIPA-ACL).
- Los protocolos de intercambio de mensajes (FIPA-protocols).

1.4.1 Arquitectura abstracta de FIPA

Especifica el modelo de referencia lógico para la creación, registro, localización, comunicación, migración y retirada de agentes. En este modelo se define la plataforma de agentes, la cual cuenta con los siguientes elementos (Huget, 2004):

- *Directorio Facilitador* (DF): es un agente que proporciona un servicio de Páginas Amarillas dentro del sistema, dando a conocer qué servicios pueden ofrecer los diferentes agentes del AMS.

- *Sistema de administración de agentes* (AMS): es un agente que controla el acceso y el uso de la plataforma de agentes, dando a conocer las direcciones de los agentes de la plataforma. Proporciona un servicio de Páginas Blancas.

- *Servicio de transporte de mensajes* (MTS): sirve para poder comunicar agentes que están en diferentes plataformas.

- *Canal de comunicación de agentes* (ACC): Soporta la interoperabilidad entre agentes de la misma plataforma, así como entre agentes de distintas plataformas.

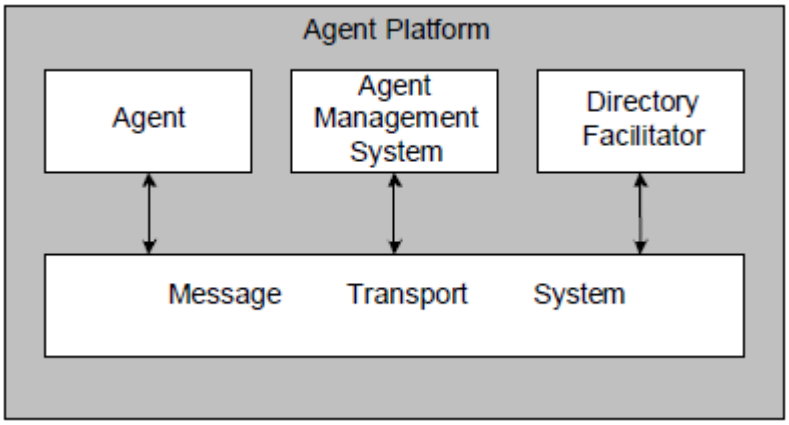


Figura 1.5 Arquitectura FIPA.

1.4.2 Lenguaje de comunicación de agentes (FIPA ACL)

Está basado en actos del habla, donde la semántica se basa en aptitudes mentales (creencias, intenciones, peticiones, etc.). Los mensajes son acciones comunicativas con una semántica formal definida con lógica modal (lenguaje SL). También se definirán unos protocolos de interacción de alto nivel, llamados conversaciones (Huget, 2004).

1.4.2.1 Estructura de un mensaje

Un mensaje se compone de una instrucción para indicar el tipo de mensaje (pregunta, petición, etc.) y de diferentes parámetros.

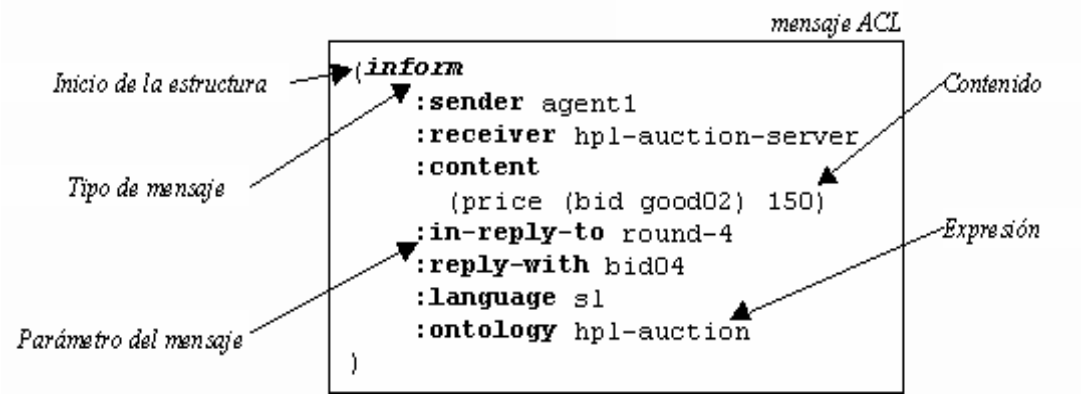


Figura 1.6 Estructura de un mensaje FIPA-ACL.

1.4.2.2 Tipos de mensaje

En la tabla 1.1 se muestran los diferentes tipos de mensaje definidos por la FIPA.

Tabla 1.1 Tipos de mensaje.

Tipo de mensaje	Significado
REQUEST	Pedir a otro agente que realice una acción.
REQUEST-WHEN	La acción se hará cuando se cumpla una condición.
REQUEST-WHENEVER	La acción se hará siempre que se cumpla una condición.
AGREE	Aceptar petición.
REFUSE	No aceptar petición.
CANCEL	Cancelar petición.
INFORM	Dar información al receptor.
QUERY-IF	Preguntar al receptor alguna cosa.
SUBSCRIBE	Apuntarse a un servicio del receptor, de modo que cuando ocurra algo que interese al emisor el receptor se lo notificará.
CFP	*Call for proposals* – pedir propuestas para realizar una acción.
PROPOSE	Hacer una propuesta.
REJECT-PROPOSAL	Rechazar una propuesta.
ACCEPT-PROPOSAL	Aceptar la propuesta.
FAILURE	Explica por qué ha fallado la acción.
NOT-UNDERSTOOD	Explica que no se ha entendido el contenido del mensaje.

1.4.2.3 Parámetros del mensaje

La Tabla 1.2 muestra los diferentes parámetros, introducidos en la estructura del mensaje, con una breve descripción de los mismos.

Tabla 1.2 Parámetros de los mensajes

Parámetro	Significado
:sender	Denota la identidad del agente que envía el mensaje.
:receiver	Indica la identidad del que recibe el mensaje. Aquí pueden haber uno o varios nombres de agente (dependiendo de a quien se quiera hacer llegar el mensaje, puede ser individual o colectivo).
:content	Contenido del mensaje, lo que se le quiere decir a los agentes receptores.
:reply-with	Introduce una expresión que será usada por el agente que responda para identificar este mensaje. Puede ser usado, por ejemplo, para seguir conversaciones entre agentes. Si un agente envía un mensaje que contiene :reply-with query1, el receptor responderá con otro que contenga :in-reply-to query1
:in-reply-to	Hace referencia a que este mensaje es una respuesta a otro anterior.
:language	Nombre del lenguaje en que está escrito el contenido del mensaje (ej. Prolog, Lisp, SL)
:ontology	Define el significado del vocabulario usado en el contenido de los mensajes.
:reply-by	Indica el tiempo en que el mensaje ha de ser respondido.
:protocol	Indica el protocolo usado por los mensajes.
:conversation-id	Expresa un identificador de conversación. Esto es especialmente útil cuando un agente mantiene varias conversaciones a la vez.

1.4.3 Protocolos

Las conversaciones entre agentes siguen en algunas ocasiones unos patrones determinados (secuencias típicas de mensajes), que se repiten en muchos casos. Aprovechando estos patrones y su repetición, FIPA ha definido protocolos, que constituyen patrones que se usan para llevar por unos cauces concretos una conversación. Son como conversaciones guiadas, en que cada agente sabe qué mensaje enviar y cuáles puede recibir (Brenner, 1998).

Para cada conversación se distinguen dos roles, el emisor y el receptor. Al usar uno de estos protocolos, los mensajes enviados, tendrán en el campo*: protocol* el tipo de

protocolo al que pertenece, para que los agentes implicados sepan que se está siguiendo un protocolo.

En FIPA hay especificados varios protocolos, entre ellos destacan los siguientes:

- *FIPA-request:* Este protocolo permite a un agente pedir a otro agente que lleve a cabo una acción.
- *FIPA-request protocol:* Se usa cuando un agente pide a otro que realice una acción. El destinatario puede aceptar o rechazar la petición, y en caso de aceptarla, deberá realizar la acción, y notificar la finalización.
- *FIPA-query protocol:* En este protocolo un agente pide algún tipo de información a otro agente, este último puede responder con la propia información, con un fallo, con un *not-understood* o con el rechazo de la petición, teniendo que alegar el motivo.
- *FIPA-ContractNet protocol:* Se utiliza cuando un agente quiere que uno o más agentes realicen una acción, realizando una especie de encuesta, pidiendo propuestas a los distintos agentes. En la petición se especifica la acción a realizar y algunas precondiciones a la hora de hacer las propuestas. Los agentes consultados envían sus propuestas al agente iniciador, también indicando las condiciones de la propuesta. Este las estudia y elige las que le convienen, rechazando el resto. El protocolo requiere que el agente iniciador sepa cuando ha recibido todas las propuestas. Como esto podría llevar mucho tiempo en caso de que un agente tarde más de la cuenta, se da un tiempo límite para responder, y las propuestas que lo hagan más tarde, serán rechazadas.

Otros protocolos a destacar son: FIPA-Brokering (interacción entre agentes a través de un intermediario), FIPA-Auction-Dutch (subasta holandesa), FIPA-Auction-English (subasta inglesa), FIPA-Iterated-Contract-Net (con varias iteraciones para mejorar las propuestas), FIPA-Propose (para hacer una propuesta de realización de una acción), FIPA-Recruiting-Interaction (para pedir a otros agentes ser intermediarios), FIPARequest-When (para pedir a otro a gente la realización de una acción si se cumplen una serie de condiciones), FIPA-Subscribe (para pedir información a cerca de cambios en el estado de un objeto).

1.4.4 Lenguaje FIPA-SL

FIPA-SL (Semantic Language) es un lenguaje general definido por la FIPA para facilitar la comunicación entre agentes. SL es una lógica multimodal con operadores modales

para creencias (B), deseos (D), creencias falsas (U) e intenciones u objetivos persistentes (PG). La semántica de cada acto comunicativo en FIPA ACL se define como un conjunto de fórmulas SL que describen: las precondiciones de factibilidad (FP, condiciones necesarias para el emisor), y el efecto racional (RE, lo que un agente espera que ocurra como resultado de una acción).

Ejemplo: El agente *a* informa al agente *b* del contenido *X*

<a, inform(b, X)>

FP: B(a, X) /\ ¬B(a, B(a, f(b,X)) \/ U(a, f(b, X)))

RE: B(b, X)

El agente *a* cree *X* (B(a, X)); pero no cree todavía que el receptor tenga algún conocimiento sobre la verdad de la proposición *X* (¬B(a, B(a, f(b,X)) \/ U(a, f(b, X)))).

Así que tiene la intención de que el agente receptor debería también llegar a creer que la proposición *X* es cierta (efecto racional B(b, X)).

1.4.5 Ciclo de vida de un agente

Los estados por los que puede pasar un agente en su ciclo de vida son activo, iniciado, en espera, suspendido, en tránsito y desconocido.

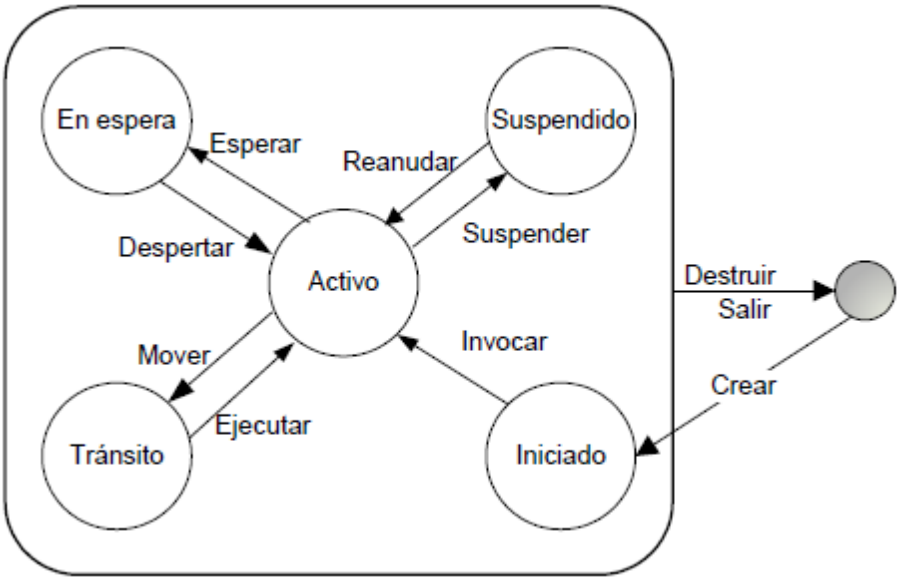


Figura 1.7 Modelo del ciclo de vida de un agente.

- Iniciado: Cuando el agente es creado y pasa al estado activo cuando es invocado.

- Activo: Cuando el agente está en ejecución, pasa al estado suspendido con la acción suspender o pasa al estado de espera con la acción esperar.
- Suspendido: Cuando el agente ha sido suspendido y no está activo, pasa al estado activo con la acción reanudar.
- En espera: El agente está en ejecución pero no realiza su tarea hasta que ocurra un evento esperado, pasa al estado activo con la acción despertar.
- En tránsito: Solo los agentes móviles entran a este estado. Las acciones del estado son mover o transportar y ejecutar. Mover realiza el tránsito de una AP origen a una AP destino, y ejecutar pone en ejecución al agente una vez realizado el tránsito pasando al estado Activo.
- Por último se tienen las acciones destruir o salir que ejecuta la terminación del agente en la AP.

1.5 Marco de trabajo de agentes

En la actualidad existe un amplio abanico de estas herramientas, ya sean comerciales o de código libre. La inmensa mayoría de estas herramientas han sido desarrolladas en el lenguaje de programación Java, aprovechándose de su cualidad de lenguaje orientado a objetos multiplataforma y multihilo. De este modo, los agentes pueden residir en máquinas diferentes, incluso con diferentes sistemas operativos, ejecutándose con varios flujos de control. De las herramientas existentes, se analizan a continuación aquellas que implementan el estándar FIPA basados en los siguientes criterios:

- Facilidad de implementación de nuevos agentes.
- Organización de forma lógica de las clases existentes en la distribución.
- Distribución en régimen de código libre o abierto.
- Robustez en su funcionamiento.
- Empleo de diferentes protocolos de comunicación (RMI, CORBA).
- Herramientas de depuración.

1.5.1 FIPA-OS

Es un marco de trabajo de agentes desarrollado por Nortel Networks[1], que trata de suministrar al usuario una plataforma cuya arquitectura destaque por su facilidad de extensión y por su modularidad, implementada en Java, es de código libre. Cumple con numerosas especificaciones FIPA, como las relativas al manejo de los agentes, la transmisión y recepción de mensajes ACL, así como a los protocolos de conversación (Poslad, 2013), (*FIPA-OS, 2013*).

Figura 1.8 Logotipo FIPA-OS.

Las comunicaciones entre los agentes de la misma plataforma se realizan mediante Java RMI y los mensajes entre plataformas se realizan mediante CORBA, RMI y TCP (Laukkanen, 2000).

FIPA-OS incluye una plantilla denominada *GenericAgent.java* que sirve de base para la implementación de agentes más complicados. Esta plantilla incluye un gestor de las tareas *(Task Manager)* a desarrollar por el agente (entendiendo por tarea como la unidad primitiva de trabajo) y un gestor de conversaciones (*Conversation Manager*) encargado de observar el cumplimiento de los protocolos FIPA relativos a las mismas. Cada tarea es ejecutada en un *thread* diferente. De este modo se aprovecha la capacidad *multithread* de Java para ejecutar diversos flujos de control de manera simultánea.

En el paquete se incluyen clases relativas a los agentes de gestión del sistema definidos por FIPA: el Agent Management System (AMS), el Directory Facilitator (DF), el Agent Communication Channel (ACC) y el Internal Platform Message Transport (IPMT).

Esta herramienta se completa con diversas utilidades como un monitor de las tareas pendientes de ejecución, un visualizador de los *threads* activos en el sistema, un generador de tareas (generador automático de código Java de todas las subclases necesarias para la ejecución de una tarea) y una interfaz gráfica que permite al usuario enviar directamente mensajes ACL de modo manual a otros agentes del sistema.

[1] http://nortelnetworks.com

1.5.2 Zeus

Ha sido desarrollado por British Telecom Labs. para proporcionar una herramienta de construcción de agentes colaborativos genéricos a nivel industrial. Para ello adopta una configuración de agentes por capas, de tal modo que los agentes son capaces de seguir un protocolo de comunicación con otros agentes de la comunidad. Los agentes implementados en Zeus tienen la habilidad de planear la secuencia de pasos necesarios para llegar a un objetivo, monitorizar la ejecución de los planes y retroceder cuando sea necesario. Son orientados a metas mediante la representación de máquina de estados (Nwana, 1999), (Hyacinth, 1999), (Darryl, 2003).

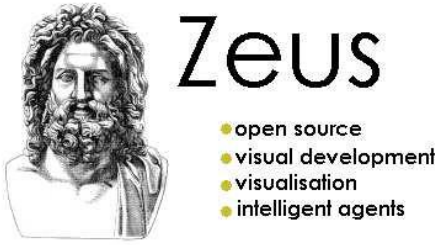


Figura 1.9 Logotipo de Zeus.

Zeus fue en principio desarrollado para cumplir con el estándar de comunicación KQML. Sin embargo, ha sido reescrito para adaptarlo al FIPA-ACL, soportando actualmente la especificación FIPA para la gestión de agentes.

Las clases de la distribución Zeus se pueden dividir en tres grupos funcionales:

- Una librería de componentes de agentes. Es un paquete de clases Java que constituye los “bloques de construcción” de los agentes. Comprende aspectos tales como la comunicación, la coordinación o la ontología.
- Una herramienta de visualización de agentes con 5 módulos (Hyacinth, 1999):
  - Un visualizador de sociedades que muestra todos los agentes y sus relaciones.
  - Un visualizador de la distribución/descomposición de las tareas activas y los estados de ejecución de las sub-tareas.
  - Un visualizador de los estados internos de los agentes.

- Una herramienta de control que permite visualizar y modificar los estados internos de los agentes, por ejemplo redefiniendo el comportamiento del agente.
- Un visualizador de las estadísticas de la sociedad de agentes.

- Una herramienta de construcción de agentes a alto nivel. Proporciona una serie de editores que permiten crear agentes, especificando de forma manual sus atributos o tareas.

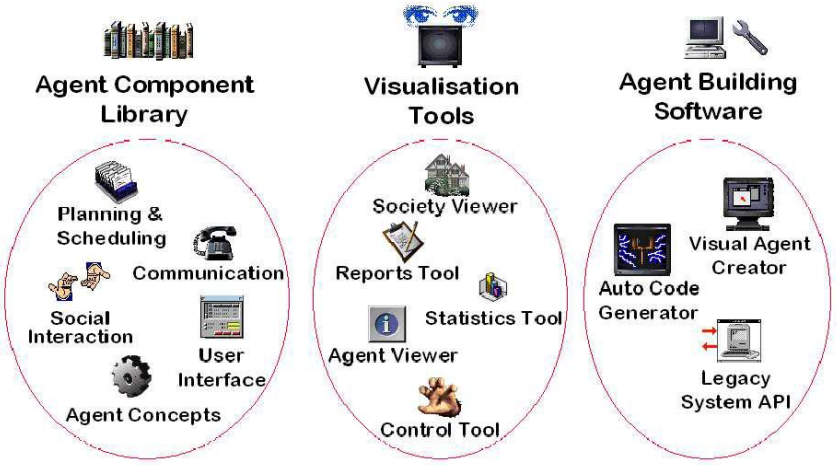


Figura 1.10 Grupos funcionales de las clases de la distribución Zeus.

Cada agente Zeus consta de tres capas:

- Capa de definición. Representa las habilidades de aprendizaje y razonamiento del agente, sus objetivos, creencias, recursos.
- Capa de organización. Describe las relaciones del agente con otros agentes.
- Capa de coordinación. Describe las técnicas de negociación y coordinación que posee el agente.

La plataforma proporciona un motor de coordinación que ejecuta máquinas de estados, denominadas grafos. Un grafo define los nodos de procesamiento de una parte del comportamiento del agente y los arcos que unen los diferentes nodos.

La parte principal del código del comportamiento del agente se divide en dos nodos llamados *exec* y *continue_exec*. El código de condición previa se define en el método *exec* del arco que determina qué nodo se va a ejecutar a continuación. Cada nodo

cuenta con un método *reset* que permite devolver el estado al inmediatamente anterior de entrar en el nodo.

El motor de coordinación gestiona la ejecución de los nodos. Los nodos son almacenados en una cola mientras se espera a que se procese todo el código requerido. Cuando este procesamiento finaliza, los nodos son colocados en otra cola para su ejecución.

Los mensajes ACL son distribuidos por una clase gestora de mensajes. Esta clase permite registrar las reglas para determinar qué objeto debe procesar un nuevo mensaje. Una vez que se dispara una determinada regla, se ejecuta el método *registered* del objeto correspondiente. Tras la ejecución de este método, el gestor de mensajes chequea la siguiente regla registrada para ver si también es satisfecha por el mensaje recibido. Este proceso continúa secuencialmente hasta que se da una oportunidad a todas las reglas activas para procesar el mensaje (Hyacinth, 1999), (Darryl, 2003).

1.5.3 JADE

Es una herramienta de código libre implementada en lenguaje Java que ha sido desarrollada en el CSELT (Centro Studi e Laboratori Telecomunicazioni) con el objetivo de facilitar la implementación de sistemas multiagentes, cuyos agentes interactúan según las normas FIPA. JADE proporciona la misma infraestructura de agentes que FIPA-OS, aunque incluye el concepto de *AgentContainer* un paradigma propio para la asociación de JVMs, agentes y hosts. Así, una plataforma de agentes implementada en JADE está formada por varios *AgentContainer* (Bellifemine, 2012), (Bellifemine, 2013).

Figura 1.11 Logotipo de JADE.

Cada *AgentContainer* es un objeto servidor RMI que gestiona de manera local a un conjunto de agentes, controlando el ciclo de vida de estos agentes mediante la creación, suspensión o eliminación de los mismos. Además, maneja los aspectos de comunicación relativos a los mensajes ACL entrantes, distribuyéndolos de acuerdo con el campo destino del mensaje (*:receiver*) y depositándolos en las colas de mensajes de

cada agente. Para los mensajes salientes, en cambio, el *AgentContainer* mantiene la suficiente información para determinar la localización del agente receptor y escoge la forma de transporte más adecuada para mandar el mismo. De este modo, las comunicaciones se agilizan, ya que entre agentes del mismo *AgentContainer* la comunicación se realiza sin ningún tipo de invocación remota. Las invocaciones RMI quedan reservadas para mensajes entre los agentes de distintos *AgentContainer* de la misma plataforma y las comunicaciones a través de CORBA para los mensajes entre agentes de diferentes plataformas.

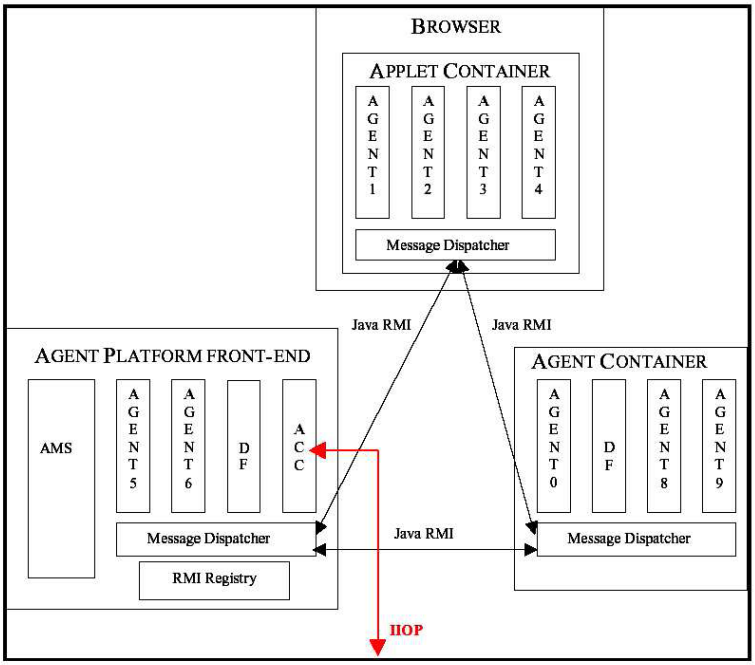


Figura 1.12 Esquema de arquitectura JADE.

Cada AP cuenta con una interfaz gráfica que permite al usuario un mayor control de la plataforma y de los agentes, pudiendo crearlos, suspenderlos o eliminarlos de forma manual. Lo interesante de esta interfaz es que ha sido implementada como si se tratase de un agente más, llamado RMA (*Remote Monitoring Agent*), de modo que la interacción de la interfaz con el resto de los agentes de la plataforma se realiza mediante mensajes ACL.

Como unidades de depuración JADE destacan el *Agente Sniffer*, que intercepta los mensajes ACL que llegan a un agente y los representa gráficamente en una notación similar a los diagramas de secuencias UML y el *Agente Introspector* que informa sobre

el ciclo vital del agente seleccionado (activo, dormido), sus mensajes ACL intercambiados y sus comportamientos implementados que se encuentran ejecutándose en la actualidad.

JADE emplea el concepto de comportamiento (*Behaviour*) para modelar las tareas que un agente es capaz de llevar a cabo. Los agentes instancian estos comportamientos de acuerdo con sus necesidades y sus capacidades. Desde el punto de vista de programación concurrente, en el caso de JADE se ha optado por la filosofía de emplear un único *thread* por agente, en vez de la opción de un *thread* por tarea como en el caso de FIPA-OS. De este modo se inicializan un menor número de threads en una AP.

En JADE cuando llega un nuevo mensaje, un gestor (implementado por la clase *Agent* y oculto para el programador) implementa una política *round-robin* entre todos los comportamientos disponibles en el agente hasta encontrar un comportamiento dispuesto a procesar el mensaje recibido. Esto se determina mediante una combinación lógica de análisis de las cadenas de caracteres (*string matching*) de los mensajes. En caso de encontrar un comportamiento interesado en procesar el mensaje, se ejecuta su método *action*. Este comportamiento que acepta el mensaje controla el flujo del agente hasta que él mismo lo libera. Por el contrario, en caso de no poder procesar el mensaje recibido, el comportamiento analizado es mandado a dormir.

1.5.4 Marco de trabajo seleccionado

Todas estas herramientas son multiplataforma, al encontrarse implementadas en el lenguaje de programación Java. En cuanto a la capacidad de depuración del flujo de mensajes, todas las herramientas cuentan con utilidades equivalentes. Brindan facilidad de implementación de nuevos agentes, se distribuyen en régimen de código libre y son robustas en su funcionamiento.

Sin embargo, se descarta a Zeus porque su programación está orientada a metas y estructurada a través de nodos y arcos, y se lleva a cabo normalmente mediante interfaces gráficas, las cuales amarran al usuario a una estructura de relleno de campos que en ocasiones no se adapta a las necesidades del problema a tratar.

FIPA-OS y JADE cuentan con utilidades equivalentes, la principal diferencia es que la primera define un hilo de ejecución por cada tarea o comportamiento del agente y la segunda solo define un hilo por agente. Queda en dependencia del uso de los agentes,

las tareas que realizan, la elección de una de estas herramientas. Para esta solución se optó por la herramienta JADE, pues los agentes estarán especializados en la ejecución de determinados algoritmos de navegación y no es necesario ningún nivel de concurrencia entre sus tareas, además disminuye el coste en cuanto a recursos dentro del sistema.

1.7 Metodología de desarrollo

Existen diversos trabajos enfocados a facilitar el trabajo con agentes tales como: BDI (Kinny et al. 1997), *Vowel Engineering* (Ricordel, 2001), MAS-CommonKADS (Iglesias, 1998), (Iglesias et al 1998b), GAIA (Wooldridge et al, 2000). Estas metodologías parten de un modelo informal en la mayoría de casos, de cómo debe ser el desarrollo de aplicaciones basada en agentes y ofrecen guías para su construcción. En las primeras metodologías, las guías consistían en una lista breve de pasos a seguir. Las más modernas, aunque han progresado en la integración con la ingeniería del software clásica, aún no muestran la madurez que se puede encontrar en metodologías convencionales como el *Unified Process*. Este trabajo se rige por la metodología proporcionada por JADE, herramienta seleccionada para el desarrollo del trabajo.

La metodología propuesta para la plataforma JADE según (Nikraz et al, 2005) y sus colegas no trata de extender técnicas orientadas a objetos, en lugar de eso, enfoca la atención en agentes y las abstracciones provistas por el paradigma del agente. La metodología propuesta trata de formalizar el análisis y fases del diseño del ciclo de vida de desarrollo del software basado en agentes.

Fase de análisis. La fase de análisis tiene como meta aclarar el problema con mínimas preocupaciones acerca de la solución, esta fase pasa por los siguientes pasos:

1. Captura de los requerimientos funcionales del sistema mediante los casos de uso.

2. Identificación inicial de tipos de agentes. Se realiza el diagrama de agentes.

3. Identificación de responsabilidades de cada tipo de agente. Observando los tipos de agentes producidos en el diagrama de agentes, se produce una tabla inicial de responsabilidades, denominada tabla de responsabilidad.

4. Identificación de relaciones de los agentes.

5. Refinamiento de los agentes, se determina claramente los tipos de agentes.

6. Información del despliegue del agente. Se produce el diagrama de implementación del agente para indicar donde los agentes serán desplegados.

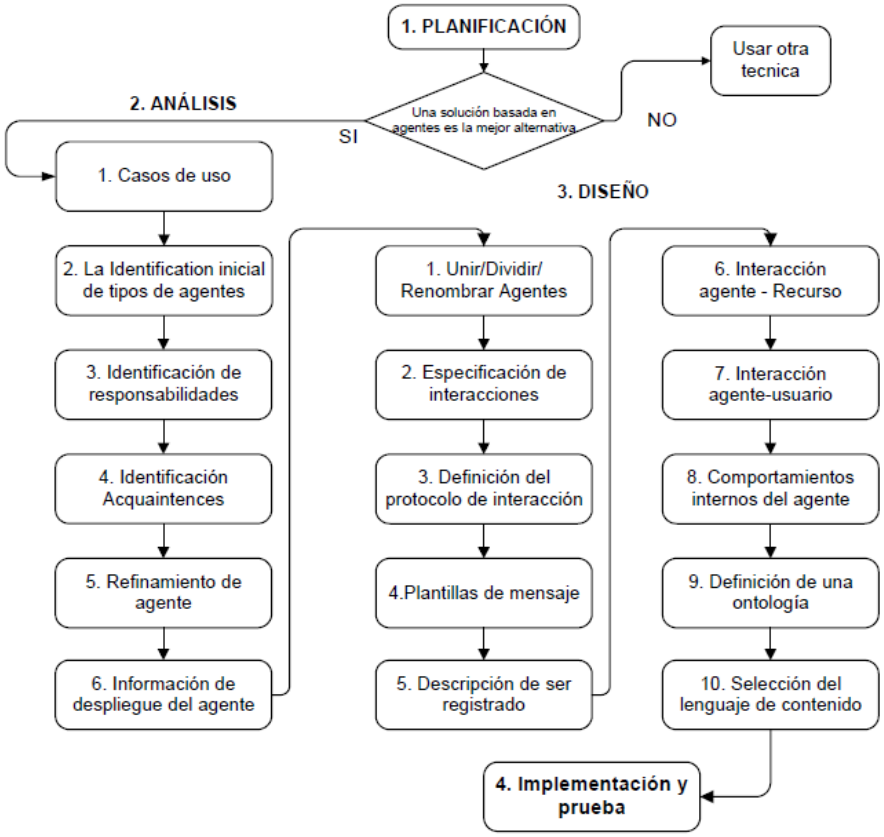


Figura 1.13 Visión general de la metodología.

Fase de diseño. Esta fase se enfoca en la plataforma JADE, representando las interacciones del agente con el usuario, recurso u otro agente y los comportamientos internos de los agentes. De igual manera que la fase de análisis se realiza una secuencia de pasos lógicos:

1. Separar, anexar y renombrar agentes. Considerando el desempeño del sistema y la complejidad en relación al diagrama de implementación del agente producido en el análisis, se determina si los agentes deberían encontrarse separados, deberían anexarse o dejar como están.

2. La especificación de interacción. Todas las responsabilidades en la tabla de responsabilidad se relacionan considerando una relación de conocimiento con otro agente, y la tabla de interacción se produce para cada tipo de agente.

3. La definición de protocolo de interacción. En el caso que un protocolo existente no es destinado para una interacción, un protocolo de interacción se define usando un formalismo adecuado.

4. Plantillas de mensajes. La tabla de interacción se actualiza para especificar adecuados objetos como plantillas de mensajes en los comportamientos para recibir mensajes entrantes.

5. Descripción para ser registrado o buscado por el servicio que realiza. Se registran los servicios que realizan los agentes, en el catálogo de páginas amarillas mantenido por el agente facilitador del directorio de JADE. Un diagrama de clase se usa como una representación.

6. Interacciones de los recursos y el agente. Basado en el diagrama del agente producido en el análisis, se identifican los recursos pasivos y activos en el sistema, y se determina cómo interactúan los agentes con estos recursos.

7. Interacciones del usuario y el agente. Basado en el diagrama del agente producido en el análisis, se identifica y detalla las interacciones del usuario y el agente.

8. Comportamientos internos del agente. Basado en la tabla de responsabilidad producida en el análisis, las responsabilidades del agente son un mapa para los comportamientos del agente. Las responsabilidades (incluyendo interacciones) requieren que los diferentes tipos de comportamientos del agente se especifiquen.

9. Definición de una ontología. Se debe especificar una ontología apropiada para el dominio.

10. Selección del lenguaje de contenido. Se debe elegir un lenguaje adecuado de contenido que utiliza el agente para comunicarse.

11. Repetir los pasos del 1-10. Desplazándose de atrás hacia adelante entre análisis y diseño cada vez que sea necesario.

1.8 Herramientas para el desarrollo

Visual Paradigm 8.0 - Es una herramienta multiplataforma, incluye el estándar UML como lenguaje de modelado y soporta completamente el ciclo de desarrollo del

software. Posibilitó el modelado de los diagramas de caso de uso, de clases y de iteración.

Netbeans7.2.1 - NetBeans es un IDE (por las siglas en inglés de Entorno de Desarrollo Integrado) libre y de código abierto, que con variados módulos brinda las herramientas necesarias para el desarrollo de aplicaciones profesionales de escritorio, web y aplicaciones móviles con la plataforma Java (Oracle Corporation and its affiliates 2011).

1.9 Conclusiones del capítulo

- Constituyen la planificación, generación y control, las etapas fundamentales de la navegación con robots móviles. En la planificación se determinó el uso de la descomposición en celdas para la representación del entorno al ser uno de los métodos más utilizados. Para la obtención de la ruta dado el entorno representado mediante la implementación de los algoritmos Bellman Ford y A* y SACO. En la generación para suavizar la ruta determinada se utilizaron las curvas ß-spline pues tienen en cuenta las restricciones cinemáticas de los robots móviles, además son continuas en posición y orientación a lo largo de toda la ruta.
- Cada una de estas etapas de la navegación se puede estructurar de manera natural mediante componentes software independientes siendo el uso de agentes adecuado para el diseño e implementación de cada una, ya que garantiza la modularidad y la independencia de ejecución de sus componentes, garantizando el grado de autonomía que se busca obtener en el desempeño de los robots.
- Como marco de desarrollo para esta solución se optó por la herramienta JADE, pues los agentes están especializados en la ejecución de determinados algoritmos de navegación y no es necesario ningún nivel de concurrencia entre sus tareas disminuyendo el coste en cuanto a recursos dentro del sistema. Además cuenta con un amplio respaldo en documentación y es ampliamente utilizada.

CAPÍTULO 2. ALGORITMOS

Como se planteó con anterioridad, la navegación se puede dividir en tres etapas: planificación, generación y control. En este capítulo se describen los algoritmos utilizados para la descomposición del entorno, la búsqueda de rutas y la generación.

2.1 Planificación

En la planificación, al robot le es necesario conocer el área de trabajo donde se desempeña, en este trabajo se brinda una interfaz que permite definirla, en la cual se colocan los puntos entre los que se tiene que desplazar el robot y los obstáculos que tiene que sortear (figura 2.1). De esta representación del espacio se obtiene una imagen en formato (png) que es la entrada para el planificador, a partir de la cual se obtiene un grafo que representa el espacio libre del área de trabajo mediante la descomposición en *celdas cuadradas* (Torres, 2010). Una vez obtenido el grafo, solo resta obtener una ruta que una el punto inicial con el punto final que el robot debe de alcanzar. Para el cálculo de esta ruta se implementan los algoritmos de búsqueda en grafos: A*, Bellman-Ford y SACO.

2.1.1 Descomposición del entorno en celdas cuadradas

A partir de los datos de entrada descritos anteriormente se obtiene una representación computacional del entorno donde opera el robot, mediante la descomposición en celdas cuadradas. El tamaño de la celda es importante, en este trabajo se toma el diámetro de la circunferencia (D) que engloba el área que ocupa el robot (Torres, 2010). El número de celdas en que se descompone el entorno se obtiene dividiendo las dimensiones de la imagen que lo representa entre (D), obteniendo una matriz de celdas. Para determinar si una celda está ocupada o libre se itera píxel a píxel analizando el color que contiene: Blanco–libre, Negro-ocupado, y definiendo que una celda está libre si todos sus píxeles son libres, la unión de todas estas celdas es el espacio libre (área que no está ocupada por ningún obstáculo).

Posteriormente la validación de que las celdas diagonales adyacentes a cada una de las celdas del espacio libre, tomando el punto medio de cada una para formar una recta que las una y cumpla que las rectas paralelas que equidistan D/2 y D/4 a ambos lados no pasan por encima de ningún obstáculo garantiza que cualquier ruta (sucesión de

celdas adyacentes que una la posición inicial con la final) encontrada esté libre de colisiones (Torres, 2010). El área del espacio libre queda mostrada en la (figura 2.1), y las condiciones para garantizar que las celdas diagonales adyacentes sean transitables en la (figura 2.2).

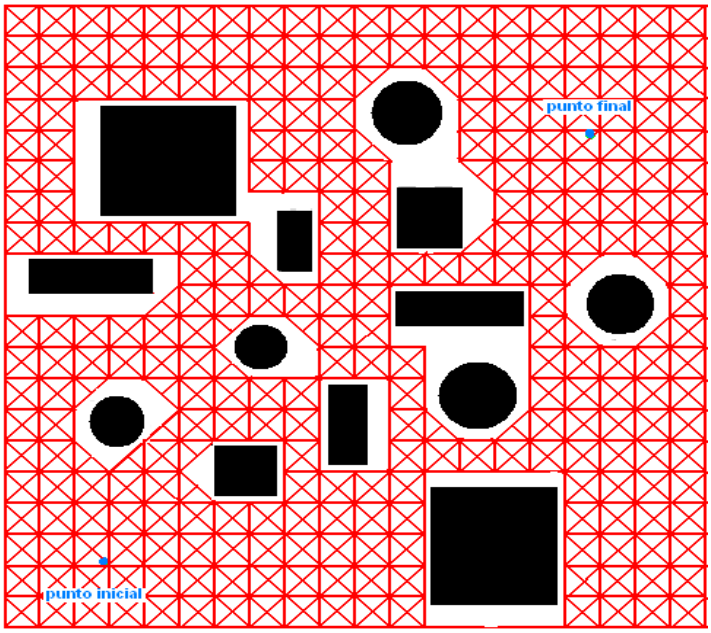


Figura 2.1 Representación del espacio libre

Cada celda será un nodo de un grafo de conectividad no dirigido, existiendo un arco entre dos celdas sí y solo sí son adyacentes. Así, una ruta que una la posición inicial pi y la final pf, consiste en encontrar una sucesión de celdas que no presente discontinuidades y no exista solapamiento, tal que la primera de ellas contenga a pi y la última a pf. Los algoritmos de búsqueda en grafos, que se describen a continuación, determinan posibles rutas.

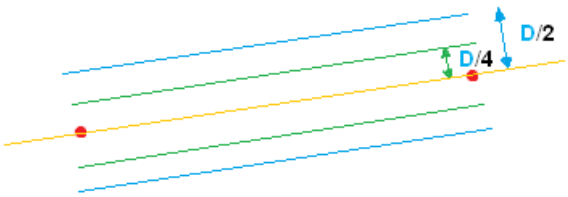


Figura 2.2 Comprobación de las celdas diagonales

2.1.2 Algoritmo de búsqueda A*

El algoritmo llamado A estrella (A*) (N. J. Nilsson, 1980), utiliza información heurística para realizar una búsqueda eficiente a través de un grafo. Dentro del área de robótica es utilizado en la planificación de trayectorias óptimas en entornos estructurados. El algoritmo encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo inicial y uno final. A* utiliza una función de evaluación f(n) = g(n) + h`(n), donde h`(n) representa el valor heurístico del nodo (n) a evaluar hasta el final, y g(n), el coste real del camino recorrido para llegar a dicho nodo (n) desde el nodo inicial. A* mantiene dos estructuras de datos auxiliares, que se denominan abiertos, implementada como una cola de prioridad (ordenada por el valor f(n) de cada nodo), y cerrados, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la f(n) de todos sus hijos, se enlazan con el nodo padre y los inserta en abiertos, y pasa el nodo evaluado a cerrados. El proceso continúa hasta que un nodo meta (x) tiene un valor f(x) menor que cualquier otro nodo en la cola y coincide con el nodo final (o hasta que el árbol ha sido completamente recorrido, en caso de no existir una ruta). Si se obtiene un nodo meta se busca, de forma recursiva, su padre sucesivamente hasta encontrar el nodo inicial y obtener la ruta buscada. El algoritmo es una combinación entre búsquedas del tipo primero en anchura con primero en profundidad. De este modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores.

2.1.3 Algoritmo de búsqueda Bellman-Ford optimizado

El algoritmo de Bellman-Ford (Cormen *et al.*, 2001) tiene el inconveniente de que en cada etapa explora todos los estados del entorno que pueden visitarse en esa etapa, lo que puede traducirse en un mayor tiempo de ejecución. Torres (Torres, 2010) propone Bellman-Ford optimizado que disminuye el tiempo de ejecución en gran medida al reducir el espacio de estados con posibilidades de ser óptimos. Esto se debe a que en cada etapa solo se exploran aquellos estados cuyo padre haya sido actualizado en la etapa anterior.

La formulación recursiva del método de Bellman-Ford optimizado se presenta de la siguiente manera:

$$F_0 = \begin{cases} 0, & si\ v = s\ \ y\ \ v \notin O \\ \infty, & si\ v\ \neq\ s\ \ \acute{o}\ v \in O \end{cases}$$

$$f_k = \begin{cases} min\{F_{k-1}(v), min_{v \in Q_k, w \in G_{k-1}}(F_{k-1}(w) + C_{vw})\}, & \forall v \notin O \\ \infty, & \forall\ v \in O \end{cases}$$

donde:

***Q*$_k$:** Es el conjunto de estados que se van a explorar en la etapa *k*.

***G*$_{k-1}$:** Es el conjunto de padres de *v* que fueron actualizados en la etapa *k*-1. Se define como padre de un estado *v* cualquier estado *w* del cual se pueda llegar a *v* en una etapa.

***S*:** Estado inicial.

***v, w*:** Estados cualesquiera.

***C*$_{vw}$:** Costo de ir del estado *w* al estado *v*.

El seudo-código del algoritmo puede ser definido como sigue:

***n*:** Cantidad de estados.

***V* :** Conjunto de todos los estados.

***Pv*:** Conjunto de padres óptimos. Un padre óptimo del estado *v* es el estado *w* que hace que se minimice el costo de *v*.

Determinar $F_0(v)\ \forall\ v \in V$ y Q_1

Mientras $Q_k \neq \emptyset$ y $k \leq n$ hacer

Calcular $F_k(v)\ \forall\ v \in Q_k$, $P_v\ \forall\ v \in Q_k$ y determinar Q_{k+1}

Fin Mientras

Si $Q_k = \emptyset$ entonces $F_k(v)$ define el costo de la trayectoria óptima desde *S* hasta *v* $\forall v$. Considerando v_{end} como el estado al cual se quiere llegar desde *S*, entonces la secuencia de estados está dada por:

$vi = v_{end}$

Mientras $vi \neq S$ hacer

$vi = P_{vi}$

Fin Mientras

Si $k = n$ y $Q_k \neq \emptyset$ entonces existen enlaces entre estados con costo negativo y no pueden ser definidas las trayectorias óptimas.

Explorar solo aquellos estados cuyo estado padre haya sido actualizado en la etapa anterior, ya que solo es de interés explorar aquellos estados que realmente tengan la posibilidad de mejorar su costo en la etapa k.

2.1.4 Algoritmo de búsqueda por Colonia de Hormigas

Dada su capacidad de organización descentralizada, las hormigas son capaces de llevar a cabo tareas complejas que en ocasiones pueden exceder las capacidades de una sola hormiga. Tal habilidad es motivo de atención para el hombre, quien a través de los años ha realizado numerosos estudios sobre el comportamiento de las hormigas y otros insectos sociales. En la última década, el comportamiento de las hormigas se ha vuelto fuente de inspiración para desarrollar una serie de algoritmos que resuelven problemas de optimización, modelando la conducta a seguir para buscar el alimento de estos insectos; los algoritmos de hormigas son entonces principalmente utilizados para resolver problemas de combinatoria sobre espacios de búsqueda discretos. Para tener una idea más clara de lo que es ACO, se han considerado dos definiciones formales:

• Algoritmo de optimización global que modela el proceso natural mediante el cual una colonia de hormigas encuentra la ruta óptima del nido al alimento.

• Técnica probabilística para solucionar problemas computacionales que pueden ser reducidos a encontrar buenos caminos mediante grafos, inspirada en el comportamiento de las hormigas para encontrar el camino hacia el alimento.

Un concepto muy importante dentro de la teoría de ACO es la ***estigmergia***, para explicar cómo trabajan los insectos sociales sin necesidad de un poder central o planificación. La estigmergia es la colaboración a través del medio físico. En el caso de las hormigas, cuya capacidad visual es bastante rudimentaria e incluso ciertas especies son completamente ciegas, la comunicación entre los individuos se realiza mediante segregación de químicos producidos por las propias hormigas, para que de esta manera otras sigan su rastro. Tales químicos son denominados feromonas. Estos mensajes químicos están más desarrollados en las hormigas que otros himenópteros, debido a que estas se encuentran en contacto con el suelo constantemente.

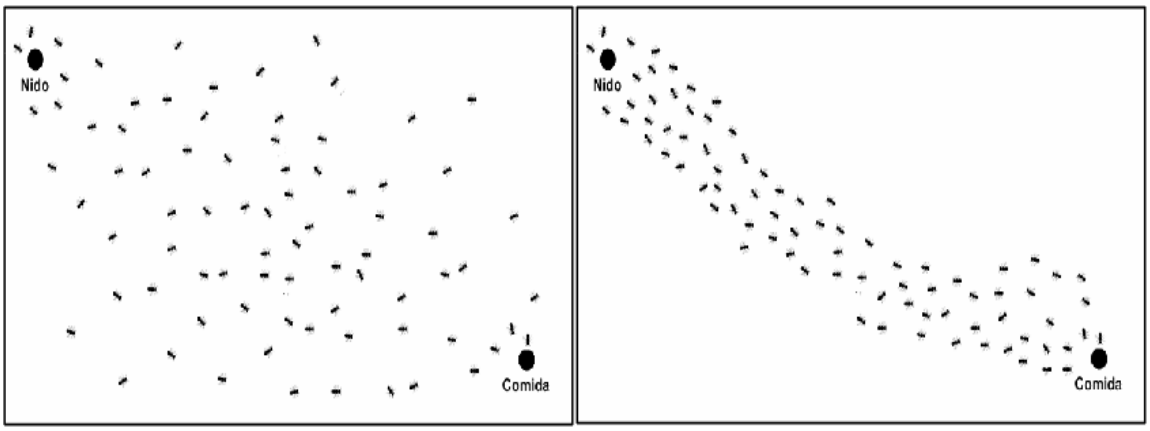


Figura 2.3 Una colonia de hormigas recién ha iniciado la búsqueda de alimento. Conforme aumenta el rastro de feromona indicando la ruta al alimento, más y más hormigas tienden a elegir este camino.

En un inicio, las hormigas presentan un patrón de actividad bastante aleatorio y desordenado en la búsqueda de la comida. En cuanto la comida es encontrada, dichos patrones se vuelven más organizados con más y más hormigas siguiendo el mismo camino hacia la fuente de alimento. Tarde o temprano, todas siguen el mismo camino, siendo este el más corto (Figura 2.3). Cuando una hormiga localiza el alimento, carga con ella un elemento de este hacia el nido y va dejando un rastro de feromona tras ella. Las demás hormigas exploradoras deciden qué camino seguir basándose en la concentración de feromona en los diferentes caminos posibles. Los caminos con mayor concentración de feromona tienen mayor probabilidad de ser elegidos. Mientras más hormigas sigan un rastro determinado, el interés por ese camino se ve reforzado por la feromona depositada por las nuevas hormigas, que atrae a su vez más hormigas. La idea de ACO es imitar este comportamiento con hormigas artificiales caminando por el grafo que representa el problema a resolver.

2.1.4.1 Meta-heurística de ACO

Se entiende por meta-heurística un conjunto de conceptos algorítmicos que pueden ser utilizados para definir métodos heurísticos que se aplican a una amplia gama de problemas. Puede verse a una meta-heurística como un algoritmo de propósito general, mientras que las heurísticas que se basan en ella son algoritmos de propósito específico. Entonces, la meta-heurística de ACO (ACO-MH, por sus siglas en inglés) resulta de las observaciones del comportamiento de las hormigas que han inspirado el

desarrollo de un gran número de algoritmos, utilizados para resolver diversos problemas de optimización sobre espacios de búsquedas discretos. Conjuntamente, a estos algoritmos se les denomina instancias de ACO-MH.

2.1.4.2 Planteamiento del problema

Sea el grafo G = (V,E), donde V es el conjunto de vértices (nodos) y E es la matriz de conexiones entre nodos. G tiene n_G = |V| nodos. Sea L^k el número de saltos en el camino construido por la hormiga *k* del nodo origen al nodo destino. Entonces hay que encontrar:

$$Q = \{qa\ ,...,q\ f \mid qi \in C\}$$

donde:

Q = conjunto de nodos que representa al menos un camino continuo sin obstáculos, *qa,…, qf* = nodos que forman dicho camino.

C = conjunto de configuraciones que especifican el espacio libre (nodos factibles). Si $x^k(t)$ denota una solución *Q*, $f(x^k(t))$ expresa la calidad de la solución.

2.1.4.3 Simple Ant Colony Optimization (SACO)

SACO es una variante algorítmica de ACO que adapta el comportamiento de las hormigas reales para solucionar problemas de caminos de costo mínimo en grafos. Cierto número de hormigas artificiales construyen soluciones para determinado problema de optimización e intercambian información acerca de la calidad de dichas soluciones haciendo alusión al sistema de comunicación de las hormigas reales (Dorigo, 2004). En general, los pasos que sigue SACO son los siguientes:

- A cada conexión (i,j) se le asocia cierta concentración inicial de feromona τ_{ij} .
- Un número de hormigas $k = 1,\ldots,n_k$ son colocadas en el nodo origen.
- En cada iteración cada hormiga construye un camino al nodo destino. Para elegir el nodo siguiente desde el nodo actual, se utiliza la fórmula de probabilidad:

$$p_{ij}^{k}(t)\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)}{\sum_{h\epsilon N_i^k}\tau_{ih}^{\alpha}(t)\xi^{ß}} & if\ \ h \in N_i^k \\ 0 & if\ \ h \notin N_i^k\end{cases}$$

donde:

N_i^k = conjunto de nodos factibles conectados al nodo *i* con respecto a la hormiga *k*.

τ_{ij} = concentración total de la feromona en el vértice *ij.*

$\xi^{ß}$ = la distancia euclidiana entre el nodo candidato y el nodo destino.

α = constante positiva utilizada para amplificar la influencia de la concentración de feromona.

ß = constante positiva utilizada para amplificar la influencia de ξ.

- Remover ciclos y calcular peso de cada ruta $f(x^k(t))$. Un ciclo puede generarse cuando ningún nodo candidato es factible, esto es, que para cualquier nodo *i* y hormiga *k.*

  N_i^k = Ø ; entonces el predecesor de dicho nodo *i* es incluido como parte del camino.

- Calcular evaporación mediante la ecuación:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t)$$

donde ρ es el factor de evaporación del rastro de feromona. La evaporación es añadida al algoritmo para forzar a las hormigas a explorar más, y no converger prematuramente a una solución (que sería el caso en que ρ = 0). Si ρ = 1, la búsqueda es completamente aleatoria.

Mientras una hormiga tarde más en recorrer un camino y regresar, más tiempo de evaporación hay de las feromonas. En cambio, en un camino corto, que se recorre más rápidamente, la densidad del rastro de feromona se mantiene mucho más alta. En otras palabras, $\Delta\tau_{ij}^{k}$ es inversamente proporcional a L_k. La evaporación evita la convergencia a óptimos locales. Sin la evaporación, los caminos de las primeras hormigas serían excesivamente atractivos para las subsecuentes. De esta manera, la exploración del espacio solución estaría muy restringida.

- Actualizar concentración de feromonas utilizando la ecuación:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t)$$

- El algoritmo puede terminar de tres maneras distintas:
  - Cuando se ha alcanzado un número máximo de iteraciones.
  - Cuando se ha encontrado una solución aceptable, con $f(x^k(t)) < \varepsilon$.
  - Cuando todas las hormigas siguen el mismo camino.

2.1.4.4 Consideraciones acerca de SACO

- Trabaja muy bien para grafos sencillos. Para grafos grandes es más sensible a los valores de los parámetros seleccionados.
- La convergencia al camino óptimo se logra para pocas hormigas; muchas hormigas provocan comportamiento no convergente.
- Para ρ = 0 el algoritmo tiende a estancarse en óptimos locales; para valores muy altos converge en soluciones sub-óptimas.
- Para valores pequeños de α el algoritmo converge al camino más corto generalmente, viceversa para valores grandes.
- Es necesario tener un balance del sentido de exploración de las hormigas y de la explotación del rastro de feromonas, para que se vean forzadas a explorar caminos alternos y a su vez no sobre explotar la concentración de feromonas, de tal manera que el algoritmo no se estanque prematuramente en soluciones sub-óptimas.

2.2 Generación del camino

Con el fin de obtener un camino suave a partir de la secuencia de puntos dada por la etapa de planificación se utilizan una serie de curvas polinomiales que interpolan o en ocasiones se aproximan a la secuencia. Entre ellas se encuentran las llamadas espirales cúbicas (Kanayama and Hartman, 1989), (*Liang et al.,* 2005), y las ß-Spline (Gómez-Bravo *et al.*, 2008), (Muñoz, 1995). En este trabajo en la etapa de generación se utiliza las ß-Spline. La meta es generar un camino factible para un robot móvil con restricciones del tipo no holonomicas, que están determinadas por el radio de giro máximo que puede describir el robot y el grado de curvatura, (figura 2.4).

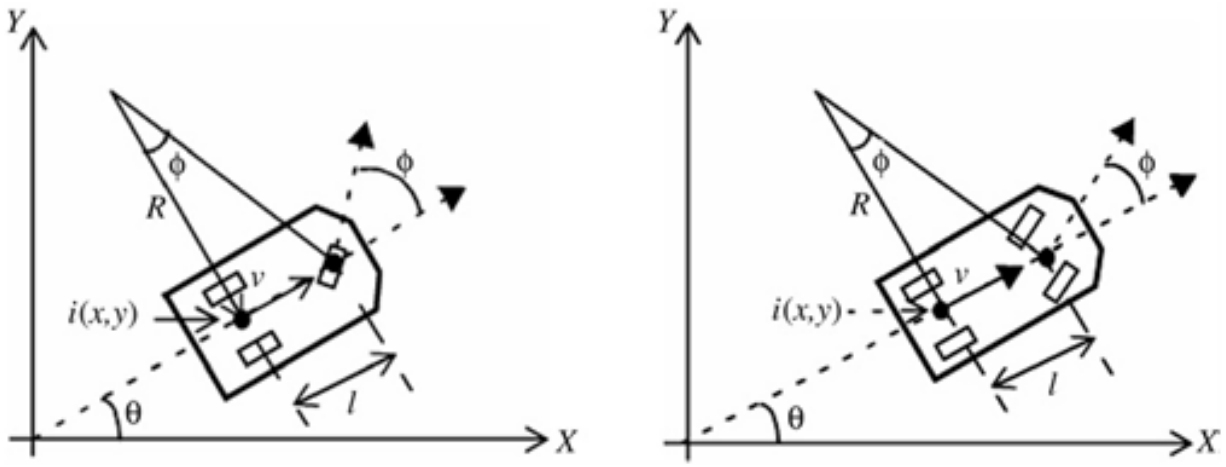


Figura 2.4 Modelo de robot móvil tipo triciclo y robot móvil tipo Ackerman o carro.

En la (figura 2.4) l representa la separación entre los ejes de las ruedas delanteras y traseras, ϕ la máxima variación del ángulo de dirección, y $\mathrm{R_{min}}$ el radio de giro mínimo que podrá realizar el vehículo en correspondencia con los valores de l y ϕ, el cual se puede calcular mediante la ecuación (1).

$$R_{min} = \frac{l}{\tan \phi} \qquad (1)$$

El radio de giro mínimo describe la capacidad de un determinado vehículo para girar. Cuanto más corto es el radio de giro de un vehículo se dice que este ofrece más maniobrabilidad. El inverso de este define el valor de curvatura máxima $K_{máx}$, que podrá tener el camino a generar para que pueda ser recorrido por el robot.

$$K_{máx} = \frac{1}{R_{min}} = \frac{\tan \phi}{l} \qquad (2)$$

Dentro de las características deseables de estas curvas están la de continuidad en la orientación. De existir una discontinuidad en la orientación, sería necesario imprimir un cambio brusco en la orientación del vehículo, el cual no puede efectuar debido a la restricción de no holonomicidad. La curvatura deberá estar acotada, y debido a las restricciones cinemáticas del vehículo, hay un radio de giro mínimo que puede realizar el mismo. La variación de la curvatura debe ser lineal, así se minimiza el esfuerzo de control que se debe ejercer sobre los actuadores del vehículo.

2.2.1 Curva ß-Spline

Las ß-Spline son un tipo de curvas paramétricas a trozos, o sea, la curva va a estar dada por un conjunto de segmentos de curvas donde el parámetro t varía de 0 a 1. La curva presenta continuidad de la primera y segunda derivada incluso en los puntos de unión de dos segmentos adyacentes. Cada segmento de la curva es especificado por 4 puntos de control a los que la curva no interpola, definidos en (Gómez Bravo et al., 2008) como vértices de control. A la conexión en secuencia de todos los vértices de control se le conoce como polígono de control, de ahí que si existen n vértices de control existirán entonces n-3 segmentos.

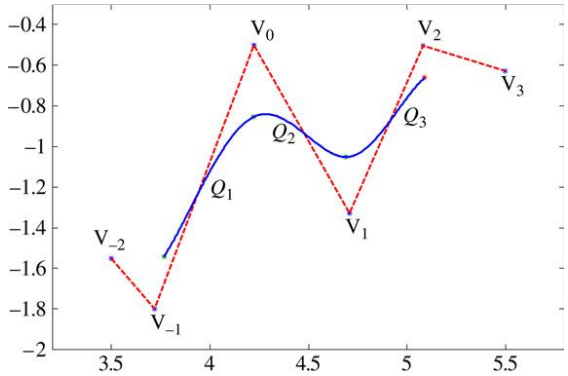


Figura 2.5 Segmentos de la ß-spline y vértices de control.

Un segmento i de la curva cuya secuencia de vértices de control es la siguiente: V_0, V_1, …, V_{n-1}, siendo cada uno de estos vértices especificado por sus coordenadas cartesianas $V_j(V_{j\_x}, V_{j\_y})$, viene definido de la siguiente manera:

$$Q_i(t) = \sum_{r=0}^{3} b_r\,(\beta_1, \beta_2, t) V_{i+r} \qquad 0 \le t < 1, \qquad i = 0, 1, 2, \qquad \dots, n-4. \qquad (3)$$

Donde b_r son funciones peso que son evaluadas para cualquier valor de t perteneciente al dominio y los valores de los parámetros β, donde el primero controla en cada segmento de la curva si esta se inclina más hacia la derecha o la izquierda, y el segundo controla la tensión de la curva (esto es si la curva pasa más o menos próxima a los vértices de control). Según (Gómez Bravo et al., 2008) los mejores resultados para cambios suaves en la curvatura se obtienen para β_1=0 y β_2=0, en este caso las funciones peso vienen definidas por:

$$\begin{bmatrix} b_0(t) \\ b_1(t) \\ b_2(t) \\ b_3(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & \frac{-1}{2} & \frac{1}{2} & \frac{-1}{6} \\ \frac{2}{3} & 0 & -1 & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \times \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \quad (4)$$

Resolviendo la ecuación anterior (4) se tiene que:

$$Q_i(t) = b_0(t) \times V_i \ + b_1(t) \times V_{i+1} \ + b_2(t) \times V_{i+2} \ + b_3(t) \times V_{i+3} \quad (5)$$

Sustituyendo b_0, b_1, b_2, y b_3, efectuando y agrupando se tiene que:

$$Q_i(t) = t^3\left(\frac{-1}{6}V_i + \frac{1}{2}V_{i+1} - \frac{1}{2}V_{i+2} + \frac{1}{6}V_{i+3}\right) + t^2\left(\frac{1}{2}V_i - V_{i+1} + \frac{1}{2}V_{i+2}\right) + t\left(\frac{-1}{2}V_i \ + \frac{1}{2}V_{i+2}\right) + \frac{1}{6}V_i + \frac{2}{3}V_{i+1} + \frac{1}{6} \quad (6)$$

Como se puede apreciar en esta última ecuación (6) $Q_i(t)$ estará definido por una función cúbica de la forma ax^3+bx^2+cx+d, donde a, b, c y d se calculan a partir de los cuatro vértices de control que definen el segmento.

Un punto $Q_i(t) = \left(Q_{i_x}(t),\ Q_{i_y}(t)\right)$ estará dado por:

$$Q_{i_x}(t) = a_x t^3 + b_x t^2 + c_x t + d_x \quad (7)$$

$$Q_{i_y}(t) = a_y t^3 + b_y t^2 + c_y t + d_y \quad (8)$$

donde:

$$a_x = \frac{-1}{6}V_{i_x} + \frac{1}{2}V_{i+1_x} - \frac{1}{2}V_{i+2_x} + \frac{1}{6}V_{i+3_x} \quad (9)$$

$$a_y = \frac{-1}{6}V_{i_y} + \frac{1}{2}V_{i+1_y} - \frac{1}{2}V_{i+2_y} + \frac{1}{6}V_{i+3_y} \quad (10)$$

$$b_x = \frac{1}{2}V_{i_x} - V_{i+1_x} + \frac{1}{2}V_{i+2_x} \quad (11)$$

$$b_y = \frac{1}{2}V_{i_y} - V_{i+1_y} + \frac{1}{2}V_{i+2_y} \quad (12)$$

$$c_x = \frac{-1}{2} V_{i_x} + \frac{1}{2} V_{i+2_x} \quad (13)$$

$$c_y = \frac{-1}{2} V_{i_y} + \frac{1}{2} V_{i+2_y} \quad (14)$$

$$d_x = \frac{1}{6} V_{i_x} + \frac{2}{3} V_{i+1_x} + \frac{1}{6} V_{i+2_x} \quad (15)$$

$$d_y = \frac{1}{6} V_{i_y} + \frac{2}{3} V_{i+1_y} + \frac{1}{6} V_{i+2_y} \quad (16)$$

Si se tiene el polígono de control se tiene la curva, bastará con buscar el valor de a_x, a_y, b_x, b_y, c_x, c_y, d_x, y d_y, para cada cuatro vértices de control consecutivos, y obtener con estos un conjunto de puntos pertenecientes a cada segmento de la curva variando el parámetro t de 0 a 1 con intervalos discretos Δt.

2.2.2 Descripción de la solución

En la bibliografía consultada, los métodos de generación de caminos usando curvas β-Spline como, es el caso de Muñoz, (Muñoz, 1995), (Gómez-Bravo et al., 2008) construyen a partir de la ruta planificada un conjunto de arcos de circunferencia y segmentos de rectas tangentes a estos, sobre los que erigen el polígono de control de la curva, con vértices de control equidistantes entre sí, una distancia δ, la separación homogénea de los vértices de control garantiza que la atracción que ejerce cada uno de ellos sobre la curva se reparta de forma equitativa, por lo que se evita saltos indeseados en la curvatura.

La ruta planificada estará dada por un conjunto de puntos consecutivos del mapa, una primera aproximación al camino deseado es la unión de estos puntos con segmentos de rectas con lo que se consigue continuidad en la posición (figura 2.6 a).

Es evidente la discontinuidad en la orientación en los puntos P2, P4, P5, P6, P7, P8, P10, P12, P13 y P14. Puntos en que los segmentos de rectas adyacentes a estos formadas por la unión de dos puntos consecutivos tienen desigual pendiente. Una solución para estos puntos de unión de segmentos de rectas con desigual pendiente es el uso de arcos de circunferencia tangentes a los dos segmentos adyacentes al punto de unión (figura 2.6 b).

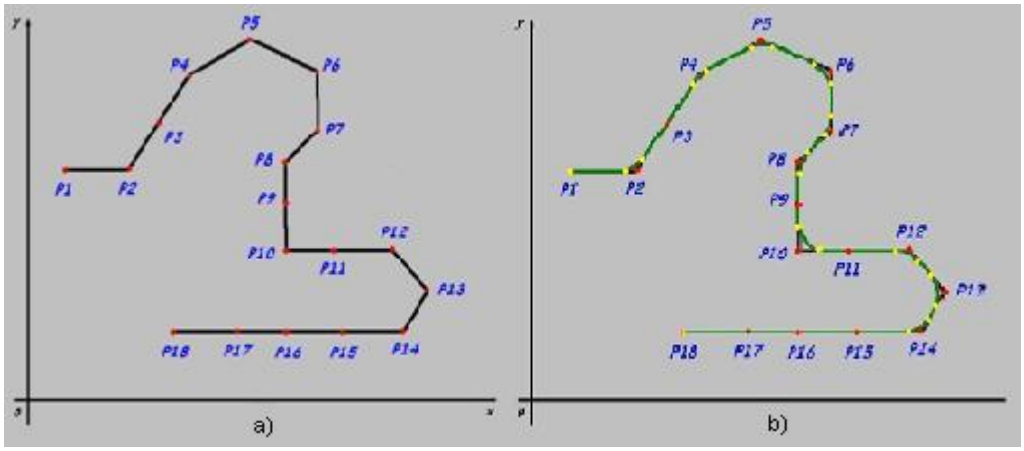


Figura 2.6 a) Ruta Planificada. b) Puntos de inicio y fin de transición

La nueva aproximación al camino deseado, se describe como un conjunto de nuevos segmentos de rectas y arcos de circunferencia, teniendo en cuenta los puntos en que sucede la transición de un segmento de recta a un arco de circunferencia, de un arco de circunferencia a un segmento de recta, y de un arco de circunferencia a otro arco de circunferencia.

El conjunto de segmentos de rectas y arcos de circunferencia no es más que una lista simplemente enlazada que una vez construida puede ser recorrida para obtener el conjunto de vértices de control (VC), equidistantes entre sí una distancia δ, que define el polígono de control de la curva. Una vez obtenido el polígono de control se puede obtener la curva β-Spline que constituye un camino de referencia factible para el robot.

Para obtener el conjunto de vértices de control se implementa el siguiente algoritmo.

El algoritmo se divide en tres etapas:

1- Obtención del conjunto de arcos de circunferencia y segmentos de recta.
2- Creación de los puntos de control equidistantes.
3- Obtención de la curva β-Spline.

A continuación se explica cada uno de estos pasos.

2.2.2.1 Obtención del conjunto de arcos de circunferencia y segmentos de recta

La entrada del algoritmo es la ruta planificada, y las restricciones cinemáticas (l y ϕ) del robot móvil para el que se desea generar el camino, pero no se trata directamente con la ruta planificada, sino con un conjunto de puntos que contendrá a la ruta planificada definida en (Torres, 2010) como camino auxiliar (CA).

Se define CA como un conjunto de puntos de un camino auxiliar que contiene la ruta planificada $\Pi = \{P_1, \dots, P_{i-1}, P_i, P_{i+1}, \dots, P_n\}$ y sirve como intermedio para obtener el camino generado. CA se construye a partir de Π de la siguiente manera:

$$ca_j = \begin{cases} P_i, & i = \dfrac{(j+1)}{2}, \forall\ j\ impar; \\ P_i + \dfrac{P_{i+1} - P_i}{2}, & i = \dfrac{j}{2}, \forall\ j\ par. \end{cases} \quad (17)$$

Por otra parte, los arcos de circunferencia serán construidos a partir de circunferencias de igual radio R; y desde esta etapa se prevé que la máxima curvatura del camino generado no vaya a sobrepasar al máximo valor de curvatura $Kmáx$ visto en la ecuación (2). Para prevenir esto, R y la distancia entre los puntos de control δ son calculados mediante las ecuaciones (18) y (19), para más detalles ver Anexo A.

$$R = \frac{R_{mín}}{0.98725} \quad (18)$$

donde $R_{mín}$ se calcula mediante la ecuación (1)

$$\delta = 0.196034 \times R \quad (19)$$

Como ya se mencionó el conjunto de segmentos de rectas y arcos de circunferencia no son más que una lista simplemente enlazada. Un segmento de recta tiene un punto inicial y un punto final, y un arco de circunferencia también, además de tener un radio R, y un centro O. Para insertar un segmento o un arco en la lista, hay que saber su punto inicial, su punto final, y en el caso de los arcos, hay que conocer además su radio R y su centro O.

Falta saber cómo calcular los puntos de tangencia de los arcos de circunferencia de radio R, que son los puntos iníciales y finales de cada arco, y cómo calcular el centro O de la circunferencia correspondiente a cada arco.

Para hallar los puntos de tangencia C_1 y C_2, de un arco de circunferencia de radio R conocido de antemano, y centro O, tangente a los segmentos $\overline{P1P2}$ y $\overline{P2P3}$, ver Figura 2.7, se utilizan las ecuaciones (20) y (21), para más detalles ver Anexo B.

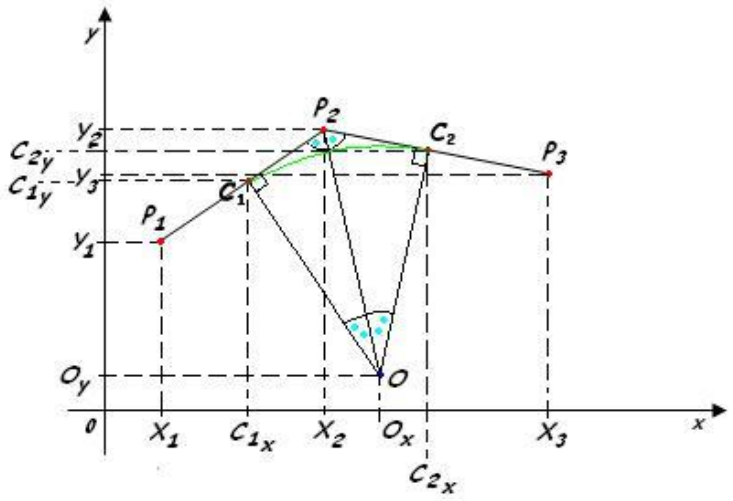


Figura 2.7 Representación geométrica de los puntos tangentes C1 y C2

$$C_1\left(C_{1_x},\, C_{1_y}\right) = (k_1(x_1 - x_2) + x_2,\;\; k_1(y_1 - y_2) + y_2\,) \tag{20}$$

$$C_2\left(C_{2_x},\, C_{2_y}\right) = (k_2(x_3 - x_2) + x_2,\;\; k_2(y_3 - y_2) + y_2\,) \tag{21}$$

Donde:

$$k_1 = \frac{d}{d_1} \tag{22}$$

$$k_2 = \frac{d}{d_2} \tag{23}$$

$$d_1 = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \tag{24}$$

$$d_2 = \sqrt{(y_2 - y_3)^2 + (x_2 - x_3)^2} \tag{25}$$

$$d = \; R \times \cot(\alpha/2) \tag{26}$$

En la ecuación (26) α representa el menor ángulo formado por los dos segmentos $\overline{P1P2}$ Y $\overline{P2P3}$, el cual por la representación del entorno usada por el planificador (descomposición mediante celdas cuadradas) puede ser 90° o 135°.

El punto O se encuentra en la intersección de dos rectas r_i y r_j que pasan por los puntos C_1 y C_2, y que son perpendiculares a las rectas r_1 y r_2 respectivamente. La recta r_1 es la recta sobre la que está el segmento $\overline{P1P2}$ y la recta r_2 es sobre la que está el segmento $\overline{P2P3}$. Las pendientes de las rectas r_1 y r_2 pueden ser calculadas fácilmente, basta elegir dos puntos que estén sobre las mismas, y se tiene que P_1, C_1, y P_2 están sobre r_1 y P_2, C_2 y P_3 están sobre r_2. Como la pendiente de una recta perpendicular a otra que se conoce su pendiente es el opuesto del inverso de esta, se pueden obtener las pendientes de r_i y r_j. Luego al tener la pendiente y un punto por el que pasa la recta, para ambas rectas r_i y r_j, se pueden obtener las ecuaciones de las mismas. Una vez obtenidas la ecuaciones de las rectas r_i y r_j, se busca el punto de intersección de las mismas que es el punto O.

El algoritmo para obtener la lista de segmentos de rectas y arcos de circunferencia se describe a continuación:

Inicialización: *R-radio, CA*-camino auxiliar*, INC=*CA_1*,* $P_1 = CA_2$*, j = 3,*

N = |*CA*|*, LSARC = { }*

Nota: si *tipomov* igual 180*,* entonces *tipomov = 1*

si *tipomov* igual 135 ó 90*,* entonces *tipomov = 2*

LSARC lista de segmentos y arcos.

N- cantidad de elementos del camino auxiliar

P_1, P_2, P_3, puntos del camino auxiliar

Mientras $j \leq N - 3$ hacer

$P_2 = CA_j$

$P_3 = CA_{j+1}$

Si (tipomov(P_1, P_2, P_3) <> 1)

inicio

Obtener puntos de tangencia conP_1,P_2,P_3,R hallar C_1, C_2

Si (INC <> C_1)

inicio

Insertarsegmento(LSARC, segmento(INC, C_1))

fin

Obtener centro con P_1,P_2,P_3,C_1,C_2 hallar O

Insertararco(LSARC, arco(C1, C2, R, O))

INC = C_2

Fin

$P_1 = P_3$

j = j + 2

Fin Mientras

Insertarsegmento(LSARC, segmento(INC, CA_N))

2.2.2.2 Obtención de los puntos de control equidistantes

La segunda parte del algoritmo consiste en construir el polígono de control que yace sobre los segmentos de rectas y arcos de circunferencias calculados en la primera parte, con vértices equidistantes entre sí, una distancia δ.

Los segmentos pertenecerán a una recta donde se calculan los valores de los incrementos en x y y, a los que se les llama Δx y Δy, que hacen falta para calcular las coordenadas de un nuevo punto que está sobre el segmento a una distancia δ del anterior.

Las coordenadas del punto P_i (x_i, y_i) se calculan así:

$$x_i = x_{i-1} \pm \Delta x \quad (27)$$

$$y_i = y_{i-1} \pm \Delta y \quad (28)$$

$$\Delta x = \delta cos(\alpha) \quad (29)$$

$$\Delta y = \delta sen(\alpha) \quad (30)$$

Los incrementos se le suman o se le restan a las coordenadas del punto anterior teniendo en cuenta las coordenadas del punto inicial y el final del segmento.

Para la obtención de los puntos equidistantes sobre un arco de circunferencia se utilizan las ecuaciones paramétricas de la circunferencia. Los arcos de circunferencia van a estar limitados por dos puntos, que conviene saber a qué valor del parámetro t

corresponden. Estos dos puntos C_1 y C_2, fueron calculados para cada arco de circunferencia en la primera parte del algoritmo. Otra cuestión que convendría saber es el incremento del parámetro t necesario para calcular las coordenadas de un nuevo punto que estará sobre el arco de circunferencia a una distancia δ del anterior.

Si se denota como t_o y t_f a los valores de los parámetros correspondientes a los puntos C_1 y C_2, y Δt como el incremento necesario para lograr una separación de δ entre dos puntos consecutivos. Las coordenadas del punto $P_i(x_i, y_i)$ se calculan así:

$$x_i = x_0 + Rcos(t_i) \quad (31)$$

$$y_i = y_0 + Rsen(t_i) \quad (32)$$

El incremento del parámetro t necesario para calcular las coordenadas de un nuevo punto que estará sobre el arco de circunferencia a una distancia δ del anterior ya ha sido calculado y es 11.25° o π/16 rad, ver Anexo A.

Para cualquier punto $P(x_p, y_p)$ de la circunferencia de centro $O(x_O, y_O)$ y radio R, puede ser obtenido su parámetro t correspondiente mediante la ecuación (33), ajustando el ángulo a su respectivo cuadrante.

$$t=arctan(y_p–y_o,x_p–x_o) \quad (33)$$

Ya casi se está en condiciones de poder construir el polígono de control con vértices de control equidistantes entre sí una distancia δ; falta solamente analizar que al moverse por un segmento de recta buscando los puntos equidistantes sobre este, no necesariamente se empieza en el punto inicial del segmento y se termina en el punto final del mismo, de igual manera sucede con los arcos de circunferencia. Por ejemplo, en una transición de un segmento de recta a un arco de circunferencia, se calculan los puntos sobre la recta a la que pertenece el segmento, si al obtener un punto, este no está sobre el segmento, es decir, se pasa del punto final del segmento, se debe buscar a partir del punto anterior que si lo está un punto sobre el arco de circunferencia que esté a una distancia δ del mismo. Esto se logra buscando los puntos de intersección de la circunferencia de radio δ con centro en el último punto calculado perteneciente al segmento y la circunferencia de radio R y centro O hallado previamente, a la que pertenece el arco de circunferencia en cuestión. De los puntos hallados se descarta

aquel cuyo correspondiente parámetro t en relación a la circunferencia de radio R no se encuentre dentro de t_o y t_f.

2.2.2.3 Creación de la curva discreta

Una vez que se tiene el polígono de control se está en condiciones de obtener la curva β–Spline que constituye un camino de referencia continuo en posición, orientación y curvatura, además de poseer esta última cambios suaves y estar acotada.

El primer y último punto del polígono de control son el primer y último punto de la ruta planificada, la curva no toca estos puntos, para lograr que lo haga se adiciona un VC adicional al inicio del polígono de control y otro al final del mismo.

Sean V_{i1}, V_{i2} los dos primeros puntos del polígono de control, coincidiendo V_{i1} con el primer punto de la ruta planificada, el punto adicional V_{io} a insertar en el polígono de control, que garantiza que la curva comience en V_{i1} y tenga en este punto la orientación del segmento de recta donde se encuentran V_{i1} y V_{i2} se calcula como:

$$V_{i0}=2V_{i1}-V_{i2} \quad (34)$$

De manera similar, si se tiene que V_{f1}, V_{f2} son los dos últimos puntos del polígono de control, coincidiendo V_{f2} con el último punto de la ruta planificada, el punto adicional V_{f3} a insertar en el polígono de control, que garantiza que la curva termine en V_{f2} y tenga en este punto la orientación del segmento de recta donde se encuentran V_{f1} y V_{f2} se calcula como:

$$V_{f3}=2V_{f2}-V_{f1} \quad (35)$$

Como se dijo anteriormente: Si se tiene el polígono de control se tiene la curva, basta con buscar el valor de $a_x, a_y, b_x, b_y, c_x, c_y, d_x, y\ d_y$, para cada cuatro vértices de control consecutivos, y obtener con estos un conjunto de puntos pertenecientes a cada segmento de la curva variando el parámetro t de 0 a 1 con intervalos discretos Δt. La elección de este Δt guardará relación con la distancia d_p a la que se quiere estén los puntos del camino discretizado.

La curva β-Spline generada es el conjunto de puntos, que junto con la información de la orientación y la curvatura en cada punto constituye el camino de referencia para el

robot móvil. Como para esta curva no hay ecuaciones que permitan hallar la orientación y la curvatura en función de su parámetro, estas deben ser calculadas de forma discreta. Para un punto P_i la orientación se calcula con este y un punto consecutivo P_{i+1}, como:

$$\theta_i = \tan^{-1}\left(\frac{P_{i+1_y} - P_{i_y}}{P_{i+1_x} - P_{i_x}}\right) \tag{36}$$

Ajustando el ángulo a su respectivo cuadrante.

La curvatura se calcula con P_i, P_{i+1} y otro punto consecutivo a este último P_{i+2}, haciendo uso de la ecuación (37).

$$K = \frac{\frac{dx}{ds}\frac{d^2y}{ds^2} - \frac{dy}{ds}\frac{d^2x}{ds^2}}{\left[\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2\right]^{\frac{3}{2}}} \tag{37}$$

donde:

$$\frac{dx}{ds} = \frac{P_{i+1_x} - P_{i_x}}{\Delta s_1} \tag{38}$$

$$\frac{dy}{ds} = \frac{P_{i+1_y} - P_{i_y}}{\Delta s_1} \tag{39}$$

$$\frac{d^2x}{ds^2} = \frac{P_{i+2_x} - 2P_{i+1_x} + P_{i_x}}{\Delta s_1 \times \Delta s_2} \tag{40}$$

$$\frac{d^2y}{ds^2} = \frac{P_{i+2_y} - 2P_{i+1_y} + P_{i_y}}{\Delta s_1 \times \Delta s_2} \tag{41}$$

$$\Delta s_1 = \sqrt{\left(P_{i+1_y} - P_{i_y}\right)^2 + \left(P_{i+1_x} - P_{i_x}\right)^2} \tag{42}$$

$$\Delta s_2 = \sqrt{\left(P_{i+2_y} - P_{i_{+1y}}\right)^2 + \left(P_{i+2_x} - P_{i+1_x}\right)^2} \tag{43}$$

2.3 Conclusiones del capítulo

- Se describe a profundidad el algoritmo de descomposición en celdas cuadradas para la obtención del espacio de configuraciones libres, así como los algoritmos

para el cálculo de ruta A*, Bellman Ford y SACO dentro de la etapa de planificación. Para la generación del camino se explicó el suavizado de la ruta mediante las ß-Spline, la obtención de los vértices de control, y la generación de la curva.

CAPÍTULO 3. IMPLEMENTACIÓN DE LA PLATAFORMA BASADA EN AGENTES

En este capítulo se describe el desarrollo de la plataforma basada en agentes, los requerimientos, análisis y diseño, una descripción de las principales características de JADE y la solución propuesta.

Se presenta la interfaz principal donde se describen las principales funcionalidades, otras interfaces donde se muestran algunas de las métricas que permiten emitir criterios del desempeño de los algoritmos de planificación. Se muestra un caso de estudio donde se ubican tres robots en la misma posición y se planifican con algoritmos diferentes, en este caso A*, Bellman Ford y SACO. Se muestran los pasos a seguir para agregar un nuevo algoritmo de planificación a la plataforma, así como su flujo de ejecución.

3.1 Desarrollo de la plataforma

El modelo de procesos es una estrategia de la ingeniería del software para resolver problemas reales de una industria, definiendo un marco de trabajo para un conjunto de áreas claves del proceso que se establece para la entrega efectiva del producto, aplicando métodos que indican como construir técnicamente el software. Por lo cual el desarrollo de la plataforma se basa en el modelo de procesos orientado a objetos descrito en (Pressman, 2005).

Para realizar la secuencia de un proceso orientado a objetos se realizan las siguientes fases:

1. Definición de requerimientos del entorno y los agentes.
2. Análisis de requisitos de los agentes.
3. Diseño de los agentes y el entorno.
4. Implementación de la plataforma.
5. Pruebas de la plataforma.
6. Evaluación de los resultados obtenidos con la plataforma.

El desarrollo de los agentes se basa en la metodología propuesta por (Nikraz et al, 2006) descrita en el capítulo I.

3.1.1 Fase de análisis

El objetivo del análisis es el reconocimiento de los elementos básicos del problema tal y como los percibe el usuario. Donde el análisis de requisitos del software permite especificar las características operacionales (función, datos y rendimiento) entendiendo el comportamiento del software en el contexto de acontecimientos que afectan al sistema, permitiendo describir el problema de manera que se pueda sintetizar un enfoque o solución global (Pressman, 2005).

3.1.1.1 Requisitos funcionales

Con la participación del equipo de investigación en robótica móvil del Departamento de Automática del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) se identificaron los Requisitos Funcionales (RF) que debe cumplir la plataforma de navegación en entornos estructurados.

- RF 1 Gestionar entorno.
  - Este requisito debe permitir crear un nuevo entorno, abrir uno creado con anterioridad, guardarlo para un análisis posterior o salir de la plataforma.
- RF 2 Ubicación de obstáculos.
  - Debe permitir brindar la posibilidad de ubicar obstáculos en el entorno, ya sean círculos o rectángulos, moverlos para lograr una mejor ubicación o borrarlos en caso de ser necesario.
- RF 3 Ubicación del robot.
  - Debe proveer la opción de ubicar varios robots en el entorno, especificando la posición inicial y final que deben alcanzar, así como visualizar las posiciones de un robot ubicado con anterioridad.
- RF 4 Algoritmo de planificación.
  - Debe permitir seleccionar un algoritmo de planificación para calcular la ruta a la hora de ubicar un robot, así como agregar un nuevo algoritmo.
- RF 5 Simular.
  - Debe iniciar la simulación mostrando los caminos generados desde las posición origen hasta la meta de cada robot con el algoritmo de planificación seleccionado.

- RF 6 Métricas.
  - Debe mostrar las métricas en forma de gráficos que permitan emitir criterios para la evaluación de los algoritmos en cuanto a costo, iteraciones, tiempo de ejecución. Además de mostrar las gráficas descritas por la ruta planificada y el camino generado para cada robot.

3.1.1.2 Casos de uso

Los casos de uso son una forma efectiva para capturar los requerimientos funcionales potenciales de un nuevo sistema. Cada caso de uso es un documento narrativo que describe la secuencia de eventos de un actor que utiliza un sistema para completar un proceso. La vista de casos de uso modela la funcionalidad del sistema, enumerando a los actores, casos de uso y sus interacciones, a través de un diagrama. En la (figura 3.1) se muestra el diagrama de casos de uso identificado para la plataforma de navegación, donde existen dos actores el usuario y el AgentePlataformaNavegacion, la descripción de cada caso se uso se encuentra en el Anexo C.

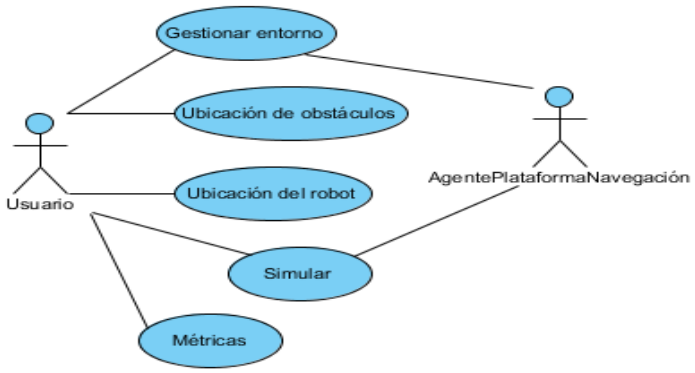


Figura 3.1 Diagrama de casos de uso de la plataforma de navegación.

3.1.1.3 Identificación de los tipos de agentes

Dada la modularidad de la navegación en la robótica móvil, se puede dividir en tres etapas fundamentales: planificación, generación y control. Para la etapa de planificación y generación se determinó un agente que se especializó en dichas funciones, así como un agente encargado del control global de la plataforma: AgentePlanificador (AP), AgenteGenerador (AG), AgentePlataformaNavegacion (APN). En la (tabla 3.1) se describe la especificación del entorno de trabajo de los agentes, denominado REAS (**R**endimiento, **E**ntorno, **A**ctuadores, **S**ensores).

Tabla 3.1 Descripciones del entorno de agentes.

Tipo de Agente	**Medidas de Rendimiento**	**Entorno**	**Actuadores**	**Sensores**
APN	1. Satisfacción de los usuarios. 2. Control de la plataforma.	-Controlador de la plataforma -Usuario.	-Control de la definición del entorno. -Mostrar las rutas generadas. -Mostrar los criterios de comparación. -Envió de mensajes.	-Teclado para iniciar la simulación. - Buzón de mensajes.
AP	1. Planificación de rutas con los algoritmos requeridos.	-APN -Robot	-Planificar la ruta de los robots de acuerdo al algoritmo seleccionado. -Envió de mensajes.	-Buzón de mensajes.
AG	1. Generación de caminos.	-APN -Robot	-Generar el camino a partir de la ruta planificada. -Envió de mensajes.	-Buzón de mensajes.

De acuerdo a la identificación de los tipos de agentes se obtiene el diagrama de agentes en la (figura 3.2), donde los agentes son representados por círculos.

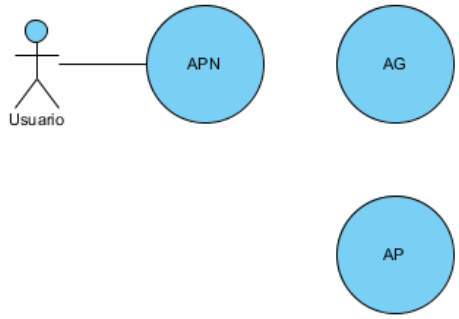


Figura 3.2 Diagrama de agentes.

3.1.1.4 Identificación de responsabilidades de los agentes

Tabla 3.2 Responsabilidades principales de los agentes.

Tipo de agente	**Responsabilidades**
APN	*1.* Crear y controlar el entorno gráfico de la plataforma. *2.* Crear los agentes AP y AG. *3.* Gestionar las solicitudes de simulación del usuario. *4.* Solicitarle al agente AP la planificación de las rutas a los robots. *5.* Solicitarle al agente AG la generación de los caminos a los robots. *6.* Mostrar en la interfaz los caminos generados. *7.* Notificarle a los agentes AP y AG cuando deben de terminar su ejecución.
AP	*1.* Gestionar las solicitudes del agente APN. *2.* Descomposición del entorno de trabajo mediante celdas cuadradas. *3.* Obtener un grafo que represente la descomposición del entorno de trabajo. *4.* Planificar la ruta para cada robot. *5.* De no poseer un servicio para el cálculo de la ruta requerida por un robot, gestionar con el Directorio Facilitador (DF) un agente que lo provea.

	6. Enviar respuesta a APN.
AG	*1.* Gestionar las solicitudes del agente APN. *2.* Cálculo de ruta auxiliar para obtención del camino. *3.* Obtener la lista de segmentos y arcos de circunferencia sobre la ruta auxiliar. *4.* Obtener los puntos de control para generar el camino. *5.* Generar el camino mediante el método β-spline. *6.* Enviar respuesta a APN.

3.1.1.5 Identificación de relaciones

El refinamiento de agentes consiste en realizar consideraciones en el soporte, descubrimiento, gestión y monitoreo de agentes. El soporte se refiere a la información necesaria para cumplir con las responsabilidades de los agentes, el descubrimiento se refiere al nombramiento único y determinado de cada agente, con su respectiva identificación y ubicación, por último la gestión y monitoreo como su nombre lo indica es realizar el seguimiento de cada agente.

Soporte: Los agentes AP y AG deben cumplir con las solicitudes del agente APN, si ocurre algún evento deben comunicárselo al agente APN.

Descubrimiento: El descubrimiento de agentes para un determinado servicio se realiza mediante el servicio de páginas amarillas de la plataforma JADE, suministrado por el agente facilitador de directorio (DF), donde cada agente ejecutado tiene un nombre único.

Gestión y monitoreo de agentes: La plataforma de agentes JADE provee una interfaz para el monitoreo de los agentes en su ciclo de vida. El agente encargado de este monitoreo es el AMS.

Dadas las responsabilidades de los agentes se determina que agentes necesitan interactuar reflejándolo en el diagrama de agentes (figura 3.3).

Las responsabilidades de interacción relacionadas entre los agentes son:

1. El agente APN solicita al agente AP la planificación de las ruta de los robots, el agente AP las planifica y le contesta.

2. El agente AP en caso de no disponer del servicio requerido para la planificación de un determinado robot, encuesta al DF preguntándole que agente lo provee, si algún agente provee el servicio el DF le contesta con su dirección, el AP lo contacta y le solicita la ejecución de dicho servicio sobre el robot que lo requiere, enviándole posteriormente la respuesta al APN.

3. El agente APN solicita al agente AG la generación del camino de los robots, el agente AG las genera y le contesta.

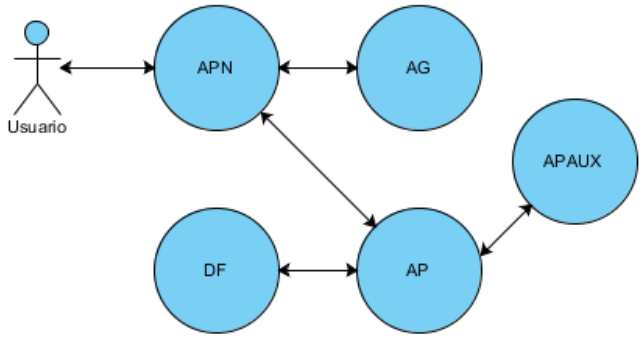


Figura 3.3 Diagrama de agentes con las relaciones.

El agente APAUX es con quien el agente AP en caso de no disponer del servicio requerido para la planificación de un determinado robot, una vez encuestado el DF preguntándole que agente lo provee se comunica para solicitárselo. Esta es una de las ventajas con las que se ha concebido la plataforma, agregarle un nuevo servicio de planificación, se reduce al desarrollo de un agente que lo provea.

En la (figura 3.4) se representa un modelo conceptual para una mejor comprensión de la solución. Mostrando la relación existente entre los diferentes elementos hasta ahora detectados. Se puede ver el agente navegador o AgentePlataformaNavegacion que controla el entorno donde se define el espacio de trabajo y se ubican los robots con sus posiciones iníciales y finales junto con los obstáculos que deberán sortear en la ejecución de su trayectoria. Por último se observa la relación de cooperación existente entre los agentes planificador y generador con el agente navegador.

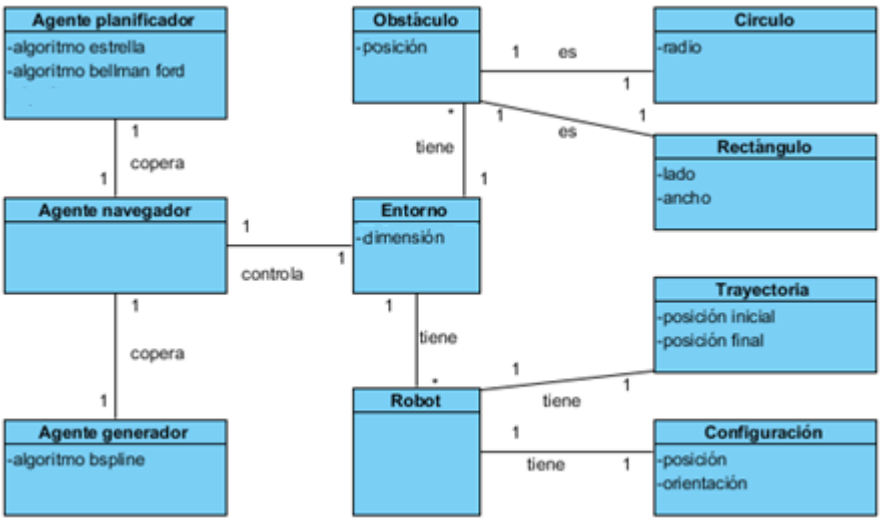


Figura 3.4 Modelo conceptual de la plataforma

3.1.1.6 Información de despliegue de los agentes

El diagrama de despliegue representa la disposición de las instancias de componentes de ejecución de los agentes en instancias de nodos, un nodo es un recurso de ejecución (tal como una computadora), como se muestra a continuación.

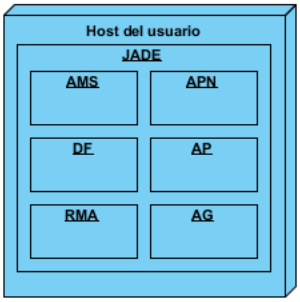


Figura 3.5 Diagrama de implementación.

En la fase de análisis fueron generados cuatro artefactos base: casos de uso, diagrama de agentes, tabla de responsabilidades y el diagrama de implementación, para la fase de diseño.

3.1.2 Fase de diseño

El diseño del software es un proceso iterativo mediante el cual los requisitos se traducen en un plano para construir el software, se representa a un nivel alto de abstracción. El diseño implementa todos los requisitos explícitos de la fase de análisis del sistema, también proporciona una imagen completa del software, enfrentándose a

los dominios de comportamiento, funcionales y de datos desde una perspectiva de implementación. Esta fase de diseño se enfoca en la plataforma JADE (Pressman, 2005), (Nikraz et al, 2006).

3.1.2.1 Especificación de interacciones

Con las responsabilidades identificadas en la (tabla 3.2), se tiene la interacción entre agentes de la plataforma JADE y los agentes APN, AP y AG que produce la (tabla 3.3) de interacción, representada por un protocolo de interacción de agentes, para cada tipo en específico. Un protocolo de interacción de agente (AIP) describe un modelo de comunicación como una secuencia de mensajes entre agentes y las restricciones del contenido de esos mensajes (Odell et al, 2000).

Tabla 3.3 Interacción del APN con AP y AG utilizando el protocolo Request.

Interacción	**Nro. de responsabilidad**	**AIP**	**Rol**	**Con el agente**	**Cuando**
1.Solicitarle la planificación de las rutas de los robot.	4	FIPA Request (petición)	Iniciar	AP	El usuario inicia la ejecución de la simulación.
2. Solicitarle la generación de los caminos de los robot.	5	FIPA Request (petición)	Iniciar	AG	Cuando el agente AP le envía los robots con la ruta planificada.
3. Notificarle a los agentes cuando deben de terminar su	7	FIPA Request (petición)	Iniciar	AP, AG	Cuando el usuario ha decidido abandonar la

ejecución.					plataforma.

Las tablas de iteración correspondientes a los agentes AP y AG se muestran en el anexo D.

3.1.2.2 Protocolos de interacción

El protocolo de interacción que utiliza el APN para comunicarse con los agentes AP y AG es especificado por la FIPA y se muestra en la (figura 3.6), donde el APN es el iniciador que envía un mensaje de tipo REQUEST (una solicitud), el agente AP es el participante que recibe el mensaje y realiza una acción determinando el tipo de respuesta que podría ser: solicitud rechazada (refuse) o solicitud acepta (agree). Si el participante acepta la solicitud debe realizar la acción para enviar el resultado de la acción con un mensaje de tipo informe (inform) o falla de operación (failure) indicando la razón. Llevando así a cabo el acto comunicativo entre los agentes iniciador y participante.

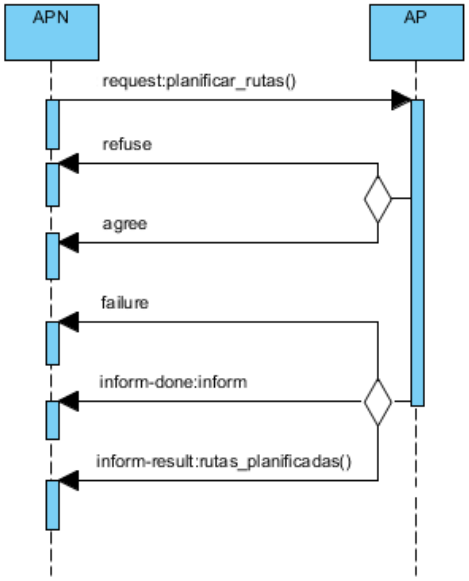


Figura 3.6 Protocolo de interacción fipa-request.

3.1.2.3 Plantillas de mensajes

En este paso, se especifica que objetos adecuados en plantillas de mensajes (MessageTemplate) son usados en los comportamientos para recibir mensajes entrantes. Las plantillas de mensajes son estructuras que validan el tipo de mensaje que se desea leer. En la (figura 3.7) se muestra el mensaje ACL del APN para el agente AP, mensajes de tipo REQUEST para iniciar el protocolo de interacción con el agente AP.

```
( REQUEST
        : sender APN
        : receiber AP
        : content (planificar_rutas(robots))
        : language FIPA-SL
        : protocol FIPA-REQUEST
)
```

Figura 3.7 Mensaje del agente APN para la planificación de rutas enviado al agente AP.

En la (figura 3.8) se muestra una porción de código de la plantilla de mensaje del agente APN que está en espera de la respuesta a la solicitud de planificar las rutas, hecha al agente AP en el mensaje mostrado en la (figura 3.7). Esta plantilla corresponde a la interacción 4 (tabla 3.3) del agente APN que solicita planificar rutas al agente AP.

```
MessageTemplate ruta = MessageTemplate.and(
        MessageTemplate.MatchPerformative(ACLMessage.INFORM),
        MessageTemplate.MatchContent("RutaPlanificada"));
```

Figura 3.8 Plantilla de mensaje del agente APN para recibir información de las rutas planificadas.

```
( INFORM-RESULT
        : sender AP
        : receiber APN
        : content (rutas_planificadas(robots))
        : language FIPA-SL
        : protocol FIPA-REQUEST
)
```

Figura 3.9 Mensaje del agente AP informando al agente APN sobre las rutas planificadas.

3.1.2.4 Descripción de registro y búsqueda

En la (figura 3.10)(A) se muestra la descripción del agente APAUX que se registra como objeto en las páginas amarillas del agente facilitador de directorio (DF).

Para obtener a los agentes proveedores de servicio *type = celdas cuadradas, name = saco* se solicita al agente DF que busque el tipo de servicio como se muestra en la (figura 3.10)(B).

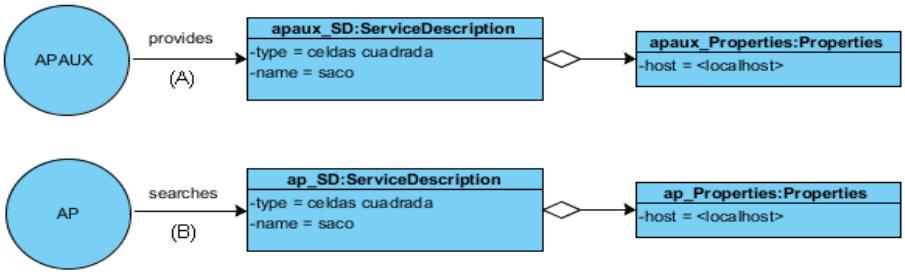


Figura 3.10 Descripción de registro y búsqueda de servicio al agente facilitador de directorio.

```
ServiceDescription sd = new ServiceDescription();
sd.setType("Celdas cuadrada");
sd.setName("Saco");
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
dfd.addServices( sd );
DFService.register( this, dfd );
```

Figura 3.11 Código correspondiente a la acción de registrar un servicio en el agente facilitador de directorio correspondiente a (figura 3.10)(A).

```
DFAgentDescription searchtemplate = new DFAgentDescription();
ServiceDescription searchtemplateSd = new ServiceDescription();
searchtemplateSd.setType(robot.getAlgPlanificacion());
searchtemplate.addServices(searchtemplateSd);
SearchConstraints sc = new SearchConstraints();
// We want to receive 1 results at most
sc.setMaxResults(new Long(1));
DFAgentDescription []results = DFService.search(myAgent, searchtemplate, sc);
```

Figura 3.12 Código correspondiente a la acción de solicitar al agente facilitador de directorio un agente que provea el servicio requerido correspondiente a la (figura 3.10)(B).

3.1.2.5 Interacción usuario-agente

En la (figura 3.13) se representan las interacciones que se producen desde que el usuario le solicita al agente APN planificar rutas hasta que se obtienen los resultados.

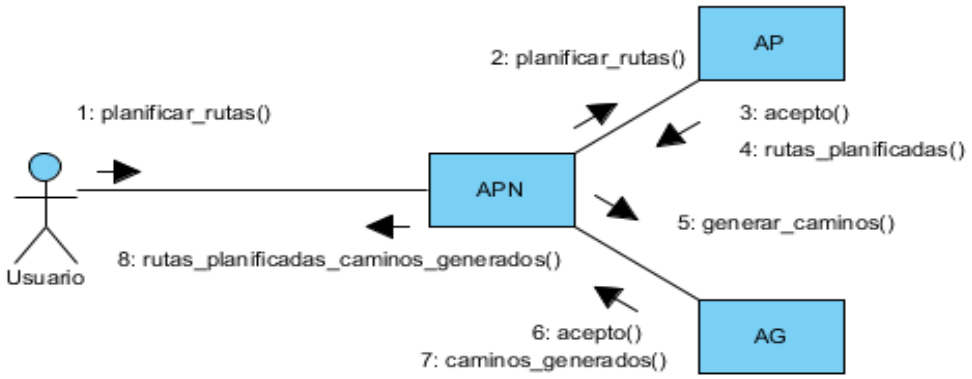


Figura 3.13 Diagrama de colaboración de los agentes.

3.1.2.6 Comportamientos internos de los agentes

Dada las responsabilidades de cada agente especificadas en la (tabla 3.2), el diagrama de relaciones de los agentes de la (figura 3.3) que se identificaron en la fase de análisis, las respectivas interacciones en la (tabla 3.3) y el anexo D se obtuvieron los comportamientos de los agentes como se muestra en las (figuras 3.14, 3.15 y 3.16).

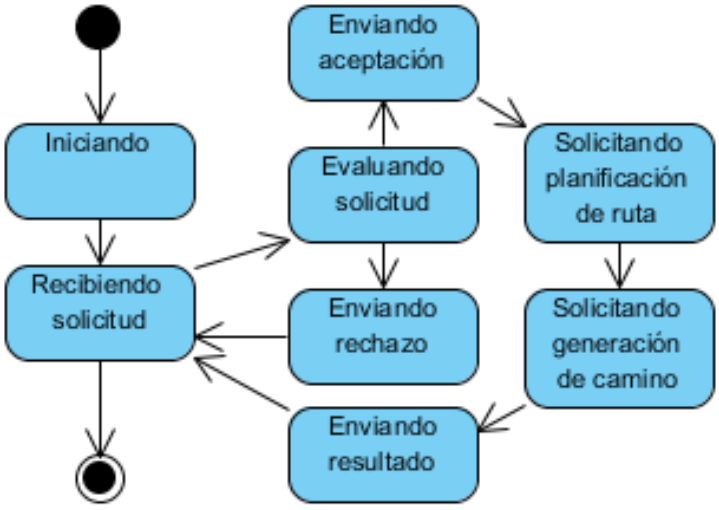


Figura 3.14 Diagrama de estados del agente APN.

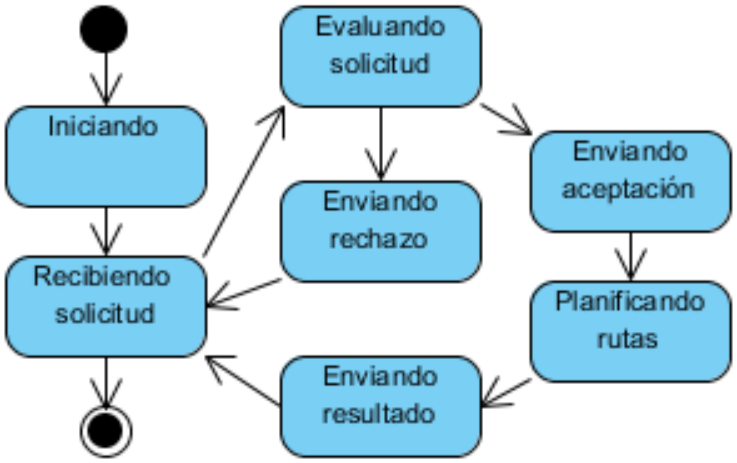


Figura 3.15 Diagrama de estados del agente AP.

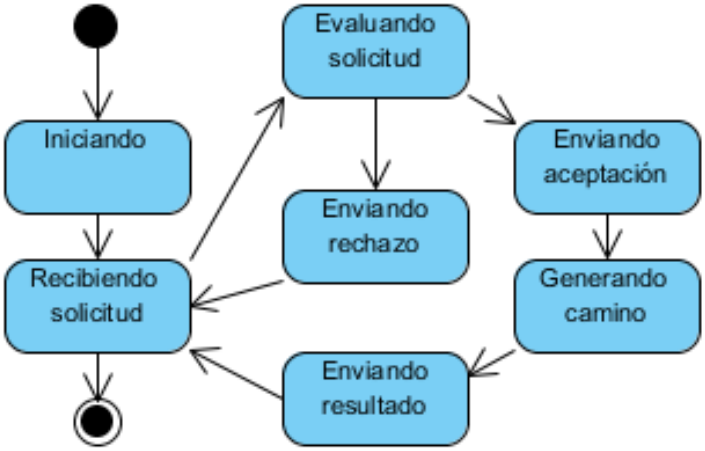


Figura 3.16 Diagrama de estados del agente AG.

3.2 JADE

Es un entorno software que simplifica el trabajo con agentes, proporcionando un *middle-ware* que cumple con las especificaciones FIPA y un conjunto de herramientas gráficas que facilita su desarrollo y depuración (Bellifemine, 2012), (Bellifemine, 2013). Está compuesto por varios paquetes Java que proporcionan al programador piezas funcionales listas para su utilización, así como interfaces abstractas adaptables a la aplicación objetivo.

Un agente JADE es una clase (*Agent*) implementada en el paquete *jade.core* que el programador debe extender adaptándola a su aplicación, es por tanto un tipo particular de objeto de Java con características específicas aportadas por JADE. En dicha clase están programadas las características básicas que permiten a cualquier agente interactuar con la plataforma, como el registro, configuración y gestión remota del agente. Además dicha clase también implementa un conjunto de métodos que pueden ser invocados para programar los comportamientos concretos propios de cada agente, como el envío y recepción de mensajes, el uso de protocolos de interacción estándar o el registro en varios dominios.

Cada funcionalidad o servicio proporcionado por el agente se programa mediante uno o más comportamientos. El agente es multitarea y sus tareas (comportamientos) se ejecutan de manera concurrente.

```
import jade.core.Agent;

public class AgentePlanificador extends Agent
{
  protected void setup()
  {
      System.out.println("Hello World. ");
      System.out.println("My GUID is "+getAID().getName());
      ........
      addBehaviour(....);
      ........
      doDelete();
  }
  protected void takeDown()
  {
       ........
  }
}
```

Figura 3.17 Declaración de agentes

Cuando un agente nace, primero ejecuta su constructor, así el agente obtiene un identificador (conforme a las especificaciones FIPA) y queda registrado en la plataforma, tras ello ejecuta su método *setup()*. Es en este punto donde empieza la actividad del agente específica a la aplicación. El programador debe añadir en este método los comportamientos necesarios (mediante *addBehaviour(Behaviour)* que el agente deba ejecutar, así como, en caso de ser necesario, modificar la información de registro en el AMS, proporcionar una descripción de sus servicios o registrarse en uno o varios dominios. Tras la ejecución de *setup()* se ejecuta el primer comportamiento de la cola de tareas listas, un planificador round-robin interno al agente y no expulsivo

oculto al programador gestiona la planificación de dichos comportamientos. Los métodos *addBehaviour(Behaviour)* y *removeBehaviour(Behaviour)* permiten gestionar la cola de tareas posteriormente.

Cualquier comportamiento puede llamar al método *Agent.doDelete()* para finalizar la ejecución del agente. Cuando un agente va a ser eliminado, ejecuta el método *takeDown()*. Este método debe ser sobrescrito por el programador para que realice las operaciones de limpieza necesarias antes de eliminar el agente, como la baja del agente en el *Directory Facilitator* (DF). Una vez finalizado este método, el agente sería dado de baja en el AMS y su hilo de ejecución sería destruido. La (figura 3.18) muestra el hilo de ejecución de un agente JADE.

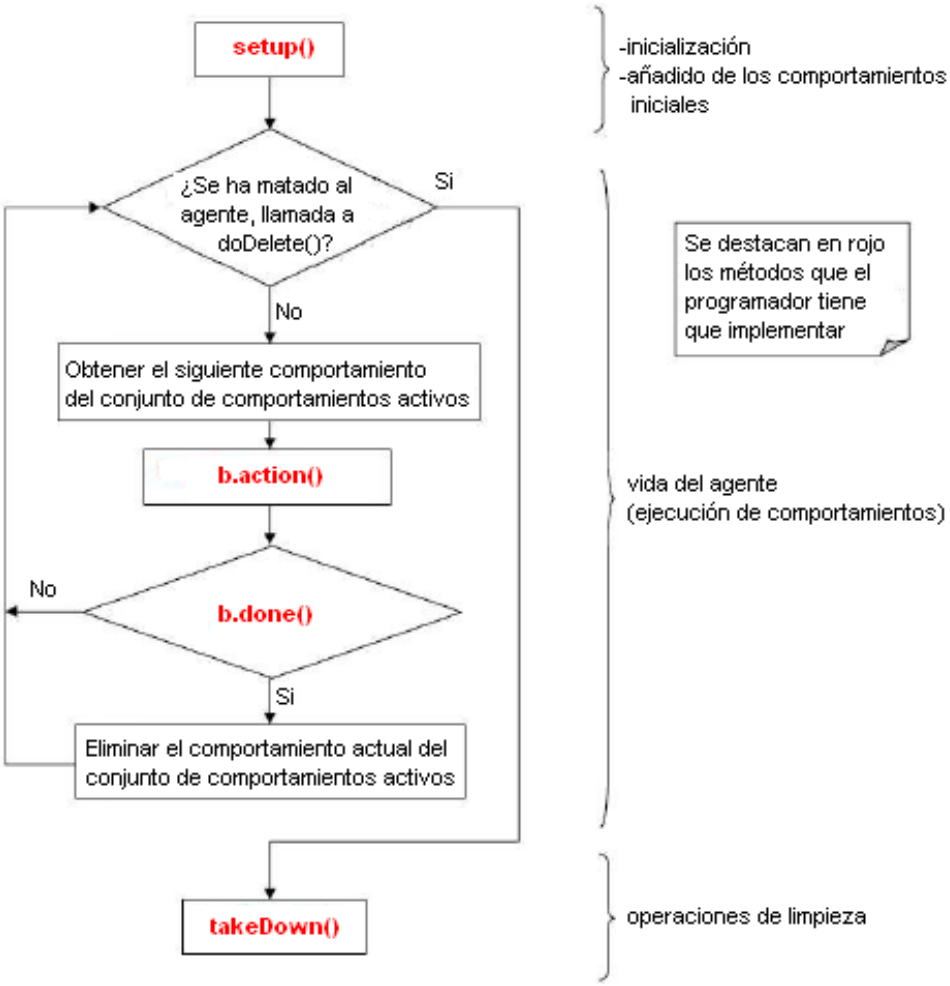


Figura 3.18.Hilo de ejecución de un agente JADE.

3.2.1 Comportamientos de agentes jade

El agente JADE no tiene un comportamiento ni reactivo ni proactivo, y no está dirigido ni por eventos ni por objetivos, no al menos de manera conceptual. Tampoco se puede hablar de actitudes mentales más allá de sus comportamientos. JADE ofrece una serie de encapsulados y clases que implementan funcionalidades básicas del agente.

Los comportamientos son la forma elegida por JADE para que un agente adquiera personalidad propia. JADE proporciona una familia de clases adecuada para definir distintos tipos de comportamientos. El programador deberá extender algunas de estas clases para generar los comportamientos deseados específicos de su aplicación. La (figura 3.19) muestra la jerarquía de clases de los comportamientos que proporciona.

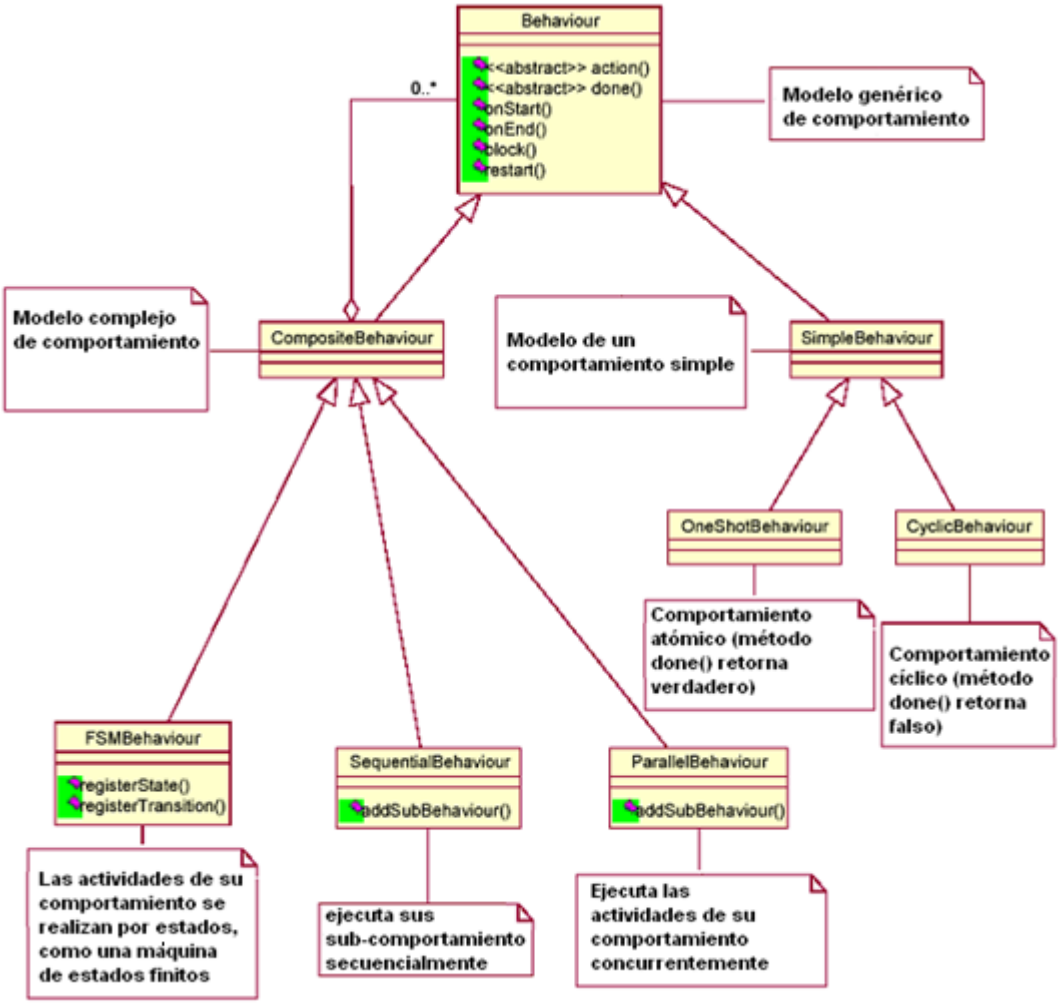


Figura 3.19 Jerarquía de Comportamientos de JADE

Cuando un comportamiento es elegido para su ejecución llama a su método *action()*, que es ejecutado hasta retornar. Si la tarea no es completada es replanificada para un siguiente ciclo de ejecución. El método *done()* es quien determina si la tarea ha sido completada o no y por tanto es llamado cada vez que se retorna de *action()*. Debido a que el planificador del agente no es expulsivo, el programador debe evitar bucles infinitos e incluso operaciones que puedan ser demasiado largas para evitar la inanición del resto de los comportamientos.

Por defecto un agente es un hilo, con lo que si un comportamiento llama a *doWait()* de la clase *Agent* está provocando la suspensión de todo el agente. Si solo se desea bloquear un único comportamiento, el método *block()* coloca el comportamiento en una cola de bloqueados una vez se vuelve de *action()*, sin detener la ejecución de todo el agente. Un comportamiento bloqueado podrá continuar su ejecución al recibir el agente un mensaje; si se definió, al cumplirse un plazo asociado al bloqueo o al invocar al método *restar()* del comportamiento.

Los comportamientos también incluyen dos métodos que pueden ser sobrescritos por el programador en caso de que sea necesario realizar algunas acciones antes o después de la ejecución del comportamiento. Estos métodos son *onStart()* y *onEnd().*

Los métodos anteriormente vistos (*action()*, *done()*, *onStart()*, *onEnd()*, *block()* y *restart()*) son definidos por la clase abstracta *Behaviour*. Las clases que hereden de *Behaviour* pueden redefinirlos para adaptarlos a sus necesidades.

La (figura 3.19) muestra la especialización de *Behaviour()* en *SimpleBehaviou* y en *CompositeBehaviour()*. Los comportamientos simples (los primeros) son aquellos que pueden ser considerados como atómicos, que no se dividen en pasos. JADE ofrece las siguientes especializaciones:

- *OneShotBehaviour()* es una clase abstracta de comportamientos que solo pueden ser ejecutados una sola vez y que no pueden ser bloqueados. Su método *done()* siempre devuelve *true*.
- *CyclicBehaviour()* es una clase abstracta de comportamientos que deben ser ejecutados siempre. Su método *done()* siempre devuelve *false*.

Para los comportamientos más complejos se usan las clases ofrecidas por *CompositeBehaviour*. Estos comportamientos se componen de otros comportamientos

hijos, que son quienes ejecutan las operaciones del comportamiento. JADE ofrece las siguientes especializaciones:

- *SequentialBehaviour* clase que ejecuta sus sub-comportamientos de manera secuencial. Resulta útil cuando una tarea compleja puede ser expresada como una secuencia de pasos atómicos. Su método *addSubBehaviour()* permite añadir los distintos sub-comportamientos.
- *ParallelBehaviour* ejecuta sus sub-comportamientos de manera concurrente y termina cuando se alcanza determinada condición en sus sub-comportamientos, cuando todos o un número determinado de ellos terminan.
- *FSMBehaviour* es una clase que ejecuta sus hijos conforme a un autómata finito. Cada uno de sus comportamientos se define como un estado mediante *regiter-State*. Además, también se definen las transiciones válidas entre estados usando *registerTransition*. El valor devuelto a través del método *onEnd()* por cada estado que se ejecuta determina la transición a recorrer en el siguiente ciclo en que el comportamiento se active. La terminación de alguno de los hijos declarado como final supone la terminación del comportamiento.

Además de estos comportamientos, JADE también ofrece el *WakerBehaviour* que dispara una tarea de un solo disparo que debe ser ejecutada tras pasar un determinado período de tiempo; y el *TickerBehaviour* que implementa tareas cíclicas que deben ser ejecutadas periódicamente.

3.2.2 Herramientas para el desarrollo y depuración de jade

Un pilar importante de JADE es el conjunto de herramientas que proporciona para el desarrollo y depuración de los sistemas de agentes. En esta sección se detallan de forma concisa esas características. Los distintos subpaquetes de *jade.tools* contienen las herramientas proporcionadas por la plataforma para su administración y desarrollo de aplicaciones:

- El *Remote Management Agent* o *RMA* es una consola gráfica para la gestión y control de la plataforma. Es posible activar varios RMAs y todos ellos mantienen la coherencia entre sí haciendo un *multicasting* al resto de los eventos que

ocurren en cada instancia. El RMA también permite activar otras herramientas de JADE.

- El *Dummy Agent* es una herramienta para la monitorización y depurado del sistema. Se compone de una interfaz gráfica y de un agente JADE. A través de su interfaz es posible componer mensajes ACL y enviarlos a otros agentes. También es posible visualizar una lista de los mensajes enviados y recibidos, con una marca temporal que permite el registro y simulación de conversaciones.
- El agente *Sniffer* es capaz de interceptar los mensajes ACL durante su transmisión, mostrándolos gráficamente mediante diagramas de secuencias similares a los de UML.
- El *Introspector* es un agente que permite la monitorización del ciclo de vida de un agente, de los mensajes ACL que intercambia y de los comportamientos que ejecuta.
- El *DF GUI* es una interfaz gráfica usada por el *Directory Facilitator* por defecto de JADE y por cualquier otro DF que el usuario necesite, de manera que se pueda crear una red de dominios y subdominios de páginas amarillas. Esta interfaz permite el control del conocimiento de un DF, mediante la federación de DFs y el acceso remoto (registros y cancelación, modificación y búsqueda) a la base de conocimiento de los DFs padre e hijos.
- El *LogManagerAgent* es un agente que permite registrar la información en un *log*, tanto para JADE como para las clases que usen *Java Logging*.

3.3 Plataforma para la navegación

En esta solución se implementaron tres agentes: AgentePlataformaNavegacion, AgentePlanificador, AgenteGenerador.

En el agente AgentePlataformaNavegacion encargado del control de la plataforma, así como de la interfaz que se muestra en la (figura 3.20), se implementaron tres comportamientos: un comportamiento de tipo OneShotBehaviour, encargado de enviarle la solicitud planificar rutas al agente AgentePlanificador una vez el usuario inicia la simulación, un comportamiento CyclicBehaviour que estará a la espera de la respuesta del agente AgentePlanificador con las rutas planificadas. Para este

comportamiento se creó una plantilla para filtrar los mensajes que en su contenido sean rutas planificadas.

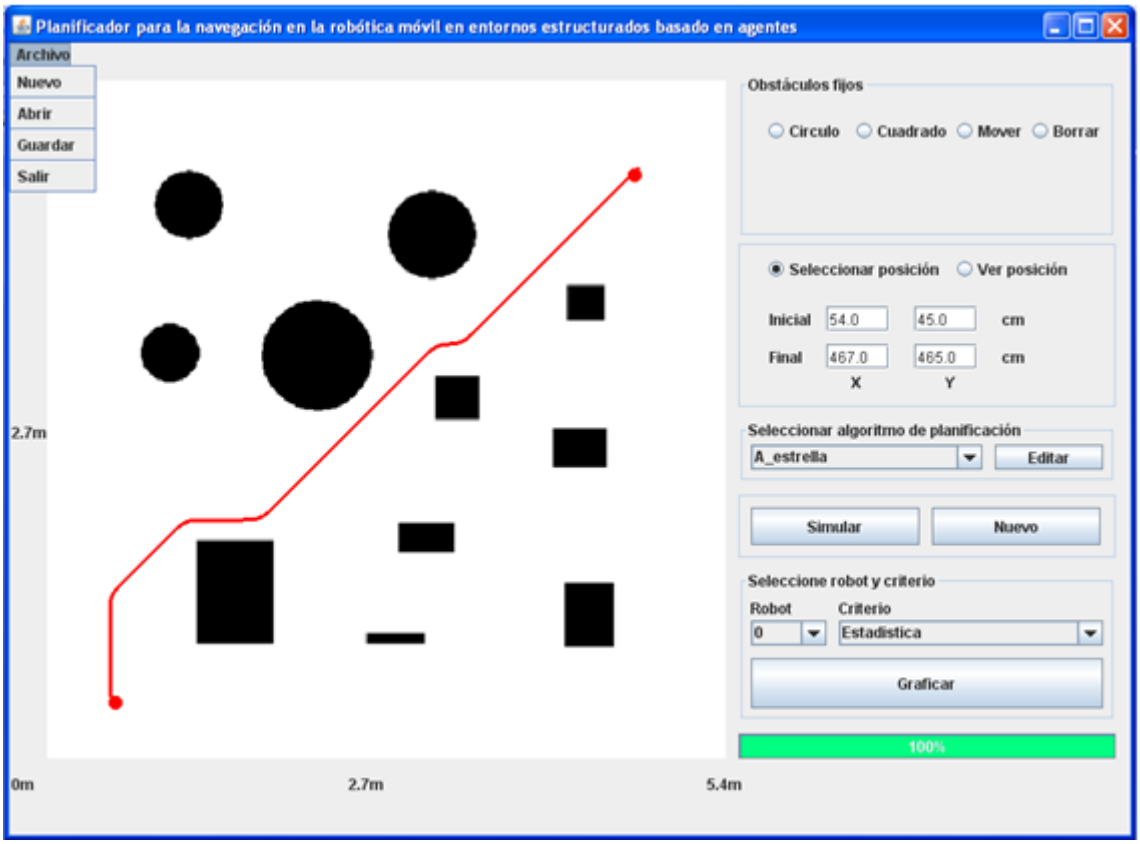


Figura 3.20 Interfaz principal del planificador

En el agente AgentePlanificador se implementó un comportamiento de tipo CyclicBehaviour que procesa las solicitudes de planificar rutas, enviadas por el agente AgentePlataformaNavegacion. Para este comportamiento se conformó una plantilla para filtrar los mensajes que en su contenido sean planificar rutas.

En el agente AgenteGenerador se definió un comportamiento de tipo CyclicBehaviour encargado de procesar las solicitudes de generar caminos, enviadas por el agente AgentePlataformaNavegacion. Para este comportamiento se creó una plantilla para filtrar los mensajes que en su contenido sean generar caminos.

En la interfaz se observa un entorno definido y el camino generado desde una posición inicial hasta su meta. Además en el menú archivo se gestiona el entorno y las diferentes acciones que se pueden realizar sobre él. En obstáculos fijos se selecciona el tipo de obstáculo a ubicar en el entorno, así como las acciones a realizar sobre él,

desplazarlo o borrarlo en caso de ser necesario. Permite seleccionar la posición donde ubicar al robot, seleccionar el algoritmo de planificación, iniciar la simulación, así como, seleccionado un robot y una métrica para obtener gráficas que visualicen algunos parámetros de desempeño de los algoritmos de planificación.

3.3.1 Métricas

Dentro de los parámetros que se tienen en cuenta para evaluar los algoritmos de cálculo de ruta se encuentran: *Plenitud,* que es la garantía de encontrar la solución si existe. *Optimización*, que es la garantía de encontrar la solución de menor costo, y *Complejidad tiempo/espacio*, que es el tiempo y la memoria usada por el algoritmo, visto como la cantidad de estados explorados dentro del grafo. Basados en estos parámetros se determinaron los aspectos a destacar por cada algoritmo en las gráficas que se pueden obtener de la ejecución de los mismos.

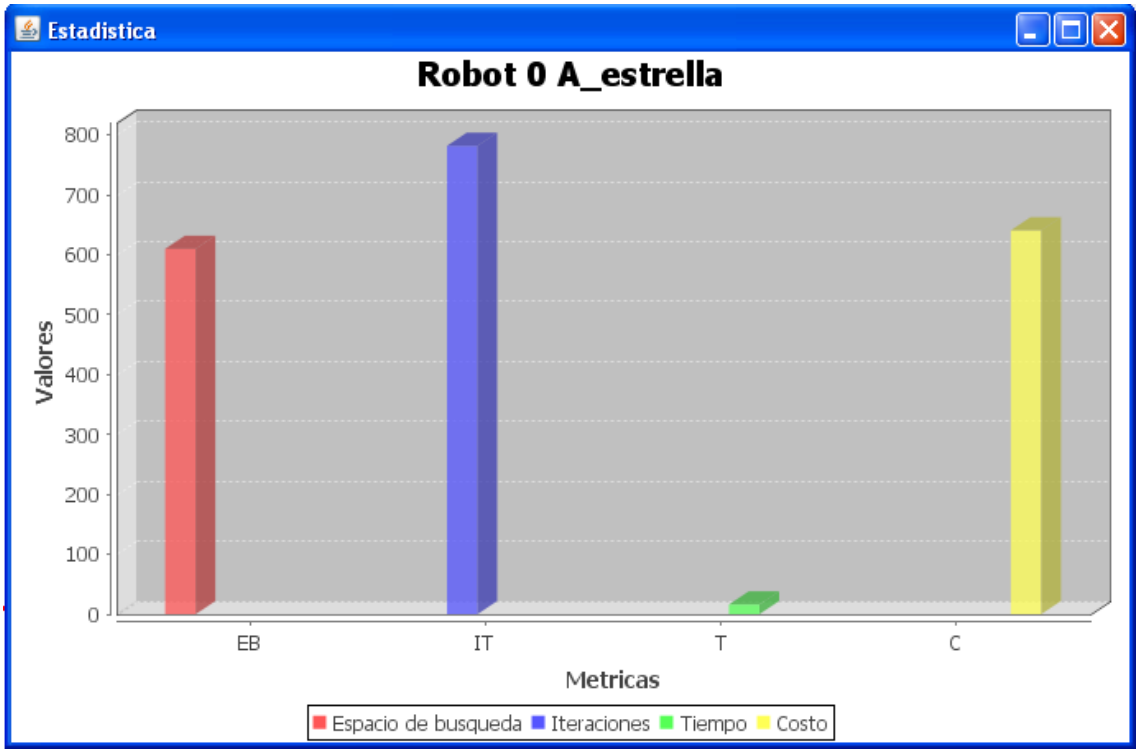


Figura 3.21 Parámetros de los algoritmos

En la gráfica se observan las métricas (EB) espacio de búsqueda, (IT) iteraciones del algoritmo para el cálculo de la ruta, (T) tiempo de ejecución del algoritmo, (C) costo de recorrido de la ruta mencionada anteriormente.

3.3.2 Caso de estudio

Para la realización de este caso de estudio el entorno escogido se muestra en la (figura 3.22), con una dimensión de 540x540 píxel. En la representación usada del entorno por el planificador (descomposición mediante celdas cuadradas) donde cada celda tiene una dimensión de 20x20 píxel, el entorno se descompone en un total de 729 celdas de las cuales (119) están ocupadas por obstáculos, el espacio libre está compuesto por (610) celdas.

Los robots fueron ubicados en la posición inicial (465,452) y final (56,45) con el objetivo de obtener una ruta primero con el algoritmo A_estrella, luego con Bellman-Ford y por último con SACO.

Estos algoritmos han sido validados por medio de simulaciones y experimentos en trabajos anteriores, Bellman-Ford en (Torres, 2010), (Torres y Moreno, 2010), (Torres y Moreno, 2011), A_estrella en (Liu and Ramakrishnan, 2001), (Stentz, 1994), (Nilsson, 1980), (Murphy, 2000), comprobándose en dichas referencias bibliográficas la validez de los mismos en cuanto a la optimización que hacen del camino a recorrer.

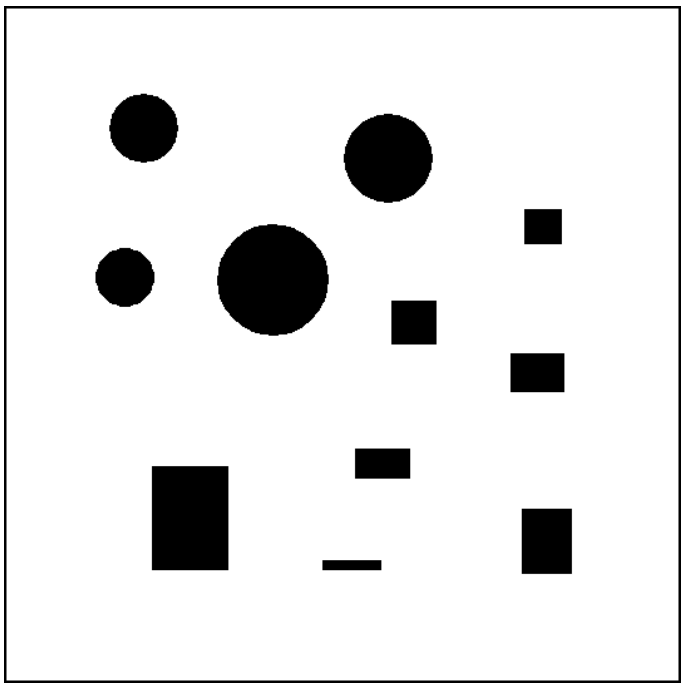


Figura 3.22 Entorno seleccionado para el caso de estudio

La pruebas se realizaron en una laptop con Sistema Operativo Windows XP, 1GB de memoria Ram, Procesador: Intel(R) Core Duo a 1.86 GHz.

En la (figura 3.23)(A) se representa el camino generado a partir de la ruta obtenida mediante el algoritmo A_estrella, en la (figura 3.23)(B) el camino generado a partir de la ruta obtenida mediante el algoritmo Bellman-Ford y en la (figura 3.23)(C) el camino generado a partir de la ruta obtenida mediante el algoritmo SACO.

Aunque a la vista los caminos generados no parecen iguales tienen el mismo costo, es decir, la suma de la distancia de los puntos encontrados en la planificación en todos los algoritmos son iguales. El espacio de búsquedas o las posiciones que pueden ser ocupadas por el robot, son las mismas para los algoritmos.

Un detalle que se puede percibir en la (figura 3.23) es el número de curvas encontradas, para iguales trayectos los caminos con menor cantidad de curvas son más óptimos, esto se traduce en menos esfuerzo de los accionamientos del robot al describir la trayectoria.

Las principales diferencias se destacan en cuanto a tiempo de ejecución y el número de iteraciones para calcular la ruta como se muestra en la (figura 3.24, 3.25, 3.26).

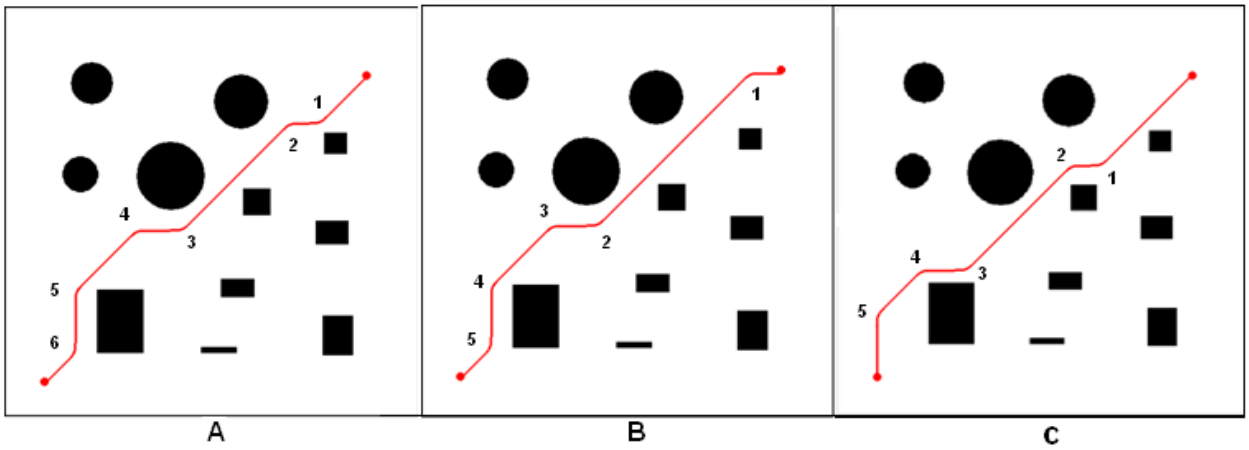


Figura 3.23 Camino generado a partir de la ruta planificada (A) con el algoritmo A_estrella, (B) con el algoritmo Bellman-Ford y (C) con el algoritmo SACO.

Las gráficas de barra que se muestran a continuación brindan una medida de los parámetros obtenidos en su ejecución para cada uno de los algoritmos.

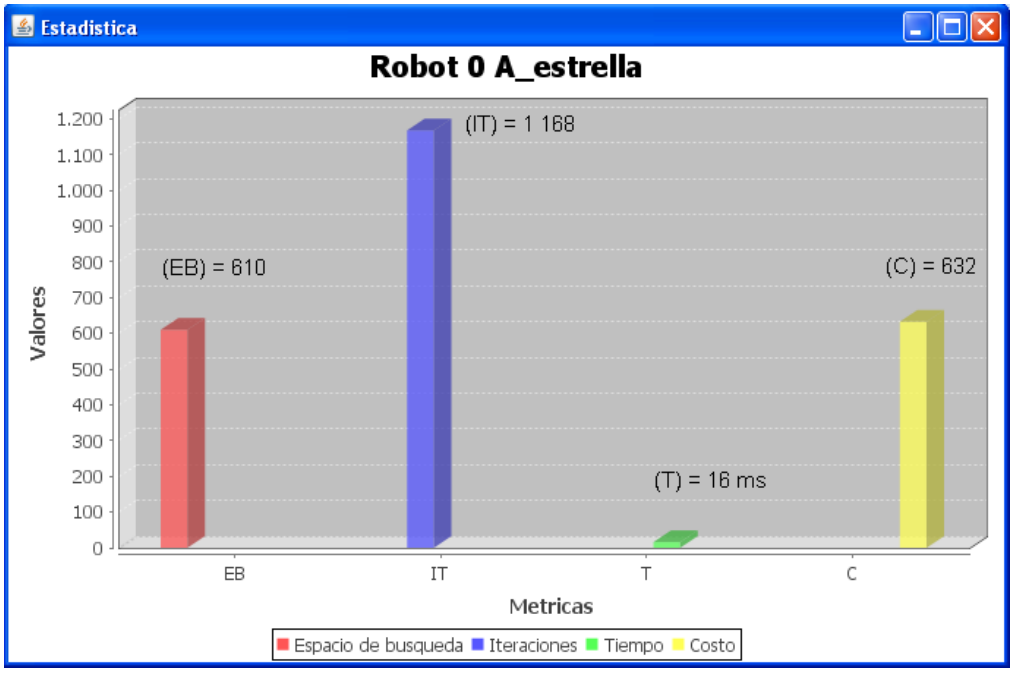


Figura 3.24 Parámetros de la ruta obtenida mediante A_estrella

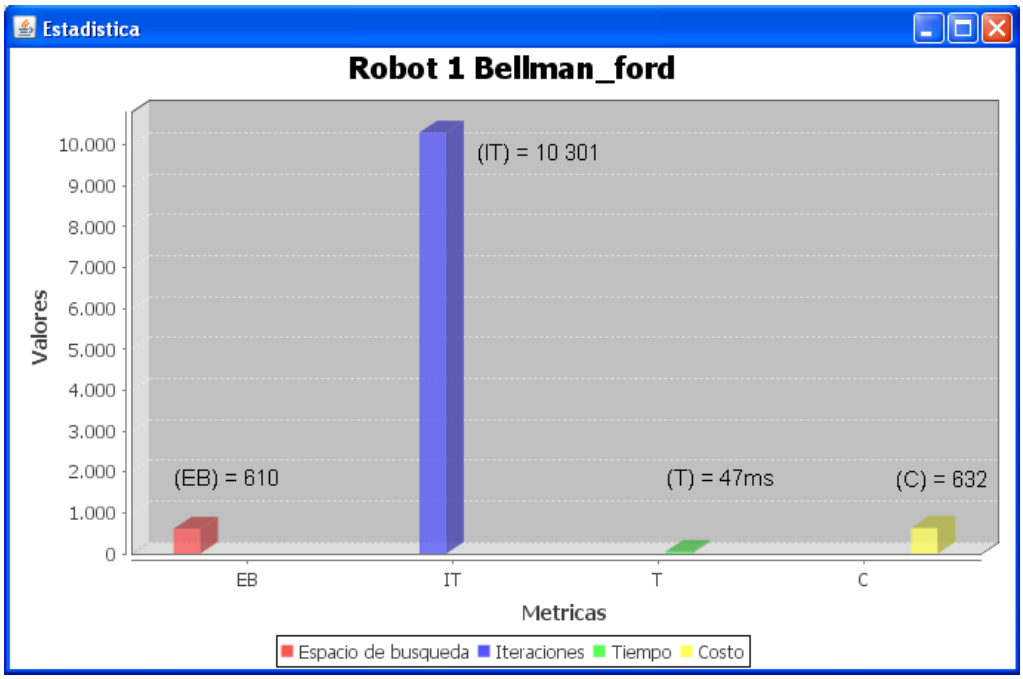


Figura 3.25 Parámetros de la ruta obtenida mediante Bellman-Ford

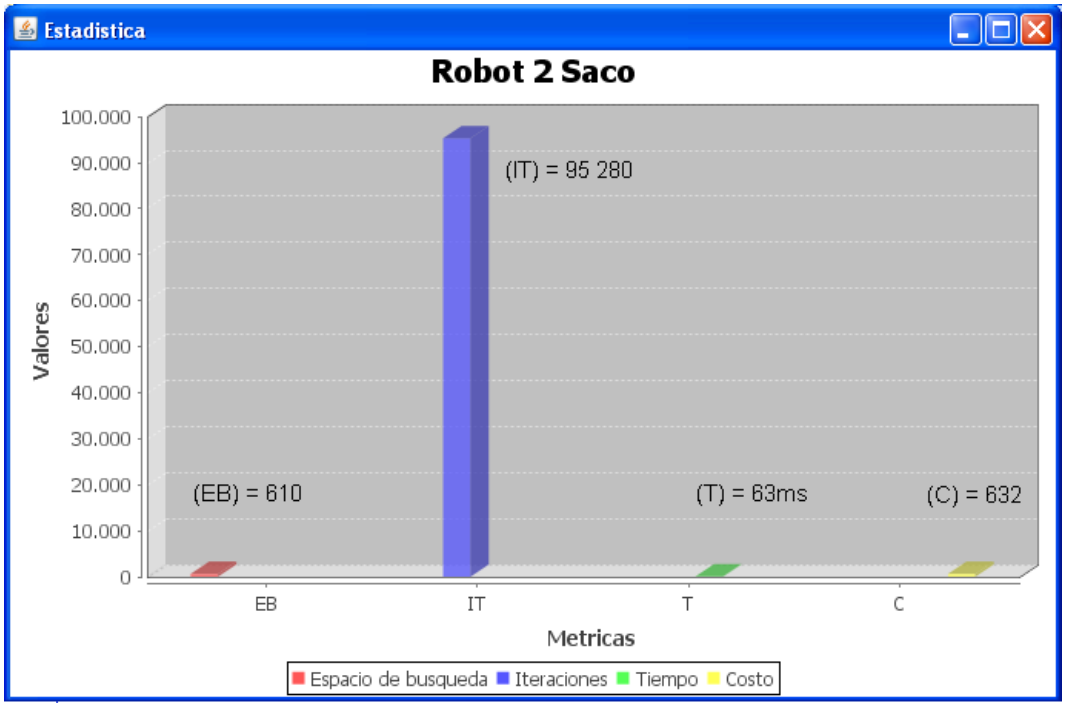


Figura 3.26 Parámetros de la ruta obtenida mediante SACO

En la (tabla 3.4) se observa que la principal diferencia entre estos algoritmos radica en el número de iteraciones necesarias para calcular las rutas y el tiempo de ejecución. El número de iteraciones está relacionado con los estados visitados.

Para este entorno teniendo en cuenta la cantidad de iteraciones en el cálculo de rutas y el tiempo de ejecución se concluye que el algoritmo A_estrella ofrece un mejor desempeño.

Tabla 3.4 Parámetros de ejecución de los algoritmos

Algoritmo \ parámetro	EB	IT	T	C
A_estrella	610	1 168	16 ms	632
Bellman-Ford	610	10 301	47 ms	632
SACO	610	95 280	63 ms	632

3.3.3 Agregar un nuevo algoritmo

Una de las facilidades que brinda la plataforma, debido a que está basada en agentes, es la de agregar un nuevo algoritmo de planificación. En el RF4 además de la posibilidad de seleccionar el algoritmo de planificación, el botón editar permite agregar

un nuevo algoritmo. Para hacerlo solo hay que programar un nuevo agente que implemente el algoritmo deseado. Este nuevo agente recibe una estructura de dato de tipo robot y establece en los campos involucrados en la etapa de planificación los datos requeridos de dicha ejecución: ruta planificada, iteraciones para calcular la ruta, tiempo y costo. Este agente al ser iniciado en la plataforma declara al directorio facilitador (DF) los servicios que él provee, permitiendo que los agentes que requieran de ese servicio lo puedan contactar.

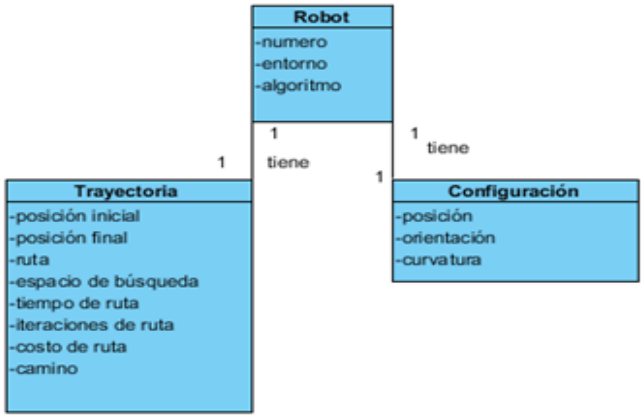


Figura 3.27 Estructura de un robot

Como se puede apreciar en la (figura 3.28), se muestra la secuencia de acciones para agregar un nuevo algoritmo. Al presionar el botón editar permite introducir el nombre del algoritmo que se desea agregar, que será el mismo del servicio que provee el agente que lo implementa. Una vez realizadas estas acciones se inicia el agente en la plataforma JADE y está disponible para ser usado.

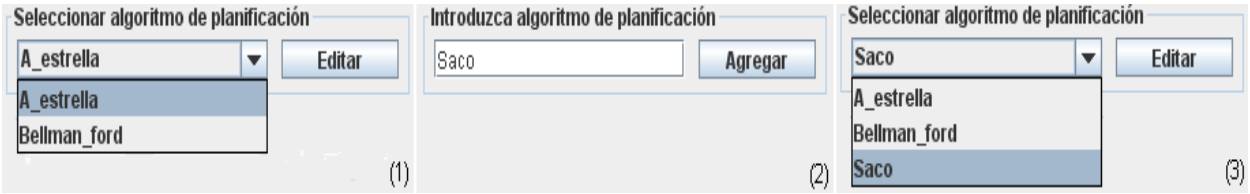


Figura 3.28 Acciones para agregar un algoritmo

Se puede observar en la (figura 3.29) el flujo de intercambio de mensajes. Cuando se inicia la simulación el agente PlataformaDeNavegacion que controla la plataforma le envía al agente Planificador el robot a planificar su ruta (1), él no provee el servicio para

dicho algoritmo, entonces pregunta al DF si algún agente provee el servicio que implementa dicho algoritmo (2).

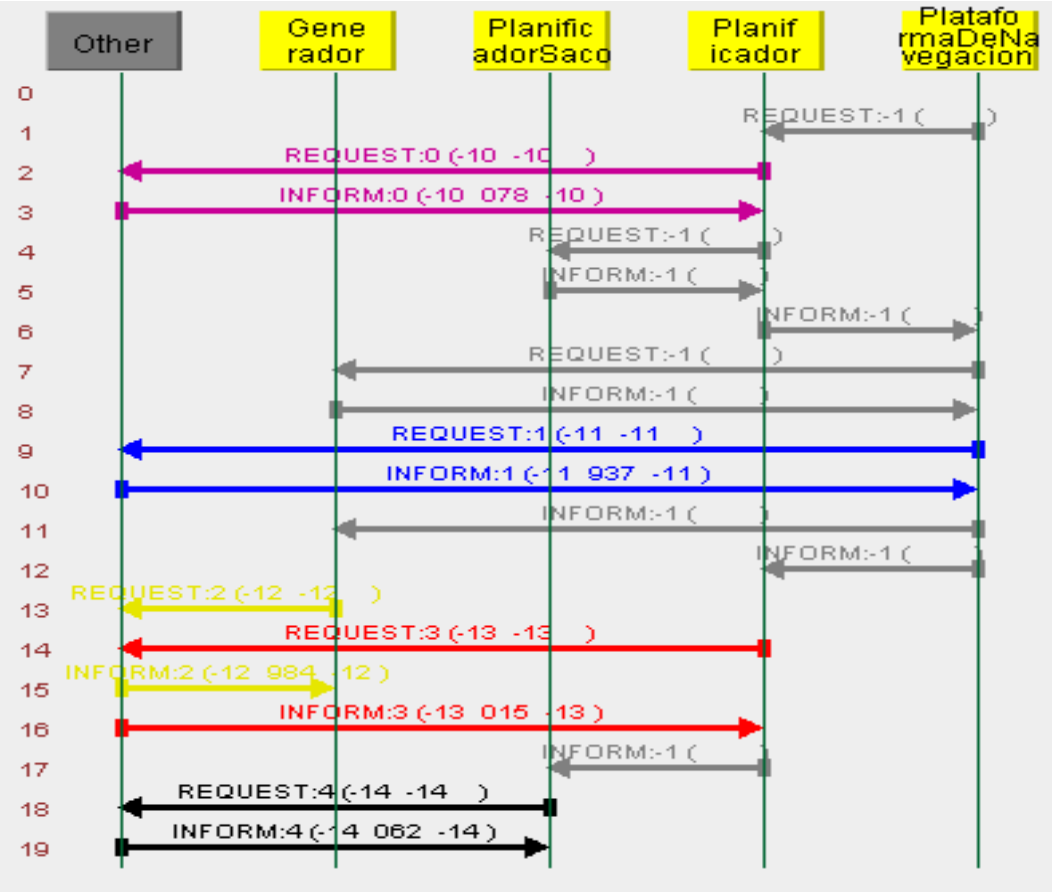


Figura 3.29 Intercambio de mensajes cuando se agrega un nuevo agente

El DF contesta que el agente PlanificadorSaco lo provee (3), entonces el planificador le solicita al agente PlanificadorSaco que le planifique el robot con dicho algoritmo (4), este lo planifica y le contesta en (5), el agente Planificador recibe la información y le notifica al agente PlataformaDeNavegacion en (6), el agente PlataformaDeNavegacion le solicita al agente Generador que genere el camino en (7), este le contesta en (8) con el camino generado. Si se sale de la plataforma el agente PlataformaDeNavegacion le notifica a los agentes Generador y Planificado en (11) y (12) para que terminen de ejecutarse. Se puede ver como el agente Planificador que es quien conoce que existe el agente PlanificadorSaco le notifica en (14) de que él terminará de ejecutarse, él puede tomar la decisión de terminar o seguir.

3.4 Pruebas de software

Las pruebas de software son un instrumento para determinar el status de la calidad de un software. Tienen como objetivo, además de descubrir errores, medir el grado en que el software cumple con los requerimientos definidos.

3.4.1 Pruebas de Caja Negra

Las pruebas de caja negra se aplican a la interfaz del software y se centran en los requisitos funcionales del sistema, permitiendo derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa.
Para realizarle las pruebas de caja negra a la plataforma, se elaboraron dos Casos de pruebas **Anexo E** donde se abarcan cada uno de los RF descritos en la sección **3.1.1.1** de este documento. Dicho caso de prueba fue evaluado por el equipo de investigación de robótica móvil del Departamento de Automática del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE).

3.4.2 Pruebas de Caja Blanca

Las pruebas de caja blanca se basan en el minucioso examen de los detalles procedimentales, se comprueban los caminos lógicos del software proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y bucles. Todo lo que se tiene que hacer es definir todos los caminos lógicos, desarrollar casos de pruebas que los ejerciten y evaluar los resultados. Para esto, primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
public void aEstrella(Robot robot)
{
 NodoAEstrella actual = null
 Vertice vInicio, vFin;
 vInicio = this.perteneceVerticeGrafo(new Vertice(robot.getPosInicial()));
 vFin = this.perteneceVerticeGrafo(new Vertice(robot.getPosFinal())); (1)
 if (vInicio != null && vFin != null) (2)
 {
  boolean ruta = true;
  actual = new NodoAEstrella (vInicio, null, 0, distanciaH(vInicio, vFin));
  abierto.add(actual);
  LinkedList<Vertice> vertAdyActual;(3)
  Vertice vertAdy;
  NodoAEstrella perteneceAbierto;
  do
  {
    actual = verticeConMenorF();
    abierto.remove(actual); (4)
    cerrado.add(actual);
    if (actual.getVerticeNodoAEstrella() == vFin) (5)
    {
     ruta = false;(6)
    }
    else
    {
     vertAdyActual = actual.getVerticeNodoAEstrella().listaVertAdy() (7)
     while (!vertAdyActual.isEmpty()) (8)
     {
      vertAdy =  vertAdyActual.removeFirst();(9)
      perteneceAbierto = perteneceAAbierto(vertAdy);
      if (perteneceAbierto == null && perteneceACerrado(vertAdy)== false)(10)
      {
       perteneceAbierto = new NodoAEstrella (vertAdy, null, 0, 0);
       setEcuacionFGH(actual, perteneceAbierto, vFin); (11)
       abierto.add(perteneceAbierto);
      }
      else
      {
       if (perteneceAbierto != null) (12)
       {
         if (perteneceAbierto.getGNodoAEstrella() > (actual.getGNodoAEstrella() + 1)) (13)
         {
           setEcuacionFGH(actual, perteneceAbierto, vFin); (14)
         }
       }
      } (15)
    }
  }
  while (!abierto.isEmpty() && ruta). (16)
  }
  obtenerRuta(robot, actual);(17)
}
```

Figura 3.30 Código fuente del algoritmo A*

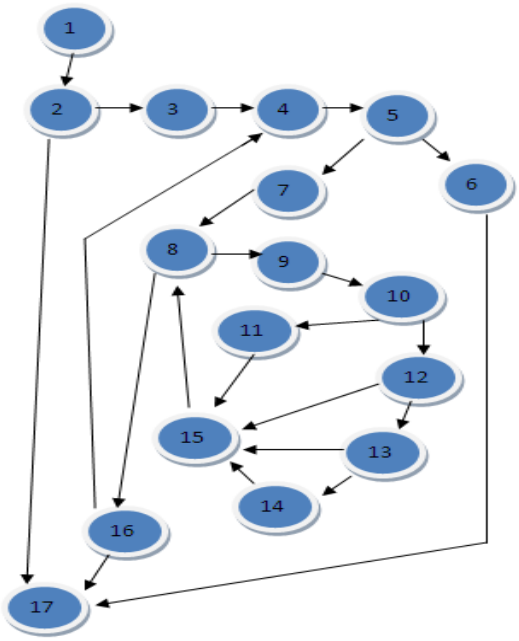


Figura 3.31 Grafo de flujo del algoritmo A*

Luego de construido el grafo, se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = (A – N) + 2

V (G) = (22 - 17) + 2

V (G) = 7

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas), **P** = 2-5-10-12-15-16.

V (G) = P + 1

V (G) = 6 + 1

V (G) = 7

Siendo “R” la cantidad total de regiones, para cada fórmula “V (G)” representa el valor del cálculo.

V (G) = R

V (G) = 7

El cálculo realizado mediante las tres fórmulas arriba al mismo valor, dando como resultado 7, lo que indica que existen 7 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Para este caso en específico, en dependencia de la longitud de la solución se puede observar que los nodos 4-*-16 y 8-*-15 representan ciclos por lo que se puede concluir que las posibles soluciones están enmarcada en los siguientes caminos básicos por los que puede recorrer el flujo:

- **Camino básico 1:** 1-2-17
- **Camino básico 2:** 1-2-3-4-*-6-17
- **Camino básico 3:** 1-2-3-4-5-*-16-17

Caso de prueba para el camino básico 1:

Descripción: Este flujo es para cuando no se ha seleccionado una posición inicial y final a alcanzar por el robot.

Condición de ejecución: Los valores obtenidos no deben ser nulos.

Obtiene: Se obtiene un mensaje que especifica que faltan parámetros.

Resultados esperados: Una ruta.

Caso de prueba para el camino básico 2:

Descripción: En caso de que se obtenga una solución desde la posición inicial hasta la final.

Condición de ejecución: Los valores obtenidos no deben ser nulos.

Obtiene: Una ruta.

Resultados esperados: Una ruta.

Caso de prueba para el camino básico 3:

Descripción: En caso de que se explore el espacio y no se obtenga una solución desde la posición inicial hasta la final.

Condición de ejecución: Los valores obtenidos no deben ser nulos.

Obtiene: Una ruta vacía, nula y se le notifica al usuario.

Resultados esperados: Una ruta.

Luego de aplicados los distintos casos de pruebas, se comprobó que el flujo de trabajo del algoritmo es correcto porque cumple con las condiciones necesarias que se habían planteado.

3.5 Conclusiones del capítulo

- En el desarrollo de la plataforma se muestra un modelo conceptual que especifica la relación existente entre los principales elementos identificados, garantizando un mayor entendimiento de la solución propuesta.
- La plataforma desarrollada cumple con los requisitos funcionales deseables para la gestión del entorno, ubicación de los robots, selección de los algoritmos de cálculo de ruta, y la muestra de estos parámetros mediante gráficas facilitan el análisis de los mismos.
- La solución fue validada mediante los casos de prueba del anexo e.
- En el caso de estudio se puede apreciar una comparación de los parámetros: iteraciones de la ruta, tiempo de ejecución, costo total que se analizan entre los algoritmos Bellman-Ford, A_estrella y SACO, concluyendo que el A_estrella ofrece mejores resultados.

CONCLUSIONES

Se creó un espacio de trabajo integrado a una plataforma basada en agentes donde se representan los robots en interacción con los obstáculos del entorno.

Se implementó el generador de caminos a partir de la ruta obtenida por el planificador, coordinándose mediante agentes de la plataforma.

Se visualizan las métricas principales que se solicitaban como requisitos y que permiten evaluar el desempeño de los algoritmos de planificación utilizados.

Teniendo en cuenta el caso de estudio analizado, con el entorno propuesto y bajo los parámetros especificados, se puede determinar que bajo esas condiciones el algoritmo A* arroja mejores resultados.

Esta plataforma constituye una herramienta didáctica para el grupo de investigación en la robótica móvil en el departamento de automática del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE).

Se recomienda extender el trabajo para facilitar la evaluación de los algoritmos de planificación en entornos dinámicos.

Los objetivos planteados al inició del trabajo se han cumplido. Se obtuvo una plataforma basada en agentes que permite realizar pruebas a algoritmos de planificación para espacios de trabajo estructurados.

BIBLIOGRAFÍA

(Alfonso, 2004)-Alfonso F.(2004). “Algoritmos genéticos aplicados al planeamiento de trayectorias de robots móviles. modelamiento y simulación”. Universidad industrial de Santander.

(Blanco Rodríguez, 2003) Blanco Rodríguez, F.J. (2003). Cálculo del C-espacio de un robot móvil: convolución jerárquica. PhD thesis. Universidad de Salamanca. Departamento de Informática y Automática.

(Bellifemine, 1999)-Bellifemine, F.; Poggi, A.; y Rimassa, G. Jade: A fipacompliant agent framework. In Proc. of PAAM, 97–108. 1999.

(Bellifemine, 2012) Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE . A FIPA-compliant agent framework. pages 97-108, Abril 2012.

(Bellifemine, 2013)-Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. JADE Programmer's Guide. Technical report, CSELT, Febrero 2013.

(Burker and Kendall, 2005)-Burker, Edmund K. and Graham Kendall (2005). Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques Springer.

(Brenner, 1998)- Brenner W., Zaenekiq, R., and Wittig, H., Intelligent Agents: Foundations and Applications, Berlin: Springer-Verlag, 1998.

(Bradshaw, 1997)- Bradshaw J. M., An Introduction to Software Agents, in Software Agents, Bradshaw, J.M., (ed.) Cambridge, MA: MIT Press, 1997.

(Cormen et al., 2001)- Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (2001). Introduction to algorithms. Second Edition, MIT Press and McGraw-Hill. ISBN 0-262-03293-7. Section 24.1: The Bellman-Ford algorithm, pg.588-592.

(Dorigo, 2004)-Dorigo M., Stützle T., Ant Colony Optimization. Bradford, Cambridge, Massachusetts, 2004.

(Darryl, 2003)-Darryl N. Davis, Yuan Luo, and Kecheng Liu. Combining KADS with Zeus to Develop a Multi-Agent E-Commerce Application. In 3-4, editor, International Journal of Electronic Commerce Research. 315-335, 2003.

(Fraichard and Scheuer, 2004)- Fraichard, Thierry and Alexis Scheuer (2004). From reeds and shepp's to continuouscurvature paths. IEEE Trans. on Robotics and Automation.

(*FIPA-OS, 2013*) *-FIPA-OS Developers Guide*. http://fipa-os.sourceforge.net/tutorials.htm, 2013.

(Guzmán, 2008) -Guzmán S, Ceballos M, Suárez N. Planificación de trayectorias para un robot tipo con restricciones dinámicas. Ciencia e Ingeniería Neogranadina. 2008; 18 (1): 75-94.

(Gómez-Bravo et al., 2007)-Gómez-Bravo, F., F. Cuestab, A. Ollero and A. Viguria

(Gómez-Bravo et al., 2008)-Gómez-Bravo, F., Cuestab, F., Ollero, A. y Viguria, A. (2008): Continuous curvature path generation based on b-spline curves for parking manoeuvres. Robotics and Autonomous Systems vol. 5. Citado en: Torres, 2010.

(Gerkey, 2003)-Gerkey, Brian P. and Vaughan, Richard T. and Howard, Andrew. The Player/Stage project: tools for multi-robot and distributed sensor systems. Proceedings of the 11th International Conference on Advanced Robotics ICAR?2003, pp. 317-323, Coimbra (Portugal).

(2007). Continuous curvature path generation based on b-spline curves for parking manoeuvres. Robotics and Autonomous Systems.

(Hyacinth, 1999)-Hyacinth S Nwana, Divine T. Ndumu, Lyndon C. Lee, and Jaron C. Collis. ZEUS: a toolkit and approach for building distributed multi-agent systems. In Oren Etzioni, J. P. Müller, and Jeffrey M. Bradshaw, editors, Proceedings of the Third International Conference on Autonomous Agents (Agents'99), Seattle, WA, USA. ACM Press.

(Huget, 2004)-Huget M-P. Especificaciones FIPA (Foundation for Intelligent Physical Agents). Geneva, Switzerland: Huget M-P; actualizada en el 2004; acceso 8 de febrero de 2013]. Disponible en: http://www.fipa.org

(Iglesias, 1998)-Iglesias, C.: *Definición de una metodología para el desarrollo de Sistemas Multi-Agente*. Tesis doctoral. Departamento de ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid. 1998.

(Iglesias et al 1998b)-Iglesias, C., Mercedes Garijo, M., Gonzalez, J. C., and Velasco, J. R., *Analysis and design of multiagent systems using MASCommonKADS*, en *Intelligent Agents IV* **.** LNAI Volume 1365 ed. SpringerVerlag: Berlin, 1998.

(Kanayamaand Hartman, 1989)-Kanayama, Y. and B. I. Hartman (1989). Smooth local path planning for autonomous vehicles. pg 1265-1270.

(K. Konolige, 1997)-K. Konolige, K.L. Myers, E.H. Ruspini, and A. Saffiotti. The Saphira architecture: A design for autonomy. Journal of experimental and theoretical artificial intelligence: JETAI, 9(1):215–235, 1997.

(Kinny et al. 1997)-Kinny, D., Georgeff, M., and Rao, A.:*A Methodology and Modelling Technique for Systems of BDI Agents*. Informe. 1997

(Wooldridge et al, 2000)-Wooldridge, M., Jennings, N. R., and Kinny, D., *The Gaia Methodology for Agent-Oriented Analysis and Design*, Journal of Autonomous Agents and Multi-Agent Systems, vol. 15 2000.

(Lozano-Pérez, 1983)- Lozano-Pérez T. (1.983) “Spatial Planning: A Configuration Approach”. IEEE Transactions on Computers. Vol. 32, pp 108-120.

(Liang et al., 2005)-Liang, T-C; Liu J.S; Hung, G.T. and Y.Z. Chang (2005): Practical and flexible path planning for car-like mobile robot using maximal-curvature cubic spiral," Robotics and Autonomous System, Vol. 52, 2005.

(Liu and Ramakrishnan, 2001)-Liu, Gang and K. G. Ramakrishnan (2001). A*prune: An algorithm for ¯nding k shortest paths subject to multiple constraints.

(Laukkanen, 2000)-M. Laukkanen. Evaluation of FIPA-OS 1.03. Technical report, Cellular System Development, Sonera Mobile Operator, Sonera Ltd., Helsinki, Finlandia, Febrero 2000.

(Muñoz, 1995)-Muñoz, V. (1995). “Planificación de Trayectorias para Robots Móviles”. Tesis Doctoral. Universidad de Málaga.

(Murphy, 2000)-Murphy, R. R. (2000). Introduction to ai robotics. MIT Press.

(Mataric, 1992)-Mataric, M. J. (1992). Integration of representation into goal driven behaviour-based robots. IEEE Transactions on Robotics and Automation 8(3), pg 304-312.

(McCrae and Singh, 2008)- McCrae, J. and K. Singh (2008). Sketching piecewise clothoid curves. EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling.

(Meidenbauer, 2007)- Meidenbauer, Kenneth R. (2007). An investigation of the clothoid steering model for autonomous vehicles. Master's thesis. Virginia Polytechnic Institute.

(Moulin, 1996)-Moulin B., Chaib-draa B. An Overview of Distributed Artificial Intelligence. In Foundations of Distributed Intelligence, (ed.) G.M.P.O'Hare and N.R. Jennings, New York: Wiley, 1996.

(MicroSoft Robotics Studio, 2009)-[url] MicroSoft Robotics Studio http://msdn2.microsoft.com/eses/ robotics Visitado 1 abril 2009

(Nagabhushan and Pai, 2001) Nagabhushan, P. and M.M. Manohara Pai (2001). Cognition of free space for planning the shortest path: A framed free space approach. Pattern Recognition Letters 22, pp 971-982.

(Nilsson, 1980)-Nilsson, N. J. (1980). Principles of articial intelligence. Tioga Publishing Company. Pg 72-88.

(Nwana, 1996)-Nwana, H. (1996) Software Agents: An Overview. En: Knowledge Engineering Review, Vol. 11, No. 3.

(Nwana, 1999)-H. Nwana, D. Ndumu, and L. Lee. Zeus: A collaborative agents toolkit. In Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pages 377.392, 1999.

(Nikraz et al, 2005)-Nikraz M., G. Caire, y P. A. Bahri, (2005), "A Methodology for the Analysis and Design of Multi-Agent Systems using JADE", School of Engineering Science and Parker Center, Murdoch University, Dixon Road, Rockingham, Western Australia 6168 Telecom Italia Lab, Via Reiss Romoli, Turin, Italy

(Odell et al, 2000)-Odell, J.; H. Van Dyke Parunak y B. Bauer (2000), "Extending UML for Agents", James Odell Associates 3646 W. Huron River Dr. Ann Arbor, MI 48103 Tel: +1 (734) 994-0833 jodell@compuserve.com www.jamesodell.com, ERIM Center for Elec. Comerse P.O. Box 134001 Ann Arbor, MI 48113 Tel: +1 (734) 623-2509 vparunak@erim.org www.erim.org/~vparunak Siemens ZT IK 6 D-81730 München.

(Player-Stage-Gazebo,2009)-[url] Player, Stage, Gazebo http://playerstage.sourceforge.net/ Visitado 1 abril 2009

(Porta, 2007)-Porta M.A . (2007). "Planeación de trayectorias para robótica móvil mediante optimización por colonia de hormigas". Instituto politécnico nacional.

(Poslad, 2013)-S. Poslad, P. Buckle, and R. Hadingham. The FIPA-OS Agent Platform:Open Source for Open Standards, Abril 2013.

(Pressman, 2005)-Pressman, Roger S. Software Engineering. A practitioner´s Approach. 7th Edition.

(Russel S, 1995)-Russel S, Norvig P, editores. Artificial Intelligence: A Modern Approach. 2ª ed. New Jersey: Pearson Education; 1995.

(Ricordel, 2001)-Ricordel, P. M.: Programmation Orientée Multi-Agents , Développement et Déploiement de Systèmes Multi-Agents Voyelles. Tesis doctoral. Institut National Polytechnique de Grenoble. 2001.

(Sleumer and Tschichold-Gurman., 1999)- Sleumer, N. and N. Tschichold-Gurman. (1999). Exact cell decomposition of arrange- ments used for path planning in robotics. Technical report. Institute of Theoretical Computer Science Zurich.

(Stentz, 1994)-Stentz, A. (1994). Map-based strategies for robot navigation in unknown environments. In AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems pp. p6 110-116.

(SRIsim, 2008)-[url] SRIsim http://www.ai.sri.com/ konolige/saphira Visitado 1octubre 2008

(Thorphe ,1984)- Thorphe C. (1.984) “FIDO: Vision and Navigation for a Robot Rover”.Ph. D. Thesis, Carnegie Mellon University.

(Torres, 2010) -Torres, M.O. (2010): Planificación y Generación de trayectorias para robot móvil recolector de objetos.

(Torres y Moreno, 2011)--Torres Piñeiro M.O, Moreno Vega, V. Un estudio de los Método de planificación de trayectorias en entornos estáticos, Revista Cubana de Ciencias Informáticas, Vol.3, No. 1-2, 2009 (publicado en 2011).

(Torres y Moreno, 2010)--Torres Piñeiro M.O, Moreno Vega, V. Planificación de Trayectorias mediante Programación Dinámica, Revista Electrónica, Automática y Comunicaciones, Vol. XXXI, No.2, 2010.

(Weiss, 1999) -Weiss G, editor. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Massachussetts: The MIT Press Cambridge; 1999.

(Wilmarth, 1998) Wilmarth, S.A., Amato N.M., Stiller, P.F. (1998) “A probabilistic roadmap planner with Sampling on the medial Axis of the Free Space”. TR-Department of Computer Science Texas A&M University.

(Weiss, 1999)-Weiss G, editor. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Massachussetts: The MIT Press Cambridge; 1999.

(Wooldridge, 2007)- Wooldridge, M.; Jennings, N.; Kinny, D. (1999) A Methodology for Agent-Oriented Analysis and Design. Disponible en: http://www.csc.liv.ac.uk/~mjw/pubs/agents99.pdf Fecha de acceso: Abril de 2007.

(Webots, 2009)[url] Webots, Cyberbotics http://www.cyberbotics.com Visitado 1abril 2009

ANEXOS

Anexo A: Acotación de la curvatura en el algoritmo β-Spline

Muñoz, (Muñoz, 1995) plantea que si se tiene que $\{V_{-2}, V_{-1}, V_0, V_1\}$ son un conjunto de puntos de control que definen una curva β-Spline, y yacen sobre un círculo de radio R, separados una distancia de $\delta \leq 0.5\pi R$, entonces la curva β-Spline definida por estos puntos estará contenida en un círculo concéntrico con radio R' definido por:

$$R' = R - s$$

Donde: $s = \frac{\delta^2 \sqrt{R^2 - \frac{\delta^2}{4}}}{3R^2}$

La propiedad anterior es planteada para 4 VC, con el objetivo de usar esta propiedad en el método empleado, se asegura que existan siempre 4 VC o más sobre los arcos de circunferencia que se crean. Esto se logra con la adecuada selección de R y δ, con los que se logra además que la curva no viole la restricción de máxima curvatura.

Selección de R y δ

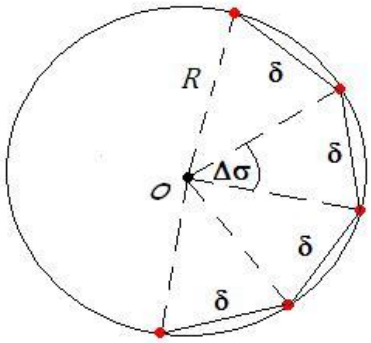


Figura A. 2 Relación de Δσ y δ en la circunferencia de centro O y radio R.

En la figura A.1 se muestra que en una circunferencia de centro *O* y radio *R* conocido, existe una relación, entre el ángulo Δσ correspondiente al arco limitado por un punto sobre el perímetro de dicha circunferencia y otro punto consecutivo separado una distancia δ.

Esta relación se halla como se muestra a continuación:

$$\Delta\sigma = \tan^{-1}\left(\frac{\delta\sqrt{4R^2 - \delta^2}}{2R^2 - \delta^2}\right)$$

Se puede a partir de la ecuación anterior llegar a

$$\delta = R \times \sqrt{2} \times \sqrt{1 - \frac{1}{\sqrt{(\tan\Delta\sigma)^2 + 1}}}$$

que se puede volver a escribir como:

$$\boldsymbol{\delta = R \times E(\Delta\sigma)}$$

donde se define la función E que depende de $\Delta\sigma$ como:

$$E(\Delta\delta) = \sqrt{2} \times \sqrt{1 - \frac{1}{\sqrt{(\tan\Delta\sigma)^2 + 1}}}$$

Los arcos de circunferencia que se deben crear serán de 90° y 45°. Como para los dos tipos de arcos de circunferencia que se deben crear se usa el mismo radio, si se asegura que existen 4 VC sobre el arco de 45°, se estará asegurando que existan más de 4 sobre el de 90°, de hecho se estará asegurando que existan 8 VC sobre este último. Para asegurar los 4 VC sobre el arco de 45° se divide 45° entre 4 y se obtiene 11.25°, entonces el valor de $\Delta\sigma$ con el que se calculará δ será de 11.25°.

Si se coge como R' el valor del radio de giro mínimo que puede realizar el vehículo R_{min}, calculado a partir de la separación entre los ejes de las ruedas delanteras y traseras l, y el máximo ángulo de direccionamiento ϕ, se tiene que:

$$R_{min} = R - \frac{\delta^2\sqrt{R^2 - \frac{\delta^2}{4}}}{3R^2}$$

Y si se sustituye $\delta = R \times E(11.25°)$, se llega a:

$$R_{min} = R - \frac{\left(R \times E(11.25°)\right)^2\sqrt{R^2 - \frac{\left(R \times E(11.25°)\right)^2}{4}}}{3R^2}$$

Que finalmente puede ser expresado como:

$$R_{min} = R\left(1 - \frac{\left(E(11.25°)\right)^2}{3}\sqrt{1 - \frac{\left(E(11.25°)\right)^2}{4}}\right)$$

de donde despejando R, y efectuando los cálculos indicados se tiene que:

$$\boldsymbol{R = \frac{R_{mín}}{0.98725}}$$

Este es el valor de R con que son creados los arcos de circunferencia, para lograr que el máximo valor de curvatura de la curva creada no sobrepase en ningún momento el valor de curvatura dado por el inverso del radio de giro mínimo del vehículo.

Relación entre Δσ y δ

Para hallar el ángulo Δσ, se dará solución al siguiente problema:

Dadas dos circunferencias, una de radio R con centro en el origen y otra de radio δ < R, con centro en el punto (R;0). Encontrar el punto de intersección de las dos circunferencias en el primer cuadrante. Una vez encontrado este punto encontrar el ángulo correspondiente a este para la circunferencia de radio R, ver figura A.2.

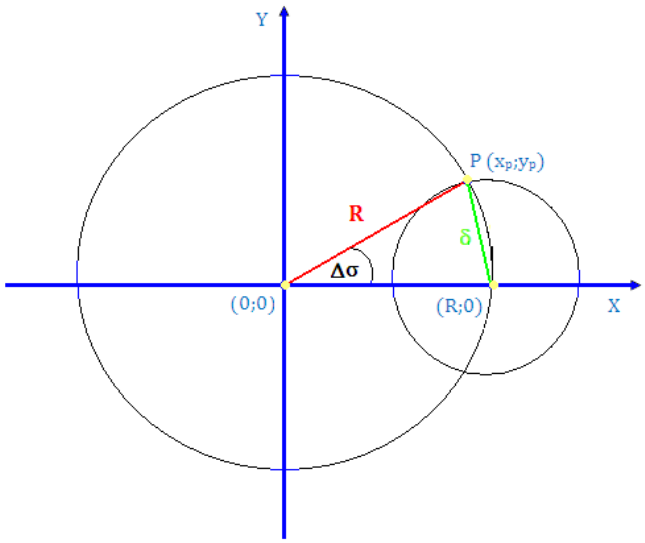


Figura A.2 Punto P, punto de intersección en el I cuadrante de la circunferencia. de radio R con centro en el origen y la circunferencia de radio δ con centro en el punto (R;0)

La ecuación de la circunferencia de radio R, con centro en el origen será:

$x^2 + y^2 - R^2 = 0$.

Y la ecuación de la circunferencia de radio δ, con centro en el punto (R;0) será:

$x^2 – 2xR + R^2 + y^2 - \delta^2 = 0$.

De restar la segunda de la primera, se tiene:

$x = (2R^2 - \delta^2)/2R$

Que corresponde a la ecuación de la recta que pasa por los puntos de intersección de las dos circunferencias. Sustituyendo $x = (2R^2 - \delta^2)/2R$ en la ecuación de la circunferencia de radio R se llega a:

$y = \delta/2R * (4R^2 - \delta^2)^{1/2}$

Luego:

$y / x = \delta/\cancel{2R} * (4R^2 - \delta^2)^{1/2} * \cancel{2R}/(2R^2 - \delta^2)$

$y / x = \delta(4R^2 - \delta^2)^{1/2} / (2R^2 - \delta^2)$

Entonces

$\Delta\sigma = \arctan (y / x) = \arctan (\delta(4R^2 - \delta^2)^{1/2} / (2R^2 - \delta^2))$.

$\Delta\sigma = \arctan (\delta(4R^2 - \delta^2)^{1/2} / (2R^2 - \delta^2))$.

$$\boldsymbol{\Delta\sigma = \tan^{-1}\left(\frac{\delta\sqrt{4R^2 - \delta^2}}{2R^2 - \delta^2}\right)}$$

Anexo B: Obtención de los puntos C_1 y C_2

El problema a resolver es el siguiente:

Dados tres puntos, P1, P2 y P3 definidos por sus coordenadas cartesianas, y una circunferencia de radio R tangente a los segmentos $\overline{P1P2}$ Y $\overline{P2P3}$, donde el menor ángulo formado por estos segmentos también se conoce, hallar los puntos donde la circunferencia es tangente a los segmentos, ver figura B.1.

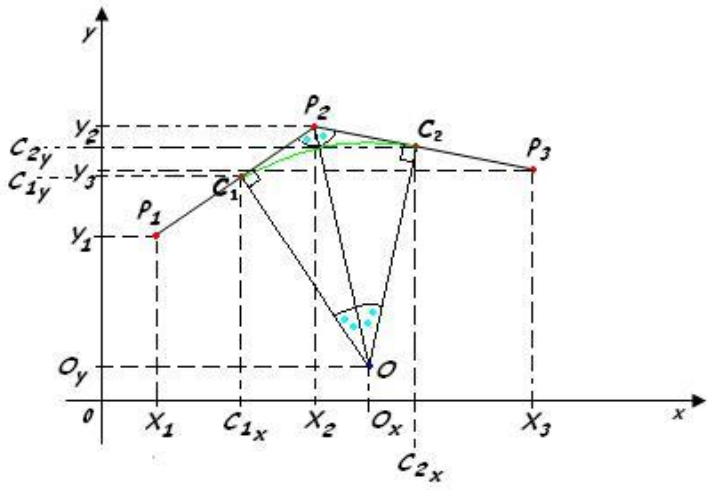


Figura B.1 Caso general, Punto O, centro de la circunferencia de radio R=$\overline{C_1O}$=$\overline{C_2O}$. C_1 y C_2 puntos de tangencia de la circunferencia en cuestión a los segmentos $\overline{P1P2}$ Y $\overline{P2P3}$ respectivamente.

Se tiene $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ y $P_3(x_3, y_3)$ y que:

$\overline{C_1O}$=$\overline{C_2O}$=*R. con, R*: Radio conocido.

Así como que $\angle P_1P_2P_3 = \alpha$.

α: Menor ángulo formado por los dos segmentos $\overline{P1P2}$ Y $\overline{P2P3}$.

Se desea hallar los puntos $C_1(C_{1\_x}, C_{1\_y})$ y $C_2(C_{2\_x}, C_{2\_y})$.

El segmento $\overline{OP2}$ es parte de la bisectriz del ángulo interior α, de un triángulo cuyo uno de sus vértices es P_2, y el ángulo asociado a este vértice es α, con lados de una longitud tal que la circunferencia de radio conocido R está inscrita en el mismo.
Lo anteriormente expuesto se basa en la siguiente propiedad geométrica: El incentro es el punto en el que se interceptan las tres bisectrices de los ángulos internos del triángulo, y es el centro de la circunferencia inscrita en el triángulo y que equidista de sus tres lados, siendo tangente a dichos lados.
Entonces el segmento $\overline{OP2}$ divide el ángulo α en dos ángulos iguales:

$\angle P_1P_2O = \angle OP_2P_3 = \alpha/2$.

Por otro lado:

$\angle OC_1P_2 = \angle OC_2P_2 = 90°$. Por ser los segmentos $\overline{OC1}$ y $\overline{OC2}$ tangentes a los segmentos $\overline{P1P2}$ Y $\overline{P2P3}$ respectivamente.

$\angle C_1OP_2 = 180° - \angle OC_1P_2 - \angle P_1P_2O$. Por suma de ángulos interiores de un triángulo, en el triángulo C_1OP_2.

$\angle C_1OP_2 = 180° - 90° - \alpha/2 = 90° - \alpha/2$.

$\angle C_2OP_2 = 180° - \angle OC_2P_2 - \angle OP_2P_3$. Por suma de ángulos interiores de un triángulo, en el triángulo P_2OC_2.

$\angle C_2OP_2 = 180° - 90° - \alpha/2 = 90° - \alpha/2$.

Los triángulos P_2OC_2 y C_1OP_2 son iguales por tener dos lados consecutivos ($\overline{C_1O}$=$\overline{C_2O}$=*R y* $\overline{OP2}$ lado común para ambos triángulos) y el ángulo comprendido iguales($\angle C_1OP_2 = \angle C_2OP_2 = 90° - \alpha/2$.).

$\overline{C1P2}$ = $\overline{P2C2}$ por ser lados homólogos en triángulos iguales (triángulos P_2OC_2 y C_1OP_2).

$$\frac{\overline{C_1P_2}}{\sin(\angle C_1OP_2)} = \frac{\overline{C_1O}}{\sin(\angle P_1P_2O)}$$

La relación anterior se establece por ley de los senos en el triángulo C_1OP_2. Sustituyendo los valores conocidos se tiene que:

$$\frac{\overline{C_1P_2}}{\sin(90° - \alpha/2)} = \frac{R}{\sin(\alpha/2)}$$

Entonces:

$$\overline{C_1P_2} = \frac{R}{\sin(\alpha/2)} \times \sin(90° - \alpha/2)$$

Sustituyendo $\sin(90° - \alpha/2)$ por $\cos(\alpha/2)$, por ángulos complementarios.

$$\overline{C_1P_2} = \frac{R}{\sin(\alpha/2)} \times \cos(\alpha/2)$$

$$\overline{C_1P_2} = R \times \cot(\alpha/2).$$

$$\overline{C_1P_2} = \overline{P_2C_2} = d = R \times \cot(\alpha/2).$$

Esta distancia d es empleada para hallar las coordenadas de los puntos C_1 y C_2. Para ello se trasladan los segmentos $\overline{P1P2}$ y $\overline{P2P3}$ de manera que las nuevas proyecciones de los puntos P_1, P_2 y P_3: P_1', P_2' y P_3' respectivamente se ubiquen en el plano de forma tal que el punto P_2' esté en el origen de coordenadas. P_1', P_2' y P_3' serán entonces:

P_1' (x_1', y_1') = ($x_1 - x_2$, $y_1 - y_2$).

P_2' (x_2', y_2') =($x_2 - x_2$, $y_2 - y_2$) = (0, 0).

P_3' (x_3', y_3') = ($x_3 - x_2$, $y_3 - y_2$).

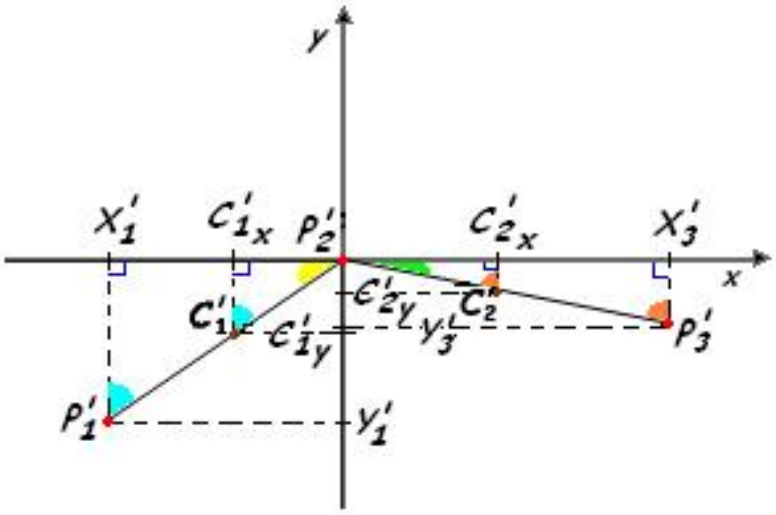


Figura B.2 Traslación de los segmentos $\overline{\mathbf{P1P2}}$ y $\overline{\mathbf{P2P3}}$

Las siguientes distancias serán usadas más adelante:

$$\boldsymbol{d_1 = \overline{P_1'P_2'} = \overline{P_1P_2} = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}.}$$

$$\boldsymbol{d_2 = \overline{P_3'P_2'} = \overline{P_3P_2} = \sqrt{(y_2 - y_3)^2 + (x_2 - x_3)^2}.}$$

$\boldsymbol{d = \overline{C_1 P_2} = \overline{P_2 C_2} = \overline{C_1{}' P_2{}'} = \overline{P_2{}' C_2{}'} = R \times \cot(\alpha/2)}$.

En la figura B.2 se puede observar como aparecen dos triángulos rectángulos semejantes con un vértice en el origen de coordenadas a ambos lados del eje de las ordenadas, pues se puede concluir que estos son semejantes por tener tres ángulos iguales entre sí.

A la izquierda del eje de las ordenadas los triángulos semejantes tienen en común el ángulo en amarillo de la figura, tienen cada uno un ángulo recto dibujado en azul, y el tercer ángulo que tienen igual por suma de ángulos interiores de un triángulo se puede apreciar pues ha sido dibujado con el mismo color en ambos triángulos.

La situación a la derecha del eje de las ordenadas es similar. Los ángulos rectos han sido dibujados también en azul, el ángulo común en verde, y el otro en anaranjado.

Por las relaciones entre los triángulos semejantes a la izquierda del eje de las ordenadas en la figura B.2, se puede decir que:

$$\frac{d_1}{d} = \frac{{x_1}'}{{C_{1_x}}'}$$

$$\frac{d_1}{d} = \frac{{y_1}'}{{C_{1_y}}'}$$

Haciendo

$$\boldsymbol{k_1 = \frac{d}{d_1}}$$

Se tienen entonces que las coordenadas del punto C_1' serán:

$C'_{1_x} = k_1 \times x'_1.$ $\qquad$ $C'_{1_y} = k_1 \times y'_1.$

De igual manera, por las relaciones entre los triángulos semejantes a la derecha del eje de las ordenadas en la figura B.2, se puede decir que:

$$\frac{d_2}{d} = \frac{{x_3}'}{{C_{2_x}}'}$$

$$\frac{d_2}{d} = \frac{{y_3}'}{{C_{2_y}}'}$$

Haciendo

$$\boldsymbol{k_2 = \frac{d}{d_2}}$$

Se tienen entonces que las coordenadas del punto C_2' serán:

$$C'_{2_x} = k_2 \times x'_3. \qquad C'_{2_y} = k_2 \times y'_3.$$

Los puntos C_1' y C_2' son las proyecciones de C_1 y C_2 al haber traslado los segmentos $\overline{P1P2}$ y $\overline{P2P3}$. Al tener las coordenadas de estas proyecciones podemos encontrar las coordenadas de C_1 y C_2.

Si:

C_1' (C_{1_x}', C_{1_y}') = ($C_{1\_x} - x_2$, $C_{1\_y} - y_2$).

C_2' (C_{2_x}', C_{2_y}') = ($C_{2\_x} - x_2$, $C_{2\_y} - y_2$).

Entonces:

C_1 (C_{1_x}, C_{1_y}) = (C_{1_x}' + x_2, C_{1_y}' + y_2) = (k_1x_1' + x_2, k_1y_1' + y_2)

C_1 (C_{1_x}, C_{1_y}) = ($k_1(x_1 - x_2) + x_2$, $k_1(y_1 - y_2) + y_2$).

Y

C_2 (C_{2_x}, C_{2_y}) = (C_{2_x}' + x_2, C_{2_y}' + y_2) = (k_2x_3' + x_2, k_2y_3' + y_2)

C_2 (C_{2_x}, C_{2_y}) = ($k_2(x_3 - x_2) + x_2$, $k_2(y_3 - y_2) + y_2$).

Anexo C: Descripción de casos de uso

Tabla C.1 Caso de uso: Gestionar entorno.

Caso de uso:	Gestionar entorno
Actor:	Usuario, AgentePlataformaNavegación
Descripción:	El caso de uso se inicia cuando se desea crear un nuevo entorno, abrir uno creado con anterioridad, guardarlo para un análisis posterior o salir de la plataforma. En el caso de salir de la plataforma se le notifica al actor AgentePlataformaNavegación.
Referencias :	RF1

Tabla C.2 Caso de uso: Ubicación de obstáculo.

Caso de uso:	Ubicación de obstáculo
Actor:	Usuario
Descripción:	El caso de uso se inicia cuando se desea ubicar obstáculos en el entorno, moverlos para lograr una mejor ubicación o borrarlos en caso de ser necesario.
Referencias :	RF2

Tabla C.3 Caso de uso: Ubicación del robot.

Caso de uso:	Ubicación del robot
Actor:	Usuario

Descripción:	El caso de uso se inicia cuando se desea ubicar un robot en el entorno especificando la posición inicial y final que deben alcanzar así como el algoritmo de planificación seleccionado. Cuando se desee visualizar las posiciones de un robot ubicado con anterioridad. Cuando se desea agregar un nuevo algoritmo de planificación.
Referencias :	RF3, RF4

Tabla C.4 Caso de uso: Simular.

Caso de uso:	Simular
Actor:	Usuario, AgentePlataformaNavegación
Descripción:	El caso de uso se inicia cuando se comienza la simulación, mostrando los caminos generados desde las posición origen hasta la meta de cada robot con el algoritmo de planificación seleccionado.
Referencias :	RF5
Precondiciones:	Previamente debe de haberse ubicado al menos un robot en el entorno.
Requerimientos especiales :	De no ser posible planificar la ruta debe de informársele al usuario.

Tabla C.5 Caso de uso: Métricas.

Caso de uso:	Métricas

Actor:	Usuario
Descripción:	El caso de uso se inicia cuando se desea mostrar las métricas en forma de gráficos que permitan emitir criterios para la evaluación de los algoritmos en cuanto a costo, iteraciones, tiempo de ejecución. Además de mostrar las gráficas descritas por la ruta planificada y el camino generado para cada robot.
Referencias :	RF6
Precondiciones:	Debe de haberse simulado con anterioridad la planificación de al menos un robot.
Requerimientos especiales :	De no haberse planificado con anterioridad un robot para poder mostrar los datos requeridos se le notificara al usuario.

Anexo D: Interacción de los agentes

Tabla D.1 Interacción del AP con APN y DF utilizando el protocolo Request.

Interacción	**Nro. de responsabilidad**	**AIP**	**Rol**	**Con el agente**	**Cuando**
1. De no poseer un servicio para el cálculo de ruta requerido por un robot gestionar con el Directorio	5	FIPA Request (petición)	Iniciar	DF	AP no provee el servicio requerido.

Facilitador (DF) un agente que lo provea.					
2. Enviar respuesta a APN.	6	FIPA Request (informe)	Responder	APN	Envió a APN resultados de la planificación

Tabla D.2 Interacción del AG con APN utilizando el protocolo Request.

Interacción	**Nro. de responsabilidad**	**AIP**	**Rol**	**Con el agente**	**Cuando**
1.Enviar respuesta a APN.	6	FIPA Request (informe)	Responder	APN	Envió a APN resultados de la generación.

Anexo E: Caso de prueba

Tabla E.1 Caso de prueba con valores correctos.

Nombre del caso de prueba	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
Simular el funcionamiento del robot introduciendo	Se describe la secuencia de acciones para crear un entorno, ubicar los	EP 1.1: Gestionar entorno.	– El menú archivo nos brinda la posibilidad de gestionar el

valores correctos.	obstáculos que debe sortear el robot, seleccionar el algoritmo que se quiere estudiar y determinar las posiciones de inicio y fin que debe de alcanzar el robot, luego presionar el botón simular.		entorno, crear uno nuevo, abrir o salir de la aplicación. – En la sección obstáculos fijos nos permite seleccionar el tipo de obstáculo que queremos ubicar en el entorno, moverlo para una mejor ubicación o borrarlo en caso de que no sea necesario. – Una vez creado el entorno se guarda y se pasa a la sección de ubicar robot.
		EP 1.2: Ubicar robot.	– Permite dado el algoritmo seleccionado ubicar un robot en el entorno definiendo su posición inicial y final.
		EP 1.3: Simular la planificación de la ruta del robot.	– Se presiona el botón simular y se nos muestra la ruta calculada para el robot con el algoritmo seleccionado. Si en el proceso de simulación no se pudo planificar la ruta para el robot el usuario será

			notificado. – Una vez planificada la ruta en la sección robot y criterio permite obtener métricas del desempeño del algoritmo seleccionado.

Tabla E.2 Caso de prueba con valores incorrectos.

Nombre del caso de prueba	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
Simular el funcionamiento del robot introduciendo valores incorrectos.	Se describe la secuencia de acciones para crear un entorno, ubicar los obstáculos que debe sortear el robot, seleccionar el algoritmo que se quiere estudiar y determinar las posiciones de inicio y fin que debe de alcanzar el robot, luego presionar el botón simular.	EP 1.1: Gestionar entorno.	– El menú archivo nos brinda la posibilidad de gestionar el entorno, crear uno nuevo, abrir o salir de la aplicación. – En la sección obstáculos fijo nos permite seleccionar el tipo de obstáculo que queremos ubicar en el entorno, moverlo para una mejor ubicación o borrarlo en caso de que no sea necesario. – Una vez creado el entorno se guarda y se pasa a la sección de ubicar robot.

	EP 1.2: Ubicar robot.	– Permite dado el algoritmo seleccionado ubicar un robot en el entorno definiendo su posición inicial y final.
	EP 1.3: Simular la planificación de la ruta del robot.	– Si presiona el botón simular y no se han ubicado algún robot al cual calcularle la ruta o no se ha guardado el entorno se le notificara al usuario que falta parámetros por especificar.