ISSN: 1692-7257 - Volumen 2 - Número 20 - 2012

Revista Colombiana de Tecnologías de Avanzada

Recibido: 26 de marzo de 2012
Aceptado: 12 de junio de 2012

ABSTRACT PLANNING SYSTEMS IN AUTONOMOUS ROBOTS

PLANIFICACIÓN ABSTRACTA EN LOS SISTEMAS DE ROBOTS AUTÓNOMOS

PhD. Jaime Guzmán-Luna, Ing. Andrés F. Gutiérrez

Universidad Nacional de Colombia. Sede Medellín, Escuela de Sistemas.
Carrera 80, No. 65 - 223, Bloque M8, Medellín, Antioquia, Colombia.
Tel.: (+57) (4) 4255350.
E-mail: {jaguzman, afgutierezs}@unal.edu.co

Abstract: This paper proposes a model of motion planning for a mobile robot that integrates within a hierarchy of abstraction of the problem a planning technique based on cases and path planning technique that uses approximate decomposition cells. The implementation of the model is evaluated within a Lego NXT robot and the results of this evaluation showed improvement over the movement planner *appCells*.

Keywords: Mobile robot, task planning, path planning, Lego Robot.

Resumen: En este artículo se propone un modelo de planificación de movimientos para un robot móvil el cual integra en el marco de una jerarquía de abstracción del problema una técnica de planificación basada en casos y una técnica de planificación de trayectorias que hace uso de la descomposición aproximada de celdas. La implementación del modelo es evaluada al interior de un robot Lego NXT y los resultados de esta evaluación muestran mejoras frente al planificador de movimientos *appCells*.

Palabras clave: Robot Móvil, Planificación de Tareas, Planificación de rutas, Robot Lego.

1. INTRODUCIÓN

Los robots móviles autónomos se desarrollan con el objetivo de llevar a cabo tareas complejas en diferentes ambientes, tanto en hábitats humanos como en ambientes difíciles tales como planetas remotos y zonas bajo el agua (Bouguerra, 2008).

Para entornos relacionados con hábitats humanos, se han propuesto robots móviles autónomos que llevan a cabo tareas solicitadas por los residentes de un edificio o los trabajadores de una planta, tales como: la entrega de correo, guiar visitantes, ir a buscar café, recoger una impresión, o simplemente entregar un mensaje u objeto de un lugar a otro (Coltin *et al.*, 2011).

Para realizar las anteriores tareas un aspecto fundamental en un robot móvil es la planificación de la trayectoria en el contexto de la navegación del robot. Particularmente en (Vélez *et al.*, 2011) se detallan las experiencias en la implementación de *appCells*, una aplicación para la navegación de un robot móvil mediante técnicas descomposición por celdas asociadas a la planificación de la trayectoria del robot. Estas técnicas aunque arrojaron buenos resultados están propensas a ser mejoradas.

En este trabajo se propone un modelo de planificación abstracta que integra una técnica conocida como planificación basada en costos y otra técnica basada en las trayectorias que hace uso de la descomposición aproximada con el fin de

mejorar la navegación de un robot que hace uso de *appCells*.

Este documento está organizado de la siguiente manera: en la sección 2 se introducen los conceptos básicos relacionados con la planificación en la robótica que soportan las técnicas de planificación utilizadas en este trabajo, en la sección 3 se presenta la arquitectura del modelo de planificación abstracta propuesto, en la sección 4 se presenta apartes de la implementación del modelo propuesto para la planificación de movimientos, en la sección 5 se detalla las pruebas de evaluación y sus resultados que se realizaron al modelo de planificación propuesto, por último, en la sección 6 se presentan las conclusiones y trabajos futuros que se desprenden de este trabajo.

2. LA PLANIFICACIÓN EN LA ROBÓTICA

La Planificación de Tareas como área de la Inteligencia Artificial (IA) es un proceso abstracto y deliberativo que escoge y organiza acciones buscando alcanzar, tan bien como sea posible, un conjunto de objetivos preestablecidos (Ghallab, 2004).

De manera formal, un problema de planificación de tareas se puede definir mediante una tupla *(L;O; I;G)*, donde cada uno de estos elementos representa lo siguiente: *L* es el conjunto de fórmulas atómicas, denominadas hechos o literales y representa el conjunto de hechos que son relevantes en el problema; *O* es el conjunto de acciones definidas en el dominio del problema; *I* es el conjunto inicial de hechos que forman la situación inicial del problema y *G* es el conjunto final de hechos que deben formar parte de la situación final del problema.

Para lograr una planificación de tareas eficiente, es importante contar con buenos lenguajes de modelado. En este sentido el PDDL (*Planning Domain Definition Language*) ha sido el que más se distingue (Ghallab *et al.*, 1998).Su objetivo principal es expresar la física de un dominio, es decir, qué predicados hay, qué acciones se pueden realizar y cuáles son sus efectos, sin proporcionar ningún conocimiento adicional sobre el mismo. Por otro lado este también permite representar un problema particular de planificación asociado a un dominio, describiendo que objetos intervienen en el problema, cual es el estado inicial del problema y cuál es su estado final. En general, en la mayoría de los planificadores de tareas se suele dividir en dos archivos la especificación anterior. El *domain.pddl* contiene las especificaciones del dominio y el *problem.pddl* contiene las características propias del problema asociado al dominio, todo esto en el marco del lenguaje PDDL (Borrajo *et al*., 1993).

Existen diversos algoritmos de planificación de tareas que emplean a su vez diferentes técnicas para la resolución de problemas entre las que se destacan la planificación heurística (Long y Fox, 2002), que se basa en la utilización de funciones de estimación para explorar el espacio de búsqueda de una forma más inteligente y la planificación basada en costos (Fuentetaja, 2010) en la cual su característica principal es que las acciones suelen tener un costo asociado y su objetivo es conseguir planes de buena calidad basado en dicho costo.

Otra área de la IA muy orientada a la robótica móvil es la planificación de trayectorias (Choset, *et al*., 2005). Esta consiste en que dado un mapa y una localización objetivo se busca identificar una trayectoria que hará que el robot llegue a la localización objetivo cuando esta se ejecuta (Gat, 1992).

En la literatura existen diferentes planificadores de trayectorias los cuales difieren principalmente en cómo efectúan la descomposición del ambiente en que se desempeña el robot (Gutiérrez, 2004).

Se pueden identificar tres estrategias generales para la descomposición (Siegwart y Nourbakhsh, 2004): la primera es el mapa de camino que identifica un conjunto de rutas dentro del espacio libre (Álvarez et al, 2011), la segunda es la descomposición por celdas que discrimina entre las celdas libres y las celdas ocupadas por obstáculos y la tercera son los campos potenciales que proponen una función matemática sobre el espacio.

En el ámbito de la descomposición por celdas vale la pena destacar la técnica de descomposición aproximada la cual tiene como objetivo descomponer el entorno donde se mueve el robot en celdas hasta que no queden más celdas mixtas, es decir celdas que contienen parcialmente una región de un obstáculo en asocio con una región vacía (Siegwart y Nourbakhsh, 2004).

La eficiencia de esta técnica radica en que sólo basta con descomponer las celdas mixtas, es decir, una vez que se tiene una celda llena, ya no se procede a dividirla.

3. MODELO PROPUESTO DE PLANIFICACIÓN ABSTRACTA

Con el fin de mejorar la navegación del robot implementado en (Vélez *et al.*, 2011) a continuación se propone la integración de una técnica de planificación de tareas basada en costos con una técnica de planificación de trayectorias que hace uso de la descomposición aproximada para descomponer el entorno en que se mueve el robot.
Las dos técnicas de planificación anteriores se realiza en el marco de una simple jerarquía de abstracción del problema (Reyes, 2001). En esta jerarquía, la planificación de tareas se ubica en el nivel superior de la abstracción del sistema considerando en la especificación del problema solo la información básica del ambiente para encontrar una trayectoria inicial abstracta.

En contraste, la planificación de trayectoria que hace uso de la descomposición aproximada, se ubica en el nivel inferior de la abstracción del sistema considerando la trayectoria inicial abstracta junto con la información detallada del ambiente para identificar una trayectoria concreta que hará que el robot llegue a la localización objetivo cuando esta se ejecuta. La propuesta anterior se ve reflejada en la Fig. 1, donde se muestra la arquitectura básica de planificación abstracta propuesta detallándose el conocimiento que se requiere en cada una sus capas.

Capa 1 (Planificacion Abstracta)	Lograr un plan abstracto que alcance localización abstracta objetivo, teniendo en cuenta: - El conocimiento de las áreas que conforman el entorno del robot - El conocimiento de los puntos de conexión entre áreas - El conocimiento relacionado con la localización abstracta del Robot - El conocimiento sobre la localización objetivo
Capa 2 (Planificación de Movimientos)	Lograr un plan concreto que alcance la localización real objetivo teniendo en cuenta: - El plan abstracto valido obtenido en la capa 1. - El conocimiento de la ubicación de los obstáculos existentes en cada una de las áreas. - El conocimiento de las áreas que conforman el entorno del robot - El conocimiento relacionado con la localización real del Robot - El conocimiento sobre la localización real objetivo

Fig. 1. Arquitectura de planificación abstracta

En la arquitectura propuesta el conocimiento requerido se obtiene por parte del usuario al ingresar la siguiente información: (*i*) el área de trabajo, (*ii*) las áreas que conforman el área de trabajo, (*iii*) los puntos de conexión entre áreas y (*iv*) la localización de los obstáculos al interior de las áreas.

La especificación del área correspondiente al ambiente de trabajo del robot al igual que el de las áreas que la conforma, queda completamente determinada, al especificar el punto del extremo superior izquierdo y el extremo inferior derecho, en un plano de coordenadas (*x*, *y*). Se asume que las áreas manejadas en esta propuesta son cuadradas y su definición se basa en la definición de su diagonal

Para la especificación de los puntos de conexión entre áreas se requiere de: (*i*) la dupla de áreas conectadas y (*ii*) la ubicación del punto de conexión, en el mismo plano de coordenadas *(x, y).*

La localización de los obstáculos hace parte de la última etapa, donde se hace necesario especificar: (*i*) el área donde está ubicado cada uno y (*ii*) los nodos, es decir puntos *(x, y)* que determinan el obstáculo. A continuación se detallan las principales características de la arquitectura propuesta.

3.1 Planificación abstracta

La planificación abstracta requiere del uso de un planificador de tareas. Particularmente la arquitectura propuesta hace uso del planificador basado en costos LPG (Gerevini y Serina, 2000; Gerevini y Serina, 2002). Este planificador permite implementar la técnica de planificación basada en costos y requiere como datos de entrada los archivos *domain.pddl* y *problem.pddl* que describen respectivamente la definición del dominio y del problema a resolver.

Con el fin de configurar los archivos anteriores se implementó un modulo traductor que convierte el conocimiento entregado por el usuario a las especificaciones que conforman estos archivos. Para crear el archivo *domain.pddl*, el traductor define dos tipos de objetos que se pueden crear: *sitio* que representa un área que conforma el entorno y *robot* que representa al robot. Así mismo define dos funciones a manejar:(i) *recorrido*, para almacenar la distancia recorrida por el robot y (ii) *distancia*, para almacenar el costo del viaje de un área a otra.

Estas funciones permiten calcular el plan con la mejor calidad, que en el caso de esta arquitectura, será el menor *recorrido* que realice el robot. Para el caso de la función *distancia*, el traductor hace uso de una función que obtiene el costo del viaje de un área a otra. Dicha función considera el centro de las áreas como el punto de referencia, por lo cual el costo del viaje se define como la distancia entre ambos centros y se calcula usando la ecuación euclidiana de distancia (Asmar, 2007):

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (1)$$

Siendo (x_1,y_1) las coordenadas del punto 1 y (x_2,y_2) las coordenadas del punto 2

El traductor también permite crearlos predicados para denotar la ubicación actual del robot, marcar un área como visitada y puntualizar la existencia de una conexión entre áreas.

Por último, para conformar el dominio,el traductor crea también la acción *viajar*, la cual siempre y cuando se cumplan la restricciones planteadas permite calcular el desplazamiento que realizará de manera abstracta el robot. En la figura 2 se detalla un ejemplo de un archivo *domain.pddl* generado por el traductor.

```
(define (domain robotviajero)
(:requirements :strips :typing :fluents)
(:types sitio robot - object)
(:functions (recorrido ?a - robot)
(distancia ?a - sitio ?b - sitio))
(:predicates  (robotEn ?x - sitio ?a - robot)
(Visitada ?x - sitio)
(conexion ?m - sitio ?n - sitio))
(:action Viajar
    :parameters (?a - robot ?from - sitio ?to - sitio)
    :precondition (and (robotEn ?from ?a)
    (conexion ?from ?to))
    :effect (and (robotEn ?to ?a) (Visitada ?to)
    (not (robotEn ?from ?a))
            (increase (recorrido ?a)
            (distancia ?from ?to))))
)
```

Fig. 2. Archivo domain.pddl generado por el traductor

De manera similar el traductor construye el archivo *problem.pddl*. Para ello primero el traductor transforma el conocimiento relacionado las áreas que conforman el entorno de trabajo del robot y la información del robot para crear los objetos del problema.

Así mismo, haciendo uso de la información de las áreas, sus conexiones, la posición abstracta inicial del robot que representa el centroide del área en la cual se encuentra el robot y que áreas ha visitado, el traductor define el estado inicial.

Para definir el estado objetivo el traductor toma del usuario el conjunto de objetivos que representan áreas que se desean sean visitadas por el robot. Junto a este conjunto de áreas el traductor incluye como métrica para minimizar el recorrido del robot. En la Fig. 3 se muestra un ejemplo de un archivo *problem.pddl* generado por el traductor.

```
(define (problem rutaapartamento)
(:domain robotviajero)
(:objects area1 area2 area3 area4 area5
    area6 - sitio nxt - robot)
(:init (robotEn area1 nxt)(Visitada area1 )
(=(recorrido nxt) 0)
(conexion area1 area4)(conexion area4 area1)
(conexion area1 area5)(conexion area5 area1)
(conexion area2 area3)(conexion area3 area2)
(conexion area2 area4)(conexion area4 area2)
(conexion area3 area4)(conexion area4 area3)
(conexion area4 area6)(conexion area6 area4)
(conexion area5 area6)(conexion area6 area5)
(=(distancia area1 area2) 14.577379737113251)
(=(distancia area2 area1) 14.577379737113251)
(=(distancia area1 area3) 27.5)
(=(distancia area3 area1) 27.5)
(=(distancia area1 area4) 14.577379737113251)
(=(distancia area4 area1) 14.577379737113251)
(=(distancia area1 area5) 25.124689052802225)
(=(distancia area5 area1) 25.124689052802225)
(=(distancia area1 area6) 33.63406011768428)
(=(distancia area6 area1) 33.63406011768428)
(=(distancia area2 area3) 16.77050983124842)
(=(distancia area3 area2) 16.77050983124842)
(=(distancia area2 area4) 15.0)
(=(distancia area4 area2) 15.0)
(=(distancia area2 area5) 34.00367627183861)
(=(distancia area5 area2) 34.00367627183861)
(=(distancia area2 area6) 34.00367627183861)
(=(distancia area6 area2) 34.00367627183861)
(=(distancia area3 area4) 16.77050983124842)
(=(distancia area4 area3) 16.77050983124842)
(=(distancia area3 area5) 35.35533905932738)
(=(distancia area5 area3) 35.35533905932738)
(=(distancia area3 area6) 25.495097567963924)
(=(distancia area6 area3) 25.495097567963924)
(=(distancia area4 area5) 20.155644370746373)
(=(distancia area5 area4) 20.155644370746373)
(=(distancia area4 area6) 20.155644370746373)
(=(distancia area6 area4) 20.155644370746373)
(=(distancia area5 area6) 20.0)
(=(distancia area6 area5) 20.0)
)
(:goal (and (Visitada area1)(Visitada area2)
(Visitada area3)(Visitada area4)(Visitada area5)
(Visitada area6)))
(:metric minimize (recorrido nxt)))
```

Fig. 3. Archivo problem.pddl generado por el traductor

Luego de conformado los anteriores archivos, se ejecuta el planificador LPG y se obtiene un archivo que contiene el plan basado en costos que permite identificar el plan abstracto con el mínimo recorrido. En la Fig. 4 se muestra un ejemplo de un del contenido generado por el planificador LPG asociado a un plan solución.

```
0:   (VIAJAR NXT AREA1 AREA5) [1]
1:   (VIAJAR NXT AREA5 AREA6) [1]
2:   (VIAJAR NXT AREA6 AREA4) [1]
3:   (VIAJAR NXT AREA4 AREA2) [1]
4:   (VIAJAR NXT AREA2 AREA3) [1]
```

Fig. 4. Muestra de un plan solución generado por el planificador LPG

El plan obtenido se almacena en un archivo de texto plano y consiste en indicar en cada paso de que área *i* a que área *j* debe viajar el robot. Es así como, cada paso del plan lo que hace es indicarle al robot que vaya del centroide de un área *i* al centroide de un área *j*.

3.2 Planificación de movimientos

Con esta planificación se busca obtener una ruta posible que tenga en cuenta la distribución del área de trabajo y los obstáculos que se encuentran en estas áreas con el fin de alcanzar la posición real objetivo.

Esta planificación hace uso del conocimiento suministrado desde la capa de planificación abstracta (el plan abstracto) y el conocimiento entregado por el usuario consistente en la ubicación de los obstáculos existentes en cada una de las áreas, la definición de las áreas que conforman el entorno del robot, la localización real del robot y la localización real objetivo. Esta tarea de planificación hace uso de la aplicación *appCells* desarrollada en (Vélez *et al.*, 2011).

En este módulo, por cada acción del plan abstracto, se realiza de manera iterativa una planificación de movimiento que tenga en cuenta los obstáculos contenidos en las áreas que conforman la especificación de dicha acción abstracta. Esta planificación de movimiento se realiza con la aplicación *appCells*.

Con el fin de configurar la información requerida por la aplicación *appCells*, se implementó un modulo traductor que convierte la información contenida en cada acción del plan abstracto y el conocimiento entregado por el usuario, a la información requerida por esta aplicación la cual consiste en (*i*) El punto inicial(ubicación actual) y el punto final(ubicación objetivo), (*ii*) Dos puntos que determinan la margen del entorno de trabajo, (*iii*) Cantidad de obstáculos, (*iv*) Nodos por cada obstáculo, (*v*) Nodos del obstáculo

Luego de la traducción se hace uso de *appCells* para planificar una ruta entre las dos áreas asociadas a la acción abstracta correspondiente con base en el conocimiento suministrado. Este proceso se realiza hasta no tener más acciones. Al final de este proceso iterativo se obtiene una trayectoria libre de obstáculos que permite al robot ir desde su posición inicial hasta su posición objetivo. El resultado de este proceso se almacena en un archivo de texto de manera similar al del plan abstracto con la diferencia que en vez de especificar dos áreas se especifican dos puntos en forma de *(x, y)* cada uno.

El archivo generado luego es traducido como un programa de ejecución en términos del sistema operativo propio del robot que en este caso al ser el mismo que se utilizó en el trabajo de (Vélez et al., 2011) corresponde al sistema Java-Lejos. Por razones de extensión la traducción de este plan no se muestra.

4. IMPLEMENTACIÓN DEL MODELO DE PLANIFICACIÓNABSTRACTA

Con el fin de evaluar el desempeño de la arquitectura de planificación abstracta propuesta y compararlo con el planificador desarrollado (Vélez *et al.*, 2011) se realizó la implementación de esta arquitectura.

La implementación se realizó en Java. En la Fig. 5 se muestra la apariencia que tiene la interfaz gráfica habilitada para la interacción con el usuario. En la columna izquierda se encuentran las opciones de inclusión de áreas, conexiones y obstáculos, mientras que en la columna derecha se habilita la selección de objetivos, generar y cargar el plan solución al igual que enviar el plan real a su ejecución.

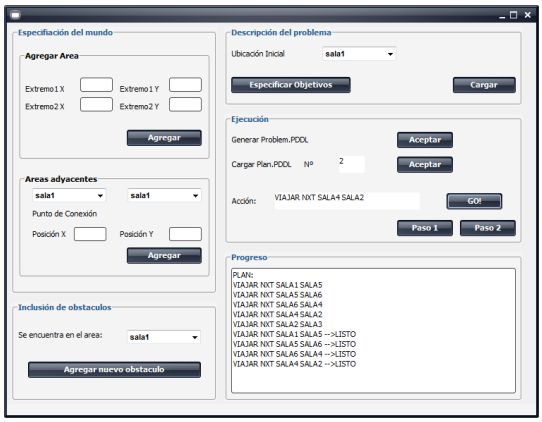

Fig. 5. Imagen de la apariencia de la interfaz gráfica de usuario.

La implementación de esta arquitectura permite que durante el cálculo del proceso de planificación de movimientos de la capa 2 el usuario pueda observar de forma gráfica su cálculo como se observa en, como en la Fig. 6, gracias a uno de los componentes de *appCells*.

Esto le facilita al usuario la interpretación de los movimientos que deberá realizar el robot al igual que le permite comprobar que el dibujo generado con las coordenadas ingresadas esté en concordancia con el sitio real.

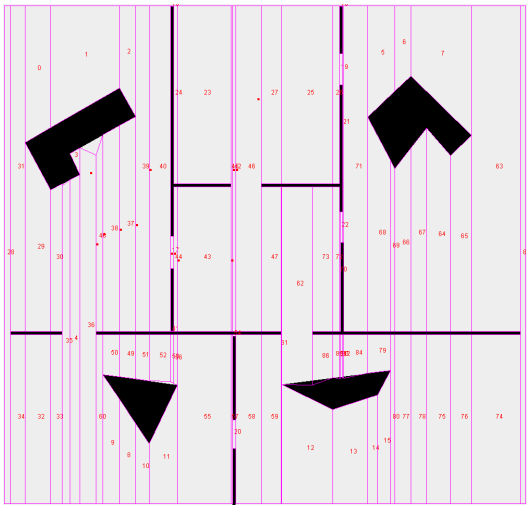

Fig. 6. Muestra de la ventana generada por appCells

Para llevar el proyecto a un nivel más práctico y poder realizar pruebas de la arquitectura propuesta su implementación se instaló en un modelo de robot móvil similar al utilizado en (Vélez *et al.*, 2011) el cual se basa en la plataforma lego NXT como se muestra en la Fig. 7.

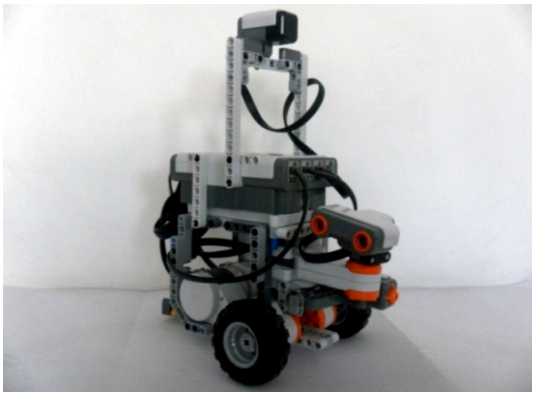

Fig. 7. Robot NXT utilizado en las pruebas

5. EVALUACIÓN DELMODELO DE PLANIFICACIÓNABSTRACTA

Con el fin de generar un problema que permita estudiar la navegación del robot móvil desarrollado, se implementó un ambientede trabajo en forma de cuadrado, el cual está dividido en 6 áreas también cuadradas, con conexiones definidas sólo entre algunas de ellas (área 1 y 4; área 1 y 5; área 2 y 3; área 2 y 4; área 3 y 4; área 4 y 6; área 5 y 6).

Adicionalmente se incluyeron algunos obstáculos en las áreas 1, 3, 5 y 6 como se detalla en la Figura 8.

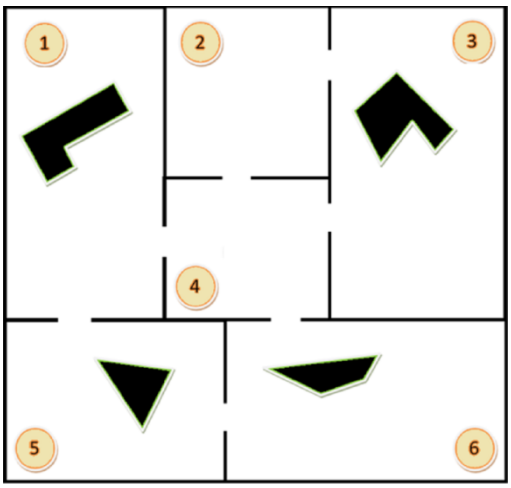

Fig. 8. Ejemplo del Entorno de Trabajo

La posible ventaja más evidente del uso del modelo de planificación abstracta propuesto frente al modelo original propuesto en (Vélez *et al.*, 2011) recae sobre la disminución del área de trabajo.

Por ejemplo, si el robot desea viajar del área 1 al área 2 de la Fig. 8, el plan abstracto le dirá que trabaje solo sobre las áreas que le interesan, es decir el área 1, 2 y 3. Por el contrario no disponer de un plan implica que un planificador de movimientos como *appCells* deba razonar sobre el área de trabajo total, incluso en zonas por las que el robot nunca va a pasar aumentando gradualmente la complejidad del problema causando en ocasiones que no se pueda llegar a una solución.

Con el fin de demostrar lo anterior, se realizó una evaluación del rendimiento del modelo de planificación abstracta propuesto frente al modelo de *appCells*. Esta evaluación consistió en proponer 10 rutas basadas en el mapa de prueba mostrado en la Fig. 8. y evaluar las diferencias en el tiempo de cómputo que le toma a cada una de las aplicaciones llegar a una solución.

De las 10 rutas se obtuvo como resultado que usando la planificación abstracta se logró una solución para cada caso, es decir una efectividad del 100%, mientras que el *appCell* no obtuvo una solución en 3 casos obteniendo una efectividad del 70 %.

Los resultados de los 7 casos en que ambos encontraron una solución se relacionan en la tabla 1 y la figura 9, donde el tiempo para calcular la ruta de navegación está expresado en milisegundos. Estos resultados muestran que el uso de la planificación abstracta presenta en promedio un menor tiempo de cómputo frente al modelo propuesto en *appCells*.

Tabla 1: Tiempo de cómputo

Ruta	Tiempo con Planificador (mseg)	Tiempo sin Planificador (mseg)
1.Area 1 - Area 2	1364	1897
2.Area 1 - Area 3	1381	1407
3.Area 1 - Area 4	732	1253
4.Area 1 - Area 5	736	1304
5.Area 1 - Area 6	1439	1917
6.Area 3 - Area 2	676	1224
7.Area 3 - Area 4	662	1382

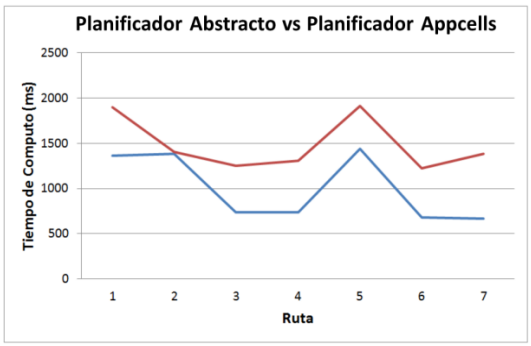

Fig. 9. Desempeño computacional del planificador abstracto (azul) y appCells (rojo)

6. CONCLUSIONES

En este trabajo se propuso una arquitectura de planificación que integra dos técnicas de planificación en el marco de una simple jerarquía de abstracción del problema. Estas técnicas fueron la planificación de tareas basada en casos y planificación de trayectorias que hace uso de la descomposición aproximada para descomponer el entorno en que se mueve el robot.

El modelo de planificación propuesto de implementó en un ambiente java el cual presenta dos capas básicas cada una asociada a una de las técnicas de planificación utilizadas. El conocimiento manejado por cada capa difiere en el nivel de detalle de los mismos.

En la evaluación del modelo de planificación abstracta propuesto se comparó con el *appCells* obteniendo que los resultados del modelo de planificación abstracta propuesto son mejores en el sentido de que siempre retornará una solución si esta existe. Adicionalmente se encontró que el modelo propuesto en este trabajo en promedio es más rápido en encontrar una solución.

Como trabajo futuro con el fin de mejorar se propone estudiar el comportamiento del modelo de planificación propuesto en ambientes inciertos para lo cual se pretende implementar una capa reactiva que haría uso de sensores que se pueden adaptar al robot con el fin detectar y evitar obstáculos que no fueron definidos en la especificación de mundo.

También se propone ampliar el rango de funcionalidades del robot implementando nuevas acciones en el proceso de planificación. De esta forma se permite que el robot sea de propósito más general y este en capacidad de transportar objetos, mensajes, interactuar con personas, puertas, ascensores, entre otros.

Los avances en el campo de la visión artificial serian de gran utilidad en un sistema autónomo como el que aquí se expone, dado que puede dotar al robot de una visualización del entorno en tiempo real, haciéndolo tolerante a ambientes dinámicos.

RECONOCIMIENTO

Este trabajo es apoyado por el proyecto "Fortalecimiento del Grupo de Investigación Sistemas Inteligentes Web - SINTELWEB" con código académico 120501004 de la Convocatoria Nacional apoyo para el fortalecimiento de Grupos de Investigación o Creación Artística que soporten programas de posgrado de la Universidad Nacional de Colombia – 2012.

REFERENCIAS

Álvarez, D. Torres, I y Guzmán, J. (2011) "Aplicando Algoritmos de Visibilidad para la Planificación de Movimientos en un Robot Lego", presentado en la XI Latín American Robotics Competition, Bogotá, Colombia.

Asmar C, A. Et al. (2007)Geometría Vectorial y Analítica. Una Introducción al Álgebra Lineal. Universidad Nacional de Colombia, Sede Medellín.

Borrajo, D., Juristo, N., Martínez, V., y Pazos, J. (1993) Inteligencia Artificial. Métodos y Técnicas. Centro de Estudios Ramón Areces, Madrid.

Bouguerra, A. (2008) "Robust Execution of Robot Task-Plans: A Knowledge-based Approach" Örebro Studies in Technology 32, 175 pp.

Coltin, B., Velsos, M. y Ventura, R.(2011) "Dynamic User Task Scheduling for Mobile Robots". AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots at AAAI 2011, San Francisco.

Choset, H. Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E y Thrun., S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press.

Fuentetaja, R. (2010). Búsqueda Heurística en Planificación Basada en Costes. Tesis de doctorado. Universidad Carlos III de Madrid. España.

Gat, E. (1992). Integrating Planning and Reaction in a Heterogeneous Asynchronous Architecture for Controlling Mobile Robots. Tenth National Conference on Artificial Intelligence (AAAI), pp. 809- 815.

Gerevini, A. y Serina, I (2000). Fast plan adaptation through planning graphs: Local and systematic search techniques. Proceedings of the Conference on AI Planning Systems (AIPS), pages 112-121.

Gerevini, A. y Serina, I. (2002). "LPG: A Planner Based on Local Search for Planning Graphs with Action Costs", Dipartimento di Elettronica per l'Automazione, Universit`adegliStudi di Brescia.

Ghallab, M., Nau, D. y Traverso, P. (2004) Automated Planning: Theory&Practice. Morgan Kaufmann Publishers, 663 pages.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D. (1998) "PDDL | The Planning Domain Defnition Language", Ecole Nationale Superieure D'ingenieur des Constructions Aeronautiques.

Gutiérrez, M. (2004). "Cálculo de caminos óptimos para manipuladores articulados", Universidad de salamanca, Departamento de Informática y Automática.

Long, D. y Fox, M. (2002). Progress in AI planning research and applications. UPGRADE. The European Online Magazine for the IT Professional, 3(5):10-24.

Reyes, B. (2001). "Guía de Conductas Reactivas Mediante Planes Abstractos en un Problema de Navegación Robótica", Instituto de Investigaciones Eléctricas.

Siegwart, R. y Nourbakhsh, I. (2004). *Introduction to autonomous mobile robots*, Editorial MIT Press. Estados Unidos de América.

Vélez, C. Guzmán, J. y Torres, I. (2011) "Experiencias en la Implementación de las Técnicas de Descomposición Aproximada y Trapezoidal para la Navegación en Robots Móviles Lego". XI Latin American Robotics Competition. Bogotá, Colombia.