TÉCNICAS DE INTELIGENCIA ARTIFICIAL PARA LA SOLUCIÓN DE LABERINTOS DE ESTRUCTURA DESCONOCIDA.

New Artificial intelligence techniques for solution of labyrinths with unknown structure

RESUMEN
Debido a que la búsqueda es el núcleo de muchos procesos inteligentes, es necesario escoger la estructura de control apropiada con el fin de que el proceso de búsqueda sea eficiente. La inteligencia artificial proporciona varias técnicas de búsqueda que tienen una formulación matemática, la cual hace posible su implementación computacional bajo el esquema de programación estructurada. En este trabajo se presentan las técnicas de búsqueda en amplitud y en profundidad, las cuales son técnicas de inteligencia artificial, para la solución de un problema de gran complejidad matemática como lo es la solución de un laberinto de estructura desconocida.

PALABRAS CLAVES: Búsqueda en espacios desconocidos, laberitos, técnicas de inteligencia artificial.

ABSTRACT
Because of the search process is a central problem in many intelligent processes, it is necessary to choose the appropriate control structure in order that the search process will be efficient. The artificial intelligence provides some search techniques which have a mathematical formulation, in which is possible the computational implementation, under the structured programming. This paper presents the techniques of amplitude and depth search, which are artificial intelligence techniques, to solve a problem with high mathematical complexity which is the solution of a labyrinth of unknown structure.

KEYWORDS: *Artificial intelligence techniques, labyrinth, search in unknown spaces.*

JASON MOLINA VARGAS
Ingeniero Electricista, M. Sc (c).
Analista
Xm los expertos en mercado
jmolina@xm.com.co

CARLOS TORRES PINZÓN
Ingeniero Electricista, M. Sc (c).
Jefe sección proyectos dpto. E&A
Ingenio la Cabaña S.A.
catorres@ingeniolacabana.com

CARLOS RESTREPO PATIÑO
Ingeniero Electricista, M. Sc.
Ph. D.(c) en Ingeniería Electrónica.
Universidad Rovira I Virgili
carlos.restrepo@urv.cat

Grupo de investigación de electrónica de potencia
Universidad Tecnológica de Pereira

1. INTRODUCCIÓN

Un laberinto es un lugar formado por calles y encrucijadas, intencionalmente complejo para confundir a quien se adentre en él. El hombre desde la antigüedad se ha preocupado por diseñar intrincados laberintos y al mismo tiempo por buscar formas para recorrerlos sin extraviarse. Un ejemplo muy claro de esto lo proporciona la mitología griega en la cual se describe cómo Dédalo construyó un laberinto muy complicado para el rey Minos en el cual escondía este un minotauro y el cual fue hallado por Teseo. En la actualidad el interés por los laberintos ha despertado nuevamente y esto se debe a la búsqueda del hombre por diseñar robots que realicen actividades de forma más inteligente.

Para solucionar un laberinto de estructura conocida se pueden emplear muchos métodos como llenar los caminos sin salida, este es un algoritmo simple, muy rápido y que no requiere mucha memoria. Otro método de solución muy conocido es el de recorrer la pared y cada vez que hay un cruce se gira a la derecha, este algoritmo es muy empleado por los robots que solucionan laberintos debido a su simple implementación. Otros métodos menos empleados son los de rehacer caminos, eliminar colisiones y moverse al azar [1]. Dado que la solución de un laberinto de estructura desconocida es un problema muy complejo se emplean técnicas de inteligencia artificial para la búsqueda de la solución. En este artículo se hace una descripción general sobre la inteligencia artificial centrándose en un sistema de producción para describir las estrategias de control empleadas para la solución de laberintos. Finalmente se presenta en los resultados una comparación en tiempo de solución entre las técnicas empleadas para diferentes laberintos.

2. DEFINICIÓN DEL PROBLEMA

El problema que se quiere abordar en este artículo es la solución de un laberinto de estructura desconocida. A pesar de que es previsible que la mayoría de la gente únicamente con esta sentencia actúe correctamente, la definición del problema tal y como está, es incompleta. Para construir un programa que solucione laberintos, primero se debe especificar la posición inicial en el

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

laberinto, las reglas que definen los movimientos legales y la posición en el laberinto que representa la solución [2,3]. Además, se debe especificar previamente el objetivo implícito de no realizar movimientos legales dentro del laberinto, sino solucionar el laberinto, si es posible. Es bastante fácil dar una descripción formal y completa del problema; en la posición inicial, el móvil se encuentra inmerso en una matriz lógica de tamaño desconocido en la cual se representa una pared por un uno y un camino por un cero. El móvil será capaz de levantar un mapa de su entorno y agregar componentes a la matriz inicial sólo en posiciones verticales u horizontales a medida que explora el laberinto y de esta forma va aumentando el tamaño de la matriz lógica [4]. El móvil sólo se podrá mover a una posición de la matriz si la posición a la que quiere llegar tiene un cero, mediante estos movimientos legales se puede llegar a la solución del laberinto partiendo de la posición inicial.

Se ha definido el problema de solucionar laberintos como un problema de movimientos a través de un espacio de estados, donde cada estado corresponde a un movimiento legal dentro del laberinto [5]. La representación como espacio de estados forma la base de la mayoría de los métodos de inteligencia artificial (IA), y su estructura corresponde con aquella de resolver problemas por dos importantes razones:

- Permite definir formalmente el problema, mediante la necesidad de convertir alguna situación dada en la situación deseada usando un conjunto de operaciones permitidas.
- Permite definir el proceso de resolver un problema como una combinación de técnicas conocidas, representadas por una regla que define un movimiento en el espacio de búsqueda. La técnica general de exploración en el espacio intenta encontrar alguna ruta desde el estado actual hasta un estado objetivo. La búsqueda es un proceso de gran importancia en la resolución de problemas difíciles para los que no se dispone de técnicas más directas.

El problema puede resolverse con el uso de las reglas en combinación con una estrategia apropiada de control para trasladarse a través del espacio problema hasta encontrar una ruta desde un estado inicial hasta un estado objetivo [6]. De esta forma, el proceso de búsqueda es fundamental en la solución de problemas. La búsqueda es un mecanismo general que puede utilizarse cuando no se conoce otro método más directo [7,8]. Si la estructura del laberinto se conoce de antemano, se podrían emplear técnicas heurísticas para solucionar el laberinto optimizando el recorrido, pero debido a que el laberinto es desconocido no se puede plantear una función objetivo para optimizar y es por esto que se debe emplear un método de búsqueda.

3. TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Una técnica de IA es un método que utiliza conocimiento expresado, de tal forma que represente las generalizaciones, que sea comprendido por las personas que lo proporcionan, que pueda modificarse fácilmente para corregir errores y reflejar los cambios en el mundo y en nuestra visión del mundo, que pueda usarse en gran cantidad de situaciones aun cuando no sea totalmente preciso o completo y que pueda usarse para ayudar a superar su propio volumen, ayudando a acotar el rango de posibilidades que normalmente deben ser consideradas [9,10].

Aunque las técnicas de IA deben diseñarse de acuerdo con las restricciones impuestas por los problemas de IA, existe cierto grado de independencia entre los problemas y las técnicas para su solución. Los programas que solucionan problemas de IA ponen en manifiesto tres importantes técnicas que son: la búsqueda, el uso del conocimiento y la abstracción [1,8]. La búsqueda proporciona una forma de resolver problemas en los que no se dispone de un método más directo, el uso del conocimiento resuelve problemas complejos aprovechando las estructuras de los objetos involucrados y la abstracción proporciona una forma de separar aspectos y variaciones importantes de aquellos otros sin importancia y que en caso contrario podrían hacer colapsar un proceso [3].

Para solucionar problemas complejos, los programas que utilizan las técnicas de IA presentan numerosas ventajas frente a los que no lo hacen. Los primeros son más robustos; no se equivocarán frente a una pequeña perturbación en la entrada. El conocimiento del programa es comprendido fácilmente por la gente. Estos programas pueden trabajar con facilidad en grandes problemas en donde los métodos directos fallan [2,8].

Los esfuerzos dedicados a construir programas que lleven a cabo tareas de la misma forma que el hombre, se dividen en dos clases. Los programas de la primera clase se encargan de aquellos problemas que una computadora puede resolver fácilmente, pero cuya solución implica el uso de mecanismos de los que no dispone el hombre; este tipo de problemas no se adapta mucho con la definición de tarea pertinente a la IA. La segunda clase de programas son los que intentan modelar el comportamiento humano, ellos realizan tareas que se adaptan claramente con la definición de tareas de IA, haciendo tareas que no son triviales para una computadora [3,10].

Hay varias razones para querer modelar la forma de trabajar humana para llevar a cabo estas tareas: verificar las teorías psicológicas de la actuación humana, capacitar las computadoras para comprender el comportamiento humano, capacitar la gente para comprender a las

computadoras y explotar el conocimiento que se puede buscar en el hombre [2]. Esta última motivación es probablemente la más importante, debido a que el hombre es el sistema del que mejor se conoce el cómo lleva a cabo las tareas con las que estamos familiarizados y por lo tanto tiene sentido fijarse en él para buscar pistas de cómo actuar. Si lo que se quiere es escribir programas que simulen el comportamiento humano ante una tarea, la forma de medir el éxito está en que el comportamiento del programa corresponda con el humano, así como mediante distintas clases de experimentos y análisis de protocolos. En este sentido no se busca un programa que simplemente sea tan bueno como sea posible, se busca un programa que falle donde la gente lo hace [8].

Es necesario aclarar que la pregunta de si una máquina posee inteligencia o puede pensar, es demasiado difusa para poder contestarla con precisión. Sin embargo, con frecuencia es posible construir un programa que encuentre un criterio de rendimiento para una tarea concreta. Esto no significa que el programa realice la tarea de la mejor manera posible, sólo significa que se entiende como mínimo una forma de hacer al menos parte de una tarea. Cuando se quiere diseñar un programa de IA, se debe intentar especificar tan bien como sea posible el criterio de éxito para el funcionamiento del programa en su particular y restringido dominio [1].

Para construir un sistema que resuelva un problema específico, es necesario realizar estas cuatro acciones:

- Definir el problema con precisión. La definición debe incluir especificaciones precisas tanto sobre las situaciones iniciales como sobre las situaciones finales que se aceptarían como soluciones al problema.
- Analizar el problema. Algunas características de gran importancia pueden tener un gran efecto sobre la conveniencia o no de utilizar diversas técnicas que resuelvan el problema.
- Aislar y representar el conocimiento necesario para resolver el problema.
- Elegir la mejor técnica que resuelva el problema y aplicarla al problema particular.

4. SISTEMA DE PRODUCCIÓN

Debido a que la búsqueda es el núcleo de muchos procesos inteligentes, es adecuado estructurar los programas de IA de forma que se facilite describir y desarrollar el proceso de búsqueda. Los sistemas de producción proporcionan tales estructuras [2]. Un sistema de producción consiste en:

- Un conjunto de reglas.
- Una o más bases de datos/conocimiento.
- Una estrategia de control que especifique el orden en el que las reglas se comparan con la base de datos, y la forma de resolver los conflictos que surjan cuando varias reglas puedan ser aplicadas a la vez.
- Un aplicador de reglas.

El proceso de solución del problema puede modelarse como un sistema de producción. El problema que se plantea es escoger la estructura de control apropiada para el sistema de producción con el fin de que el proceso de búsqueda sea lo más eficiente posible [1,2].

4.1 Estrategias de control

El primer requisito que debe cumplir una buena estrategia de control es que cause algún cambio, las estrategias de control que no causen cambio de estado nunca alcanzan la solución. El segundo requisito que debe cumplir una buena estrategia de control es que sea sistemática [10].

Para tener una buena claridad de las estrategias de control en la solución de laberintos, se presenta en la Figura 1, a modo de ejemplo un laberinto que un móvil, representado por un círculo negro, desea solucionar; siendo la solución del laberinto el triángulo negro. Las paredes del laberinto son cuadros grises y las posiciones en el laberinto se representan por una letra acompañada por un número, de esta forma la posición inicial se localiza en E9 y la posición final en H1. En la figura 2 se presenta un árbol de búsqueda para el laberinto de la Figura 1. En la raíz del árbol se encuentra el estado inicial, todas las ramificaciones de la raíz se generan al aplicar cada una de las reglas al estado inicial. Cada bifurcación en una rama representa un punto del laberinto con varios caminos y una rama de la que no sale ninguna bifurcación puede ser un callejón sin salida o la solución.

4.1.1 Búsqueda en amplitud

Este método va construyendo un grafo de estados explicito mediante la aplicación de los operadores disponibles al nodo inicial, después aplica los operadores disponibles a los nodos sucesores directos del nodo inicial, y así sucesivamente [5,6]. Este procedimiento de búsqueda actúa de manera uniforme a partir del nodo inicial.

Este tipo de búsqueda consiste en ir explorando el árbol por ramas del mismo nivel, es decir, no se podrá explorar una rama superior si la rama inferior no se ha explorado por completo.

La búsqueda en amplitud no queda atrapada explorando callejones sin salida, además, si existe una solución la búsqueda en anchura garantiza que se encuentre. Si existen múltiples soluciones se encuentra la solución mínima, es decir, la que requiera el mínimo número de pasos. Esto está garantizado por el hecho de que no explora una ruta larga hasta que se hayan examinado todas las rutas más cortas que ella.

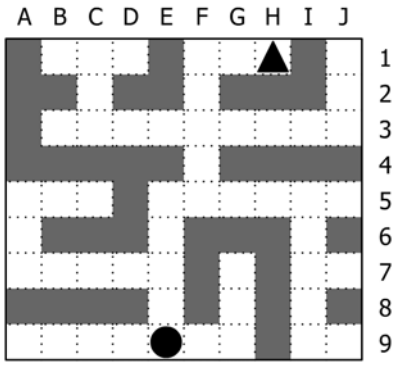

Figura 1. Laberinto.

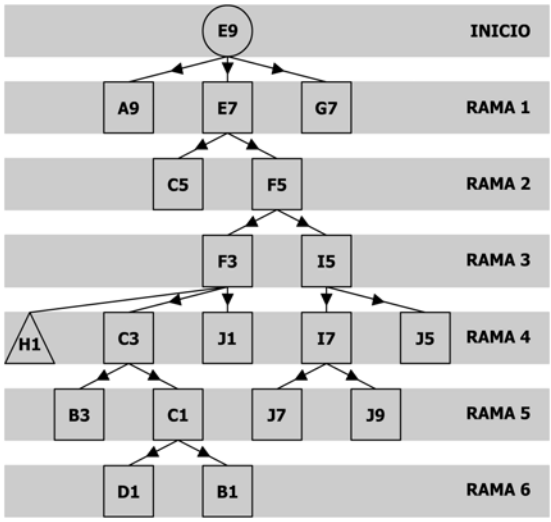

Figura 2. Árbol de búsqueda

4.1.2 Búsqueda en profundidad

En este proceso de búsqueda se genera sólo un sucesor del nodo en cada paso, es decir, cada vez que se obtiene un nuevo sucesor, se le aplica a este un nuevo operador y se obtiene un nuevo sucesor, y así sucesivamente.

En este tipo de búsqueda se avanza por una sola rama del árbol hasta que se encuentre una solución o hasta que se llegue a un callejón sin salida [3]. En el caso de llegar a un callejón sin salida se retorna hasta la raíz para iniciar una nueva búsqueda. La búsqueda en profundidad necesita menos memoria ya que sólo almacena los nodos del camino que se siguen en ese instante. Esto contrasta con la búsqueda en anchura en la cual debe almacenarse todo el árbol que haya sido generado hasta ese momento.

5. SOFTWARE DESARROLLADO

Se desarrollo utilizando Wx Dev C++ un software para el estudio de los diferentes métodos de solución de laberintos llamado Teseo. En la Figura 3 se presenta la interfaz gráfica de Teseo con la cual se pueden construir diferentes tipos de laberintos como el que se muestra, el cual equivale al de la Figura 5.

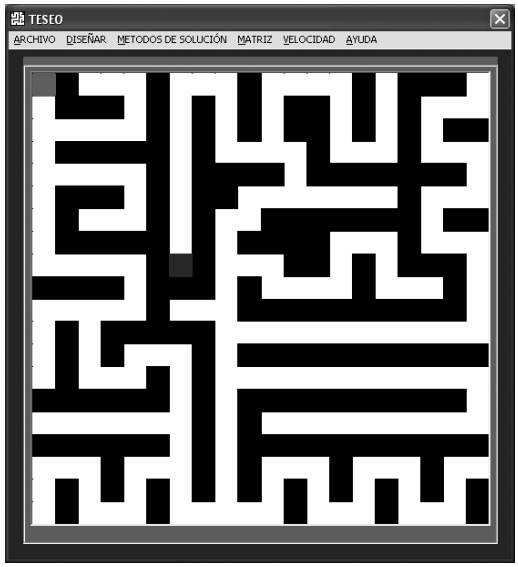

Figura 3. Panel inicial de Teseo.

En la Figura 4 se ilustra las diferentes funciones de Teseo, entre las que se destaca la posibilidad de construir cualquier tipo de laberinto, almacenarlo, cargar laberintos previamente diseñados, escoger un método de solución para el laberinto como la velocidad a la que se quiere solucionar, también es posible visualizar las matrices de diseño la cuales contiene las coordenadas de cada uno de los muros del laberinto y la del recorrido en la cual se muestra la solución del laberinto dada por el método de solución seleccionado.

6. RESULTADOS

En la Figura 5 se muestra un laberinto que contiene circuitos cerrados conocido como simplemente conectado. Se presentan en la tabla 1 los tiempos de cada una de las técnicas de búsqueda en amplitud y profundidad para cinco diferentes intentos. En la Tabla 1 se observa que la búsqueda en amplitud es un método que consigue los mejores tiempos en promedio y en todos los intentos se observa que es una técnica con tiempos muy uniformes. Para llegar a la solución del laberinto la búsqueda en amplitud realizó un recorrido completo del laberinto.

En la Figura 6 se muestra un laberinto que contiene muros separados conocido como conexiones múltiples. Se muestran en la Tabla 2, los resultados para cinco intentos de solución empleando la búsqueda en amplitud y profundidad. Este tipo de laberinto no puede ser solucionado empleando el usual método de recorrer una pared y girar siempre en una misma dirección en las

bifurcaciones ya que su estructura de muros separados ocasiona un recorrido en círculos sobre la misma trayectoria sin llegar a la solución.

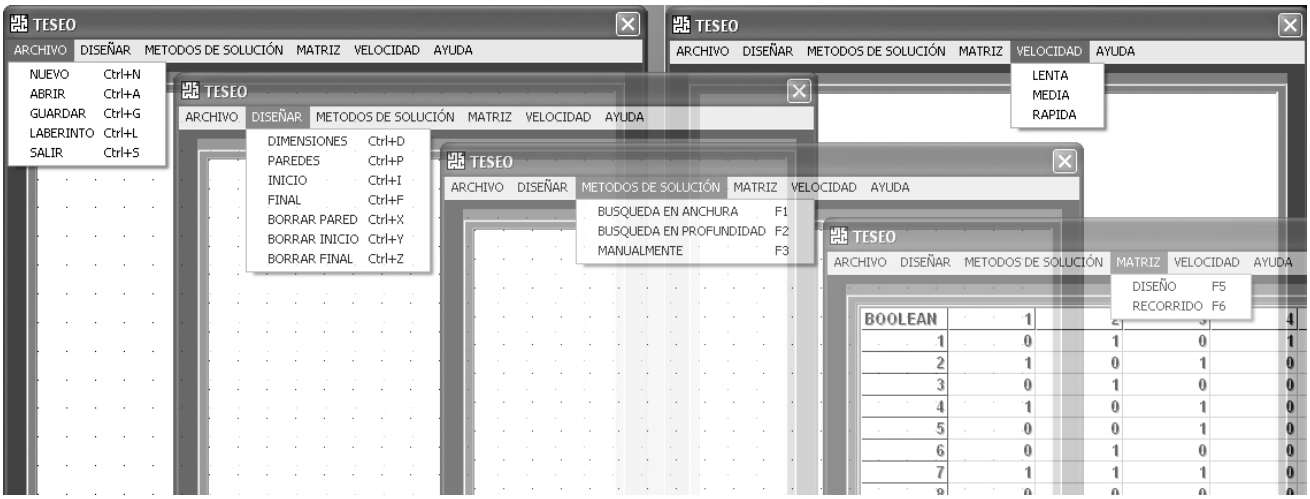

Figura 4. Funciones de Teseo.

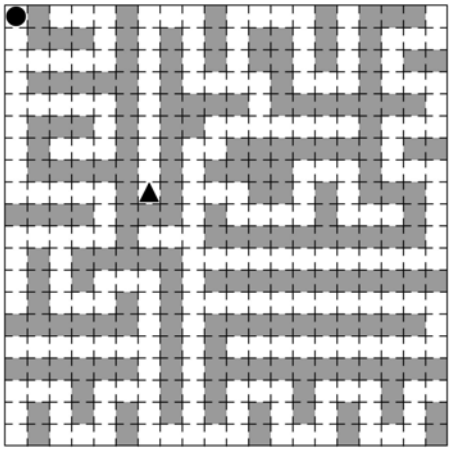

Figura 5. Laberinto del tipo simplemente conectado

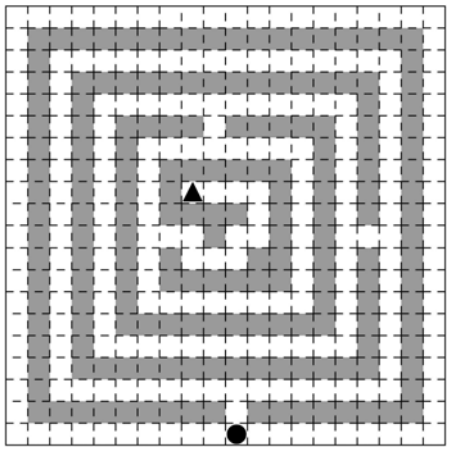

Figura 6. Laberinto del tipo conexiones múltiples.

Amplitud Tiempo [s]	Profundidad Tiempo [s]
17.09	33.12
15.76	42.35
16.46	37.95
16.71	31.95
17.55	38.17
16.71±0.67	36.70±4.21

Tabla 1. Resultados para el laberinto *simplemente conectado*

Se observa de la Tabla 2 que la búsqueda en amplitud obtiene en promedio el mejor tiempo para la solución del laberinto, aunque también se observa que el mejor tiempo de un intento lo obtuvo la búsqueda en profundidad, con un tiempo muy inferior a los empleados por la búsqueda en amplitud. Este último tiempo se obtuvo recorriendo la ruta mínima entre el objetivo y el punto inicial.

Amplitud Tiempo [s]	Profundidad Tiempo [s]
20.74	36.93
21.18	21.35
19.82	3.94
19.60	28.45
21.67	26.93
20.60±0.88	23.52±12.30

Tabla 2. Resultados para el laberinto *conexiones múltiples*

7. APORTES Y CONCLUSIONES

Se desarrolló un software con el cual se pueden diseñar diferentes tipos de laberintos para hacer una comparación entre las diferentes técnicas de inteligencia artificial de búsqueda y de esta forma lograr un mejor entendimiento de las mismas. Además fueron planteados dos métodos de solución de laberintos usando técnicas de inteligencia artificial, los cuales son la búsqueda en profundidad y amplitud.

El método de búsqueda en amplitud obtiene en promedio los mejores tiempos de solución a pesar de tener que recorrer todo el laberinto empleando más memoria para realizar la búsqueda, por otro lado, la búsqueda en profundidad es más aleatoria en cuanto al tiempo de solución, no obstante puede obtener mejores tiempos que la búsqueda en amplitud y menor cantidad de memoria, pero en promedio sus tiempos de búsqueda son mayores. Dependiendo del tipo de laberinto ingresado, cada método empleado entrega una solución mejor que la otra.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] S. J. Russell, P. Norvig, “Inteligencia Artificial: un enfoque moderno,” Mexico, Prentice Hall Hispanoamericana, S.A. 1996.

[2] R. S. Aylett, G. J. Petley, P. W. H. Chung, B. Chen, D. W. Edwards, “AI planning: solutions for real world problems,” Knowledge-Based Systems, Elsevier, Vol. 13, No. 2, 2000, pp. 61-69(9).

[3] D. C. Dracopoulos, “Neural robot path planning: The maze problem,” Neural Computing & Applications, Springer London, ISSN0941-0643 (Print) 1433-3058, Vol. 7, No. 2, June, 1998, pp.115-120, April 06, 2005.

[4] S. Srinivasan, D.P. Mital, S. Haque, “A novel solution for maze traversal problems using artificial neural networks,” Computers and Electrical Engineering, Vol. 30, 2004, pp. 563–572.

[5] W. Shenga, Q. Yang, J. Tan, N. Xi, “Distributed multi-robot coordination in area exploration,” Robotics and Autonomous Systems, Elsevier, 2006, doi:10.1016/j.robot.2006.06.003.

[6] S. Carpin, G. Pillonetto, “Motion Planning Using Adaptive Random Walks,” IEEE Transactions On Robotics, Vol. 21, No. 1, Febrero 2005.

[7] B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, S. Hutchinsona, “Planning exploration strategies for simultaneous localization and mapping,” Robotics and Autonomous Systems, Elsevier, Vol 54, 2006, pp. 314–331.

[8] N. Nilsson, “Inteligencia Artificial, una nueva síntesis,” España, McGraw-Hill/Interamericana de España, S.A. 2001.

[9] A.J. Bagnall, Z.V. Zatuchna, “On the Classification of Maze Problems,” Applications of Learning Classifier Systems, Studies in, Bull, Springer, pp. 307-316, 2005.

[10] P. Gerard, O. Siguad, “YACS: Combining Dynamic Programming with Generalization in Classifier Systems,” Advances in Learning Classifier Systems. Springer, 2001, pp. 52–69.