DISEÑO, IMPLEMENTACIÓN Y APLICACIÓN DE UNA ESTRATEGIA DE BÚSQUEDA PREFERENTE POR AMPLITUD, PARA USO MULTIDIRECCIONAL SOBRE SISTEMAS DISTRIBUIDOS O DE PROCESAMIENTO EN PARALELO, USANDO UN SIMULADOR DE ESCENARIOS, CONSTRUIDO PARA EL TRAZADO DE RUTAS EN ROBÓTICA MÓVIL

ALEJANDRO GONZÁLEZ OSPINA
HUGO BALDOMIRO CANO GARZÓN

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE CIENCIAS BÁSICAS
MAESTRIA EN INTRUMENTACIÓN FÍSICA

Pereira, Noviembre de 2011

DISEÑO, IMPLEMENTACIÓN Y APLICACIÓN DE UNA ESTRATEGIA DE BÚSQUEDA PREFERENTE POR AMPLITUD, PARA USO MULTIDIRECCIONAL SOBRE SISTEMAS DISTRIBUIDOS O DE PROCESAMIENTO EN PARALELO, USANDO UN SIMULADOR DE ESCENARIOS, CONSTRUIDO PARA EL TRAZADO DE RUTAS EN ROBÓTICA MÓVIL

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE CIENCIAS BÁSICAS
MAESTRIA EN INTRUMENTACIÓN FÍSICA

Por:
ALEJANDRO GONZÁLEZ OSPINA
HUGO BALDOMIRO CANO GARZÓN

M. Sc. WILLIAM ARDILA URUEÑA
Director de tesis

Tesis presentada como requisito para optar al título de
Magister en Instrumentación Física

Pereira, Noviembre de 2011

Nota de aceptación:

_____

_____

_____

_____

_____
Director

_____
Jurado 1

_____
Jurado 2

Pereira, Noviembre de 2011

DEDICATORIA

"Señor que tanto me has dado, sé misericordioso y concédeme algo más: Un corazón agradecido".

Apóstol Pablo

"El agradecimiento es la memoria del corazón".

Lao-Tsé

Sea la dedicatoria de esta tesis, un humilde reconocimiento y agradecimiento, a todas las personas -unas vivas y otras ya fallecidas-, que de una u otra forma han sido apoyo, inspiración y motor de mi trabajo.

Gonzalo, Inés, Betty, Heriberto, Lola, Fabio, Amanda.

De forma especial a mi esposa Diana Patricia, a mis hijos María Alejandra, Juan Manuel y Santiago.

Alejandro González Ospina

Dedico este trabajo especialmente a mi esposa Claudia María, a mis lindas hijas Laura Elena y Manuela, quienes son la razón de mi vida.

Igualmente a mi padre Arturo Antonio y mi hermana Inés Lucía, por ser las personas que me acompañan desde siempre con sus buenos deseos.

Hugo Baldomiro Cano Garzón

AGRADECIMIENTOS

Gracias a Dios por estar vivos…

Son varias las personas a las que tenemos que agradecer su ayuda y apoyo durante la realización de esta maestría.

Comenzando con aquellas más directamente relacionadas con el proyecto: M.Sc. William Ardila Urueña, quien a pesar de sus grandes compromisos, en ningún momento dejó que ellos fueran un impedimento para la dirección de esta tesis; a los profesores M.Sc. Ricardo López, M.Sc. Hugo Armando Gallego, M.Sc. Hoover Orozco, M.Sc. José Carlos Moreno, Ing. Gilberto Vargas y demás personas que colaboraron con su dedicación y esfuerzo en la culminación del postgrado.

Por último a todos los compañeros de la maestría, con los cuales se creó un gran grupo con fuertes lazos de amistad.

Alejandro González Ospina

Hugo Baldomiro Cano Garzón

RESUMEN

Para la construcción de un robot móvil autónomo se debe disponer de una gran cantidad de elementos funcionales, entre los cuales se destaca un sistema de procesamiento que ayude a resolver los problemas de desplazamiento, para ir de un lugar a otro dentro de un entorno o ambiente atestado. Inicialmente se plantea la construcción de un simulador de escenarios en dos dimensiones, el cual permitirá que un robot de software (llamado Softbot) y los diferentes elementos que conforman un ambiente, se utilicen como sistema de pruebas en la búsqueda de una ruta apropiada para ir de un punto a otro.

El simulador de escenarios y sus características, se construye mediante el uso de bases de datos capaces de almacenar grandes cantidades de información y con tiempos de acceso pequeños, esta información contiene la descripción de los elementos y de los cambios que se puedan producir en el modelo del mundo virtual. De esta manera es posible hacer que el Softbot realice un "recorrido" por el escenario.

El modelo del escenario, construido dentro del aplicativo, se presenta como un “plano” en dos dimensiones (2D), el cual contiene la distribución física de elementos; mientras que el agente de software cuenta con una representación virtual del ambiente, basada en artificios computacionales que le permite recorrer el entorno y simular las lecturas de los sensores virtuales de proximidad en él incorporados.

Como herramienta para la determinación de la ruta se utilizan técnicas de Inteligencia Artificial (IA), aplicadas en el diseño de sistemas instrumentados soportados por software avanzado. Este tipo de aplicaciones tienen en común, la necesidad de ir de un punto o estado inicial a otro, lo que se logra mediante la sucesión de cambios de estado a través de acciones o transformaciones, el uso de estrategias de búsqueda y heurísticas generales.

Finalmente se implementa la capacidad de interactuar no solo uno, sino varios agentes de software, trabajando en forma cooperativa, de tal forma que con la ayuda de ellos y aplicando la estrategia de búsqueda multidireccional sea posible establecer una ruta o camino para ir de un punto a otro en el escenario.

ABSTRACT

For the construction of an autonomous mobile robot must have a large number of functional elements, among which stands out a processing system to help solve the problems of displacement, to go from one place to another within an environment or atmosphere overcrowded. Initially there is the construction of a simulator scenario in two dimensions, which allow a software robot (called Softbot) and different elements that are part of an environment, used as a test system in finding a suitable route to get from one point to another.

The simulator scenarios and their characteristics, is constructed by using databases capable of storing large amounts of information and access times smaller, this information includes the description of the elements and changes that may occur in the model the virtual world. Thus it is possible to make the Softbot perform a "walk" across the stage.

The model of the stage, built into the application, is presented as a "flat" two-dimensional (2D), which contains the physical distribution of elements, while the software agent has a virtual representation of the environment, based on artifacts computer that allows you to traverse the environment and simulate the virtual sensor readings proximity incorporated therein.

As a tool for determining the route will use techniques of Artificial Intelligence (AI), applied in instrumented system design supported by advanced software. Such applications have in common, the need to move a point or initial state to another, which is achieved by the sequence of state changes through actions or transformations, the use of search strategies and general heuristics.

Finally, implement the ability to interact not only one but several software agents, working cooperatively, so that with the help of them and applying the multi-search strategy is possible to establish a route or way to get from one point another on stage.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

La presente tesis tiene como propósito realizar el análisis, diseño e implementación de una estrategia de búsqueda preferente por amplitud que genere una ruta apropiada, por la cual un agente de software (Softbot[1]), se desplazará para ir de un punto inicial hasta un punto final o meta deseada, haciendo uso de un simulador de escenarios (**SIRUM**), construido y diseñado de forma virtual con la capacidad de representar ambientes atestados. Un aspecto importante en la robótica móvil es la definición de una trayectoria o ruta, que le permita a un robot desplazarse por escenarios o ambientes previamente construidos.

Como primera consideración, el Softbot parte de un lugar cualquiera dentro del escenario, como el mostrado en la Figura 1 (posición de inicio); y con base en la información obtenida mediante la dotación de sensores virtuales, va trazando un camino que lo conducirá a la meta (Ruta inicial), cuando pase por un punto cualquiera del escenario (punto A) y perciba que un objeto (plataforma O) se mueve describiendo una trayectoria de colisión (cerrándole el paso), es razonable esperar que reconsidere el camino, de tal forma que describa un nuevo recorrido (Ruta recalculada).

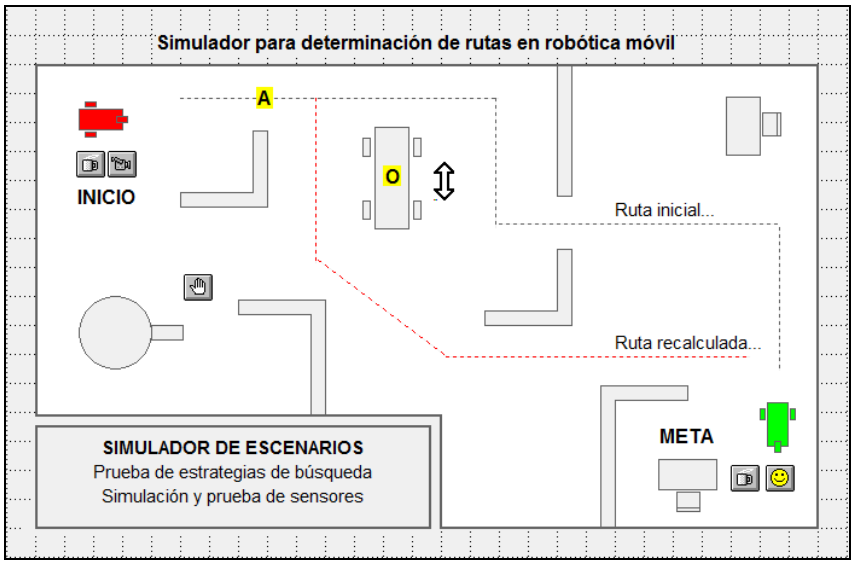


Figura 1: Trazado dinámico de ruta.

[1] Softbot: Es un agente de software basado en **IA**. (Ver marco teórico para más detalles).

Igualmente se plantea la construcción del agente inteligente que es el equivalente a un robot virtual o de software, el cual se dota de:

a. Una estrategia de búsqueda para determinar la secuencia de pasos necesarios para ir desde un punto inicial hasta la meta.

b. Sensores virtuales, que emulan la percepción de sensores físicos, para que el Softbot pueda responder a cambios inesperados en el entorno.

c. La capacidad de desplazamiento en el escenario, reaccionando a situaciones imprevistas durante su recorrido.

Una vez se tenga construido lo mencionado anteriormente, se procederá a establecer un camino para ir desde el origen hasta la meta, no solamente con uno, sino con la ayuda de varios agentes inteligentes, desarrollando la estrategia de búsqueda preferente por amplitud para uso multidireccional que opera en sistemas distribuidos o de procesamiento en paralelo. La estrategia propuesta, aprovecha la capacidad de las redes de computadores para “dar vida” a un conjunto de agentes que trabajan cooperativamente en la búsqueda de la ruta.

El presente documento de tesis se estructura de la siguiente manera:

1. Descripción del proyecto
2. Marco referencial
3. Diseño metodológico
4. Componentes del proyecto
    - Generador de escenarios
    - Modelo de los sensores
    - La estrategia de búsqueda
        - Sin sensores
        - Con sensores
        - El agente de software como articulador
        - Búsqueda multidireccional
5. Pruebas y resultados
6. Conclusiones
7. Referencias bibliográficas
8. Glosario de términos
9. Anexos

Capítulo 1

DESCRIPCIÓN DEL PROYECTO

1.1 PLANTEAMIENTO DEL PROBLEMA

Teniendo en cuenta la carencia de recursos en las pruebas de nuevas tecnologías en el campo de la instrumentación física, como es el caso de sistemas inteligentes que involucran hardware y software de alto nivel; así como los altos costos de los dispositivos para la robótica móvil y la necesidad de dar respuesta en tiempo real, a cambios del entorno por donde deambula un robot para ir de un punto inicial a un punto objetivo llamado meta, se realiza en este proyecto un trabajo que articula los anteriores aspectos.

La complejidad de los problemas que se presentan en robótica móvil como serían los diferentes entornos (ambientes) y sus características impredecibles, hacen muy atractiva la necesidad de contar con una herramienta virtual que permita realizar, a nivel de prueba, la ejecución de acciones, que hechas en el mundo físico, serían de muy alto costo y que podrían ser riesgosas. Por tanto tener disponible dicha herramienta, que hace uso de la inteligencia artificial, será un recurso importante en la solución de este tipo de problemas.

Para este propósito, se contempla básicamente lo siguiente:

- Diseño, construcción e implementación de una estrategia de búsqueda para resolver problemas que implican ir de un punto o estado inicial, a otro deseado, mediante la aplicación de operadores o transformadores.

- Diseño y creación de un agente inteligente (Softbot), mediante tecnología de objetos, con la capacidad de moverse en escenarios diseñados para tal fin.

- Diseño de un software de alto nivel, para crear escenarios atestados; lugar donde se probará tanto la estrategia de búsqueda, como las percepciones simuladas del agente de software, las cuales están contenidas y registradas en tablas dinámicas.

- Comprobación de la estrategia de búsqueda para uso multidireccional con base en la creación de agentes inteligentes auxiliares que, de forma cooperativa, ayudan a encontrar una ruta para ir del origen a la meta, dentro de un ambiente atestado.

1.2 HIPÓTESIS DEL PROBLEMA

La posibilidad de poseer un simulador de escenarios, donde se puedan crear diferentes ambientes, se convierte en un recurso imprescindible en el campo de la robótica móvil, puesto que es allí donde se probarán las estrategias de búsqueda que tienen como objetivo solucionar problemas de desplazamiento.

La obtención de una primera ruta para ir de un punto a otro en un ambiente atestado, no necesariamente es considerada como la mejor, puesto que la estrategia de búsqueda que se utilice como técnica de la inteligencia artificial, puede ser ajustada y mejorada, de tal manera que sus resultados se acerquen a los óptimos.

La dotación de sensores es un recurso indispensable para encontrar el camino por donde se desplazará el agente de software (Softbot) dentro de un escenario, del cual no tiene conocimiento, pero el sólo disponer de sensores, no garantiza el hallazgo de la ruta; sin embargo, sí facilita el trabajo de búsqueda, puesto que se considera que lo reportado por ellos es lo único que existe en el escenario.

La heurística desarrollada para la búsqueda de la ruta, hace parte de la estrategia que ayudará a encontrar el camino o vía para ir al objetivo. La necesidad de resolver una situación o imprevisto que se presente en el ambiente por donde navegará el agente, requiere del uso de la inteligencia artificial como parte de la solución en la toma de decisiones, sin embargo estas condiciones atípicas requerirán de pequeños ajustes que hacen converger a una solución.

Con el aprovechamiento de una estrategia de búsqueda y consecuentemente el uso de uno o varios procesadores en paralelo es posible resolver, de forma cooperativa, los problemas de búsqueda de rutas en robótica móvil, consistentes en ir desde un punto inicial a otro llamado meta, teniendo en cuenta evitar dobles trabajos y bajar la carga de procesamiento.

Cuando se plantea el trabajo cooperativo, es necesario definir inicialmente las tareas que cada agente auxiliar debe desarrollar, mismas que son redefinidas a medida que se avance en el proceso de búsqueda de la solución.

1.3 DELIMITACIÓN DEL PROBLEMA

El ambiente o escenario generado se limitará a dos (2) dimensiones y se construirá con formas simples: líneas, aristas, vértices.

Al ir dibujando el escenario por medio de la herramienta de edición, se generará de forma automática su plano correspondiente, el cual se podrá utilizar como conocimiento previo por parte del *agente autónomo*[2], para que descubra los caminos por donde navegará hasta su objetivo.

Los sensores simulados con los que se dota el Softbot, producirán una exactitud uniforme a través de todo el entorno, ya que son sensores virtuales simples de posición y contacto; adicionalmente, tratarán el ambiente de forma simplificada.

Se tomará en consideración el diseño de una heurística[3] para la implementación de la estrategia de búsqueda preferente por amplitud, la cual permitirá tomar decisiones en cuanto a los movimientos de navegación dentro del escenario. Así mismo, está prevista la creación de los agentes auxiliares que harán uso de la estrategia de búsqueda; de tal forma que la cantidad de ellos esté relacionada con la capacidad computacional (disponibilidad de procesadores) y su pertinencia definida por el universo o espacio de búsqueda.

La ruta que se encuentre para ir desde el origen hasta la meta, haciendo uso de la estrategia multidireccional, será la primera, es decir, otras soluciones no serán tenidas en cuenta, puesto que se trata de encontrar convergencia en la solución del problema.

Con el ánimo de aprovechar las últimas tecnologías y modelos informáticos; así como el hacer uso de los conocimientos adquiridos en la maestría de Instrumentación Física, se realiza el desarrollo de un software sobre la plataforma Microsoft FoxPro. Esta plataforma garantiza entre otras cosas: acceso a alto y bajo nivel, diseño cliente servidor para la programación de la estrategia de búsqueda y desarrollo tanto para ambiente de escritorio, como de dispositivos móviles y web.

En este punto se resalta que la realización de este proyecto, podrá extrapolarse al estudio de otras técnicas de Inteligencia Artificial (IA) aplicadas a la instrumentación física; ayudando a cerrar la llamada Brecha Digital[4].

[2] Agente autónomo, sinónimo de agente inteligente (Ver definición en el numeral 2.1.1).
[3] Ver definición en el numeral 2.2.3.
[4] Ver definición Glosario de términos

1.4 OBJETIVOS

1.4.1 Objetivo general

Diseñar, desarrollar e implementar una estrategia de búsqueda preferente por amplitud para uso multidireccional en un simulador de escenarios, para pruebas de determinación de rutas de un robot de software, al que se le simulan percepciones del ambiente.

1.4.2 Objetivos específicos

- Diseñar y construir un software para la simulación de escenarios atestados en el que se pueda desplazar un robot de software (Softbot).

- Diseñar, construir e implementar una estrategia de búsqueda para resolver problemas que implican ir de un punto o estado inicial a otro deseado, mediante la aplicación de operadores o transformadores.

- Realizar pruebas de la estrategia de búsqueda con diferentes modelos de escenarios, mediante trabajo en paralelo, usando uno o más procesadores.

1.5 JUSTIFICACIÓN

Dando un vistazo a la tendencia del diseño de equipos que usan intensivamente la tecnología, se observa fácilmente que cada vez es más común, el uso de herramientas inteligentes para planear, gobernar y ejecutar todo tipo de sistemas; (Desde pilotos automáticos en aviones, hasta dispositivos para predecir el clima). Todos ellos se caracterizan por recoger datos del entorno, hacer uso de bases de conocimiento y realizar procesos de gestión y control.

Es en los dos (2) últimos frentes, donde se encuentra la mayor diferenciación a nivel de diseño y construcción de equipos. Por esta razón es de especial interés el estudio de técnicas de inteligencia artificial (IA), que llevan a plantear la presente tesis, como contribución al diseño de sistemas instrumentados soportados por software avanzado.

Como ejemplos de aplicación, se tiene que la **IA** y el desarrollo de simuladores por software, son fundamentales para resolver problemas relacionados con:

- Determinación de secuencias de ensamble.
- Distribución de conexiones en la construcción de circuitos integrados de muy alta escala de integración (VLSI).
- Determinación de rutas en robótica móvil.
- Cálculo de soluciones en problemas que satisfacen restricciones.

Este tipo de aplicaciones tienen en común la necesidad de ir de un punto o estado inicial a otro denominado meta, lo que se logra con una sucesión de cambios de estado a través de acciones o transformaciones sucesivas. La solución consiste fundamentalmente en:

a) Expresar el problema como un espacio de estados, que se puede representar mediante un diagrama de árbol invertido, como el mostrado en la Figura 2.

b) Encontrar un camino que une la raíz y el destino o meta.

c) Emplear una estrategia de búsqueda preferente por amplitud, la cual permitirá la toma de decisiones con respecto a los diferentes estados generados.

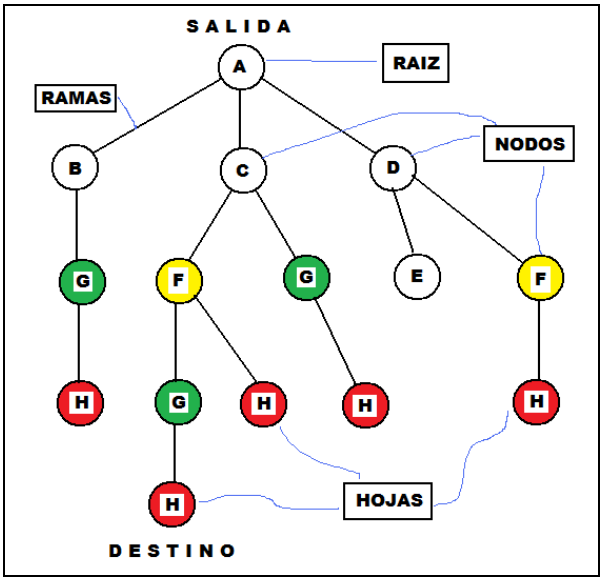


Figura 2: Árbol de estados.

Lograr la solución de este tipo de problemas, permitirá que se disponga de una herramienta computacional de gran complejidad que, al final, facilitarán la tarea de los diseñadores y creadores de dispositivos que combinan hardware y software de alto nivel.

1.5.1 Beneficios que conlleva

Con el desarrollo del simulador de escenarios para la búsqueda de rutas en robótica móvil (SIRUM), se esperan los siguientes impactos:

- Mejorar la calidad de la enseñanza y la aplicación de la **IA** y su integración con el hardware.

- Contar con una herramienta para pruebas de software de **IA**. (específicamente simulación de sensores y métodos de búsqueda).

- Extrapolar los conocimientos y experiencias obtenidas del SIRUM a las aplicaciones derivadas de la instrumentación física, entre otros, el manejo de los diferentes transductores y tarjetas de adquisición.

- Acercar a los estudiantes de las diferentes ramas del saber al uso de dispositivos de procesamiento en paralelo, entre los que se cuentan, las redes de computadores y los supercomputadores.

- Generar desarrollos tecnológicos que permitan integrar la **IA** con la instrumentación física.

- Avanzar en el campo de la robótica móvil, dotando a los agentes inteligentes contaran con la capacidad de tomar decisiones a cambios imprevistos del entorno.

De igual manera, la creación de la estrategia de búsqueda preferente por amplitud para uso multidireccional, permitirá desarrollar modelos que resuelven problemas complejos como lo sería la búsqueda de una ruta entre dos puntos en un ambiente atestado de difícil análisis.

La coordinación de varios agentes de software en la solución de problemas de desplazamiento en ambientes atestados, será un nuevo recurso que la robótica móvil podrá utilizar, de tal manera que una vez implementada en los modelos físicos, se convierte en una estrategia para el uso de robots cooperativos.

Capítulo 2

MARCO REFERENCIAL

El proyecto abarca tres (3) disciplinas bien diferenciadas de la robótica móvil que tienen que ser articuladas adecuadamente para darle estructura y soporte, ellas son:

a) La inteligencia artificial (IA) asociada al diseño y construcción de la estrategia de búsqueda, que ayuda en la solución de problemas que consisten en ir de un estado inicial a uno final, previamente definidos.

b) Los sistemas y la informática para la implementación de la ***estrategia de búsqueda*** en ambientes multitarea y multiusuario, así como la construcción del simulador de escenarios, bajo tecnologías que permitan trabajo a alto y bajo nivel.

c) La instrumentación para el estudio y comprensión de los transductores que se desean emular. A tal punto que en proyectos futuros, los transductores emulados puedan ser directamente reemplazados por transductores físicos mediante el uso de software de acople (conocidos como drivers[5]).

2.1 ESTADO DEL ARTE

A continuación se presentan algunos fundamentos teóricos y el estado del arte asociados a cada una de las tres disciplinas mencionadas anteriormente.

Para el caso específico de la fundamentación en Inteligencia Artificial (**IA)**, se utiliza la descripción de un agente inteligente basado en metas, que junto con el agente reflejo simple, el agente reflejo con estado interno y el agente basado en utilidad, conforman el marco teórico en el que se enfoca el estudio y la investigación sobre las diferentes áreas de impacto de la IA.

Posteriormente se muestra el método de búsqueda respaldada con información, como herramienta para la solución de problemas de búsqueda de rutas en robótica móvil.

[5] Drivers: ver definición Glosario de términos

2.1.1 Agentes inteligentes

Un agente es aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores. [1]

Se define al agente inteligente como una entidad software que, basándose en su propio conocimiento, y la percepción del ambiente en el que se encuentra, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere.

En la Figura 3, se presenta el diagrama de un ***agente completo basado en la utilidad,*** el cual utiliza la información que percibe del ambiente para la toma de decisiones. Se destacan los siguientes aspectos en el esquema:

- En un instante cualquiera, el estado interno es la codificación y mapeo de todas las características que el agente conoce del ambiente, abarca todos los elementos en él contenido -incluido el propio Softbot-.

- El agente debe tener en cuenta que el mundo o entorno evoluciona independientemente de lo que él haga.

- El agente debe conocer de qué forma las acciones que él emprenda afectarán el entorno.

- Ante una variedad de posibles acciones que propendan por alcanzar la meta, el agente debe contar con criterios que le permitan decidir sobre las acciones a emprender; con base en el discernimiento del grado de utilidad y la satisfacción que cada una de ellas pueda tener.

- Cuando un agente tiene más de una alternativa por escoger y a su vez cada una de ellas se ramifica en otras más, puede indagar por dichas ramificaciones hasta n pasos de profundidad, tratando de prever los resultados a futuro de sus posibles acciones; y regresando a niveles anteriores (el ahora) donde efectivamente ejecutará la acción que le puede reportar una mayor satisfacción o conformidad con la meta propuesta.

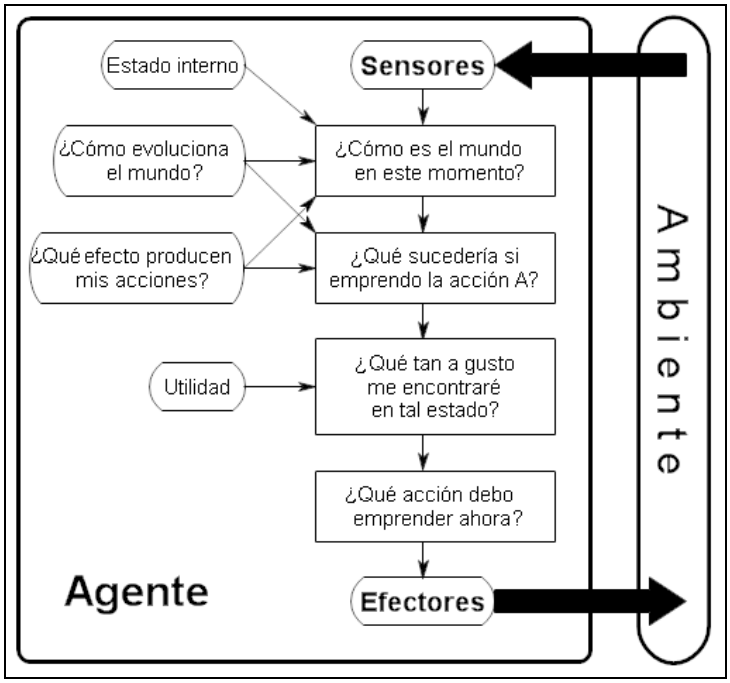


Figura 3: Agente completo basado en la utilidad. [1]

Todos los agentes inteligentes son programas, pero no todos los programas que realizan búsquedas son agentes inteligentes. Los agentes en sí mismos pueden ser considerados como entidades individuales (partes de programa que tienen control sobre sus propias vidas y movimientos). Continuamente están realizando procesos como: percibir una variable externa, comunicarse con otros agentes o entidades, análisis y proyecciones para finalmente tomar decisiones -realizar un trabajo-.

De acuerdo con el punto de vista de la inteligencia artificial, un agente posee las siguientes propiedades: autonomía, sociabilidad, capacidad de reacción, iniciativa, benevolencia y racionalidad (*Wooldridge y Jennings, 1995*).

***Autonomía*:** Actuar sin ningún tipo de intervención humana directa y tener control sobre sus propios actos.

***Sociabilidad*:** Comunicarse por medio de un lenguaje común con otros agentes e incluso con los humanos.

***Capacidad de reacción*:** Percibir su entorno y reaccionar para adaptarse a él.

Benevolencia: Se define como la capacidad de comprender aunada a la tolerancia.

***Iniciativa*:** Emprender las acciones para resolver un problema.

Racionalidad: Un agente racional es aquel que hace lo correcto, es decir, que obtiene el mejor desempeño.

Una vez dicho esto, ya no se hablará más de agentes inteligentes para la búsqueda de rutas en robótica móvil, sino que simplemente se hablará de ellos como **agentes de búsqueda**.

El agente está diseñado específicamente para reconocer un ambiente o escenario, su propia ubicación inicial, la meta o punto de llegada y realizar percepciones a cambios dinámicos en el ambiente, es decir, tendrá capacidad de proceso, conocimiento del entorno, movilidad y una estrategia de búsqueda que le permita tomar decisiones para llegar a su objetivo.

Un agente tiene capacidad de proceso, puesto que toma decisiones con base en la meta propuesta y las percepciones que va recogiendo conforme se desplaza. Su conocimiento del entorno le viene dado por su propio conocimiento y por el de otros agentes que se comunican con él (el conocimiento puede ser adquirido: del mismo usuario o de otros agentes con los que interactúa mientras realiza una tarea determinada y de lugares previamente explorados). En todo momento debería saber a qué información acceder o a qué otro agente dirigirse para obtenerla. Un agente puede tener también acceso a un dominio o información de un modelo (estrategias alternas), si se asocia con la estructura de éste.

En la construcción del agente inteligente se requiere incorporar un programa que permita pasar de las percepciones a las acciones, lo que implica, realizar mapas de percepciones del ambiente o escenario, dotando al programa con la capacidad de construir árboles de estados, sobre los cuales realizar búsquedas orientadas al alcance de la meta, apoyado o dirigido por funciones de decisión o de utilidad, diseñadas mediante heurísticas que puedan ser ajustadas dinámicamente.

La subrutina de búsqueda debe estar diseñada para indagar por cambios en el ambiente y responder a interrupciones por condiciones o llamados especiales que la pausen, le redefinan los parámetros o suspendan definitivamente su ejecución.

El diagrama de flujo mostrado en la Figura 4, representa el algoritmo del agente inteligente en la búsqueda de un objetivo final.

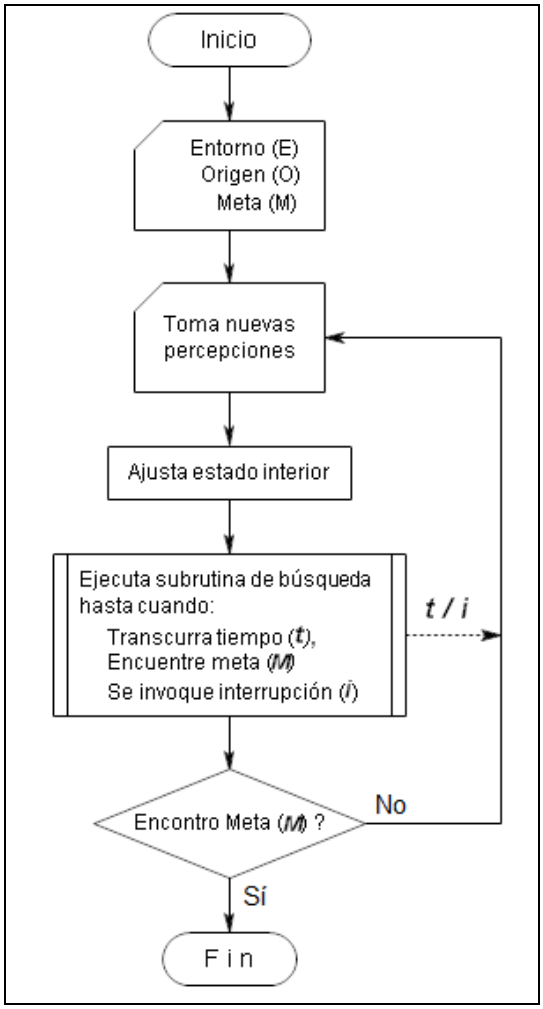


Figura 4: Algoritmo del agente inteligente.

Se destaca en el algoritmo el punto de partida o estado inicial del entorno, allí se incluye el origen y la meta. El ambiente o entorno lo conforman tanto elementos móviles como elementos fijos; del entorno se toma la información que es percibida y que será analizada en la toma de decisiones. Todo lo anterior con el propósito de encontrar la meta que previamente fue establecida.

2.1.2 Agentes cooperativos

Cualquier objetivo o problema se puede descomponer en sub-objetivos o en sub-problemas. Por tanto, cualquier búsqueda se podrá dividir en sub-búsquedas (o en tantos agentes como sub-búsquedas tenga la búsqueda), para lo cual se usan los llamados ***agentes cooperativos***. Para que un agente pueda ser autosuficiente y conocedor del entorno en el que se encuentra, debe coordinarse y cooperar con cada uno de los otros agentes. Existen varias formas para hacer esto.

En un sistema compartido un agente cualquiera descompone la búsqueda y asigna las sub-búsquedas a otros agentes. Cada uno conoce las capacidades y limitaciones del resto. No existe un "agente maestro"; el grupo de agentes recibe las sub-búsquedas y todos ellos trabajan por igual para encontrar la solución.

En un sistema contractual los agentes siguen teniendo todos el mismo estatus. Sin embargo, el agente que inicia la búsqueda no asigna las sub-búsquedas al grupo de agentes, sino que las envía a todos ellos para que éstos le indiquen quién puede resolverlas y selecciona aquellos cuyo dominio sea más adecuado.

Por el contrario, un sistema federado es una estructura jerárquica de agentes controlada por un facilitador o agente principal (también llamado agente coordinador). Los agentes federados se comunican sólo con su agente principal, el cual conoce las capacidades y limitaciones de cada uno de sus agentes. Una vez inicia la búsqueda, el facilitador principal se comunica con el resto de facilitadores con el fin de seleccionar los agentes locales más adecuados de cada federación para resolver las sub-búsquedas que permitan resolver la búsqueda completa (*Haverkamp y Gauch, 1998*).

Los agentes móviles, se basan en el principio organizador de redes de comunicación entre ordenadores, conocido como *Control de Procedimientos Remotos* (RPC). Cuando un ordenador cliente de una red (no importa su tamaño) dirige una petición al servidor de archivos para ejecutar una aplicación, el cliente debe realizar al menos dos comunicaciones, una solicitando la ejecución de un programa determinado y otra informando al servidor que la operación se ha completado con éxito. La alternativa a este procedimiento es la Programación Remota (RP), consistente en acordar por adelantado las tareas que pueden realizar los clientes sin ningún tipo de verificación ni confirmación por parte de los servidores. De esta forma un cliente enviaría una instrucción al servidor de archivos y una vez allí ejecutará un programa en concreto. Este procedimiento (remoto) que es una orden realizada por el cliente pero ejecutada en el servidor (local) recibe el nombre de operación o instrucción móvil, por tanto, una orden remota es ejecuta localmente.

2.1.3 Plataforma de desarrollo para simuladores

En la seguridad de procesos industriales el factor humano tiene un papel trascendental y en particular, en el campo de la robótica móvil, es esencial evitar los errores en el manejo del entorno, para ello se hace necesario el uso de simuladores que reproducen con un alto grado de fidelidad física y funcional de todos los elementos presentes en un ambiente. Lo anterior conlleva a que los modelos de simulación tengan en cuenta los principios de ingeniería básicos y avanzados para reproducir el comportamiento del entorno en estado normal.

Por otra parte la arquitectura y entornos computacionales que conforman los ambientes simulados, corresponden a programas modulares de alta complejidad que son capaces de manejar los procesos desde una red de ordenadores. Desde el punto de vista de diseño de la interacción con los sistemas de control distribuido utilizados en la mayoría de plantas industriales, se pueden distinguir varios tipos de simuladores, por ejemplo el estimulado y el emulado. El simulador estimulado es muy caro, no permite parar la simulación y requiere de extensas modificaciones para que los sistemas de control puedan atender las operaciones normales de carga de una condición inicial. Como ventajas se señalan que la interacción hombre máquina es exactamente igual a la planta real y que adicionalmente puede servir para el entrenamiento en los equipos de control.

Por su parte los simuladores totalmente emulados son menos costosos y permiten al instructor realizar una serie de operaciones como lo son grabar una condición inicial, volver atrás, parar, avanzar y generar gráficos de tendencias.

Otro tipo de simuladores son los utilizados principalmente para la formación del personal de ingeniería, conocidos como simuladores de escritorio, que se ejecutan en un ordenador personal y que contienen, además del entorno de simulación, todos los elementos de control y seguimiento dispuestos en “paneles virtuales” a través de los cuales el alumno puede interactuar como si estuviera en la sala de control. Este tipo de simuladores tienen la ventaja adicional que son flexibles en su configuración pudiendo ofrecer la opción de modificar la disposición de los elementos del ambiente. [2]

2.2 ANTECEDENTES

Tal como se expresó en la sección 1.5, existen una gran cantidad de problemas que implican partir de unas condiciones iniciales y aplicar una serie de pasos o transformaciones para llegar a un estado deseado.

La solución del problema se logra mediante la configuración de un espacio de estados, que pueda incluir el estado final deseado y la estrategia de búsqueda, como método de análisis, para la obtención de una la sucesión de transformaciones que lleven al estado final o meta deseada.

La estrategia de búsqueda se define como el conjunto de procedimientos y operaciones que se realizan con el fin de obtener la información necesaria para resolver un problema. [3]

En la actualidad, las estrategias de búsqueda existentes son:

- Búsqueda preferente por amplitud.
- Búsqueda de costo uniforme.
- Búsqueda preferente por profundidad.
- Búsqueda limitada por profundidad.
- Búsqueda por profundización iterativa.
- Búsqueda bidireccional.
- Búsqueda especial para problemas que satisfacen restricciones.

La implementación de los anteriores tipos de búsqueda, puede darse bajo una de las dos siguientes modalidades:

- Sin información, llamadas también búsquedas ciegas.
- Búsquedas con información, para las cuales podemos aplicar técnicas como:

    - Preferentes por lo mejor.
    - Mediante heurísticas.
    - Limitadas por la capacidad de la memoria.
    - Algoritmos de mejoramiento iterativo.
    - Búsqueda por ascenso de cima.
    - Endurecimiento simulado.
    - Aplicación a problemas que satisfacen restricciones.

No es prohibido la combinación de diferentes estrategias para producir una nueva y mejorada forma de realizar las búsquedas; es el caso de la búsqueda mostrada en la Figura 5, donde se presenta un esquema de búsqueda preferente por amplitud bidireccional, el cual está a punto de tener éxito, cuando una rama del nodo inicio haga contacto con una rama del nodo meta.

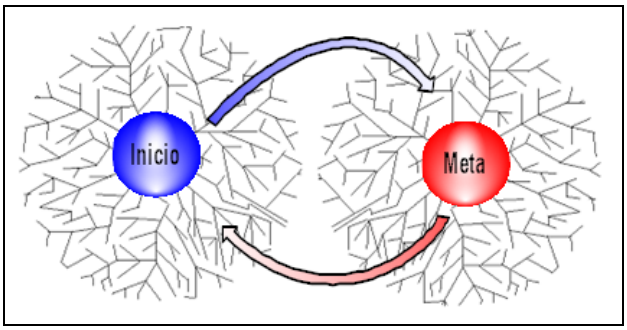


Figura 5: Búsqueda bidireccional. [1]

La conexión o enlace entre dos ramas de distintos nodos, proporcionará un camino por el cual se pueda ir del inicio u origen a la meta y viceversa.

2.2.1 Especificación de espacios de búsqueda

El aplicar técnicas de búsqueda en grandes espacios (ambientes o escenarios), se considera en principio una tarea demasiado extensa y compleja; siendo necesario ilustrar de forma práctica y sencilla, como se hace la planificación de las secuencias de acciones para alcanzar un estado final.

Un caso típico de explicar es el famoso juego del puzzle de 8 piezas, donde el objetivo es encontrar las secuencias de movimientos, que transforme una disposición inicial en una configuración final deseada. Suponga que se desea mover las casillas como se muestra en la Figura 6.

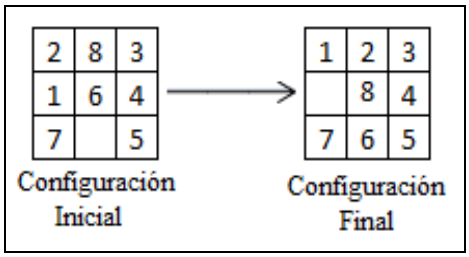


Figura 6: Configuración inicial y final del puzzle.

Una representación inicial obvia, es tomar una configuración matricial de 3x3, donde cada celda contiene un número del 1 al 8 o un carácter que lo represente.

Nótese que tanto la configuración inicial como la final en la Figura 6, corresponden a estados del gran conjunto que resulta de mover las fichas en todas las direcciones posibles.

Así mismo, si se analizan los posibles cambios a partir de un estado cualquiera, se obtiene una serie de nodos o estados, derivados del estado objeto de análisis.

Ordenar y concatenar todas las ramificaciones posibles a partir de un estado cualquiera, conduce a obtener un árbol invertido como el mostrado anteriormente en la Figura 2. Teniendo cuidado de no expandir estados previamente expandidos, el problema se puede centrar en encontrar una ruta que lleve del estado inicial al estado meta previamente determinados.

Una consideración particular en el puzzle para determinar los posibles movimientos, es mover la casilla libre (celda vacía sin datos) hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha, sin permitir movimientos diagonales.

La secuencia en el árbol de estados, con los diferentes nodos o estados representados, indicara la ruta que se deriva para llegar al objetivo final, partiendo del nodo inicial. En el puzzle de ocho piezas se obtendría un conjunto de nodos igual a nueve factorial $(9! = 3628880)$, este valor tan grande se podría dividir a la mitad planteando dos problemas; uno con búsqueda de la configuración inicial a la final y otro de forma contraria, es decir, partiendo de la configuración final a la inicial.

2.2.2 La búsqueda en amplitud (anchura)

Los procedimientos de búsqueda sin conocimiento, se hacen aplicando los operadores (movimientos) disponibles a los nodos, es decir, sin ninguna idea sobre el problema en su contexto, este procedimiento es conocido como búsqueda en amplitud, donde el árbol de estado se construye mediante la aplicación de los operadores disponibles al nodo inicial, después aplica los operadores disponibles a los nodos que le preceden directamente al nodo inicial y así sucesivamente, cada que se aplique este procedimiento se obtendrá una función de estados sucesores que dan como resultado final la expansión del nodo.

En la Figura 7, se muestra este procedimiento, donde se pueden ver los nodos creados por la búsqueda en anchura para el puzzle de ocho piezas.

La forma como se llega al nodo final, es comenzando la expansión del nodo inicial e indicando mediante el número superior izquierdo cada movimiento próximo del puzzle. Los nodos que están situados a la misma profundidad se van expandiendo según un orden prefijado de antemano; para cada nodo, el orden de aplicación de los operadores es el siguiente:

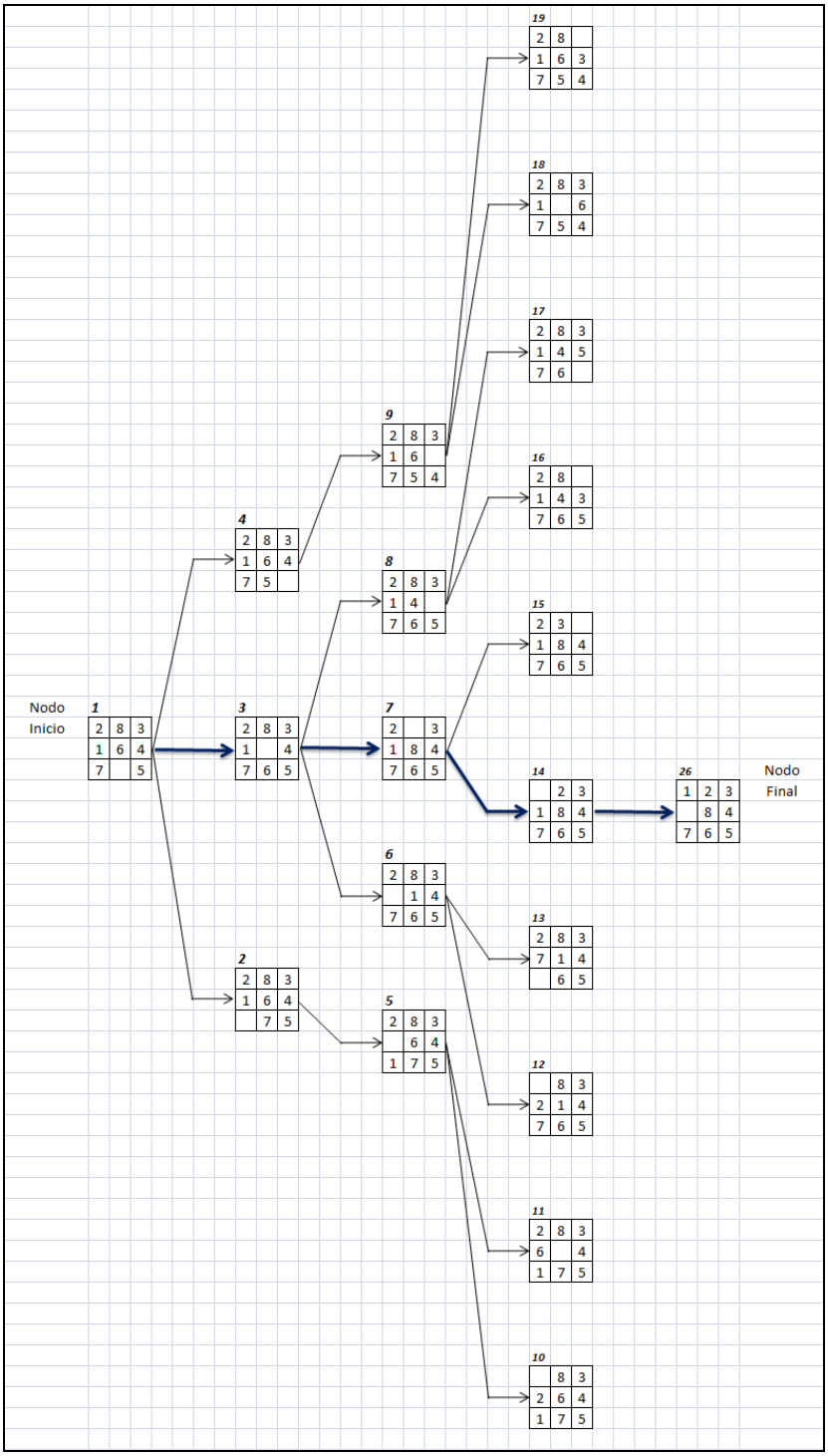


Figura 7: Búsqueda por amplitud para el juego del puzzle.

El nodo inicial (1), se expande en tres nodos (2, 3, 4), obtenidos mediante los movimientos válidos de la celda vacía a la izquierda (2), arriba (3) y hacia la derecha (4). Aunque cada movimiento es reversible, en el ejemplo se omiten los nodos que van de los sucesores a los predecesores. El camino que lleva al objetivo final se encuentra marcado en la Figura 7, por la línea gruesa de color azul.

La ruta para ir de la configuración inicial a la configuración final deseada será a través de los nodos: 1, 3, 7, 14 y 26 con nivel de profundidad 4.

Paso del estado 1 al 3: movimiento hacia arriba
Paso del estado 3 al 7: movimiento hacia arriba
Paso del estado 7 al 14: movimiento hacia la izquierda
Paso del estado 14 al 26: movimiento hacia abajo.

La búsqueda por amplitud tiene la cualidad de que una vez encontrado el nodo final, se tiene también el camino de menor longitud. Sin embargo, el principal inconveniente de este método es que requiere la generación y almacenamiento de todo el árbol de estados, cuyo tamaño crece de manera exponencial respecto a la profundidad del nodo objetivo menos profundo, sin involucrar la variable tiempo, la cual dependiendo de la velocidad de procesamiento podría llegar a alcanzar valores extremadamente grandes.

En la tabla 1, se muestra el tiempo y memoria necesarios para una búsqueda preferente por amplitud para varios valores de profundidad, cuyo factor de ramificación es 10; 1000 nodos/segundo; 100 bytes/nodo.

Profundidad	Nodos	Tiempo	Memoria
0	1	1 milisegundo	100 bytes
2	111	0.1 segundos	11 kilobytes
4	11.111	11 segundos	1 megabyte
6	10^6	18 minutos	111 megabytes
8	10^8	31 horas	11 gigabytes
10	10^{10}	128 días	1 terabyte
12	10^{12}	35 años	111 terabytes
14	10^{14}	3500 años	11.111 terabytes

Tabla 1: Tiempos y memoria en la búsqueda preferente por amplitud. [1]

El número de nodos en memoria es un indicador que permite medir la complejidad de espacio para un determinado tipo de búsqueda.

Relación de complejidad: (O)

La ecuación (2.1), muestra relación de complejidad, la cual está dada por el tamaño del árbol de estados.

$$O = b^d \tag{2.1}$$

b = Factor de ramificación
d = Longitud de ruta

La cantidad máxima de nodos expandidos antes de encontrar la solución es:

$$1 + b + b^2 + b^3 + \cdots + b^d$$

Existen dos medidas de la profundidad: la profundidad máxima del espacio de estados, la cual indica el nivel máximo hasta el cual se expande el árbol; y la profundidad de solución de menor costo, que indica la profundidad en la cual se encontró la solución.

2.2.3 La búsqueda mediante heurística

Como se expresó en la subsección 2.2.2, una búsqueda que expanda e indague exhaustivamente el árbol de estados, resulta ser poco deseable por la cantidad de tiempo y memoria requerida. Una alternativa a este problema consiste en hacer uso de una de las técnicas para búsquedas basadas en información; que para el caso es la búsqueda mediante heurística.

Se define **heurística**[6] como:

a. Técnica de la indagación y del descubrimiento.

b. Búsqueda o investigación de documentos o fuentes históricas.

c. En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

[6] Definición de heurística, tomada de la RAE (Real Academia de la Lengua Española).

“Cualquier técnica que permita mejorar el desempeño de caso promedio en una tarea de resolución de problemas, aunque no necesariamente permita mejorar el desempeño del peor de los casos”.

Una función heurística podrá encontrarse como la resultante de determinar en cada estado, la probabilidad de mejorar, dado que se produzcan a continuación n cambios de estados consecutivos.

La búsqueda mediante heurística se asemeja a las búsquedas en amplitud y en profundidad, pero se diferencia en el hecho de que no se explora el árbol de estados de forma uniforme, ya que primero se intenta examinar aquellos nodos que de acuerdo con cierta información heurística especifica del problema, están situados en el mejor camino hacia el objetivo final.

El procedimiento en este tipo de búsqueda es el siguiente:

a) Haciendo uso de la función de evaluación heurística real $f(n)$, definida sobre las descripciones de los estados, se toma la decisión de que nodo se debe expandir.

b) Se expandirá el nodo n, para el que se obtenga el menor costo de $f(n)$.

c) Se terminará el proceso cuando el nodo a expandir sea el nodo objetivo.

Para el problema del puzzle de ocho piezas, se podría usar una función de evaluación heurística como la medida de la cantidad de piezas que están descolocadas con respecto al nodo final (estado objetivo).

$f(n)$ = Número de piezas descolocadas respecto del estado final deseado.

Con la función de evaluación se tienen dos compromisos, uno es ahorrar el esfuerzo de la búsqueda y otro el calcular el costo de la heurística en cada nodo expandido.

Al usar esta función heurística, se obtiene el un árbol como el de la Figura 8, donde se especifica, el valor que toma la función para cada nodo expandido y el camino hacia el objetivo final, partiendo de la comparación de cada uno de los nodos con el estado o nodo final. Allí solamente se muestran tres niveles de profundidad, donde se observa que para el nodo inicial la función heurística tiene un costo de cinco (*5)*, puesto que son cinco las piezas que se encuentran descolocadas respecto a la configuración final deseada y así sucesivamente se calcula el costo para los siguientes niveles de profundidad.

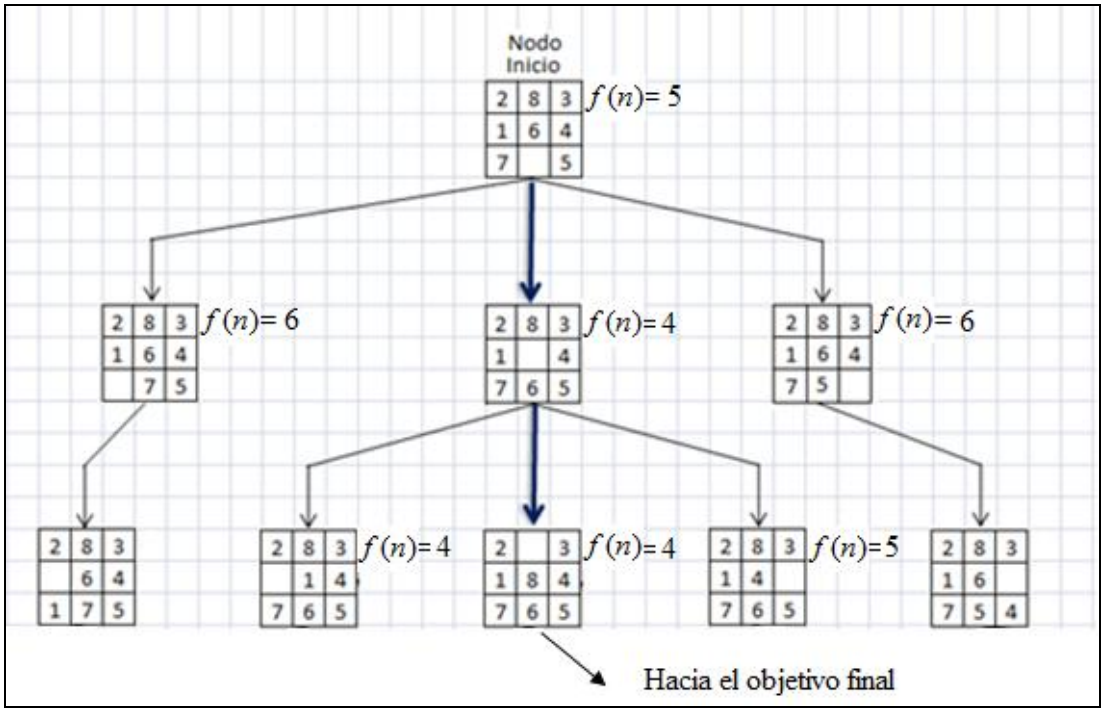


Figura 8: Valores de la función heurística para el puzzle.

El diseño de las heurísticas es un área muy interesante, cuyo estudio permite obtener funciones que son producidas como una sumatoria de subfunciones que interpretan al mismo tiempo más de un criterio de selección.

La función heurística se definiría como la mostrada en la expresión (2.2). Por lo anterior, existen métodos para la elección de las funciones de evaluación heurística, de los cuales se dan referencias. [1][4]

$$f(n) = g(n) + h(n) + i(n) + \cdots \tag{2.2}$$

donde:

$g(n) + h(n) + i(n)$: Subfunciones que permiten mejorar el criterio de selección

Como comentario, se expresa que existen las denominadas estrategias ***metaheurísticas***[7]**,** que si bien es cierto, se salen del alcance de la presente tesis, se advierte de su existencia y que son heurísticas de muy alto nivel para solución de problemas complejos.

[7] Estrategia de alto nivel que guía a otras heurísticas para buscar soluciones factibles en dominios donde la tarea es compleja.

2.3 LA PROPUESTA: BÚSQUEDA MULTIDIRECCIONAL

La estrategia de búsqueda multidireccional propuesta en el proyecto, es una extrapolación conceptual de la búsqueda bidireccional, mencionada en la sección 2.2, como una de las estrategias existentes.

En ella, se plantea la búsqueda de objetivos comunes por parte de una serie de agentes auxiliares, los cuales aportarán a la solución del problema, de acuerdo a su capacidad de expansión dentro del universo de estados. Cada agente auxiliar ayuda a encontrar la solución, de manera que comparta información con los demás agentes involucrados.

2.3.1 Estrategia de búsqueda bidireccional

En la búsqueda bidireccional, el problema consiste en trazar una ruta para ir desde el origen hasta la meta; mientras de forma paralela y simultánea, se intenta trazar otra ruta que conduzca de la meta al origen. En este ejercicio, dado que la búsqueda se está realizando en un espacio de estados comunes -*al origen y a la meta*-, puede darse el caso en el que la ruta que se está construyendo de origen a meta, se cruce (conecte) con la que se está construyendo para ir de la meta al origen, caso en el cual, se habrá establecido un camino para unir los dos puntos.

En la Figura 9, se representa de forma simplificada el algoritmo de la búsqueda bidireccional, donde el origen (inicio) en el diagrama de flujo de la izquierda, se convierte en la meta para el de la derecha y de forma contraria, la meta en el de la izquierda es el origen en el de la derecha. Nótese en el diagrama lo siguiente:

- Una instancia es una copia completa del programa simulador -*aunque también puede ser solo la parte que ejecuta la búsqueda*-, que se pone a correr (ejecutar) en otro procesador. Es bueno señalar que aunque todas las instancias comparten información -como su repositorio de nodos-, existe información que es de uso exclusivo de cada Instancia.

- La búsqueda que va del origen a la meta, se vale del “Repositorio_1”, que es el lugar donde se registran todos y cada uno de los estados o nodos que se van expandiendo. De forma similar, la búsqueda que va de la meta al origen, se vale del “Repositorio_2”.

- Se indaga la existencia en la Instancia_2 de todo nodo generado en la Instancia_1; lo que se representa mediante una flecha bidireccional.

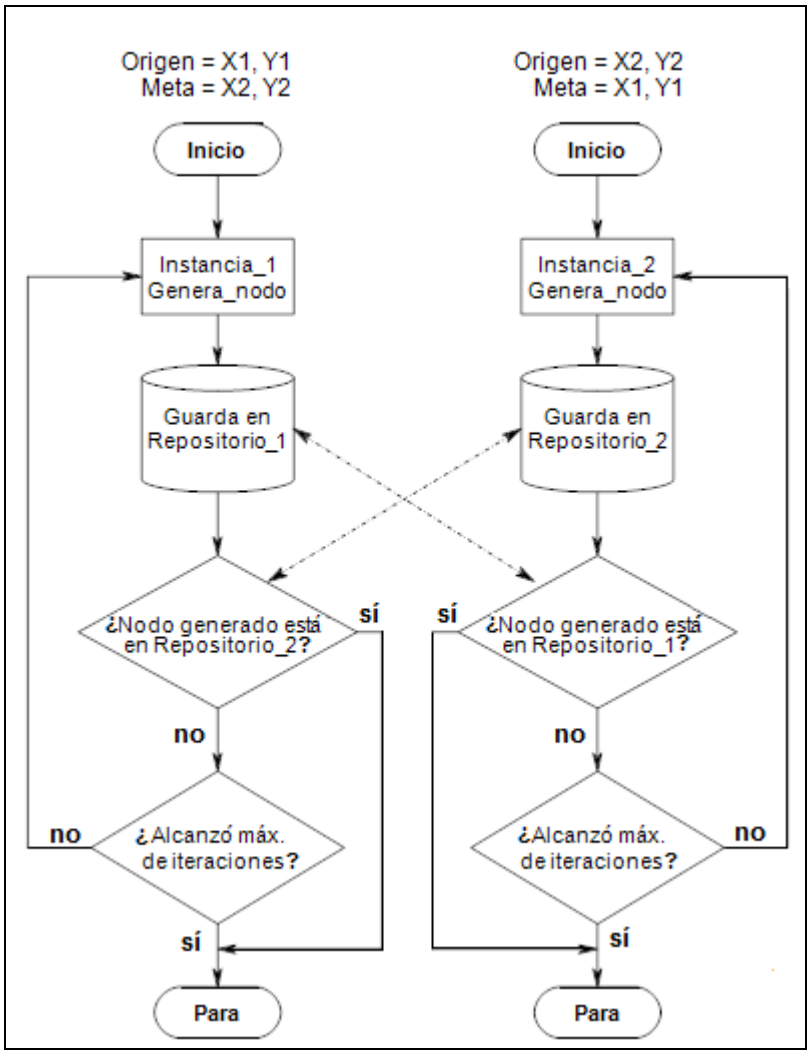


Figura 9: Algoritmo de la búsqueda preferente por amplitud bidireccional.

- Si un nodo generado en una de las dos (2) instancias se encuentra en el repositorio de la otra instancia, es porque se estableció una ruta o conexión entre los nodos -origen y meta-; de lo contrario, si no se ha llegado al número máximo de iteraciones propuesto, se genera un nuevo nodo en la instancia correspondiente y se repite el proceso.

- Cuando se alcance el máximo número de iteraciones sin encontrar solución, es porque se tiene un espacio sin solución; por tanto, el algoritmo se detiene y no hubo convergencia en el problema.

2.3.2 Estrategia de búsqueda multidireccional

Teniendo como base la estrategia de búsqueda bidireccional anteriormente expuesta, se presenta en la Figura 10, el funcionamiento de la estrategia de búsqueda multidireccional propuesta en la presente tesis, teniendo como información lo siguiente:

Nodo inicial: Rojo

Nodo meta: Verde Brillante

Nodos auxiliares: Gris, Violeta, Amarillo, Marrón, Azul, Verde.

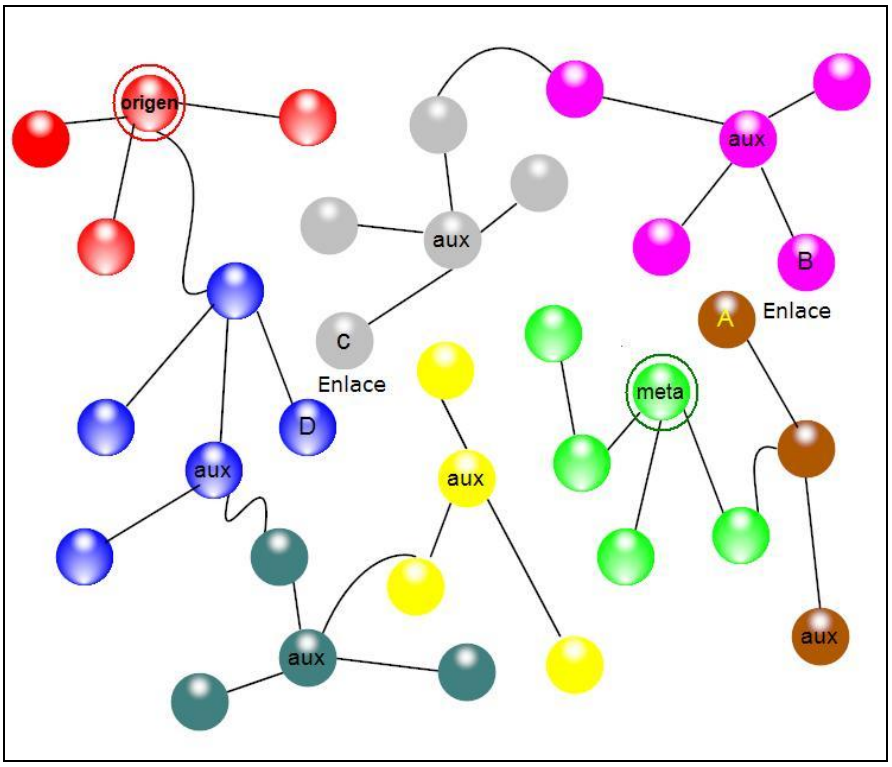


Figura 10: Representación de búsqueda multidireccional.

Si el ramal marcado como **A** en el nodo marrón, se enlaza al nodo violeta, en el punto **B**, entonces el nodo gris y todos sus ramales adquirirán el “conocimiento” (la

ruta) para llegar al nodo meta. Si posteriormente el ramal marcado con una **C** en el nodo gris, hace contacto con **D** en el nodo azul, se habrá completado un camino para ir del nodo inicial al nodo final.
Si sobre el nodo inicial, el nodo meta y cada uno de los nodos auxiliares, se "implanta" un *agente inteligente*, se puede hacer uso de un procesador para ejecutar una búsqueda por cada agente instanciado[8]. Una de las mayores ventajas de este arreglo, es la capacidad de utilizar procesadores en paralelo, uno por cada agente.

Al implementar un tipo de solución donde sean varios los procesadores o hilos de programación en paralelo, la disminución del tiempo de procesamiento es significativa, logrando aplicaciones que se puedan ejecutar en tiempo real, no obstante la complejidad implícita en la búsqueda de la(s) solución(es) si es que existe (ver hipótesis en la sección 1.2).

La estrategia de búsqueda multidireccional se cimenta en dos pilares fundamentales:

a) Generación de nodos auxiliares que produzcan una rápida convergencia en la solución del problema, teniendo en cuenta que cada nodo es en realidad un agente inteligente con capacidades idénticas a las del Softbot. Para lo anterior se necesita:

- Disponer de una adecuada cantidad de nodos auxiliares.

- Garantizar que los nodos auxiliares se encuentren en el espacio de estados dentro del cual se circunscribe el problema.

- Propender porque los nodos auxiliares seleccionados, sean los que ayuden de la mejor forma posible a la convergencia rápida en la solución del problema.

La teoría de los algoritmos evolutivos dentro de los cuales se ubican los algoritmos genéticos, ofrece un marco de trabajo que, partiendo de un "universo inicial" y realizando cruces y mutaciones sucesivas de los elementos de dicho universo, se pueden reproducir nuevas entidades, de las cuales se van escogiendo mediante un proceso de selección aquellas cuyas características estén más cercanas a las deseadas. En consecuencia, la solución del cálculo de la cantidad y obtención de los nodos auxiliares, puede hacerse mediante la aplicación de un algoritmo genético, en donde la función de prueba o selección,

[8] Una copia de un mismo programa puede ser ejecutada en un procesador independiente. A cada copia con su conjunto de datos y variables asociadas se les conoce como instancia.

es la que garantiza que una de las características de las entidades escogidas, esté dentro del espacio de estados en el cual se circunscribe el problema.

b) Uso recursivo de una estrategia equivalente a la de la búsqueda bidireccional, aplicada simultáneamente a todos y cada uno de los nodos proveídos mediante algoritmos genéticos -o algún otro método de fácil implementación-, según el numeral anterior.

Resumiendo, se señala que la cantidad y ubicación de los nodos auxiliares se puede determinar con técnicas que dependen del tipo de entorno del problema, yendo desde la simple distribución geométrica -*como es el caso de la presente tesis, donde el usuario podrá escoger la cantidad de nodos o agentes auxiliares*-, o la utilización de algoritmos genéticos.

En la búsqueda multidireccional se tendrán en cuenta los siguientes aspectos:

- El entorno es un lugar común a todos los agentes y las acciones de uno cualquiera afectan a todos los demás.

- Cada agente contribuye con sus propias percepciones al levantamiento del entorno común.

- Los agentes se diseñan con las capacidades de comunicación necesaria para compartir conocimiento -en especial los caminos o rutas entre ellos-, desechando información no relevante que no conduzca a la solución del problema-.

- Los agentes tendrán sus propios objetivos, es decir, algunos ayudaran a encontrar la meta mientras que otros ayudarán a encontrar al origen.

- La solución de la ruta para ir del origen a la meta será la primera que se encuentre, descartando otras posibles soluciones.

- La ruta hallada, como solución por la contribución de los agentes, podrá contener caminos con derivaciones, los cuales son producto de las indagaciones de otros agentes, por tanto, se depuran para mostrar una ruta más simplificada del origen a la meta.

Capítulo 3

MARCO REFERENCIAL

Inicialmente se construye el simulador de escenarios (SIRUM), como un aplicativo donde el usuario puede crear diferentes entornos y probar la estrategia de búsqueda, la cual posibilita la toma de decisiones de tal manera que el agente autónomo se pueda desplazar por el ambiente, reaccionando ante posibles colisiones y buscando el destino o meta previamente definida.

La forma de programación del software permite la adición de módulos funcionales a medida que se avanza en el desarrollo del aplicativo, lo cual hace que el software crezca de forma estructurada sin perderse el control sobre el mismo.

Se dota al ***agente inteligente (Origen)*** con sensores virtuales, los cuales entregan información del ambiente, a medida que el Softbot se desplaza. La información recogida por los sensores se deposita en tablas y se utiliza para realizar un levantamiento interno del escenario.

Las pruebas se enmarcan sobre escenarios de diferentes niveles de complejidad y se realizan experimentos conducentes a ajustar tanto los algoritmos como la heurística empleada. Adicionalmente, se hace uso de un sistema de huellas descritas por el agente inteligente, con el fin de que el usuario del sistema pueda observar como es el trabajo de la estrategia de búsqueda.

3.1 GENERALIDADES

La creación del simulador implica la confección de un generador de escenarios que permite la modelación de ambientes atestados, con alto nivel de complejidad y funcionalidad, permitiendo incorporar al ambiente, paredes, puertas móviles, maquinaria, equipo de oficina y vehículos o plataformas móviles.

El agente inteligente, que se encuentra incorporado en el simulador, se diseña con ocho sensores virtuales que le permiten percibir el entorno del ambiente donde está ubicado, incluido la percepción de objetos que se desplacen cerca de él.

La información recabada se registra en un sistema o arreglo de capas matriciales implementadas físicamente en tablas contenidas en una base de datos. Si bien es cierto que el objetivo último es la navegación autónoma del Softbot dentro del escenario simulado, se partió de una navegación manual, donde el usuario a su gusto, puede hacer que el Softbot se mueva paso a paso en direcciones válidas.

Con el ánimo de demostrar el funcionamiento en paralelo de los agentes inteligentes, y la capacidad del simulador de responder a eventos que en el mundo real, se construyeron las funcionalidades necesarias para el trabajo en red de n ordenadores.

3.2 MODELO DE DESARROLLO EN ESPIRAL

La ingeniería del software, se estructura con base en una serie de modelos que muestran las distintas etapas y estados por los que pasa un software, desde su concepción inicial, pasando por su desarrollo, puesta en marcha y posterior mantenimiento, hasta la retirada del producto. A estos modelos se les denomina *modelos de ciclo de vida del software*. El primer modelo concebido fue el de Royce [5], más comúnmente conocido como desarrollo en cascada o desarrollo lineal secuencial. Este modelo establece que las diversas actividades que se van realizando al desarrollar un producto software se suceden de forma lineal.

Boehm [5], autor de diversos artículos de ingeniería del software; modelos de estimación de esfuerzo y tiempo que se consume en hacer productos software; y Modelos de Ciclo de Vida; ideó y promulgó un modelo desde un enfoque distinto al tradicional en Cascada:

El ***Modelo Evolutivo Espiral:*** Su modelo de ciclo de vida en espiral tiene en cuenta fuertemente el riesgo que aparece cuando se desarrolla software. Para ello, se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgo más asumible y se hace un ciclo de la espiral. Si el cliente quiere seguir haciendo mejoras en el software, se vuelven a evaluar las distintas nuevas alternativas y riesgos y se realiza otra vuelta de la espiral. Así hasta que llegue un momento donde el software desarrollado sea aceptado y no necesite seguir mejorándose con otro nuevo ciclo.

Este modelo fue propuesto por Boehm en 1988 [5]. Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. Se suele interpretar como si en cada ciclo de la espiral se sigue un modelo en cascada, pero no necesariamente debe ser así. En cada vuelta o iteración hay que tener en cuenta los siguientes aspectos:

Los objetivos: Cual es la necesidad que debe cubrir el producto, de tal manera que se pueda dar solución a un problema real.

Alternativas: Las diferentes formas de conseguir los objetivos de forma exitosa, desde diferentes puntos de vista a saber:

- Características, experiencia del personal, requisitos a cumplir, etc.
- Formas de gestión del sistema.
- Riesgo asumido con cada alternativa.

Desarrollar y Verificar: Programar y probar el software. Si el resultado no es el adecuado o se necesita implementar mejoras o funcionalidades.

Luego se planificarán los pasos siguientes y se comienza un nuevo ciclo de la espiral. La espiral tiene una forma de caracol y se dice que mantiene dos dimensiones, la radial y la angular:

- Angular: indica el avance del proyecto software dentro de un ciclo.

- Radial: indica el aumento del costo del proyecto, ya que con cada nueva iteración se pasa más tiempo desarrollando.

Este sistema es muy utilizado en proyectos grandes y complejos como por ejemplo, la creación de un sistema operativo (OS) que ejecutan la gestión de equipos informáticos.

Al ser un modelo de ciclo de vida orientado a la gestión de riesgo (*que se involucra en todo proyecto por factores tanto previsibles como imprevisibles)*, se dice que uno de los aspectos fundamentales de su éxito radica en que el equipo que lo aplique tenga la necesaria experiencia y habilidad para detectar y catalogar correctamente los riesgos. En la Figura 11, se presenta el modelo en espiral.

Para cada ciclo habrá cuatro actividades:

a) Fijar los productos definidos a obtener: requerimientos, especificación, manual de usuario.

b) Fijar las restricciones.

c) Identificación de riesgos del proyecto y estrategias alternativas para evitarlos.

d) Planificación inicial o previa (actividad que se realiza solo una vez).

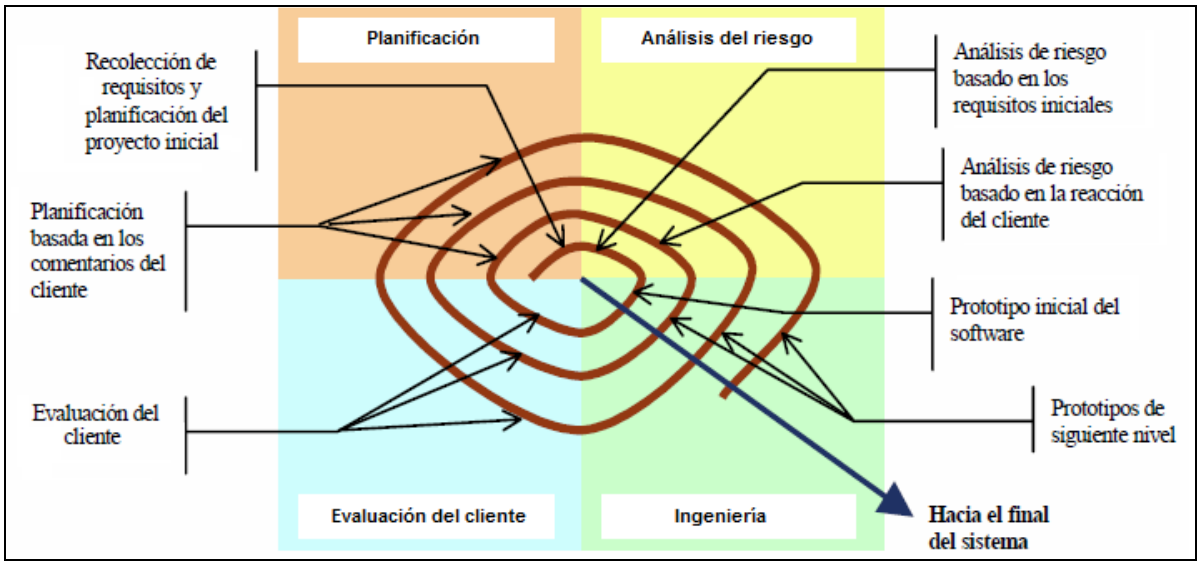


Figura 11: Representación modelo en espiral [5].

El modelo en espiral descrito hasta este punto, fue precisamente el que se siguió en la planificación, diseño, implementación, prueba y ajuste del simulador de escenarios para la búsqueda de rutas en robótica móvil, propósito de la presente tesis.

Capítulo 4

COMPONENTES DEL PROYECTO

4.1 CONFORMACIÓN

El proyecto está conformado por cinco (5) componentes funcionales que trabajan de forma integrada y armónica en la consecución de los objetivos planteados:

- Generador de escenarios.
- Panel de visualización:
  - Visualizador de tablas.
  - Representación dinámica del escenario, el Softbot y la ruta.
- Los sensores virtuales.
- La estrategia de búsqueda.

El diseño plantea un sistema totalmente integrado, pero bajo *criterios de desacople* [6], que facilitan la configuración de los cinco módulos de tal forma que se puedan soportar, mantener o reemplazar por otros.

Los criterios de acople y desacople, garantizan que un componente pueda ser reemplazado por otro que se desee probar, como por ejemplo, la estrategia de búsqueda preferente por amplitud para uso multidireccional, que podrá ser reemplazada por cualquier otro tipo de búsqueda que cumpla con los criterios de interface definidos en el módulo del Softbot, el cual es, en sí mismo, el articulador de todo el sistema.

Igualmente, se pueden adicionar nuevos sensores o actuadores virtuales que pueden ser reemplazados por sensores físicos, cuando se desee migrar el software de control ya depurado, a un robot móvil, mediante el uso de **Drivers** como mecanismo de interface software-máquina.

En la Figura 12, se representa esquemáticamente del **SIRUM** (**si**mulador de escenarios para la búsqueda de **ru**tas en robótica **m**óvil).

El diseño modular desacoplado del proyecto, permite que los subsistemas puedan operar con algún grado de independencia (flexibilidad), lo que finalmente se traduce en una facilidad para introducir o reemplazar componentes con nuevas o mejoradas prestaciones; de igual manera, queda preparado el terreno para la materialización de componentes mediante hardware.

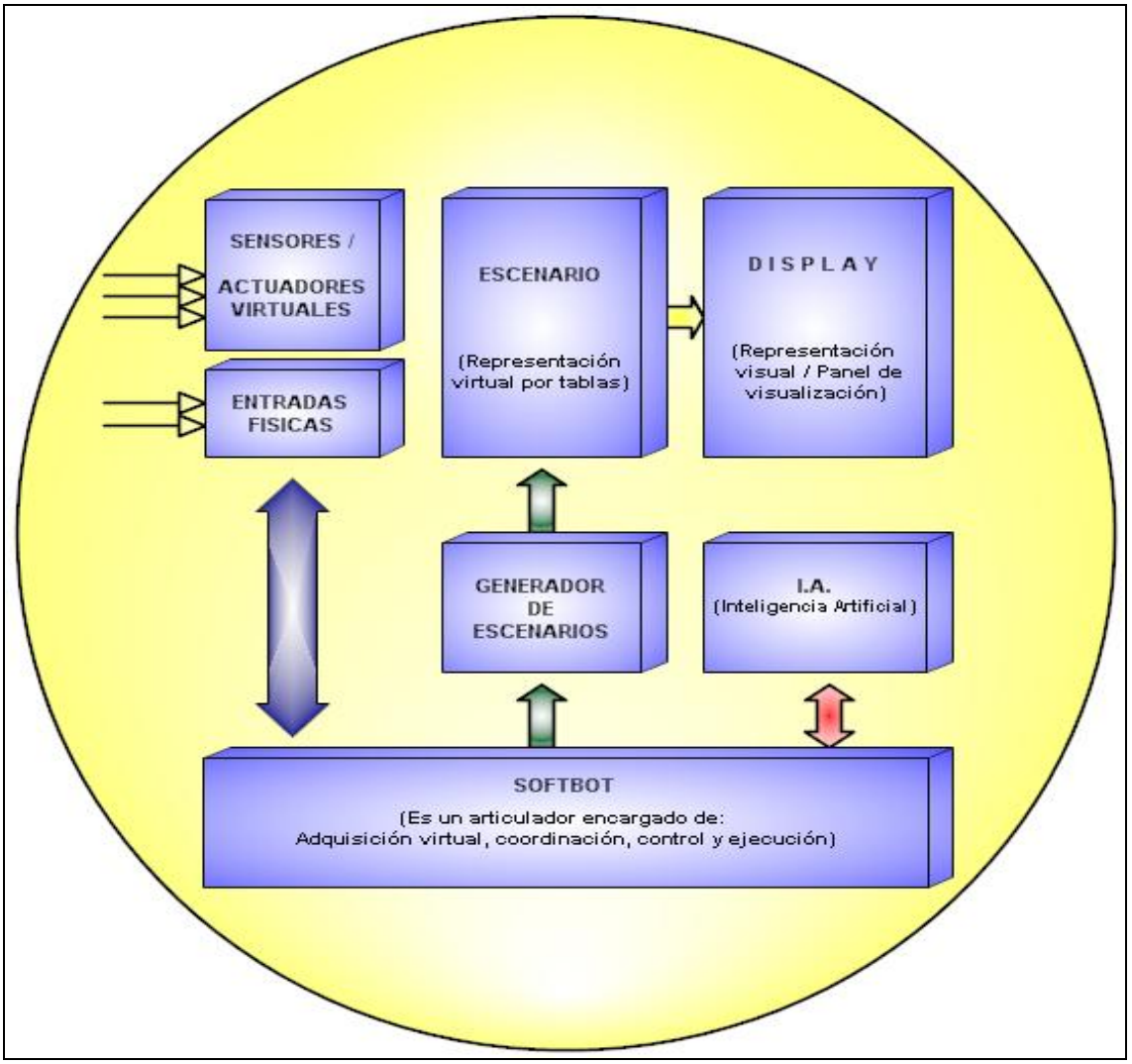


Figura 12: SIRUM.

En el esquema del SIRUM, se encuentran integrados los módulos mencionados, teniendo en cuenta que cada uno se construye con base en el manejo de tablas dinámicas y desarrollo de software de alto nivel. Se destaca en el esquema la existencia del modulo correspondiente a inteligencia artificial, en el cual se encuentra inmersa la estrategia de búsqueda planteada.

4.2 GENERADOR DE ESCENARIOS

Por escenario se entiende cualquier región bidimensional, delimitada por paredes, en la cual se moverá el agente inteligente (Softbot). El escenario eventualmente podrá incluir obstáculos ya sean fijos o móviles, los cuales tendrán características propias dentro del entorno, como sería por ejemplo sus dimensiones.

Un componente importante del proyecto **SIRUM,** es la construcción de un software de alto nivel para la creación y edición de escenarios de dos (2) dimensiones, aplicativo donde el usuario tiene la posibilidad de generar diferentes ambientes. En dichos ambientes o escenarios, se realizarán las pruebas de la estrategia de búsqueda, la cual tiene como finalidad encontrar un camino para ir de un punto a otro.

4.2.1 Construcción de ambientes

Los escenarios construidos pueden ser arbitrarios y están limitados sólo por la imaginación del usuario. Un caso particular son los “laberintos”, que interconectan lugares a través de pasillos, en este caso el uso de obstáculos o paredes que encierren completamente una región extensa del escenario, no tendrían posibilidad de ser navegadas; por tanto, se tomarán como un objeto u obstáculo de gran tamaño. En la Figura 13, puede observarse el área de un escenario con dimensiones de M filas por N columnas, construido a partir de una unidad básica llamada **UMR** (Unidad Mínima de Representación), la cual puede ser modificada según el área y los pixeles que se quieran. Inicialmente ésta unidad se tiene referida a un tamaño de 10 x 10 pixeles.

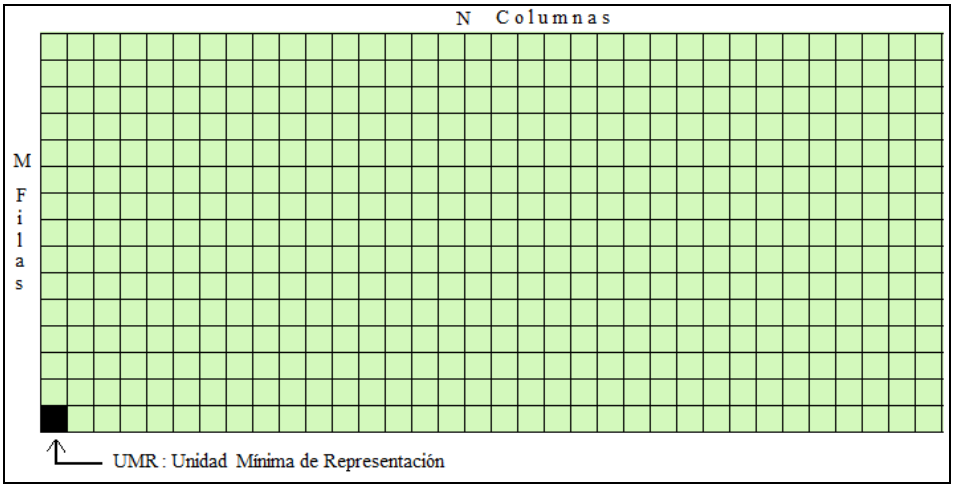


Figura 13: Área del escenario.

La Figura 14, es un ejemplo de un ambiente delimitado por paredes (el color negro describe un obstáculo fijo), con algunas divisiones internas. Por otra parte, debe considerarse el uso de pasillos horizontales o verticales, cuyo ancho mínimo sea aproximadamente el mismo del robot de software, el cual inicialmente se considera con las dimensiones de una UMR.

El ejemplo muestra un segmento del escenario con coordenadas:
(X_j , X_k) para las columnas (X_j = X0035, X_k = X0069)
(Y_j , Y_k) para las filas (Y_j = Y0001, Y_k = Y0015)

Al escenario se agregan con iconos que representan objetos o contienen información, como por ejemplo:

- Una pequeña esfera verde del tamaño de una UMR, para representar la meta.
- Una esfera roja para representar el origen, ubicación del agente inteligente.
- Un segmento del escenario compuesto por una o más UMR de color negro indica un obstáculo, como es el caso de las paredes.
- Un segmento de color naranja *-una vez más conformada a partir de las UMR-* representará un objeto móvil, como es el caso de una puerta corrediza.

Una vez que el agente inteligente comience su desplazamiento por el escenario, realizará un levantamiento del mismo, mediante la representación unívoca del lugar, lo que se realiza a través de planos de software.

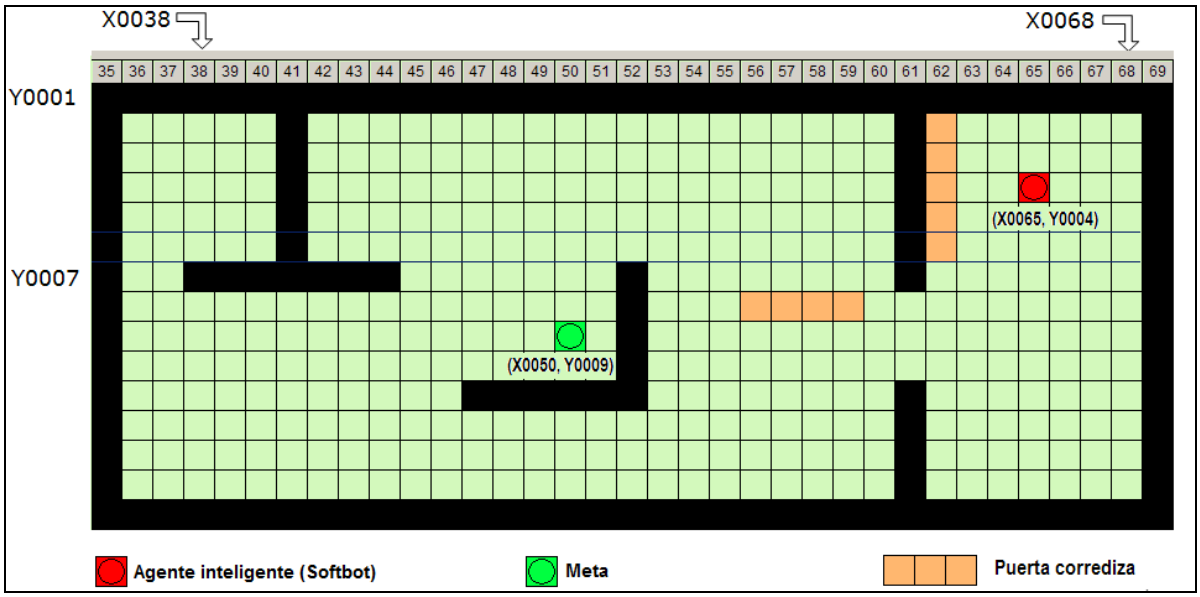


Figura 14: Ejemplo de un escenario con paredes, la meta y el Softbot.

A medida que el usuario construye un escenario, el software automáticamente va alimentando las tablas de datos que contienen la representación de los elementos que se incluyen en el ambiente. La construcción del escenario es un proceso dinámico, en el sentido de que podría ser modificado durante la simulación, agregando o eliminado elementos, mediante el panel de herramientas -*descrito en el anexo Manual del usuario*-, usando el mouse del computador.

El Softbot (*agente de software*) es una entidad capaz de desarrollar una técnica y mostrar un comportamiento dentro de un escenario, reconociendo objetos y tomando decisiones más o menos complejas, igual a lo que realizaría un robot físico que resuelve laberintos usando por ejemplo redes neuronales[9].

Mediante una interfaz gráfica simple, el usuario tiene la posibilidad de generar el ambiente de simulación, indicando las características del escenario y definiendo el punto de inicio de donde comenzará el Softbot su recorrido, así como el punto de meta al que debe llegar. Las simulaciones virtuales podrán almacenarse como un archivo para reproducción futura, u obtención de atajos, siendo la base del análisis del llamado *conocimiento de máquina*.

4.2.2 Estructuración de un ambiente

Una facilidad que tiene el aplicativo es el selector de proyectos, allí se tiene la alternativa de escoger un ambiente previamente guardado o crear uno nuevo con características particulares.

Para crear un nuevo proyecto se define un nombre, el número de filas y el número de columnas del área rectangular que constituye el escenario (área de trabajo); de tal forma que dicha información es utilizada por el software, para definir las características y el tamaño de las diferentes tablas que integran un proyecto, además sirven entre otras cosas, para “digitalizar” el plano del escenario construido, así como los diferentes objetos que involucra: el origen, la meta, el Softbot, objetos móviles, etc.

En términos prácticos, unas tablas de datos se utilizan para dibujar y conservar el escenario gráfico de interacción con el usuario (Display) y otras como estructura matricial o vectorial para realizar los análisis de ingeniería a que haya lugar, con el propósito de que el Softbot navegue por el escenario y encuentre la ruta del origen a la meta.

[9] Redes nneuronales artificiales: Son una simulación de las redes neuronales biológicas, haciendo uso de las matemáticas y los sistemas computacionales.

El tener un proyecto guardado, mejora los tiempos de simulación, porque ahorra tiempo en la creación o modificación del escenario. Se debe garantizar que no haya limitaciones físicas ni posibles casos no permitidos, como sería la ubicación de elementos en posiciones no válidas (por ejemplo, un elemento en medio de un muro).

La Tabla 2, presenta un ejemplo de la información que podrían tener diferentes elementos constitutivos de un escenario cualquiera y de los objetos o entidades que lo conforman:

Elemento / Cadena	Significado
X	Muro fijo
S	Objeto móvil (Softbot)
M	Objeto móvil (vehículo, plataforma, etc.)
H	Huella o señalización.
000255000	Color verde.
255000000T0800C	Punto del escenario donde se tiene una temperatura (T) de 800 °C, representada con color rojo mediante el código internacional.
S0##	Softbot en el origen
S2##	Softbot en la meta
EP03670240	Obstáculo reportado por escaneo de proximidad en Y=0367, X=0240

Tabla 2: Ejemplo con la información de elementos.

La sola inspección de la tabla, muestra que algunos elementos constitutivos del escenario, requieren diferentes tamaños de las cadenas o campos de almacenamiento. Lo anterior, permite que mediante códigos (cadenas de caracteres), se pueda identificar cada celdas que constituyen la matriz MxN, que representa un entorno a nivel interno.

4.2.3 Escenario representado en capas o niveles

Las diferencias en el tipo y tamaño de los datos que describen objetos como son el Softbot, la meta y el ambiente mismo, hacen recomendable construir un sistema de capas de información que optimice el almacenamiento y la búsqueda. De este modo, mediante capas, también se representan los diferentes estados por los que se atraviesa un escenario y los elementos en él contenidos.

El modelo de capas mostrado en la Figura 15, provee diferentes niveles de representación o abstracción y permite resolver el problema de representar gran cantidad de información heterogénea, agrupándola según su afinidad.

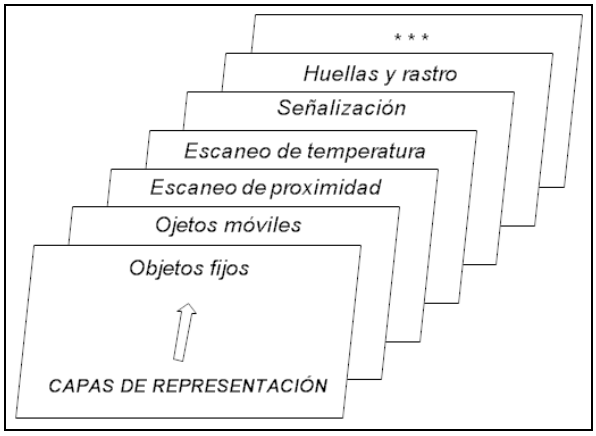


Figura 15: Esquema en capas.

La representación de la información mediante capas (tablas), separadas por cada tipo de información afín, hace posible un diseño que ahorra espacio en las celdas de las tablas. Esta representación del escenario mediante el modelo de capas, facilita diseñar un esquema de compresión de datos, el cual reduce de gran manera la cantidad de información para representar un escenario y los diferentes estados[10]por los que transita conforme transcurre el tiempo. Mientras un objeto o grupo de objetos cambia en el tiempo *-ya sea de posición o de cualquiera de sus características-*, el resto del escenario muy probablemente permanezca invariante.

Los cambios de estado del escenario o de algunos de los objetos o entidades en él contenidos, podrán ser representados como variaciones específicas de la tabla que los representen, con la sola adición al estado anterior, de vectores de cambio. (Este es un principio de compresión de video muy utilizado en la actualidad).

Debido a la necesidad de implementar un sistema de representación en el tiempo, tanto del escenario como de los eventos que ocurren, se ideó un sistema de compresión que funciona de la siguiente forma:

Se registra completamente en tablas el escenario en un instante de tiempo (t_o) y a medida que transcurre el tiempo se hacen nuevos registros, de forma que para expresar la diferencia entre unos y otros sólo es necesario almacenar en tablas lo que cambió, puesto que lo demás permanece igual.

[10] Los estados son producidos por variaciones en el tiempo del escenario y los elementos en él incluidos.

Tomando como ejemplo la variación de la posición de un objeto móvil en el escenario, dicha variación se registra mediante las coordenadas inicial y final; y no es necesario registrar todos los puntos intermedios.

Lo anteriormente expresado, significa que un determinado estado E_n, se puede representar como el equivalente al estado anterior E_{n-1} más la sumatoria de los cambios entre dicho estado y el estado E_n, como se muestra en la expresión (4.1).

$$E_n = E_{(n-x)} + \sum_{j=n-x}^{n} cambio_j \qquad (4.1)$$

4.3 EL SIMULADOR Y LA EDICIÓN DE ESCENARIOS

El simulador está configurado a partir de módulos funcionales que trabajan de forma integrada para aceptar parámetros del usuario, realizar procesos automáticos y ejecutar tareas de búsqueda de rutas. También se podrá observar el funcionamiento de la estrategia de búsqueda multidireccional donde interactúan varios agentes de software.

La Figura 16, muestra los componentes del simulador repartidos en tres (3) áreas básicas a saber:

a) Área de selección principal, donde se definen los parámetros básicos de un proyecto. Allí se podrá seleccionar un proyecto previamente creado, el cual tendrá información del tamaño del escenario y algunas características particulares para la simulación.

b) Área de monitoreo direccional del Softbot, en cada instante se muestran los movimientos válidos que puede realizar el agente de software en función de su posición y de los posibles obstáculos dentro del escenario. Haciendo un simple clic en cada botón de navegación es posible hacer que el Softbot avance un paso en dicha dirección, esta opción se tiene para la navegación manual.

c) Área de trabajo configurada mediante pestañas, dependiendo del proceso o tarea a realizar, se selecciona mediante un clic la pestaña de interés. Algunas de las pestañas, como es el caso del Display, se subdividen en dos (2) sub-áreas una de ellas para selectores de parámetros y otra para los botones de comandos.

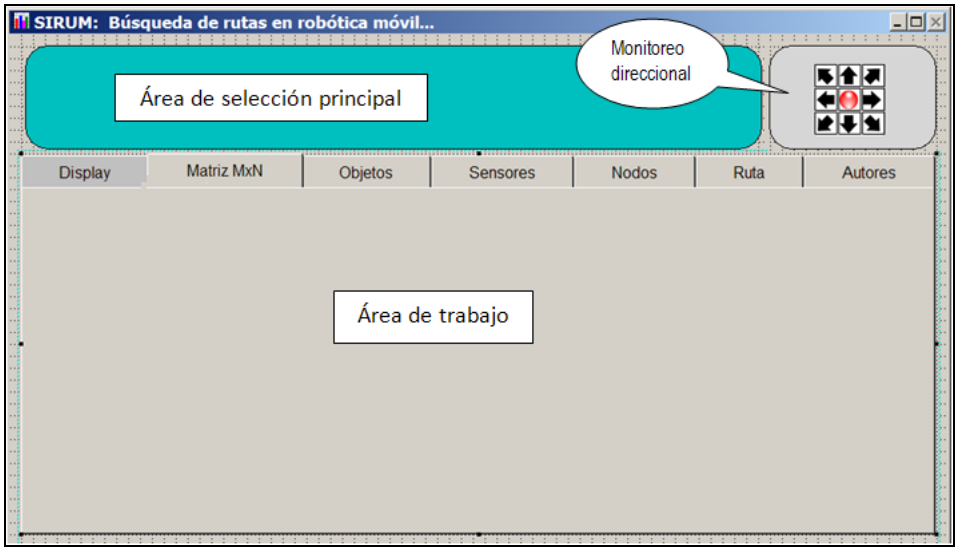


Figura 16: Componentes de simulador.

4.3.1 Panel de visualización (*Display*)

El panel de visualización es el lugar donde se representa gráficamente el escenario, los objetos y los eventos. Así mismo, es el área de trabajo con la cual interactúa el usuario del SIRUM. La Figura 17 muestra un ejemplo del panel de visualización, en la "pestaña *Display"* del área de trabajo.

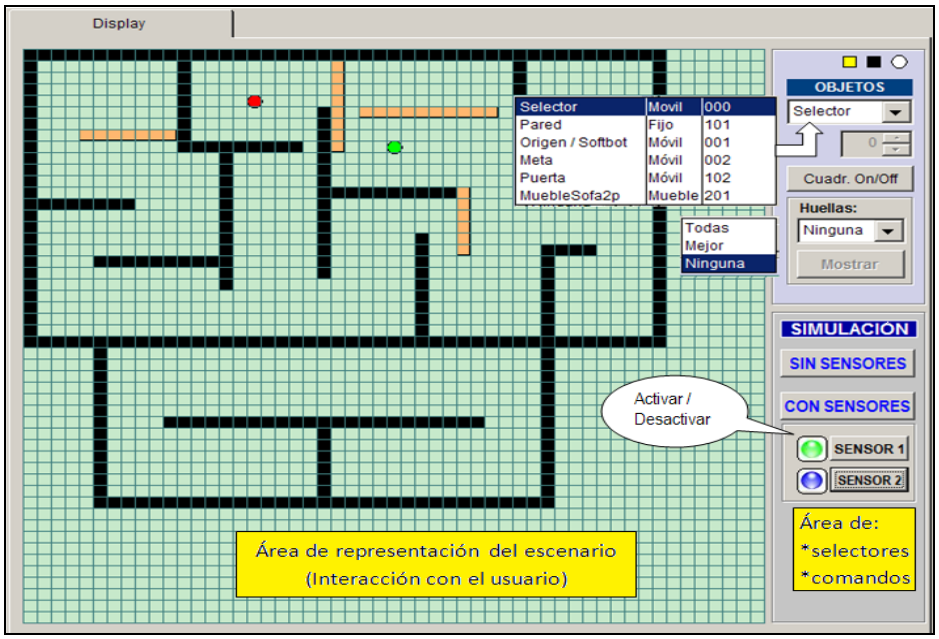


Figura 17: Display (visualizador del escenario).

El panel de visualización esta subdividido en dos (2) áreas:

- Área de representación del escenario y visualización de eventos.
- Área de comandos y selectores de objetos.

A medida que se construye un escenario y se le incorporan elementos tales como el Softbot (origen), meta, objetos fijos o móviles, etc., se actualizan las tablas asociadas a los diferentes componentes del proyecto. Como se expresó anteriormente, el generador de escenarios es un software que permite representar un área delimitada dentro de la cual se encuentran:

Objetos fijos: Representan paredes, máquinas, equipos de oficina, etc.

Objetos móviles: Son todos aquellos que cambian su localización en un instante de tiempo t determinado. Un ejemplo para un objeto móvil es el icono que representa al Softbot (robot de software), el cual se desplaza por el escenario.

Para una mejor comprensión de como se traza un escenario y se le incorporan elementos, en el anexo (guía del usuario) se tiene mayor claridad.

Una vez trazado el escenario incluyendo el origen (Softbot en color rojo) y la meta (en color verde), se procede a seleccionar el tipo de búsqueda a realizar:

a) Búsqueda sin sensores.

b) Búsqueda con sensores emulados.

c) Búsqueda multidireccional.

En cualquier caso, es posible optar por seleccionar que el simulador dibuje las huellas en los lugares por donde el agente de software realiza la búsqueda.

4.3.2 Escenario generado en tablas

Al crear un nuevo proyecto -como por ejemplo el número “006”-, se generan varias tablas del tipo DBF (Data Base File), donde el nombre de todas comienza por la raíz “Cp”, seguido de tres dígitos que indican el número del proyecto y un guion bajo (“_”) más dos (2) dígitos que indican el número de la capa de información contenida en la tabla.

En la Tabla 3, se muestra un ejemplo de las tablas DBF que se podrían generar en un proyecto determinado.

Nombre	Significado
Cp006_01.dbf	Matriz MxN con una celda por cada UMR
Cp006_02.dbf	Tabla de objetos codificados vectorialmente
Cp006_03.dbf	Tabla con las lecturas de los sensores emulados

Tabla 3: Archivos **DBF** creados por un proyecto.

4.3.2.1 *Representación de la matriz MxN*

Al dibujar un escenario como el mostrado en la Figura 18, donde aparte de las paredes de color negro, se adicionaron el origen de la búsqueda (posicionamiento inicial del Softbot) de color rojo y la meta a encontrar de color verde. Se genera automáticamente el llenado de la tabla **Cp006_01.dbf**.

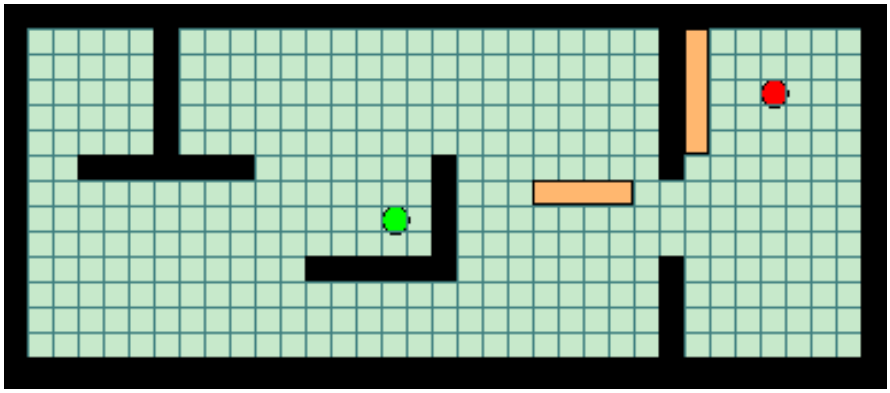


Figura 18: Modelo de un escenario.

En la Figura 19, se muestra la codificación de la tabla, en la que por defecto, las celdas de la Matriz MxN están vacías, es decir, no contienen información codificada y sólo se registran datos en las celdas donde existen objetos, ya sean fijos o móviles.

La codificación del escenario, se utiliza como información en la búsqueda del camino que lleva del origen a la meta, su interpretación es la siguiente:

- Cuando en una UMR cualquiera del escenario se dibuja una pared, la celda correspondiente de la Matriz MxN será almacenada con la cadena "**NG**". (se colocó fondo negro por conveniencia de representación).

- Los objetos móviles -como una puerta que se desliza-, se registran mediante el código "**NA**".

- Al ubicar el Softbot sobre una UMR del Display, la celda correspondiente en la tabla Cp006_01.dbf es alimentada con el código "**RJ**".

- Al ubicar la meta sobre una UMR del Display, la celda correspondiente en la tabla Cp006_01.dbf es alimentada con el código "**VB**".

Figura 19: Modelo de un escenario codificado en tabla.

Por último, si después de dibujar el escenario en la pestaña Display del área de trabajo, que se tiene incluida como uno de los componentes del simulador, se hace clic con el mouse en la "pestaña Matriz MxN", se presentará la codificación del escenario, haciendo uso de la tabla Cp006_001.dbf, tal como se muestra en la Figura 20.

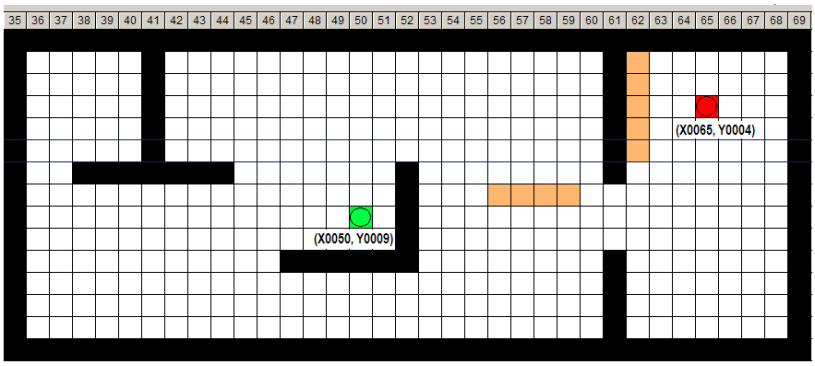


Figura 20: Escenario visto en el Display.

Cuando un objeto móvil -incluido el Softbot- cambia de posición, de forma automática se actualizan tanto las celdas que dejan de estar ocupadas como las que pasan a estarlo.

4.3.2.2 *Tabla para representación de objetos*

En la Figura 21 se muestra el contenido de la tabla Cp006_02.dbf, que corresponde a la capa de codificación de objetos (Capa “02”) de un proyecto enumerado como el “006”. En dicha tabla, aparecen todos y cada unos de los objetos que conforman o están dentro del entorno.

Los objetos denominados obj001 hasta obj0012, corresponden a segmentos de recta con sus respectivas coordenadas en pixeles y en UMRs.

El objeto fgRojo corresponde al Softbot ubicado en las coordenadas X=0011 y Y=0012. (Los otros valores definen las coordenadas Xini, Yini y Xfin, Yfin medidas en pixeles para su representación gráfica).

El objeto fgVerd corresponde a la meta a la cual el Softbot debe llegar, ella está ubicada en las coordenadas X=0041 y Y=0005. (Los otros valores definen las coordenadas Xini, Yini y Xfin, Yfin que son medidas en pixeles para su representación gráfica).

Objnom	Objxini	Objyini	Objxfin	Objyfin	Xini	Yini	Xfin	Yfin
obj0001	10	10	505	10	1	1	45	1
obj0002	505	10	505	285	46	1	46	25
obj0003	10	285	516	285	1	26	46	26
obj0004	10	21	10	296	1	2	1	26
obj0005	241	65	241	230	22	6	22	20
obj0006	252	142	351	142	23	13	31	13
obj0007	395	98	395	208	36	9	36	18
fgRojo	120	131	395	208	11	12		
fgVerd	450	54	395	208	41	5		
obj0008	395	21	395	65	36	2	36	5
obj0009	131	21	131	98	12	2	12	8
obj0010	131	98	230	98	12	9	20	9
obj0011	230	98	230	87	21	9	21	7
obj0012	318	186	318	296	29	17	29	26

Figura 21: Representación de los objetos en la tabla.

Al contar con la facilidad de cambiar de ubicación algunos objetos móviles y de eliminar elementos en el escenario, en la tabla se mantendrá un registro con la información del objeto descartado con su ubicación inicial.

4.3.3 Características especiales y facilidades del simulador

Se establecen tres (3) aspectos interesantes del simulador a saber:

- Producción de huellas o rastros de la actividad de generación de nodos y desplazamiento del Softbot, para brindarle al usuario una percepción muy clara del comportamiento del agente durante el proceso de búsqueda.

- Simulación y procesamiento de objetos móviles, los cuales pueden variar de posición en el escenario y por tanto, generar una reacción del agente de software.

- Capacidad de trabajar en modo ***cliente-servidor*** con una red de ordenadores (en la búsqueda multidireccional); de tal forma, que cada ordenador se registra como un agente auxiliar que ayuda en la búsqueda de la ruta para ir del origen a la meta.

4.4 SENSORES VIRTUALES DIRECCIONALES

Al tener el diseño del simulador como parte del análisis de la búsqueda de una ruta en un escenario completamente conocido, es decir, que se cuenta con toda la información de los elementos y demás objetos en el escenario, no es menos cierto, que se considera como una primera aproximación a la solución del problema planteado, cuyo propósito es ir del origen a la meta; por cuanto posteriormente se dota al Softbot de sensores que le permiten navegar y buscar la ruta en ambientes atestados desconocidos, en los cuales solo se establece la información del origen y la meta.

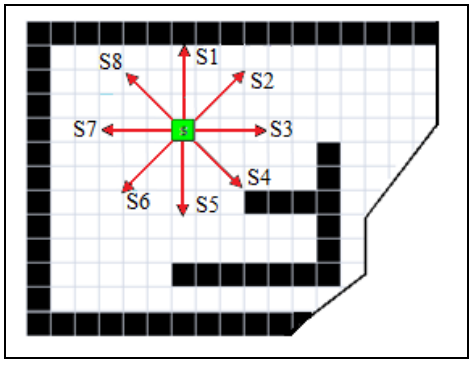


Figura 22: Direccionamiento de los sensores.

La segunda forma de trabajar con el simulador, implica la incorporación al Softbot de ocho (8) sensores direccionales de proximidad (S1, S2, S3,…, S8), ubicados a intervalos de 45º, tal como se muestra en la Figura 22. La ubicación de los sensores permite de cierta manera que el Softbot pueda percibir el ambiente; sin embargo, existirán zonas ciegas comprendidas entre dos sensores adyacentes (celdas o UMRs no detectadas), las cuales solo percibirá a medida que él avance por el escenario.

El Softbot se beneficia de contar con sensores en la medida que le permite percibir el ambiente y posibles cambios que se den de forma dinámica al momento de navegar por el escenario en busca de la meta.

A medida que el Softbot con sus sensores se desplaza, van quedando al descubierto zonas que antes estaban ocultas tras obstáculos, fuera del ángulo de visión, o que simplemente no eran visibles por limitaciones en el alcance de los sensores, de tal forma que integrando la información percibida durante el desplazamiento, se puede realizar un levantamiento del plano del escenario.

Para el caso de los sensores de proximidad reales, las lecturas que un sensor entrega, corresponden a diferencias de potencial en función de la distancia que exista entre él y el obstáculo, produciendo el "rebote" de la señal; lo que al emular por software, permite registrar directamente en una tabla (ejemplo la información contenida en Cp006_03.dbf) las coordenadas (X_j,Y_j) de los puntos donde haya interferencia u obstáculo.

Una facilidad con que cuenta el usuario del simulador es que puede variar el alcance de los sensores de forma arbitraria, con el propósito de verificar la incidencia del alcance en la calidad de la solución o la rapidez con que dicha solución es encontrada.

Por otra parte, la "recolección de datos" es de manera permanente puesto que el agente esta en movimiento y cada que exista el desplazamiento habrá cambio de coordenadas que traerán como resultado un cambio en las distancias.

4.4.1 Modelo físico de los sensores

Existe una amplia gama de dispositivos diseñados para percibir la información externa de una magnitud física y transformarla en un valor eléctrico que sea posible introducir al circuito de control, de modo que el robot físico sea capaz de cuantificarla y reaccionar en consecuencia.

En general, los sensores pueden compararse a los receptores de los órganos sensoriales, que también realizan la conversión de valores físicos, por ejemplo, la luz, el calor o el sonido a una sensación neurofisiológica [4].

Los sensores tienen unas propiedades que se deben tener en cuenta en cualquier desarrollo, algunas de ellas son:

- **La velocidad de operación**: se refiere a la velocidad a la que el sensor genera nuevas medidas. Esto hace que unos sensores sean apropiados para trabajar en tiempo real y en continuo, y otros sólo se usen en momentos muy específicos.

- **El costo**: es una barrera a la hora de fabricar un robot, ya que el precio de los sensores es una parte importante del costo total del robot.

- **Tasa de error**: incluye el número de medidas erróneas que da un sensor, el error medio de medida y el número medio de medidas perdidas.

- **Robustez**: se refiere a la tolerancia que tiene un sensor a cambios en el medio de funcionamiento.

- **Requerimientos computacionales**: aspecto que se convierte en una barrera cuando se fabrica un robot, puesto que los sensores requieren gran capacidad computacional, obligando a unas condiciones mínimas del robot de las que no puede disponer. Este aspecto suele ir unido al costo y la velocidad de operación.

- **Potencia, peso y tamaño**: son aspectos muy importantes a tener en cuenta, ya que influyen en la autonomía y el tamaño del robot.

Para lograr una buena capacidad de adaptación, lo primero que necesitan los robots es conocer el entorno donde se encuentran, siendo clave para realizar cualquier acción en el ambiente. Los robots deben poseer sensores que les permitan saber dónde están, cómo es el lugar donde están posicionados, a qué condiciones físicas se enfrentan, dónde están los objetos con los que deben interactuar y algunos otros sus parámetros físicos y operar de manera similar a los animales y los humanos.

A continuación se explican los sensores más utilizados en un robot móvil, atendiendo a la clasificación anteriormente mencionada.

4.4.2 Sensor basado en ultrasonido

Los sensores de ultrasonido son una *tecnología de medida activa* en donde se emite una señal ultrasónica en forma de pulso, para posteriormente recibir el reflejo de la misma o eco. Se pueden explotar diferentes aspectos de la señal reflejada: el tiempo de vuelo o la atenuación [2].

Los sensores de ultrasonido están formados por una cápsula emisora y otra receptora situada al lado de la emisora o bien por un transductor que actúa de emisor y receptor. En los robots móviles se suelen montar en la periferia, de forma que los sensores se encuentren separados a intervalos uniformes a lo largo del contorno del robot. Esto se hace así porque estos sensores son baratos. Una estrategia alternativa es colocar un sensor montado en una plataforma rotatoria, obteniendo así un barrido de 360º.

La forma estándar de usar un sensor ultrasónico es dar un impulso corto, pero de gran voltaje y a alta frecuencia, a la cápsula emisora para producir una onda ultrasónica. Si la onda ultrasónica viaja directamente contra un obstáculo, rebota, y vuelve directamente hasta el receptor. La distancia que hay entre el sensor y el objeto es la mitad de la distancia que ha recorrido la señal y se calcula, mediante la expresión (4.2).

$$D = \frac{1}{2}ct \tag{4.2}$$

Donde D es la distancia al objeto, c es la velocidad del sonido en el aire y t es el tiempo que tarda la señal desde que se emite hasta que se recibe.

4.4.3 Sensor basado en el espectro infrarrojo

Se incluyen en esta sección los sensores de infrarrojo y láser. A través de estos sensores se pueden estimar las distancias a las que se encuentran los objetos en el entorno. Hay diferentes métodos para medir la distancia a un objeto:

Triangulación: usa relaciones geométricas entre el rayo de salida, el de entrada y la posición del sensor. Como se muestra en la Figura 23, cuanto mayor sea el ángulo a, mayor será la distancia al objeto.

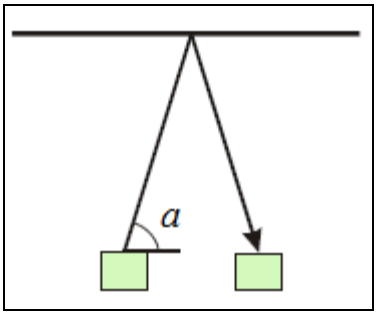


Figura 23: Medición del sensor.

Tiempo de vuelo: Mide el tiempo que transcurre desde que sale el rayo de luz hasta que se recibe, después de haber rebotado en un objeto. La precisión que se obtiene con estos sensores es muy elevada, debido a que son muy direccionales al ser muy pequeña su longitud de onda. La distancia máxima de medida depende de la potencia que se aplica al rayo de salida. Se suele montar un solo láser en una plataforma móvil (pan-tilt), o con un espejo móvil que permita direccionar la señal a diferentes zonas del entorno. Los sensores de infrarrojo se suelen instalar de forma similar a los sensores ultrasonido.

Las ventajas de este tipo de sensores se pueden resumir en:

- El láser puede generar un millón de medidas en un segundo, con una precisión de pocos mm en medidas de 30m. Son sensores ideales para medidas de profundidad, ya que el ángulo de medida es infinitesimal en un láser y muy pequeño en los sensores de infrarrojo.

Por otro lado existen un conjunto de desventajas a tener en cuenta:

- En el caso del láser el precio es muy elevado. El consumo del láser es elevado para llegar a obtener medidas de 30m. Al ser tan direccionales no detectan obstáculos ni por encima, ni por debajo del plano horizontal de medida [2].

- En el caso de los sensores de infrarrojo las medidas de profundidad son muy limitadas (típicamente < 80 cm).

4.4.4 Sensor de aproximación

Los sensores de proximidad suelen tener una salida binaria que indica la presencia de un objeto dentro de un intervalo de distancia especificado. En condiciones

normales, los sensores de proximidad se utilizan en robótica para un trabajo en campo cercano para agarrar o evitar un objeto. Cualquier sensor para medir distancia se puede usar como sensor de proximidad.

El sensor de proximidad es un transductor que detecta objetos o señales que se encuentran cerca del elemento sensor. La Figura 24 muestra un ejemplo de sensor de aproximación.

Tipo SMT/SME8M	Salida de conexión	Función elemento conmutación	Tensión	Longitud cable [m]	Descripción
Detectores proximidad 	Contacto hermético tipo Reed Sin contacto PNP, NPN	Contacto normalmente abierto Contacto normalmente cerrado	24 V CC/CA 250 V CC/CA	2,5 5 7,5 Máx. 30	• Ejecución con cable • Ejecución con conector • Ejecución bifilar • Apto para utilizar con cadenas de arrastre y robots

Figura 24: Sensor de proximidad. [7]

4.4.5 Sensor inductivo

Los sensores basados en un cambio de inductancia debido a la presencia de un objeto metálico están entre los sensores de proximidad industriales de uso más frecuente. El principio de funcionamiento consiste fundamentalmente en una bobina arrollada, situada junto a un imán permanente empaquetado en un receptáculo simple y robusto. El efecto de llevar el sensor a la proximidad de un material ferro-magnético produce un cambio en la posición de las líneas de flujo del imán permanente. En condiciones estáticas no hay ningún movimiento en las líneas de flujo y, por consiguiente, no se induce ninguna corriente en la bobina. Sin embargo, cuando un objeto ferro-magnético penetra en el campo del imán o lo abandona, el cambio resultante en las líneas de flujo induce un impulso de corriente, cuya amplitud y forma son proporcionales a la velocidad de cambio de flujo. [8]

4.4.6 Sensor basado en fotodiodos

Los sensores de luz visible y de infrarrojos cubren un amplio espectro de complejidad. Las fotocélulas se encuentran entre los más sencillos de todos los sensores para hacer su interfaz con el microprocesador, y la interpretación de la salida de una fotocélula es directa. Las cámaras de vídeo, por el contrario, requieren una buena cantidad de circuitería especializada para hacer que sus salidas sean compatibles con un microprocesador.

Los sensores de luz posibilitan comportamientos de un robot tales como esconderse en la oscuridad, jugar con un flash y moverse hacia una señal luminosa. Los sensores de luz simples son fotorresistencias, fotodiodos o fototransistores. Las fotorresistencias son simplemente resistencias variables con la luz en muchos aspectos parecidos a los potenciómetros, excepto en que estos últimos varían girando un botón. El fototransistor es un transistor con la corriente de base generada por la iluminación de la unión base-colector (ver Figura 25). La operación normal del transistor amplifica la pequeña corriente de base ($\mathbf{I_B}$).

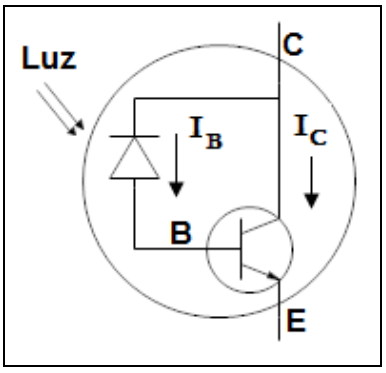


Figura 25: Fototransistor.

Los fotodiodos tienen una gran sensibilidad, producen una salida lineal en un amplio rango de niveles de luz, y responden con rapidez a los cambios de iluminación. Esto les hace útiles en los sistemas de comunicación para detectar luces moduladas; el mando a distancia de casi todos los TV, equipos estéreos y reproductores de CD.

4.4.7 Emulación virtual de los sensores en el SIRUM

La dotación de sensores, como un recurso para la percepción del entorno por donde navega el Softbot, se asemeja a los “ojos” que dispone un robot físico. Sin embargo, en la emulación no se tendrán en cuenta factores externos (ruido) que afecten su funcionamiento.

4.4.7.1 *Generalidades de los sensores virtuales*

Los sensores virtuales tratarán el escenario en forma simplificada, es decir, tendrán un alcance por defecto, por ejemplo 10 UMR, y estarán posicionados alrededor del Softbot en las ocho posiciones de movimiento (a intervalos de 45°).

Los sensores virtuales producen una secuencia de datos como la mostrada en la Figura 26. Cada renglón corresponde a una posición distinta del Softbot, descrita por los campos **Posx** y **Posy** así como las lecturas de todos y cada uno de los demás sensores.

SIRUM: Búsqueda de rutas en robótica móvil...

Posx	Posy	S01x	S01y	S02x	S02y	S03x	S03y	S04x	S04y	S05x	S05y	S06x	S06y	S07x	S07y	S08x	S08y
11	12	11	1	14	9	22	12	23	24	11	24	1	22	1	12	1	2
18	12	18	9	21	9	22	12	22	16	18	24	6	24	6	12	15	9
20	12	20	9	22	10	22	12	22	14	20	24	8	24	8	12	17	9
21	10	21	9	22	9	22	10	22	11	21	22	9	22	9	10	20	9
21	11	21	9	22	10	22	11	22	12	21	23	9	23	9	11	19	9

Figura 26: Contenido de la tabla con información de sensores.

El ejemplo corresponde a algunas mediciones de los sensores en un ambiente cualquiera, donde el Softbot parte de unas coordenadas iniciales, el primer renglón establece que son Posx=0011 y Posy=0012. Así mismo, el sensor número uno (**S01**) presenta un hallazgo en S01x=11 y S01y=1, lo que se puede corroborar a simple vista para el escenario del la Figura 27.

El mismo análisis se extrae para los sensores en las direcciones 2 a 8, donde se reportan los hallazgos que en su indagación cada uno encontró; es importante tener en cuenta que el alcance del sensor limita su espacio y por tanto de forma temporal se considera que a partir de allí, lo que hay después es un obstáculo, como es el caso del sensor **S05** que reporta hallazgo en Posx=0011 y Posy=0024. El agente inteligente en su funcionalidad, será quien interprete el contenido de la tabla, puesto que el reporte del sensor es simplemente del hallazgo de “algo” en ciertas coordenadas del escenario.

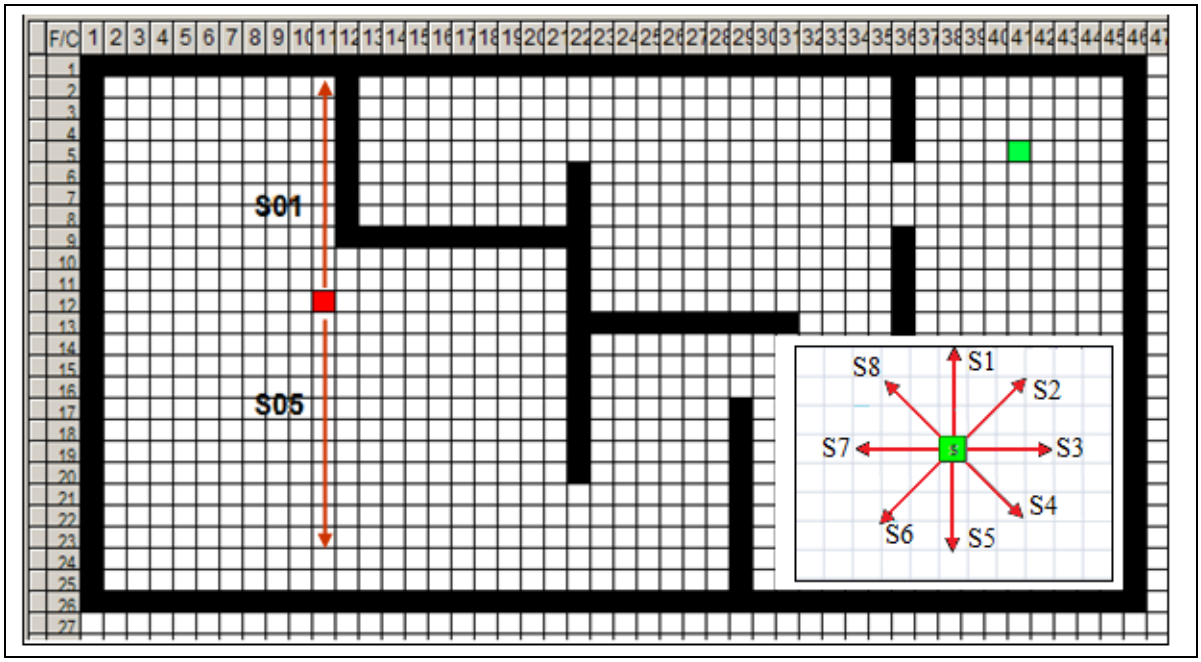


Figura 27: Disposición de los sensores virtuales.

El sensor virtual construido computacionalmente es una emulación del sensor físico de proximidad, para lo cual se usa una estructura FOR_NEXT, que permite dentro del algoritmo, realizar un escaneo de las celdas en la dirección de ubicación del sensor.

En la Tabla 4 se muestra, a manera de ejemplo, parte del algoritmo para el caso del sensor virtual "**S03**". El sensor virtual opera haciendo un barrido direccional de celdas. La explicación del programa emulador del sensor se muestra por cada línea o instrucción, de manera idéntica se tendría el algoritmo para los restantes sensores direccionales. Una vez se perciba un obstáculo, el sensor reporta las coordenadas X y Y correspondientes a lo encontrado.

CÓDIGO FUENTE	EXPLICACIÓN DE LA LÍNEA / COMANDO (Comentario)
K_inicial = 12	// Define la posición inicial de escaneo del sensor
NGrande = Alcance_sensor	// Característica del sensor emulado
For K = K_inicial To NGrande	// Busca en todas las celdas
Do Case	// Define la columna
Case K <= 9	
xCol = "X000" + Str(K,1)	// Cuando K = 3, xCol = "X0003"
Case K <= 99	
xCol = "X00" + Str(K,2)	
Case K <= 999	
xCol = "X0" + Str(K,3)	
OtherCase	
xCol = "X" + Str(K,4)	
EndCase	
xLectura = &xCol	// Se refiere al contenido de la columna "X0003", registro 4
Do Case	// Comienza el análisis de posibles hallazgos
Case xLectura = "NG"	// Si el contenido de la celda en la tabla Cp006_01.dbf es "NG", // se sale del ciclo de escaneo porque se trata de un muro
Exit	
Case xLectura = ""	// Si la celda está vacía, continua con el escaneo
*continuar…	
Case xLectura = "S1##"	// Señal que indica que en esa celda (posición), // se encuentra la meta
Exit	
***	
***	
OtherCase	// Para casos no previstos
Situación no prevista…	
EndCase	
***	
***	
Next K	// En este caso, ir a la siguiente celda

Tabla 4: Algoritmo para emular un sensor.

Al final de recorrido el algoritmo, se podría tener como resultado un mensaje reportado por el sensor así:

"Muro en la posición (X0013, Y0005)", la distancia estará dada por D = k – kin, en UMR. Un sensor virtual se comportará según se programe, lo que supone una gran

ventaja al momento de trasladar la caracterización de un sensor real a su correspondiente simulado.

4.4.7.2 *Limitaciones del sensor*

Básicamente los sensores virtuales están limitados por el alcance, y es el usuario quien lo establece y decide si lo aumenta o lo reduce; sin embargo, tendrá una distancia establecida por defecto (default) en UMR, de 10.

Dado que los sensores son virtuales, se podrán emular desde el punto de vista del alcance, con las siguientes dos características:

- Limitados en su alcance: sólo llegan hasta un objeto / obstáculo que le impiden "ver" más allá.

- Penetrantes: sólo están limitados por el alcance que se les programe, pidiendo indagar más allá del obstáculo.

Por considerar el simulador que los objetos son sólidos, no se aplica la función de sensores penetrantes, como se podría tener en el mundo real para situaciones donde se presentan vidrios en el ambiente y el uso de sensores tipo láser.

Como se ha mencionado, el escenario se trata de forma simplificada en dos dimensiones (2D), por tanto, no existirán condiciones que afecten los sensores como sucede en el mundo real, tales como el ruido, la temperatura y otros factores que determinan la precisión y confiabilidad del sensor.

Se aclara que en sí misma la idealización del ambiente no es de ninguna manera una desventaja del simulador, sino que permite gradualmente ir adicionando características como sería la introducción de un generador de ruido simulado que afecte el ambiente para planear y diseñar modelos de manejo del ruido.

4.5 ESTRATEGIA DE BUSQUEDA

No obstante que la robótica móvil se desarrolla en entornos o ambientes desconocidos, de tal forma que determinar caminos seguros hacia el objetivo o meta se convierte en un aspecto fundamental, la presente tesis ha sido desarrollada siguiendo un modelo de desarrollo en espiral, que posibilita partir de la idealización del escenario, así como del conocimiento del mismo.

Esta idealización permite suponer inicialmente que todo el escenario, los objetos en él incluidos y la información o datos asociados se encuentran definidos y que dicha

información está disponible justo cuando se necesite. De esta manera los esfuerzos se concentran en la formulación, diseño, construcción y prueba de una estrategia de búsqueda para la solución en un ambiente ideal; posteriormente, subir por la espiral (diseño metodológico) mediante la inclusión de sensores que permitan operar en ambientes desconocidos y por ultimo comprobar la estrategia multidireccional. El diagrama de bloques de la Figura 28, muestra el camino que la estrategia toma a partir del manejo del espacio de estados que se generan en la solución del problema.

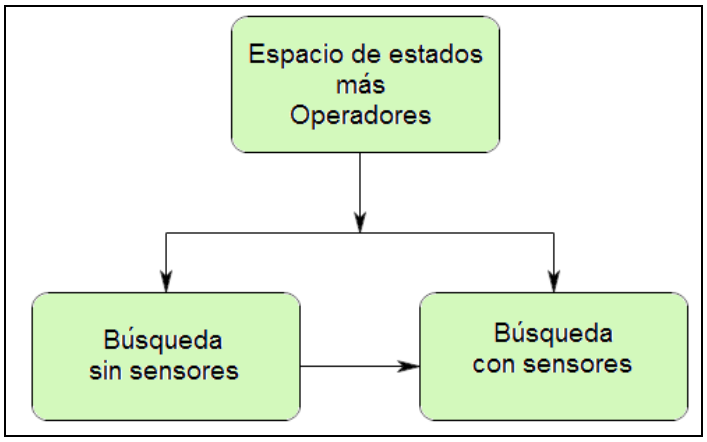


Figura 28: Diagrama de la estrategia.

Mientras que la búsqueda de una ruta en ambientes conocidos se asemeja un poco a un problema de planificación; la búsqueda de dicha ruta en un ambiente desconocido, donde solo se cuenta con la información de los sensores, se transforma en una solución que aunque esté gobernada por algún método de planificación, se vuelve completamente reactiva.

4.5.1 Búsqueda sin sensores

La búsqueda sin sensores, plantea como información inicial el conocimiento total del ambiente, es decir, se conocen todas las tablas dinámicas, que se fueron construyendo en la creación del entorno. Por tanto, la búsqueda de la ruta que lleva del origen a la meta se podrá conocer y posteriormente se mostrará la navegación por el escenario, en caso de no haber ruta, se indicará mediante un mensaje que no es posible hallar el camino.

El análisis y el procesamiento de la búsqueda sin sensores se realizan en un tiempo corto, puesto que el conocimiento de todo el escenario, ayuda a encontrar una posible solución de forma segura.

4.5.1.1 *Operadores de navegación*

Partiendo del hecho de que se conoce tanto la posición inicial del Softbot, como de la meta y los demás objetos dentro del escenario, se realiza el siguiente proceso:

Se indaga en la tabla MxN el contenido de las ocho celdas adyacentes a la que ocupa el Softbot; determinando cuáles de ellas están vacías, de forma que se admite el desplazamiento del Softbot a dichos lugares.

Para indagar las celdas adyacentes, se deben aplicar los operadores de movimiento de (1 a 8), tal como se muestra en la Figura 29.

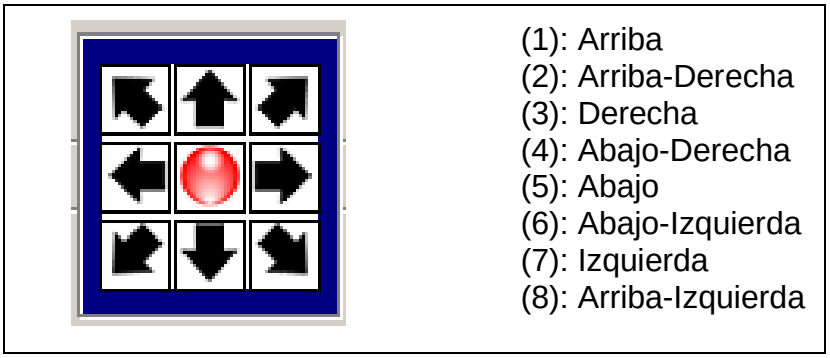


Figura 29: Operadores para la navegación.

En la Figura 30, se muestra una parte de la visualización de la tabla MxN que corresponde igualmente a un segmento del escenario; se observa el Softbot en color rojo, la meta en color verde, algunas paredes de color negro y las ocho direcciones posibles.

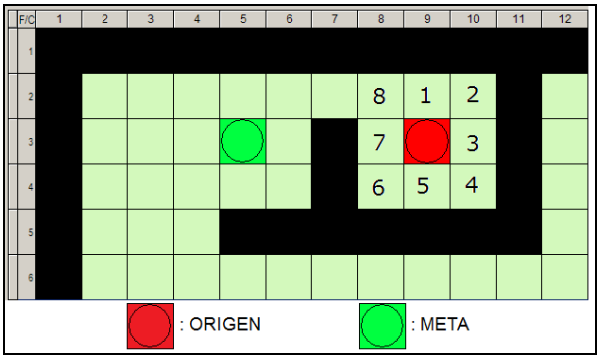


Figura 30: Movimientos posibles del Softbot.

Para saber si las celdas adyacentes al Softbot (UMR de color rojo) están ocupadas o no, debe ser posible escanearlas mediante un algoritmo que indague en cada una de las ocho posiciones comparando su contenido con la cadena “NG” (codificación que identifica una UMR perteneciente a una pared); en caso de hallar dicho código, se dice que un movimiento en dicha dirección no es válido.

4.5.1.2 *La heurística*

Utilizando una función heurística como la definida en el numeral 2.3.3, es posible construir una función de evaluación de estados que incluya el costo para alcanzar un estado determinado, así como una estimación sobre el costo para encontrar el objetivo final (meta). Se definen los siguientes costos:

$f(n)$ es el costo heurístico asociado a un estado n.

$g(n)$ es el costo real para alcanzar un estado n a partir del estado actual[11].

$h(n)$ es el costo heurístico para alcanzar un objetivo desde el estado n.

La relación que existe entre los costos está dada por la expresión (4.3).

$$f(n) = g(n) + h(n) \tag{4.3}$$

Como una primera aproximación al estudio del SIRUM, se puede usar la heurística $f(n) = g(n)$, donde no se tiene en cuenta inicialmente el costo real de ir desde el nodo origen hasta el nodo n. Esta primera aproximación es muy importante cuando se utilice la búsqueda con sensores, donde una vez se haya avanzado, el costo hasta el punto actual no es posible de usar para minimizar el costo total de ruta; contrario a la búsqueda sin sensores donde el Softbot realmente no se mueve hasta que la ruta total haya sido calculada.

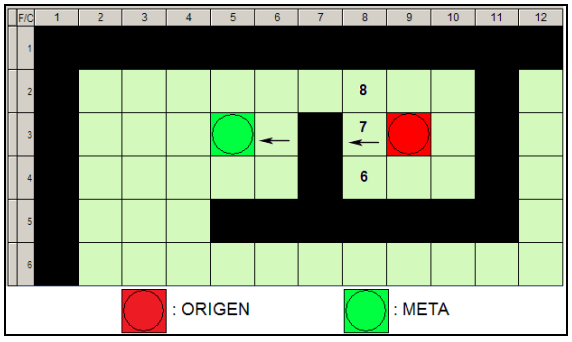


Figura 31: Parte inicial del escenario.

[11] En general, el costo entre dos estados vecinos puede ser fijado arbitrariamente.

En la Figura 31, se muestra parte de un escenario donde se ha definido la posición inicial del Softbot con coordenadas X=0009, Y=0003 y la meta con coordenadas X=0005, Y=0003 (ubicación a donde el Softbot debe llegar), se observa que desde la ubicación actual (estado o nodo inicial) no es posible alcanzar la meta en línea recta, puesto que existen obstáculos (paredes) con coordenadas como la X=0007, Y=0003, que impiden el paso del Softbot.

La búsqueda en tiempo real utiliza la estrategia del menor costo para los movimientos. Es decir, la información obtenida sólo se utilizará para determinar el próximo movimiento. La razón es que luego de ejecutar la acción, se supone que la frontera de búsqueda se expandirá, lo cual puede llevar a una elección para el segundo movimiento diferente a que la que arrojó la primera búsqueda.

La heurística planteada será la que determinará el movimiento válido adecuado, que en el caso de ejemplo en la Figura 31, se deberá elegir entre los movimientos (6), (7) y (8), puesto que la función heurística básicamente toma la distancia más corta.

Como el escenario fue construido a partir de filas y columnas, la forma de describirlo es de manera matricial, es decir, cada punto en el escenario tendrá coordenadas **(X_j,Y_j)**; por tanto, la distancia más corta entre la posición del Softbot y la meta podrá determinarse aplicando el ***Teorema de Pitágoras***, el cual establece que en un triángulo rectángulo como el mostrado en la Figura 32, el cuadrado de la longitud de la hipotenusa (el lado de mayor longitud del triángulo rectángulo) es igual, a la suma de los cuadrados de las longitudes de los dos catetos (los dos lados menores del triángulo rectángulo: los que conforman el ángulo recto). Si un triángulo rectángulo tiene catetos de longitudes a y b, y la medida de la hipotenusa es *c*, se establece en la expresión (4.4).

$$c = (a^2 + b^2)^{1/2} \tag{4.4}$$

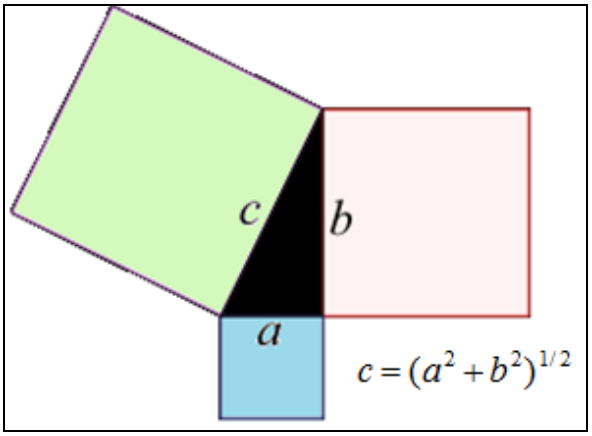


Figura 32: Teorema de Pitágoras.

Por tanto en el escenario se establece la distancia más corta D, mediante la expresión (4.5).

$$D = \sqrt{(X_j - X_k)^2 + (Y_j - Y_K)^2} \quad (4.5)$$

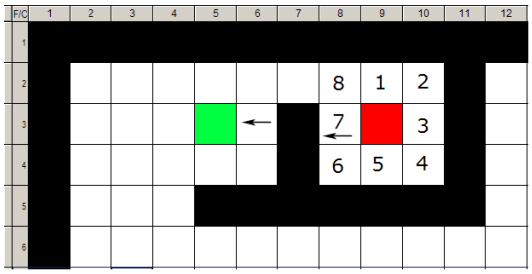
El algoritmo que desarrolla la búsqueda utiliza como heurística el resultado de evaluar los movimientos posibles, que no están restringidos; y calcular la distancia hasta la meta desde cada uno de las posiciones adyacentes a la posición del Softbot. Una vez calculada cada una de las distancias, se elige la más corta y allí será donde el Softbot ejecutará el movimiento.

La anterior secuencia de decisiones, se puede ver en la Figura 33, donde se observa cómo avanza el Softbot por el escenario, así también la funcionalidad de cada uno de los operadores usados, los cuales dependen de la heurística y el valor calculado de D.

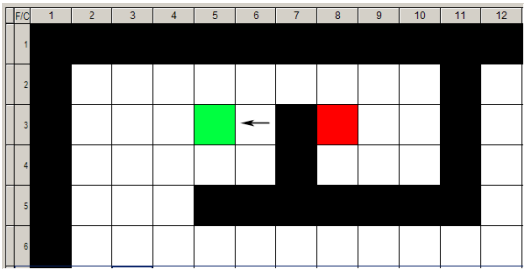
Para realizar una secuencia de decisiones no es suficiente con la información devuelta por el algoritmo, la estrategia básica de repetir el algoritmo para cada decisión resulta inadecuada al ignorar la información relacionada con los estados anteriores. El problema radica en que, al volver a un estado anteriormente visitado, se entrará en un ciclo infinito, esta situación podrá suceder con frecuencia, debido a que las decisiones se basan en información limitada y por lo tanto, direcciones que al principio parecían favorables, pueden resultar equivocadas al reunir más información durante la exploración.

En general, se desea evitar entrar en ciclos infinitos y a la vez permitir volver a estados ya visitados cuando parezca favorable. Los obstáculos del entorno definen una serie de restricciones sobre el movimiento del Softbot, por tanto, se restringe la toma de decisiones en aquellas direcciones que pueden provocar la colisión del Softbot con un obstáculo.

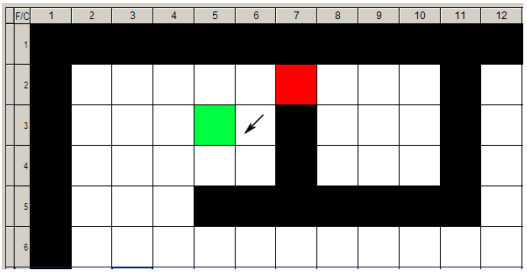
Teniendo en cuenta que las indagaciones, se realizan en las ocho direcciones posibles de movimiento, puede suceder el caso de que se encuentren valores idénticos en dos nodos; sin embargo, tal como se menciona en la heurística, se podrán realizar ajustes que tiene como finalidad converger en una solución. En caso de encontrarse dos valores, se puede observar una oscilación de posiciones, situación que no conviene por el tiempo de procesamiento que conlleva.



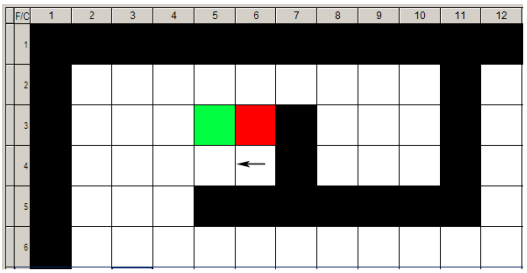
Estado inicial:
Softbot en X=0009, Y=0003
Meta en X=0005, Y=0003
La heurística propende por un movimiento a la izquierda. Valor de $f(n)=4$



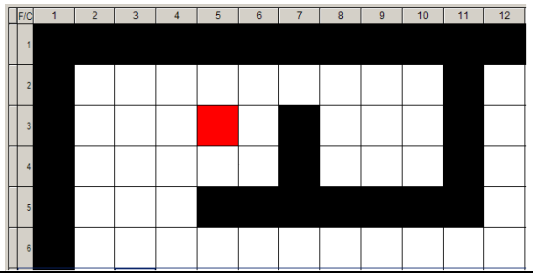
Operador: 7 (Movimiento Izquierda)
Softbot en X=0008, Y=0003
La pared impide el desplazamiento a la izquierda. (De los movimientos válidos la dirección Arriba-izquierda presenta el menor costo). Valor de $f(n)=3$



Operador: 8 (Movimiento Arriba-Izquierda)
Softbot en X=0007, Y=0002
La heurística produce dos costos iguales. (El agente toma uno de ellos).
Valor de $f(n)=2316$



Operador:6 (Movimiento Abajo-Izquierda)
Softbot en X=0006, Y=0003
Existe línea de vista a la meta.
Valor de $f(n)=1$



Operador: 7 (Movimiento Izquierda)
Softbot en X=0005, Y=0003 (llegada física a la meta). Valor de $f(n)=0$

Figura 33: Secuencia de movimientos.

Teniendo en cuenta que las indagaciones, se realizan en las ocho direcciones posibles de movimiento, puede suceder el caso de que se encuentren valores idénticos en dos nodos; sin embargo, tal como se menciona en la heurística, se podrán realizar ajustes que ayuden a converger en una solución.

Igualmente se puede observar que el camino hacia el objetivo puede estar bloqueado en determinadas situaciones (paso por aristas), aunque la condición de paso libre proporcione un resultado favorable. Para asegurar un camino libre es necesario definir un ancho, que tenga en cuenta el espacio necesario para que el Softbot evite los obstáculos hacia la meta, que para el caso deberá ser una UMR.

4.5.1.3 *Expansión de nodos (nuevos estados)*

Cuando el Softbot está ubicado en cualquier punto válido sobre el escenario, es posible aplicarle -*dependiendo de los obstáculos que se encuentren en las celdas inmediatamente adyacentes*-, uno de ocho operadores de navegación descritos en la subsección 4.5.1.1.

La Tabla 5, corresponde a la información de la "pestaña *nodos"*, ubicada en el área de trabajo del simulador, la cual se genera para el caso del escenario mostrado anteriormente en la Figura 31. La información allí contenida, corresponde al árbol de estados con los nodos expandidos por donde se realiza la búsqueda de la ruta.

op	nodo	xfin	yfin	meta	cdo	desde	costo	expandió	ruta
O	0000000000	9	3		S		4,0000	1, 2, 3, 4, 5, 6, 7, 8	S
1	0000000001	9	2	N		0000000000	4,1231		
2	0000000002	10	2	N		0000000000	5,0990		
3	0000000003	10	3	N		0000000000	5,0000		
4	0000000004	10	4	N		0000000000	5,0990		
5	0000000005	9	4	N		0000000000	4,1231		
6	0000000006	8	4	N		0000000000	3,1623		
7	0000000007	8	3	N	S	0000000000	3,0000	9	S
8	0000000008	8	2	N		0000000000	3,1623		
8	0000000009	7	2	N	S	0000000007	2,2361	10, 11	S
6	0000000010	6	3	N		0000000009	1,0000		S
7	0000000011	6	2	N		0000000009	1,4142		
5	0000000012	6	4	N		0000000010	1,4142		
6	0000000013	5	4	N		0000000010	1,0000		
7	0000000014	5	3	S		0000000010	0,0000		S

Tabla 5: Expansión de nodos.

En la tabla 6, se presenta una explicación del contenido de cada una de las columnas generadas en la "pestaña *nodos*".

Columna	Descripción
op	Indica el operador utilizado para realizar la expansión del nodo enumerado en la columna "**desde**"; a excepción del primer registro que es el correspondiente al origen ("O"). Cada número del uno (1) al ocho (8), indica la dirección que debe tomar el Softbot para producir el nuevo **nodo**.
nodo	En el primer registro, siempre se ubica el nodo inicial. En los demás registros aparecerá el número consecutivo de nodo expandido virtualmente.
xfin	Abscisa "X" de la posición del Softbot correspondiente al nodo.
yfin	Ordenada "Y" de la posición del Softbot correspondiente al nodo.
meta	"S" cuando el nodo expandido es la meta; "N" en los demás casos.
cdo	Contiene una letra "S" cuando el nodo del registro haya sido completamente expandido (cerrado); es decir, que a partir del **nodo** se hayan generado todos los nodos posibles. Esta información es útil para no intentar volver a expandir nodos que ya lo fueron.
desde	Indica el número del nodo desde el cual se genero el nuevo **nodo**.
costo	Es un número que mide la distancia al nodo **meta**.
expandió	Contiene la lista de nodos adicionados al espacio (árbol) de búsqueda a partir de cada uno de los nodos de la columna **nodos**.
ruta	Contiene una letra "S" como identificador de los nodos necesarios para ir del **origen** a la **meta**.

Tabla 6: Explicación de los campos de la expansión de nodos.

4.5.1.4 *Tabla solución de la ruta*

En la expansión de nodos se describió cómo a medida que se genera el árbol de búsqueda, se indaga mediante una heurística aquellos nodos que son más prometedores de conducir a la meta, mediante una ruta de bajo costo. A medida que los nodos expandidos se van registrando en la tabla (*Traza.dbf)*, el agente inteligente mediante el cálculo de la heurística, decide cuales nodos expandir. No todos los nodos explorados en un paso, son necesariamente expandidos en los siguientes.

La Figura 34, muestra la información de la tabla ***Secuencias.dbf***, la cual contiene un resumen de la tabla *Traza.dbf*, para aquellos registros marcados con "S" en la columna ruta.

La tabla ***Secuencias.dbf*** es muy importante porque:

- Aplicando desde el origen o posición inicial del Softbot, los operadores correspondientes, el Softbot recorre el camino para llegar hasta la meta.

- En sí misma se puede considerar como "*conocimiento de máquina*", para nuevas simulaciones en ambientes donde la ruta o parte de ella sea solución a un nuevo problema de búsqueda. El conocimiento previo será objeto de futuros estudios.

OP	RUTA	X	Y	Costo
O	0000000000	9	3	4.0000
7	0000000007	8	3	3.0000
8	0000000009	7	2	2.2361
6	0000000010	6	3	1.0000
7	0000000014	5	3	0.0000

Figura 34: Información de la tabla de secuencias.dbf.

En la Tabla 7, se presenta una explicación del contenido de cada una de las columnas en la tabla de secuencias.

Columna	Descripción
OP	Indica el operador utilizado para realizar la expansión del nodo enumerado en la columna "**desde**"; a excepción del primer registro que es el correspondiente al origen ("O"). Cada número del uno (1) al ocho (8), indica la dirección que debe tomar el Softbot para producir el nuevo "**nodo**".
RUTA	Estado solución, indica el nodo que fue expandido y que contribuyo a encontrar el camino del origen a la meta.
X	Abscisa "X" de la posición del Softbot correspondiente al nodo.
Y	Ordenada "Y" de la posición del Softbot correspondiente al nodo.
Costo	Valor correspondiente a la heurística $f(n)$.

Tabla 7: Explicación de los campos de secuencias.

4.5.1.5 *Trazado de huellas*

Con el fin de presentar de una forma fácil como el agente indaga su entorno, se implementa la muestra en el Display de esta tarea. La indagación realizada permite que junto con la estrategia de búsqueda se opte por un camino, es decir, se decida previa evaluación cual es la dirección que paso a paso tomará el agente para llegar a su objetivo (la meta).

En la Figura 35, se presenta dentro del escenario la indagación del entorno del agente y las huellas representadas por círculos blancos, que se dejan una vez indagada las celdas en las direcciones de los operadores de navegación.

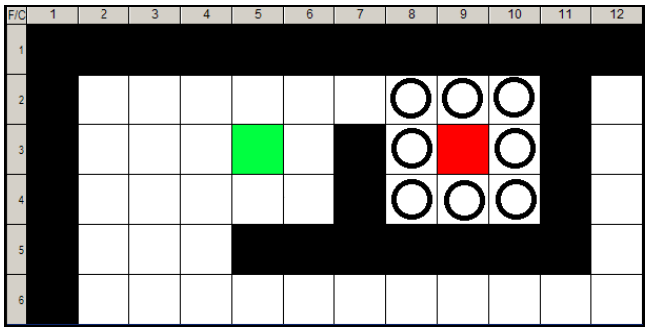


Figura 35: Muestra de las huellas.

La implementación de la técnica de huellas presenta tres (3) alternativas:

- Mostrar todas las huellas: presenta en el Display el rastro de la indagación de las celdas adjuntas y adyacentes a la posición del Softbot.

- Mostrar la huella mejor: se restringe a presentar en el Display el rastro de solo la mejor alternativa de la indagación de celdas adjuntas y adyacentes.

- Inhabilitar la muestra de huellas (por defecto): no se muestra ningún tipo de huellas en el Display.

4.5.2 Búsqueda con sensores

Al no contar con la información completa del escenario donde realizar la búsqueda, se tendrá que hacer el levantamiento del área con puntos específicos reportados por los sensores direccionales del Softbot. Así, asumiendo que únicamente lo que reportan los sensores es todo lo que existe en el escenario, se procede a encontrar una ruta, desde la posición actual del Softbot (Xini, Yini) hasta la posición de la meta (Xfin, Yfin) y a medida que el Softbot avanza, el camino obtenido respeta los obstáculos percibidos y obviamente, no considera los obstáculos que él no ha podido percibir.

En la Figura 36, se muestra el diagrama de flujo a nivel de la estrategia para la búsqueda con sensores; donde una vez realizada una ***búsqueda virtual*** *-en la que el escenario está conformado únicamente con base en las percepciones de los sensores-*, el Softbot avanza un solo paso (una UMR) sobre un camino previamente determinado; entonces se procede a realizar nuevamente las lecturas con los

sensores para obtener un nuevo plano del escenario que provea más puntos de información y repetir la búsqueda e iterar la estrategia. De igual manera, se puede activar la muestra de huellas en el Display para observar cómo se realiza la indagación estados en la búsqueda.

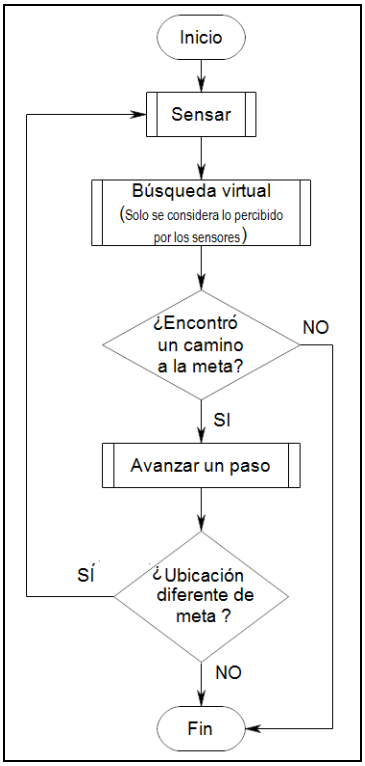


Figura 36: Algoritmo estrategia con sensores.

En la Figura 37, se ilustra la secuencia del proceso de la búsqueda con sensores, allí se observa como lo que es reportado por los ocho sensores direccionales se pinta de color azul. En la tabla MxN las celdas cambian su codificación, es decir, se remplaza el contenido de **NG** por **AZ**, permitiendo realizar un levantamiento del escenario con puntos específicos a medida que se avance por él.

El alcance de los ocho sensores direccionales por defecto es de 10 UMRs y es el usuario del aplicativo quien pude variar este parámetro, el tener un alcance mayor o menor puede resultar más o menos ventajoso, puesto que dependiendo de la forma del escenario es posible tener más rápidamente el levantamiento del mismo.

Como se mencionó anteriormente, existirán zonas ciegas, las cuales no han podido ser percibidas por los sensores; sin embargo, como la estrategia está basada en que lo reportado es lo que existe, dichas zonas ciegas, al igual que el resto del escenario, serán ignoradas al momento de tomar una decisión de desplazamiento.

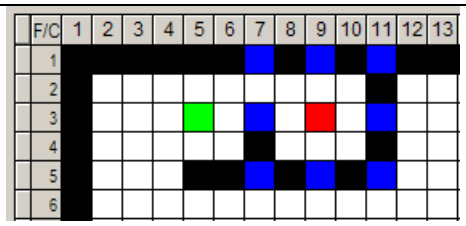
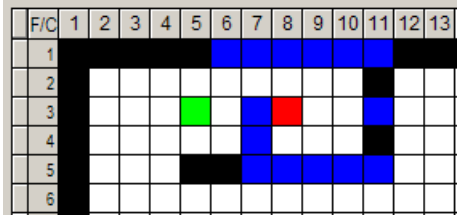
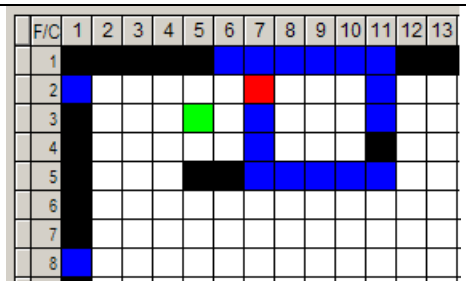
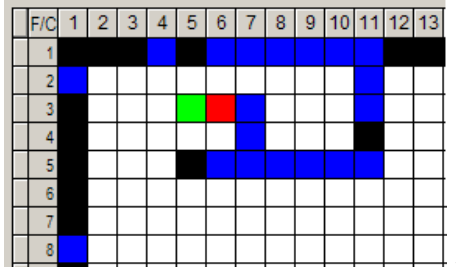
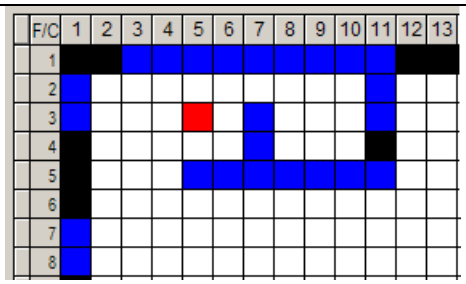
 37a	Desde el punto de origen, Posx=9 y Posy=3 (en color rojo), el Softbot comienza por activar los ocho (8) sensores direccionales, de tal forma que lee obstáculos según la Figura 37a. (Lo marcado con color azul es lo que percibe cada sensor como obstáculo).
 37b	Mediante la búsqueda virtual sin sensores, se encuentra un camino que sólo incluye lo percibido en el paso anterior. El Softbot se desplaza una UMR ubicándose en Posx=8 y Posy=3, para posteriormente escanear el escenario detectando nuevamente obstáculos (segundo registro de la Figura 37b) marcando con azul las nuevas UMR.
 37c	Se puede observar en la Figura 37c, como desde la nueva posición del Softbot (Posx=7, Posy=2), los sensores reportan los nuevas coordenadas con obstáculos en las posiciones (1, 2), (11, 2), (1, 8).
 37d	Se puede observar en la Figura 37d, como desde la nueva posición (Posx=6, Posy=3), los sensores reportan los nuevos puntos (4, 1), (6, 5) respectivamente. Lo que ya fue detectado permanece con color azul.
 37e	En la Figura 37e se observa el Softbot ubicado sobre la meta. Así mismo registra en color azul los nuevos puntos visibles desde la ubicación final Posx=5 y Posy=3, su punto de llegada.

Figura 37: Estrategia con sensores.

La Figura 38, muestra las lecturas de los ocho sensores para las posiciones recorridas por el Softbot (9,3), (8,3), (7,2), (6,3) y (5,3).

Cp002_03

Posx	Posy	S01x	S01y	S02x	S02y	S03x	S03y	S04x	S04y	S05x	S05y	S06x	S06y	S07x	S07y	S08x	S08y
9	3	9	1	11	1	11	3	11	5	9	5	7	5	7	3	7	1
8	3	8	1	10	1	11	3	10	5	8	5	7	4	7	3	6	1
7	2	7	1	8	1	11	2	10	5	7	3	1	8	1	2	6	1
6	3	6	1	8	1	7	3	7	4	6	5	1	8	1	3	4	1
5	3	5	1	7	1	7	3	7	5	5	5	1	7	1	3	3	1

Figura 38: Información de sensores.

Nótese que la secuencia de coordenadas de las columnas Posx y Posy, es precisamente, la ruta seguida por el Softbot para ir del origen a la meta. Por tanto, con la implementación de la estrategia de búsqueda sin sensores, fue posible su uso y adecuación en la búsqueda con sensores. El funcionamiento de ambas estrategias se resume en la Tabla 8.

Tipo de búsqueda	**Tabla de nodos** (Ver subsección 4.5.1.3)	**Secuencias** (Ver subsección 4.5.1.4)
Búsqueda sin sensores	Se construye una sola vez.	Resume los puntos a unir como ruta.
Búsqueda con sensores	Se borra su contenido si existe, para generar una nueva cada vez que se invoca la búsqueda virtual.	Se borra la existente para construir una ruta (camino) por cada vez que se invoca la búsqueda virtual.

Tabla 8: Resumen de las búsquedas.

La búsqueda con sensores ante la presencia de elementos móviles en el escenario, se realizará de acuerdo con el siguiente planteamiento:

Dado que el proceso de escaneo mediante sensores reporta lo que se encuentra en un instante de tiempo en el que se realiza el barrido, un elemento que entra dentro del rango de alcance de los sensores en el momento justo en que se escanea, será percibido de la misma forma que se perciben los elementos estáticos.

4.5.3 El agente inteligente como articulador

Una vez presentado la totalidad de componentes del proyecto SIRUM, se puede percibir el agente inteligente como un articulador de todos ellos.

Un agente inteligente se puede caracterizar con base en sus Percepciones, Acciones, Metas y Entorno (PAME), tal como se muestra en la Tabla 9.

Tipo de agente	Percepciones	Acciones	Metas	Entorno
Simulador para determinación de rutas en robótica móvil (SIRUM).	Lecturas del entorno mediante sensores virtuales.	Trazar una ruta del origen al destino y recorrerla. Mostrar el móvil y su desplazamiento.	Llegar al punto de destino en condiciones óptimas.	Escenario virtual.

Tabla 9: Agente y descripción PAME.

Es fácil comprender la función y alcance del agente inteligente como articulador si consideramos que a un robot físico (antes Softbot), se dota con motores paso a paso que harían la función de desplazamiento, sensores físicos -por ejemplo basados en ultrasonido- en lugar de virtuales y unas subrutinas con los algoritmos de búsqueda; todo esto requiere ser coordinado y gobernado por un software de alto nivel que es a quien se denomina ***agente inteligente***.

La Tabla 10, realiza una comparación entre un robot físico o de hardware y un robot de software:

Componente	Softbot	Robot físico
Escenario	Representación matricial (tablas DBF) Representación gráfica (en UMRs).	Planta física con paredes, muebles, zonas despejadas, etc.
Dispositivos de censado	Sensores emulados (virtuales). Lecturas del escenario representado en tablas.	Sensores de ultrasonido, proximidad, etc. La lectura es de medidas físicas (ambiente).
Búsqueda de ruta	Software de I.A.	Software de I.A.
Movimiento	Representación gráfica y matricial (en tablas) del desplazamiento del Softbot. Avance en pixeles o UMRs.	Servomotores, ruedas, GPS para detección y corrección de errores de desplazamiento.
Dispositivos de procesamiento y control	Computador o red de computadores.	Procesador, memoria, micro-controlador, bus de datos, tarjetas de adquisición.

Tabla 10: Comparación Softbot vs Robot físico.

4.5.4 Desarrollo de la búsqueda multidireccional

Para realizar la ***extrapolación conceptual*** de la búsqueda bidireccional hacia la multidireccional, es necesario retomar el concepto de ***Agente Inteligente*** establecido en la subsección 2.1.1, y definir los que se utilizaran en el simulador según la Tabla 11.

Tipo de agente	Características
Agente_origen	Está ubicado el origen, con coordenadas previamente definidas. En el escenario se define con una UMR de color rojo.
Agente_objetivo	Está ubicado en la meta, con coordenadas previamente definidas. En el escenario se define con una UMR de color verde.
Agente_auxiliar	Se ubica un agente_auxiliar, sobre cada uno de los puntos previamente definidos en el entorno de color azul claro. Tienen acceso a la representación del entono.

Tabla 11: Tipos de agentes inteligentes.

4.5.4.1 *Implementación con uno o más procesadores*

Al hacer uso de agentes auxiliares, es posible tener una red de ordenadores, donde cada procesador hace las veces de un agente auxiliar. Para esta metodología un procesador debe registrarse en el proyecto del aplicativo y así interactuar en el escenario.

En la Figura 39, se muestra la funcionalidad que el sistema operativo Windows ofrece, en el cual se puede tener acceso a configurar los procesos que desarrolla con prioridad el procesador de un computador, de tal manera, que se de mayor prioridad a los procesos de la búsqueda multidireccional, permitiendo un manejo mas óptimo de los ciclos de máquina.

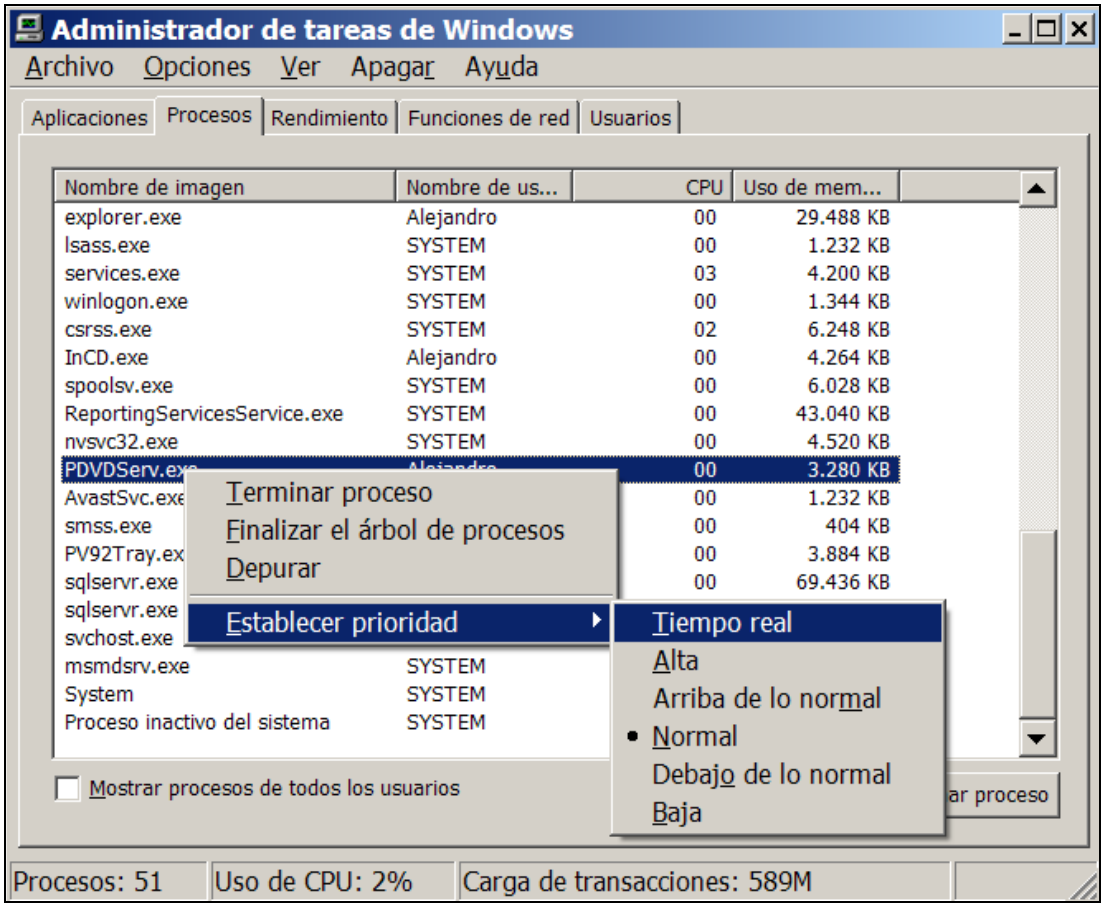


Figura 39: Establecimiento de prioridad de procesos en el sistema Windows.

Cuando se usa más de un procesador, se puede asignar un agente auxiliar por cada uno de ellos. El agente coordinador puede ser ejecutado en un procesador independiente, o compartiendo ciclos de máquina con algún agente auxiliar en un mismo procesador.

El escenario al que se enfrenta un agente auxiliar puede ser modificado en tiempo real desde otro procesador. Los cambios en el ambiente, como por ejemplo, el desplazamiento de una puerta o plataforma móvil, serán percibidos por cualquier agente activo cuyo alcance de sensores se lo permita.

Cuando no hay una red de computadores y todo el trabajo se desarrolla en un solo computador, la metodología que se dispone es de alternar ciclos de máquina para cada agente auxiliar habilitado en la búsqueda. Cada agente realiza un proceso de búsqueda, en el cual avanza un paso (una UMR) en el escenario; cuando un agente auxiliar termina su tarea, el agente coordinador selecciona el siguiente agente auxiliar para que realice un nuevo proceso de búsqueda.

4.5.4.2 *Funcionalidad del agente coordinador*

Haciendo uso de la formulación conceptual y la extrapolación dada en la sección 2.3, con la inclusión de nuevas instancias, una por cada nodo o agente auxiliar, se vio la necesidad de contar con un ***agente coordinador,*** el cual realizará la tarea de analizar los hallazgos que los demás agentes reportan; al punto de decidir si ya se encontró una ruta.

Como cada agente auxiliar cuenta con los mismos recursos que usa el Softbot, se debe tener en cuenta aquellos aspectos que se modifican en cuanto a la metodología de operación. Un caso particular es la forma de cómo los sensores reportan los hallazgos. En la búsqueda con sensores, percibir la meta es sinónimo de haber encontrado la ruta y solución al problema, mientras que en la búsqueda multidireccional, sólo se puede afirmar que una agente conoció la meta, pero no necesariamente se ha configurado la totalidad de la ruta.

Como la estrategia implica encontrar un camino entre el origen y la meta, con ayuda de nodos auxiliares, se plantea la generación de varias tablas entre ellas una con las características de los agentes y otra con el reporte de conexiones, en la cual se registran de forma inmediata los eventos que cada nodo auxiliar reporta, como podría ser el hallazgo del origen, de la meta o el hallazgo de otro agente auxiliar.

En la Tabla 12, se presenta un ejemplo de la tabla de agentes, allí aparecen n agentes, tal que el *agente_origen está* numerado como 00, el *agente_objetivo*, es decir, la meta es el 01 y a partir del tercero hasta el n-ésimo son *agentes_auxiliares*. (lo que se indica en la columna Tipo).

Agente	Tipo	Busca
00	origen	Meta
01	meta	origen
02	auxiliar	meta
03	auxiliar	origen
04	auxiliar	meta
05	auxiliar	origen
***	***	***
***	***	***
$n-1$	auxiliar	origen
n	auxiliar	meta

Tabla 12: Características de los agentes auxiliares.

También se observa en la tercera columna llamada **Busca**, que hace referencia al objetivo inicial de cada uno de los agentes. Es de señalar que el objetivo a buscar por parte de cada uno de los agentes auxiliares puede ser ingresado al software de simulación según los siguientes criterios:

- Aleatorio (meta / origen).

- Alternados entre meta y origen conforme se creen (opción por defecto implementada).

- Según distancia a la meta y al origen: calculada la distancia en línea recta desde la posición del agente hasta la meta (Dm) y hasta el origen (Do), se escogerá como objetivo la meta, si Dm es menor o igual a Do; en caso contrario escoger como objetivo el origen.

Cuando un agente reporta que pudo encontrar ya sea el origen o la meta, en el simulador se observa un cambio de su color azul claro a un color semejante al objetivo encontrado, verde para la meta y rojo para el origen, es decir, se puede saber en todo momento que información disponible tienen los agentes auxiliares.

El **agente coordinador** tabula la información de los hallazgos reportados por los diferentes agentes auxiliares, ordenándolos por el número de agente tal como se muestra en la Tabla 13. Dado que la función principal del agente coordinador es la de conformar la ruta con base en los hallazgos de los agentes auxiliares, se construyó un algoritmo basado en el concepto de búsqueda en profundidad, para encontrar el camino que une al origen con la meta.

En la Tabla 13, se presenta un caso exitoso donde el origen (agente_00) se conecta con la meta (agente_01), con la ayuda de los agentes auxiliares 07, 08, 10 y 03.

La columna denominada (Crix, Criy) contiene las coordenadas de ubicación del Agente_i, mientras que las coordenadas (Corx, Cory) son del Agente_j, puntos donde se encontraban los dos agentes en el momento en que se percibieron mediante sus sensores. Con las coordenadas comunes de los agentes involucrados en la tabla de conexiones, es posible construir mediante una rutina basada en la búsqueda en profundidad los enlaces que conforman la ruta para que el Softbot (origen), partiendo desde su posición llegue a la meta.

SIRUM: Búsqueda de rutas en robótica móvil...

Ag_i	Ag_j	Crix	Criy	Corx	Cory	Cdo	Nmrg	Cant
02	08	48	8	44	8	N	19	3
03	01	18	38	10	38	N	18	2
03	07	26	31	19	38	S	15	8
03	10	43	19	43	20	N	7	1
03	12	27	30	27	38	N	23	1
04	00	93	8	92	8	N	1	12
04	05	67	23	67	32	N	17	6
05	04	60	38	55	33	N	20	4
05	06	85	35	92	35	N	5	1
06	00	92	18	92	8	N	11	2
06	05	92	38	87	33	N	4	1
07	01	20	38	10	38	N	14	1
07	03	19	38	25	32	N	16	6
07	08	32	38	24	30	S	8	1
08	01	10	41	10	38	N	2	2
08	02	44	8	49	8	N	21	2
08	03	28	21	28	29	N	12	1
08	09	24	28	24	19	N	10	12
09	01	10	29	10	38	N	3	2
09	03	27	25	27	30	N	13	1
09	08	24	18	24	28	N	9	10
10	00	82	8	92	8	S	24	1
10	03	42	20	43	19	S	6	2
12	03	31	38	24	31	N	22	1

Tabla 13: Conexiones generadas.

La Tabla 14, contiene únicamente los registros de la Tabla 13 que constituyen la ruta encontrada. Se precisa que la construcción de la tabla resumen es una de las tareas más importante del agente coordinador.

SIRUM: Búsqueda de rutas en robótica móvil

Ag_i	Ag_j	Crix	Criy	Nmrg	Cdo	Nniv	Ruta
08	01	10	41	2	S	1	1
08	07	32	38	8	N	2	2
07	03	26	31	15	N	3	3
03	10	42	20	6	N	4	4
10	00	82	8	24	S	0	5

Tabla 14: Resumen de la ruta encontrada.

La columna **NmRg** contiene el número del registro en la tabla de conexiones; el cual es un consecutivo que se genera en tiempo real a medida que los agentes auxiliares perciben otro agente, el origen o la meta.

Las columnas **Cdo** y **Nniv,** hacen parte de los mecanismos que el agente coordinador emplea para encontrar la ruta.

La columna **Ruta**, indica el orden en que se deben tomar los registros en la consecución del camino que lleva del origen a la meta.

La búsqueda multidireccional se realiza únicamente con los agentes seleccionados o habilitados, de tal forma que cada uno de ellos recibe del agente coordinador el control de la búsqueda por una unidad de tiempo suficiente y necesaria para realizar las percepciones a que haya lugar, reportar dichas percepciones y trazar una ruta con la información disponible hasta ese instante, para finalmente dar un paso sobre dicha ruta y devolver el control al agente coordinador.

En la Figura 40, se ilustra lo expuesto en el párrafo anterior y se precisa como cada agente auxiliar aplica el mismo algoritmo, cada vez que tiene el control de la búsqueda.

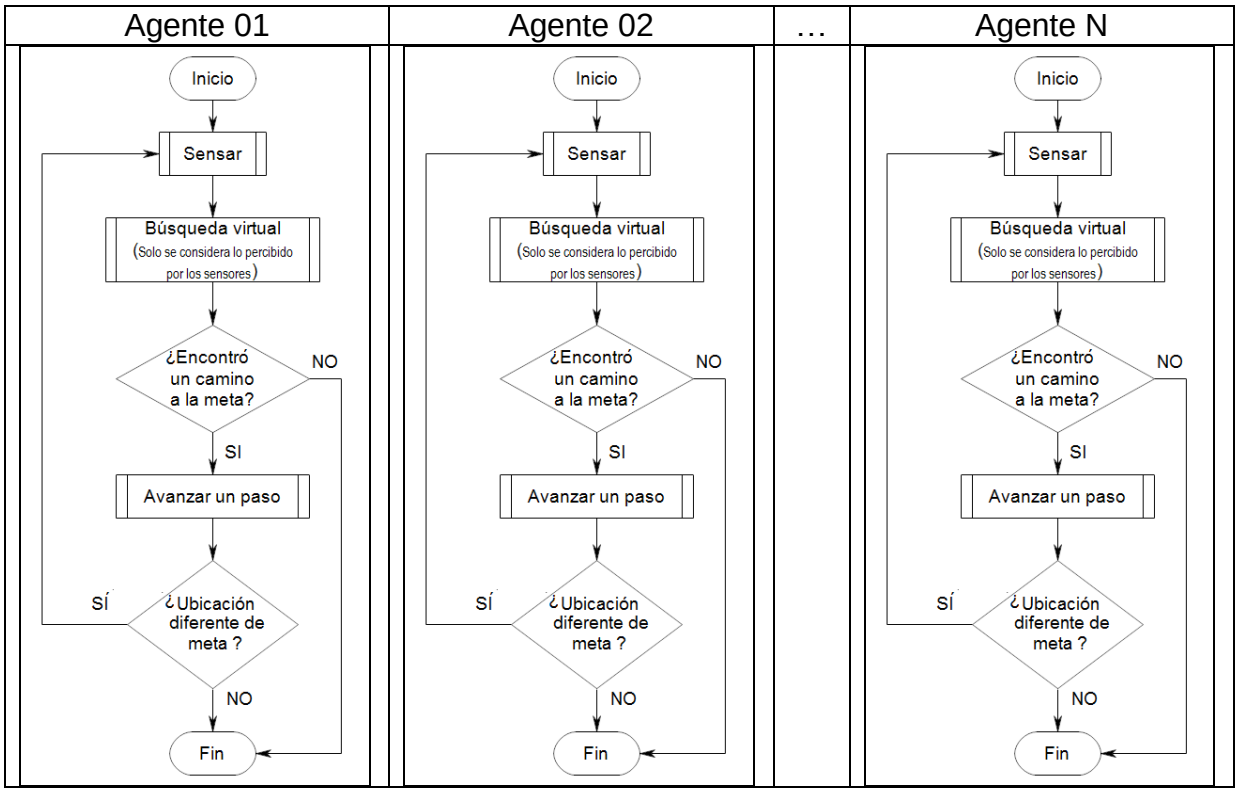


Figura 40: Algoritmo de los agentes auxiliares.

Dado que las percepciones de los agentes auxiliares son similares a las del Softbot, realizadas mediante cada uno de sus ocho (8) sensores, estas se ven reflejadas en un escenario en común, es decir, todos ellos se benefician de los hallazgos reportados.

Al desarrollar la simulación y encontrarse una posible ruta del origen hasta la meta, dicho camino se construye mediante los tramos que los agentes auxiliares involucrados aportan a la solución. Para describir y presentar la ruta en el ***Display***, se debe realizar una depuración por parte del agente coordinador, de tal forma que se descarten segmentos de ruta que parten un punto y regresan al mismo punto (lazos cerrados).

En la Figura 41, se muestra el algoritmo que ejecuta el agente coordinador, donde debe usar una metodología de búsqueda en profundidad para encontrar los tramos de la ruta aportados por cada agente auxiliar, según la tabla de conexiones.

El agente coordinador en su funcionalidad tiene en cuenta las siguientes premisas:

- Si un agente auxiliar percibe la meta o el origen, toma nota de dicho hallazgo de tal forma que pueda informar de él a otro agente con el que se encuentre más adelante.

- En un momento dado el objetivo de un agente auxiliar puede ser el origen o la meta. Una vez el agente auxiliar percibe su objetivo, por ejemplo la meta, cambia de objetivo y comienza un nuevo proceso de búsqueda, pero ahora su nuevo objetivo es el origen.

- Cuando dos agentes auxiliares se perciben y llevan la misma información (igual objetivo encontrado), el agente coordinador procede a desactivar uno de ellos, con el fin de evitar dobles esfuerzos.

- Un agente auxiliar desactivado con información, puede trasferir dicha información a otro agente que no la contenga.

El proceso de búsqueda termina cuando se dé alguno de los siguientes casos:

a) Un mismo agente auxiliar percibe el origen y la meta.

b) Dos o más agentes auxiliares se conectan todos entre sí, de tal forma que al menos uno de ellos encuentre la meta y otro el origen.

c) Cuando se alcance un determinado límite de iteraciones (100.000) en el algoritmo de búsqueda sin encontrar una ruta.

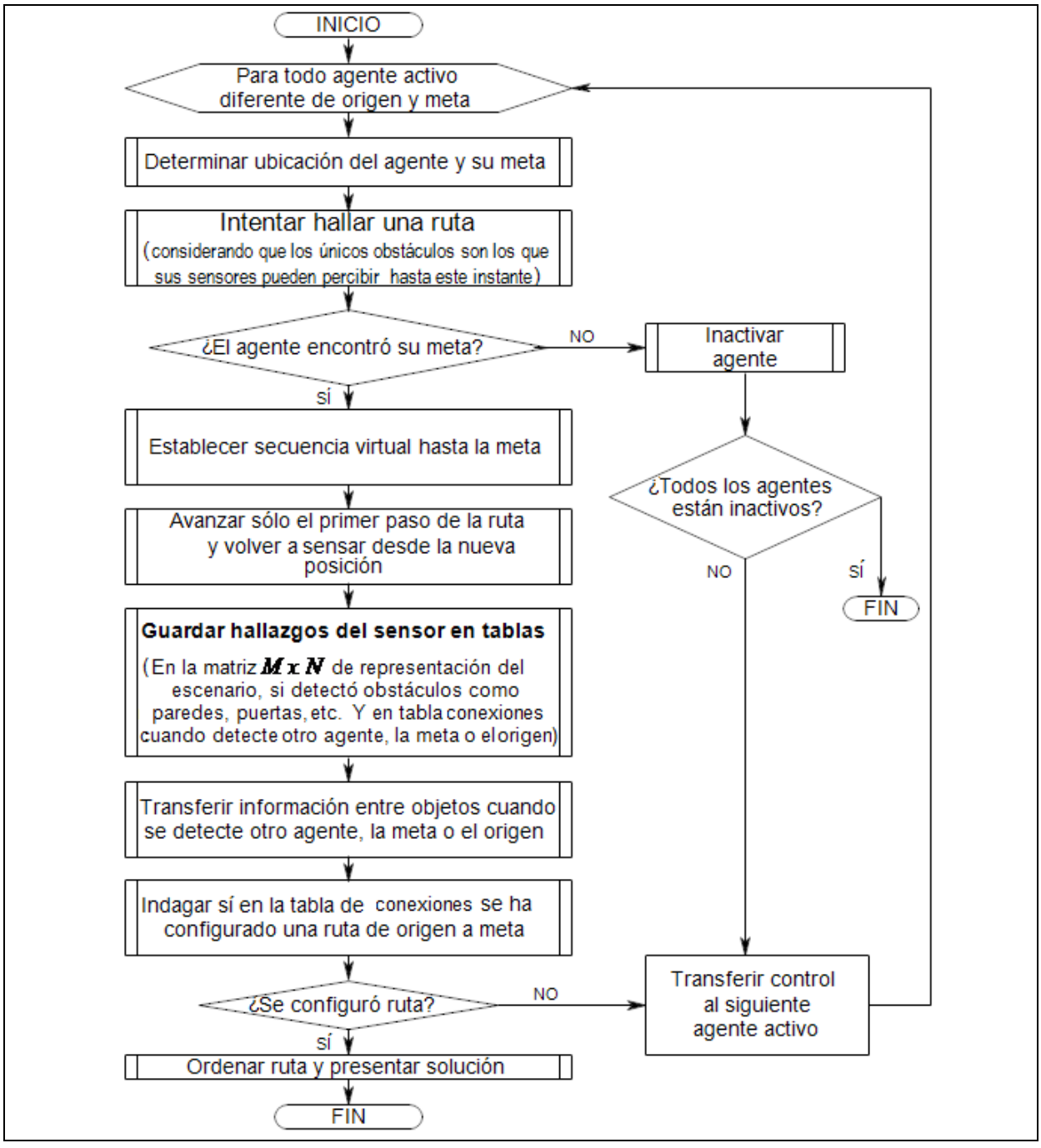


Figura 41: Algoritmo del agente coordinador

Una vez se termine la ejecución del algoritmo del agente coordinador y se tenga la tabla resumen, se hace simplificación de la ruta para finalmente mostrar el desplazamiento del Softbot.

Capítulo 5

PRUEBAS Y RESULTADOS

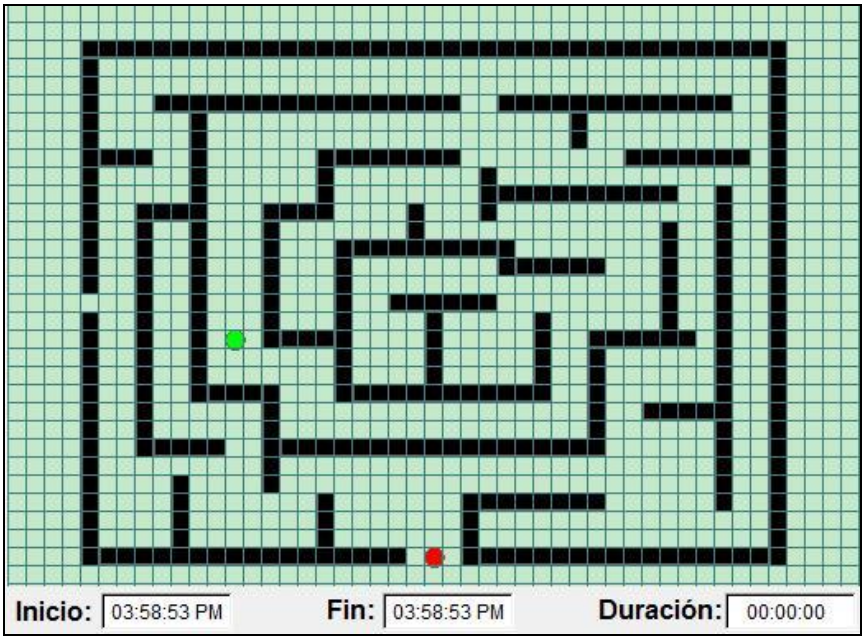
5.1 EJECUCIÓN DE LA BÚSQUEDA SIN SENSORES

En este modo de funcionamiento se tiene todo el conocimiento del ambiente y se presenta el desarrollo de la estrategia de forma rápida. La ruta obtenida es una ruta muy próxima a la óptima.

Ejemplo No.1:

Se plantea un escenario tipo laberinto, donde el agente inteligente (Softbot) se encuentra incorporado en el nodo origen (UMR de color rojo), por tanto, conoce la información de las bases de datos que incluyen todos los elementos del escenario. Haciendo uso de la esta información, junto con la estrategia de búsqueda, realiza la indagación de todo el árbol de estados, hasta encontrar la solución.

Una vez ejecutada la búsqueda la ruta, la información de las coordenadas queda almacenada en una tabla de datos, la cual se muestra al final del proceso.

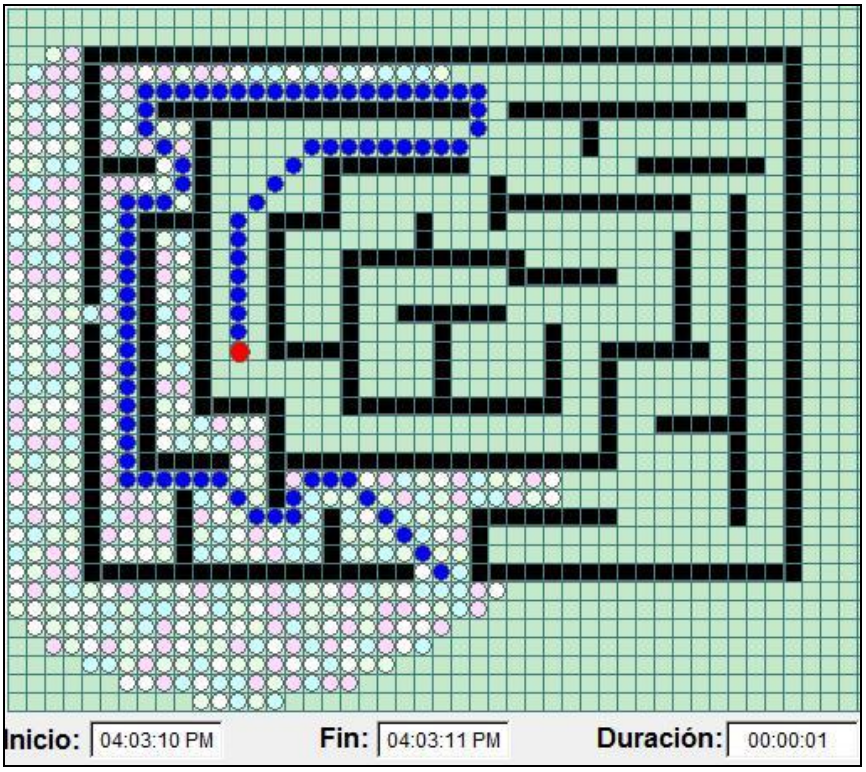


Condiciones iniciales del escenario:

Escenario en forma de laberinto con paredes fijas (UMRs de color negro).

Origen: (UMR de color rojo).

Meta: (UMR de color verde).



Proceso de búsqueda en ejecutado.

Se muestra el escenario, con las huellas, producto de la indagación del árbol de estados.

La ruta encontrada siguiendo la estrategia de búsqueda sin sensores, se detalla en color azul oscuro.

Tiempo de duración del proceso de búsqueda:

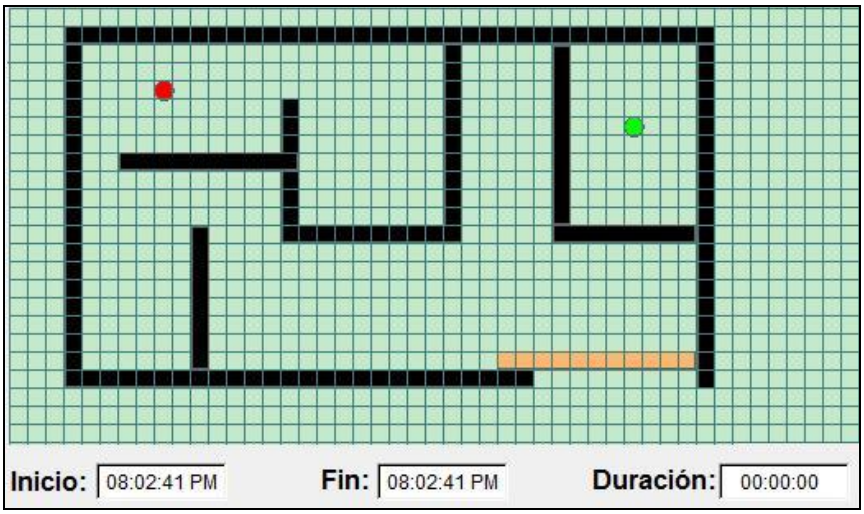
(hh:mm:ss)
00:00:01

La siguiente tabla contiene los datos de la ruta encontrada:

OP	RUTA	X	Y	Costo
O	0000000000	24	32	16.5529
8	0000000008	23	31	15.2315
8	0000000012	22	30	13.9284
8	0000000016	21	29	12.6491
8	0000000021	20	28	11.4018
8	0000000026	19	27	10.1980
8	0000000030	18	26	9.0554
7	0000000032	17	26	9.0000
6	0000000033	16	27	10.0499
6	0000000062	15	28	11.1803
8	0000000066	14	27	10.4403
1	0000000067	14	26	9.4868
8	0000000072	13	25	8.9443
6	0000000104	12	26	10.2956
7	0000000107	11	26	10.8167
7	0000000108	10	26	11.4018
7	0000000110	9	26	12.0416
7	0000000112	8	26	12.7279
8	0000000115	7	25	12.8062
1	0000000116	7	24	12.2066
1	0000000120	7	23	11.6619
1	0000000122	7	22	11.1803
1	0000000124	7	21	10.7703
1	0000000126	7	20	10.4403
1	0000000128	7	19	10.1980
1	0000000130	7	18	10.0499
1	0000000132	7	17	10.0000
1	0000000134	7	16	10.0499
1	0000000136	7	15	10.1980
1	0000000138	7	14	10.4403
1	0000000140	7	13	10.7703
1	0000000142	7	12	11.1803
2	0000000145	8	11	10.8167
3	0000000149	9	11	10.0000
2	0000000151	10	10	9.8995
1	0000000154	10	9	10.6301
1	0000000156	10	8	11.4018
8	0000000159	9	7	12.8062
8	0000000202	8	6	14.2127
2	0000000204	9	5	14.4222
3	0000000209	10	5	13.8924
3	0000000212	11	5	13.4164
3	0000000214	12	5	13.0000
3	0000000216	13	5	12.6491
3	0000000218	14	5	12.3693
3	0000000220	15	5	12.1655
3	0000000222	16	5	12.0416
3	0000000224	17	5	12.0000
3	0000000226	18	5	12.0416
3	0000000228	19	5	12.1655
3	0000000230	20	5	12.3693
3	0000000232	21	5	12.6491
3	0000000234	22	5	13.0000
3	0000000236	23	5	13.4164
3	0000000238	24	5	13.8924
3	0000000240	25	5	14.4222
4	0000000243	26	6	14.2127
6	0000000248	25	7	12.8062
5	0000000250	25	8	12.0416
4	0000000452	26	9	12.0416
6	0000000456	25	10	10.6301
6	0000000459	24	11	9.2195
7	0000000463	23	11	8.4853
6	0000000465	22	12	7.0711
6	0000000469	21	13	5.6569
7	0000000472	20	13	5.0000
7	0000000474	19	13	4.4721
6	0000000476	18	14	3.1623
6	0000000479	17	15	2.0000
5	0000000483	17	16	1.0000
5	0000000488	17	17	0.0000

Ejemplo No.2:

Se plantea un escenario en el cual no hay solución, puesto que no hay forma de llegar a la meta; por tanto el software reporta que se han cumplido el número máximo de iteraciones sin encontrar la ruta.



Condiciones iniciales del escenario:

Escenario con paredes fijas (UMRs de color negro).

Puerta deslizante: (UMRs de color naranja).

Origen:(UMR de color rojo).

Meta:(UMR de color verde).

Proceso de búsqueda en ejecución.

Se muestran las huellas, producto de la indagación del árbol de estados.

Un mensaje con el texto correspondiente, que indica que no se encontrar la meta.

Todo el escenario fue indagado por el agente inteligente, sin converger en una solución.

Tiempo de duración del proceso de búsqueda:

(hh:mm:ss)
00:00:01

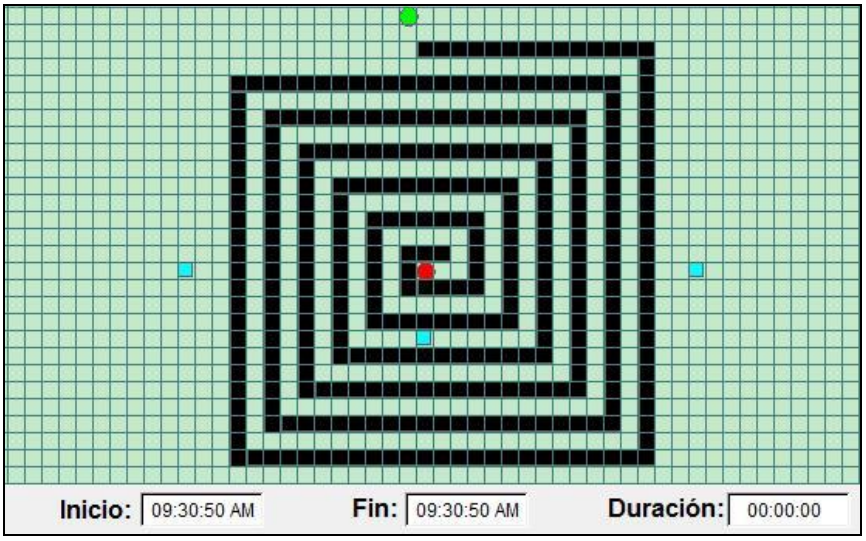
5.2 EJECUCIÓN DE LA BÚSQUEDA MULTIDIRECCIONAL

Teniendo en cuenta que la estrategia de búsqueda multidireccional es una aplicación de la búsqueda con sensores, sobre cada uno de los agentes auxiliares, se presenta tres ejemplos con entornos construidos y que hacen uso del trabajo cooperativo de ellos.

Ejemplo No. 1:

Para la prueba de la búsqueda se ubican dos agentes auxiliares por fuera del escenario (tipo laberinto), mientras que uno de ellos está ubicado por dentro. En la estrategia de búsqueda, algunos agentes auxiliares tendrán como objetivo inicial buscar el origen y otros la meta.

Origen: UMR en forma de círculo de color rojo.
Meta: UMR en forma de círculo de color verde.
Agentes Auxiliares: UMRs en forma de cuadro de color azul claro.
Obstáculos fijos: UMRs de color negro (paredes).



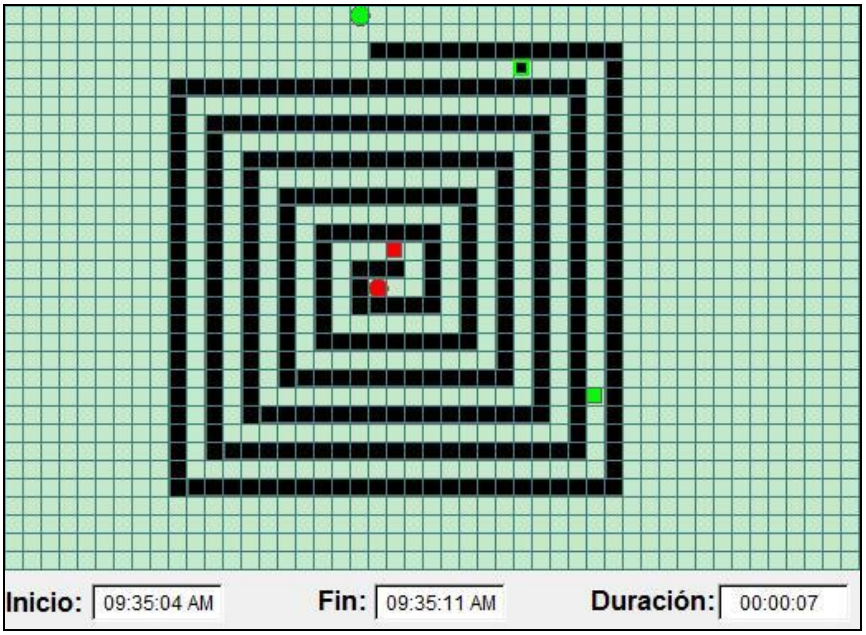
Condiciones iniciales del escenario:

Origen: (rojo).

Meta: (verde).

Tres agentes auxiliares sin información (azul claro).

En la parte inferior del escenario, se encuentran los valores del tiempo para el proceso de búsqueda. Inicialmente antes de comenzar la ejecución, los relojes de inicio y fin contienen la misma información horaria.



Proceso de búsqueda en marcha, se destacan:

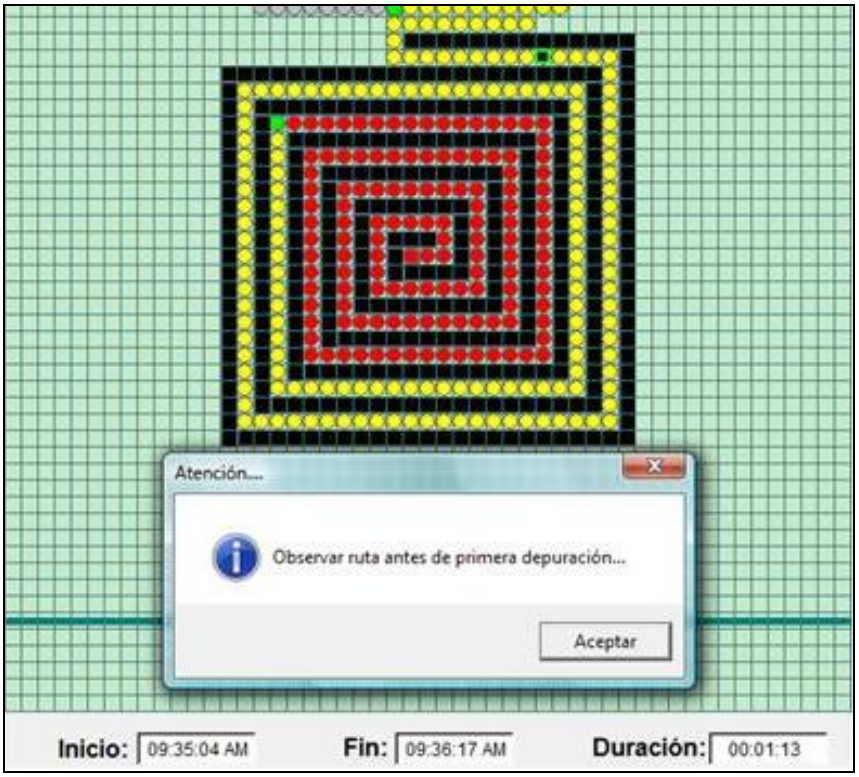
Origen (círculo rojo).
Meta (círculo verde).

Tres agentes auxiliares:

Con información del origen (cuadro rojo).

Con información de la meta (cuadro verde).

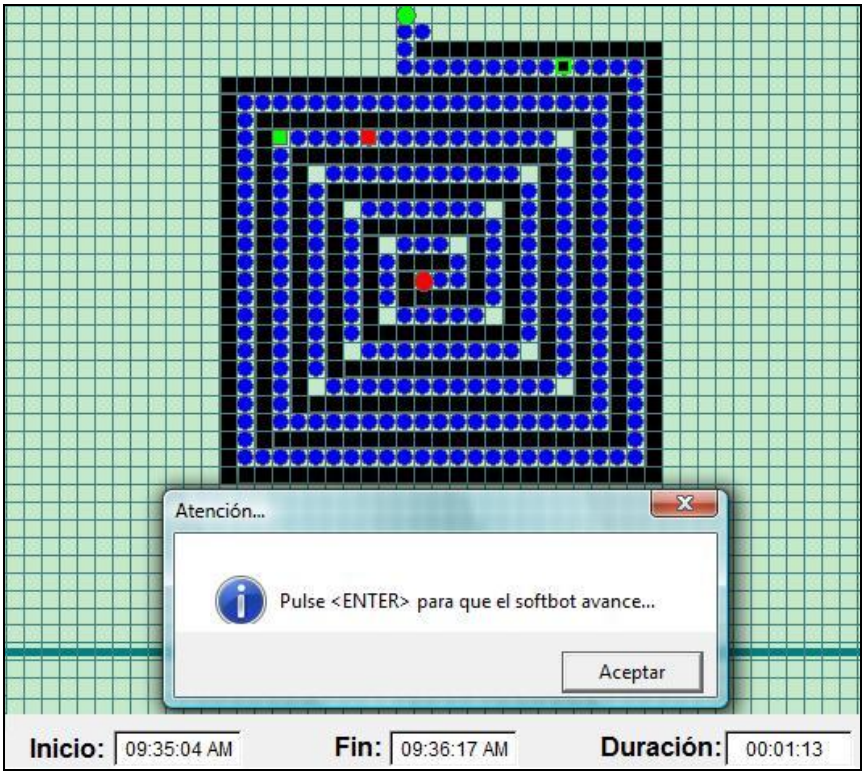
Con información de la meta e inactivo (cuadro verde con negro).



Escenario:

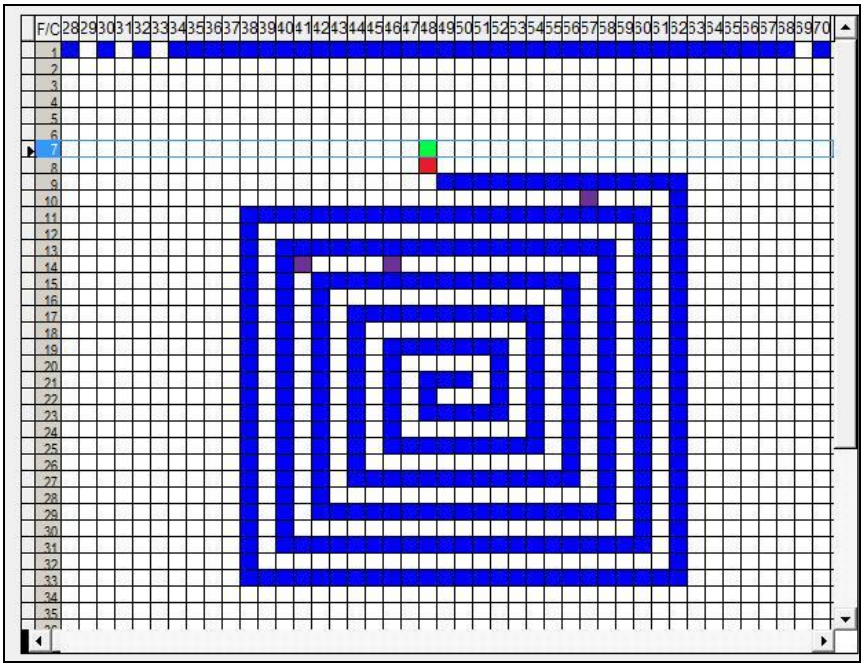
Ruta hallada por la contribución de los agentes auxiliares.
En color rojo se muestra la ruta que uno de los agentes aporta, otro ayuda con parte de ella de color amarillo y de color gris otra sección de ruta aportada por el último agente.

Tiempo de duración: (hh:mm:ss)
00:01:13



Escenario:

Ruta final depurada, se obtiene el camino de color azul oscuro que lleva del origen a la meta.



Matriz MxN **:**

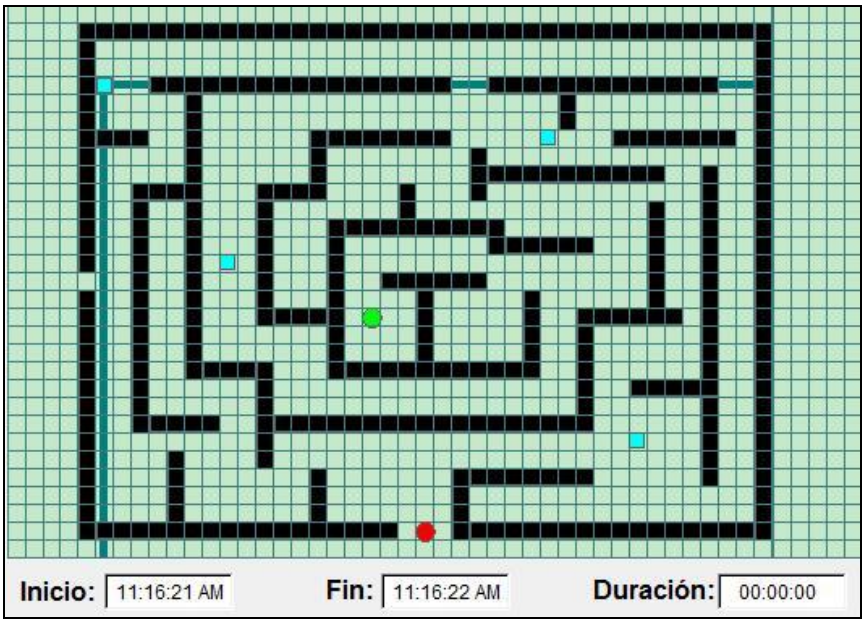
Escenario percibido por los agentes auxiliares.

Se muestra en color azul el levantamiento realizado por los sensores del escenario.

En la implementación de la búsqueda multidireccional, es necesario tener en cuenta que la cantidad de agentes auxiliares es determinada por el usuario. Se pudo obtener en pruebas aparte realizadas, que tener una cantidad excesiva de agentes auxiliares, hace que el proceso de búsqueda sea lento y que se realicen esfuerzos duplicados.

Ejemplo No. 2:

En el ejemplo se usa el mismo escenario que fue utilizado en la búsqueda sin sensores, de tal manera, que con la ayuda de cuatro (4) agentes auxiliares, trabajando cooperativamente, se pueda tener una ruta que enlace el origen con la meta. En la aplicación multidireccional el origen y la meta estarán fijados en sus coordenadas iniciales.



Escenario inicial:

Origen: (círculo de color rojo).

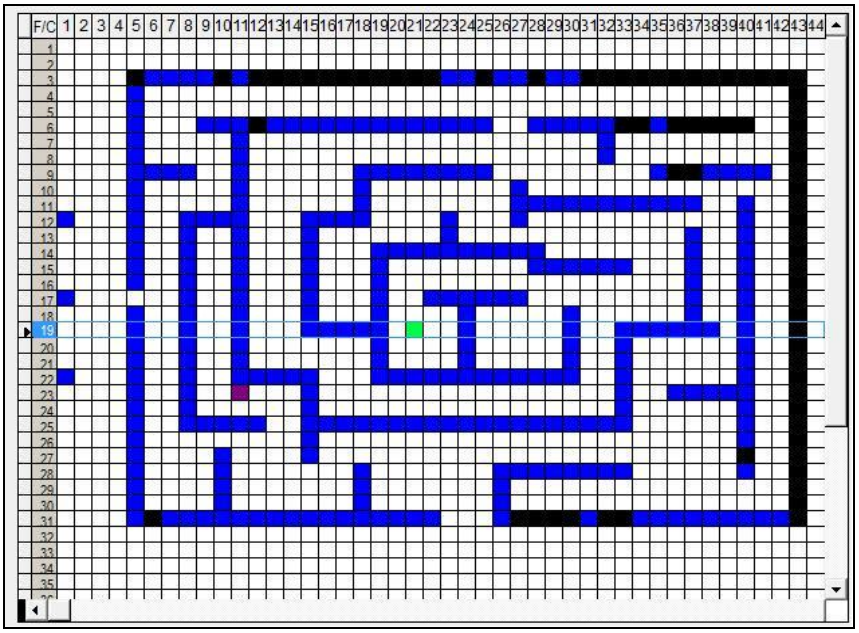
Meta: (circulo color verde).

Tres agentes auxiliares sin información (cuadros de color azul claro).

Escenario:

Ruta hallada por la contribución de los agentes auxiliares.
En color verde se muestra la ruta que uno de los agentes aporta, otro ayuda con parte de ella de color amarillo.

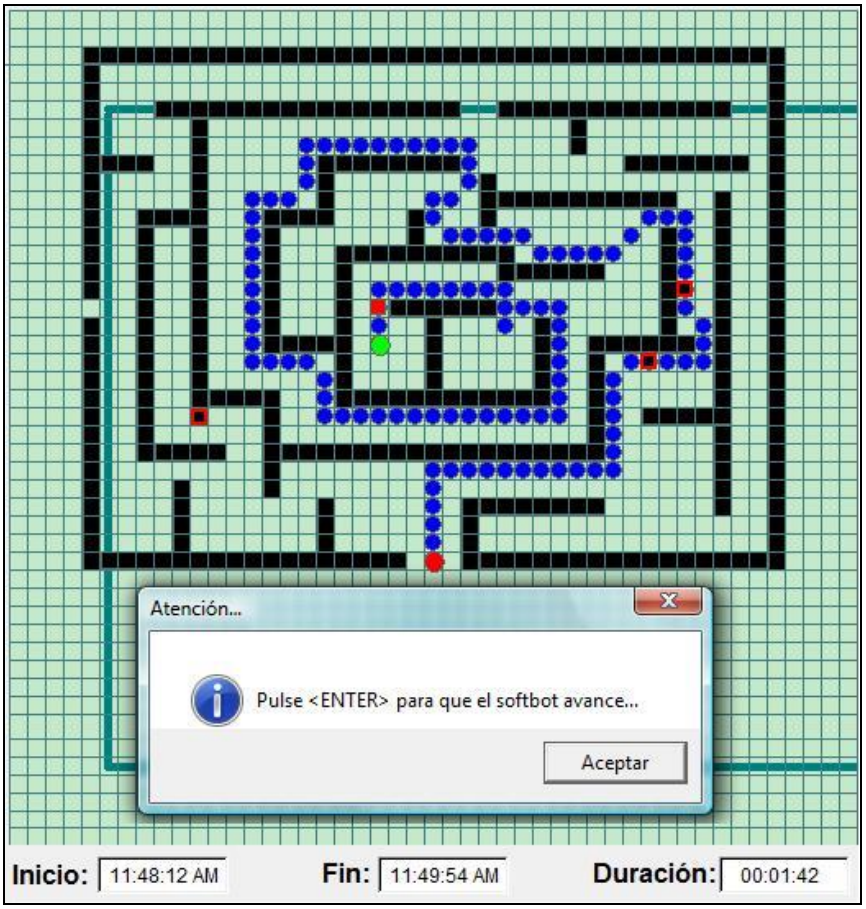
Tiempo de Duración:
(hh:mm:ss)
00:01:42



Matriz MxN **:**

Escenario percibido por los agentes auxiliares.

Se muestra en color azul el levantamiento del plano del escenario, realizado por los agentes auxiliares haciendo uso de los sensores.



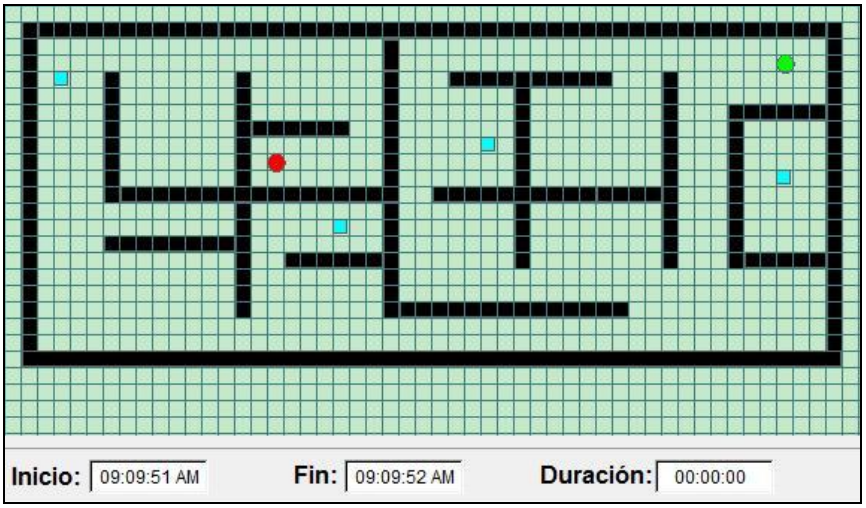
Escenario:

Ruta final depurada, se obtiene el camino de color azul oscuro que lleva del origen a la meta.

En el segundo ejemplo se ilustra la conveniencia de los agentes auxiliares, y como cada uno de ellos aportan en la solución de la ruta. Igualmente al final se tiene en la capa de la matriz MxN un escenario percibido por los agentes en un alto porcentaje, permitiendo que para nuevas pruebas, se pueda contar con información previa.

Ejemplo No. 3:

Se plantea un escenario donde no es posible encontrar la ruta por parte de un agente auxiliar, puesto que se encuentra encerrado (sin salida) en una parte del entorno, pero los otros agentes contribuyen a la solución.



Escenario inicial:

Origen (círculo rojo).

Meta (círculo verde).

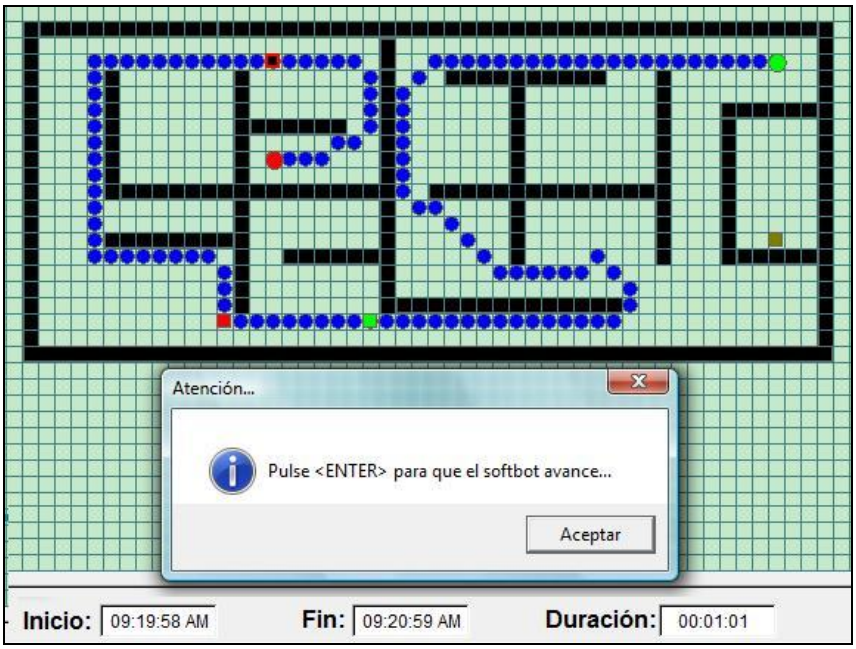
Cuatro agentes auxiliares sin información (cuadros azules).

Escenario:

Ruta hallada por la contribución de los agentes auxiliares.
En color amarillo se muestra la ruta que uno de los agentes aporta, otro contribuye con parte de ella de color rojo y en color oscuro la parte final, también aportada por otro agente.

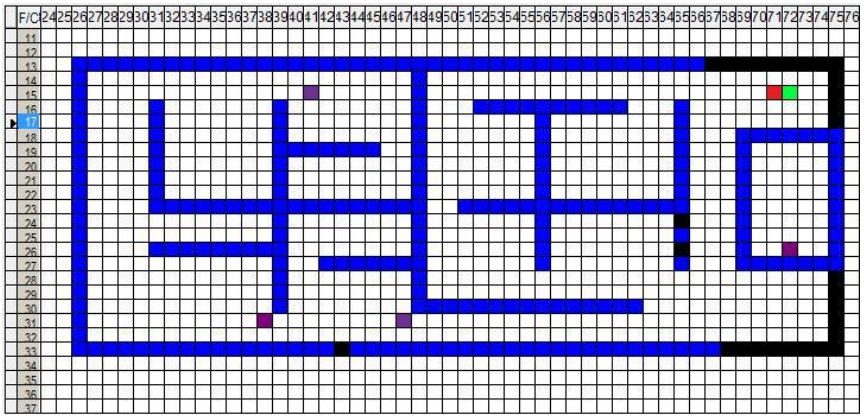
AE: Agente auxiliar encerrado, no contribuye en la búsqueda de la ruta.

Tiempo de duración:
(hh:mm:ss)
00:01:01



Escenario:

Ruta final depurada, se obtiene el camino de color azul oscuro que lleva del origen a la meta.



Matriz MxN **:**

Escenario percibido por los agentes auxiliares.

Se muestra en color azul el levantamiento del plano del escenario, realizado por los agentes auxiliares haciendo uso de los sensores.

Como se puede observar en la solución del ejemplo, la contribución cooperativa de los agentes auxiliares hace que se tenga un levantamiento de gran parte del escenario y con ello encontrar la ruta que une el origen con la meta.

Capítulo 6

CONCLUSIONES Y RECOMENDACIONES

- El proyecto SIRUM es una herramienta informática de simulación, que permite la construcción simplificada de ambientes atestados en dos dimensiones, tales como plantas industriales y laberintos, que incluyen objetos especiales como puertas deslizantes y plataformas móviles; lo anterior, usando una interfaz gráfica simple para la navegación de un agente -robot de software- que de forma virtual encuentra una ruta para ir de un punto a otro.

- Se pudo comprobar que en la construcción del modelo del agente inteligente, se requiere incorporar funcionalidades de software que permitan pasar de las percepciones a las acciones. Dicho software se apoya en la construcción de árboles de estados, sobre los cuales realiza búsquedas orientadas al alcance de la meta, apoyado o dirigido por funciones de decisión o de utilidad, las cuales son diseñadas mediante heurísticas que puedan ser ajustadas dinámicamente mejorando el tiempo de ejecución.

- Tanto la estrategia de búsqueda como la heurística empleada, se construyeron como módulos de software que se pueden reemplazar por otros, con fines de experimentación, siempre y cuando cumplan con los criterios de interface del simulador.

- Durante la etapa de planificación, es fundamental restringir la profundidad de la búsqueda para poder responder dentro de los límites de tiempo. Si bien el tiempo de ejecución de los algoritmos de búsqueda es exponencial respecto a la frontera de búsqueda, la misma está acotada por una constante (100.000 iteraciones) y, por lo tanto, también lo está el tiempo de ejecución.

- Comparado el rendimiento en velocidad de las búsquedas implementadas: *sin sensores, con sensores y la multidireccional*, es claro que la sin sensores produce mejores resultados, pero supone un escenario completamente conocido.

- El trabajo cooperativo de los agentes auxiliares para realizar entre otras, tareas el levantamiento incremental del escenario, permite compartir la información de áreas del entorno, en beneficio del proceso de búsqueda que cada agente realiza. Este modelo conlleva a que el trabajo se realice en tiempos cortos.

- El uso de más de un agente inteligente evita los atascamientos que se pueden dar cuando se cuenta con un único agente. La ayuda que otro agente presta no sólo se limita a la transferencia de información, sino que se ve reflejada en el levantamiento del escenario en común.

- Considerar que lo único que existe en el entorno es lo percibido por los agentes, mediante sus sensores, en un instante dado fue decisivo para obtener una ruta. Dicha técnica permite usar la búsqueda sin sensores y obtener una solución rápida.

- El alcance de los sensores incide en el desempeño y calidad de la búsqueda; con mayor alcance se puede realizar un levantamiento del escenario en menor tiempo. El simulador cuenta con la facilidad de cambiar el alcance, variando en el menú el tamaño de UMRs para los sensores.

- La inhabilitación de agentes redundantes -*que siendo cercanos cuentan con la misma información y tienen en mismo objetivo*- evita el procesamiento innecesario de agentes que ya no mejoran el desempeño de la búsqueda.

- En el simulador es posible adicionar y probar otro tipo de sensores virtuales, ejemplo sensores de temperatura, olfativos y de humedad. De igual manera, los sensores virtuales pueden incorporar las lecturas de sensores físicos, mediante el diseño y construcción de ***Drivers*** como mecanismo de interface software-máquina, permitiendo emplear a SIRUM como un módulo inteligente de control en varias aplicaciones.

- La estrategia de búsqueda multidireccional requirió manejar un nivel de abstracción superior, que coloca en un mismo plano de interacción conceptos como: *agente inteligente, sensor virtual, búsqueda por amplitud, procesamiento en paralelo y trabajo cooperativo;* dejando puertas abiertas para trabajos futuros. En la página 29, se enuncian claramente las estrategias de búsqueda existentes, concluyéndose que la estrategia de búsqueda multidireccional diseñada, construida y probada en la presente tesis, *es un aporte innovador* en el estado del arte, que no ha sido considerado en ningún otro trabajo del que se tenga noticia.

6.1 TRABAJOS FUTUROS

Siendo la búsqueda de rutas, una aplicación de la Inteligencia Artificial, se pueden desarrollar trabajos futuros a partir de la realización de la presente tesis, entre ellos se mencionan:

- **Integración y reconocimiento de planos de AutoCad al simulador.**

  Parte de las investigaciones previas a la formulación, llevaron a que el software AutoCad, especializado en la construcción de todo tipo de planos que incluyen plantas industriales, zonas de carga y descarga marítima, terrestre y aeroportuaria, registra o exporta sus planos como información vectorial en formato de Excel (xls); el cual puede ser importado fácilmente al SIRUM, gracias a que su diseño tuvo en cuenta esta facilidad.

- **Optimización de la ruta obtenida.**

  Una primera optimización es la de construir una función de línea_vista para utilizar atajos dentro del escenario (caminos directos de un punto a otro); igualmente el diseño y construcción de nuevas heurísticas que involucren aspectos como la aleatoriedad y la predicción.

- **Implementación del algoritmo para el manejo del Hardware (Robot físico).**

  Incluye el diseño y construcción de Drivers que permiten el acople del software desarrollado con el hardware de un robot físico.

- **Desarrollo del simulador en 3D.**

  Es posible incorporar un nuevo eje (Z) para incluir solución a problemas que implique una tercera dimensión, como es el caso de la navegación aérea y el control de equipos automáticos de transporte y almacenamiento de mercancías.

  Igualmente se plantea la utilización de software comercial de virtualización de escenarios.

- **Incorporación de conocimiento previo.**

  Teniendo en cuenta los resultados de la búsqueda multidireccional, y que la información se encuentra almacenada en las tablas, se puede usar para otros casos, segmentos de la ruta previamente establecida. Lo anterior, puesto que la ruta es un conjunto de puntos sucesivos en el plano cartesiano y cada vez que se inicie un nuevo proceso de búsqueda de, es posible acudir al “banco” de soluciones antes calculadas.

- **Adición de capacidades predictivas al simulador:**

  Adicional a la capacidad de comportamiento reactivo del simulador ante cambios del entorno, se puede adicionar una capa de software predictiva y proyectiva, que registre cómo evoluciona el ambiente en el tiempo. Es el caso especial de los objetos móviles, para los cuales se pueden registrar sus movimientos en un marco temporal, de tal forma que dicha información se pueda usar para realizar proyecciones de posibles colisiones o despejes de áreas particulares del entorno.

Capítulo 7

REFERENCIAS

[1] Russell, S. J., Norvig, P., "*Inteligencia Artificial un enfoque moderno*" (p. 48-80-87). Ciudad de México, México: Prentice Hall, primera edición (1996).

[2] Desarrollo y utilización de simuladores en la industria nuclear. http://www.iiisci.org/journal/CV%24/risci/pdfs/P541543.pdf

[3] Estrategias de búsqueda. http://evirtual.lasalle.edu.co/info_basica/nuevos/guia/resumenNivel1.pdf

[4] Nilsson, J. N., "*Inteligencia artificial",* (c. 10). Madrid, España: McGraw-Hill, primera edición (2001).

[5] Pressman, R. S., *"Ingeniería del software un enfoque practico"*, (c. 4) Madrid, España: McGraw-Hill, quinta edición (2002).

[6] Davis, G. B., Olson M. H., "*Sistemas de información gerencial",* (p. 290). Ciudad de México, México: McGraw-Hill, segunda edición (1995).

[7] Sensores de proximidad. http://www.festo.com/cms/es-co_co/16457.htm

[8] Festo Didactic. "*Sensores para la manipulación de procesos*". (p. 11), primera edición (2005).

[9] A. Villar y J. Pastor. "*Entorno de simulación y control de un robot velocista". TFC.* Escuela Politécnica Superior. Universidad de Alcalá: España (2005).

[10] Real Academia de la Lengua Española. http://www.rae.es(diccionario de la lengua española - vigésima segunda edición).

Capítulo 8

ANEXOS

A. GUÍA DEL USUARIO

A continuación se presentan la forma de realizar una simulación, indicando cómo se hace la representación dinámica del escenario, la conformación de un ambiente con los diferentes elementos que interactúan en él.

A.1 Entorno del simulador

Para realizar una simulación se puede hacer uso de las siguientes alternativas:

- Construir el escenario desde cero.
- Recuperar un proyecto o escenario previamente construido y guardado en el sistema de archivos que lo contiene.

La creación de una simulación implica un conjunto de pasos e información que debe conocerse de antemano. En primer lugar, deberá asignársele un nombre al proyecto. También se le debe establecer la ruta o ubicación del proyecto dentro del sistema de carpetas del sistema operativo. (Se propone aceptar por defecto el sugerido por el sistema).

Manual del software SIRUM (CD incluido en el documento):

Pasos para ejecutar el software:

1) Copiar la carpeta ***CasaUtp*** contenida en el CD, a la raíz del disco duro (C:) de su computador.
2) Copie el acceso directo Sirum_2011.lnk (contenido en el directorio ***CasaUtp***) al escritorio de su computador.
3) Ejecutar el acceso directo (haciendo doble clic en el icono).

Una vez hecho el doble clic en el icono de acceso al software, ubicado en el escritorio del computador, aparece el siguiente menú para acceso al simulador:

En este formulario se puede hacer un clic en:

a. **CREACION DE PROYECTOS** para generar un nuevo proyecto o escenario.
b. **SIMULADOR** para acceder al software de simulación.
c. **Res. 1280 x 800** o **Res.1280 x 960** para cambiar la resolución de pantalla.
d. **Res. Inicial** para restaurar la resolución que tenía el computador antes de cargar el simulador.
e. **SALIR** para abandonar la aplicación.

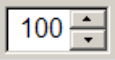
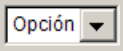
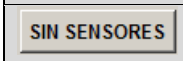
A.2 Creación de proyectos

Use el siguiente formulario para crear un nuevo proyecto o eliminar uno existente. El botón *Importar plano desde Auto-CAD* esta deshabilitado porque sólo sirve para indicar que en una próxima versión del software, será añadida la funcionalidad de "cargar" automáticamente un escenario digitalizado en Auto-CAD.

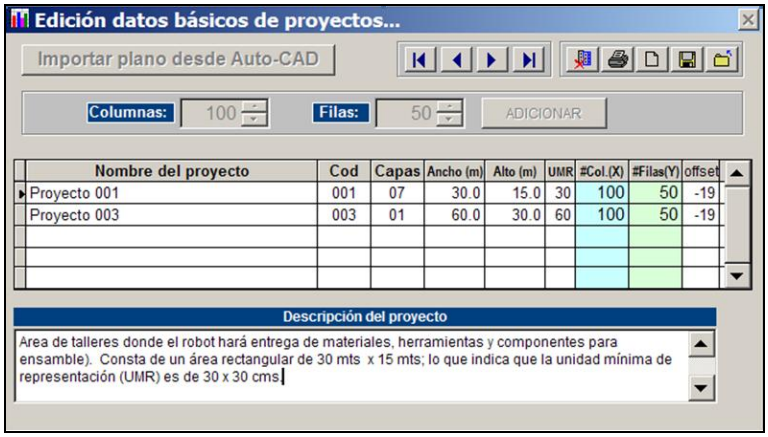
A.2.1 *Convenciones generales de botones para realizar acciones*

Donde quiera que aparezcan los botones de comando que se relacionan a continuación, tienen un significado o acción similar a las descritas al frente de cada uno de ellos:

	Ir al primer registro de una tabla
	Ir al registro anterior
	Ir al registro siguiente
	Ir al último registro
	Eliminar registro actual
	Imprimir
	Agregar un nuevo registro / proyecto
	Grabar información en el disco.
	Salir del formulario.

	Introducir un valor numérico con el teclado o haciendo uso de las puntas de flecha para aumentar o disminuir el valor.
	Hacer clic en la punta de flecha permite seleccionar una opción.
	Botones de comando que describen brevemente la acción a realizar.

Para adicionar un nuevo proyecto, haga clic en el botón agregar. Como respuesta a esta acción, se habilitan las celdas para definir el número de columnas y de filas mediante las cuales se estructurará el escenario.



A.2.2 *Generación de proyectos*

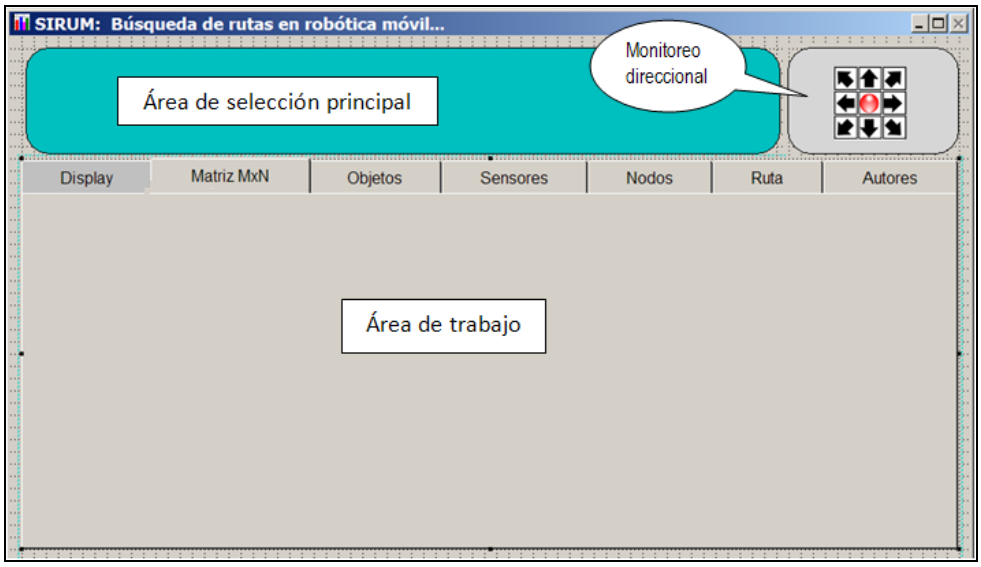
Una vez definido el número de columnas y de filas, haga clic en el botón ADICIONAR para agregar el nuevo proyecto. Puede cambiar el nombre por defecto del proyecto por uno que exprese mejor de que se trata.

En la versión actual el número de capas no es un dato relevante. (El software lo deduce automáticamente). Las columnas ***Ancho*** y ***Alto***, expresadas como números positivos, corresponden a las dimensiones en metros del área del escenario. El **Offset** es un valor entero numérico positivo o negativo para ajustar el software a diferentes tipos de pantallas (resoluciones).

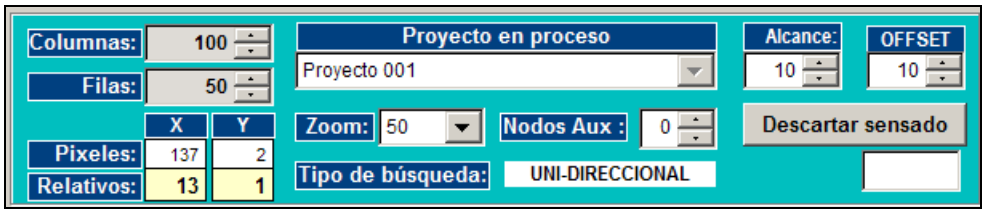
La columna UMR (unidad mínima de representación) correspondiente a una de las celdas que conforman el escenario, es el resultado de dividir el valor del ***Ancho*** por el número de columnas que tiene el escenario digitalizado. Si bien es cierto que el software se diseñó para que el número de columnas varíe entre 0001 y 9999, permitiendo escenarios tan grandes como un terminal marítimo o aeroportuario, se tiene reservada esta capacidad para la versión industrial del software. La versión académica alcanza un máximo de 100 columnas y 50 filas.

A.3 Alternativas de selección en el simulador

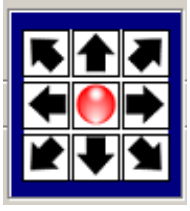
En términos generales el simulador se extructura como se muestra en el gráfico siguiente:



El *área de selección principal* corresponde al siguiente gráfico:



- El selector *Proyecto en proceso* permite seleccionar un proyecto antes definido.
- Zoom: el selector permite cambiar el tamaño de las celdas mostradas en la página “Matriz MxN” del área de trabajo.
- Nodos Aux: permite seleccionar el número de agentes auxiliares con los que desa trabajar. (Una vez establecidos los agentes, podrá cambiar su posición haciendo un clic sobre ellos y desplazándo el ratón del computador a una nueva posición donde los soltará haciendo doble clic.
- El alcance corresponde al número de UMRs que es capaz de sensar el sofbot en una determinada dirección.
- Los campos ***Columnas*** y ***Filas*** son los establecidos cuando se creó el proyecto.
- ***Pixeles*** corresponde a la posición en esa medida del puntero del mouse o de un objeto seleccionado.
- ***Relativos*** corresponde a las coordenadas cartesianas X e Y de un objeto o posición señalada con el ratón.
- ***Monitoreo direccional:*** una vez ubicado el Softbot (origen) dentro del escenario, es posible hacer que se desplace con el navegador haciendo clic en las flechas direccionales.

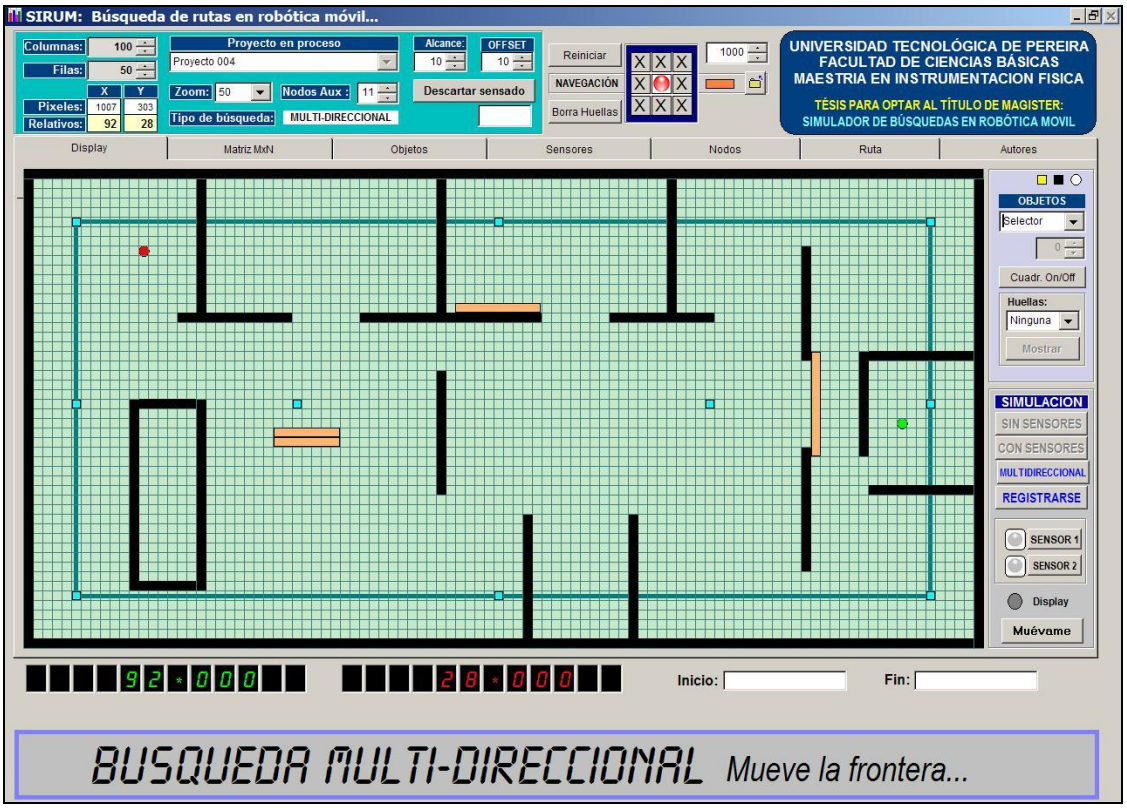


(1): Arriba
(2): Arriba-Derecha
(3): Derecha
(4): Abajo-Derecha
(5): Abajo
(6): Abajo-Izquierda
(7): Izquierda
(8): Arriba-Izquierda

A.4 Ventana del Display

Al hacer clic en la pestaña ***Display***, se presenta un área de trabajo, *en fondo verde,* con un ***área de selección y comandos*** en fondo gris, en lado derecho de dicha página.

En esta pestaña se podra observar como funciona la simulación en tiempo real, al igual que sus resultados.



- La cuadrícula sobre el área con fondo verde se puede mostrar u ocultar según se explica en el siguiente cuadro.
- Las líneas de color negro representan las paredes (objetos fijos) dentro del escenario.
- Las líneas de color naranja corresponden a puertas delizantes o plataformas móviles
- El círculo de color rojo es el Softbot u origen de la búsqueda.
- El círculo de color verde es la meta.

- Cada UMR (cuadro) de color azul claro, representa un agente inteligente auxiliar.
- Los números ***tipo LED*** ubicados en la parte inferior muestran las coordenadas cartesianas del objeto señalado con el ratón del computador.
- ***Inicio*** y ***Fin*** corresponden al tiempo transcurrido para cada búsqueda.

A.5 Área de selección y comandos

A continuación se explica el contenido del *área de selección y comandos* ubicada en el margen derecho del display:

	Sofbot, Meta, Señalador, Pared, Agente en su posición por defecto.
	Selector de objetos para construir el escenario.
	Tamaño de un objeto mueble. (Se activa una vez seleccionado el objeto apropiado)
	Visualiza o esconde la cuadrícula que conforma el escenario en el display.
	Huellas permite seleccionar Todas, Mejor o Ninguna. Todas: muestra un set completo de huellas. Mejor: sólo muetra la mejor de las huellas en cada paso. Ninguna: no muestra huellas.
	Sin sensores: realiza una búsqueda 100% virtual. (Se conoce el escenario). ***Con sensores***: realiza una búsqueda con sensores. (Se realiza levantamiento del escenario a medida que el sofbot lo recorre). ***Multidireccional:*** realiza una búsqueda con sensores con la ayuda de los agentes inteligentes auxiliares. ***Registrarse***: para aplicaciones con más de un procesador, permite que un agente sea identificado por el agente coordinador.
	Sensor_1 y Sensor_2 corresponden a la activación / desactivación de los sensores de proximidad y temperatura, respectivamente.
	El LED ***Display*** parpadea cuando el simulador esta en ejecución.

Podrá trazar objetos fijos (***Pared***) y objetos móviles (***Puerta***) se hace clic en el *punto de inicio* y doble clic en el *punto final* con el mause. Puede trazar líneas verticales en cualquier sentido (de arriba hacia abajo o de abajo hacia arriba).

Tanto el ***origen/softbot*** como la ***meta*** deben ser descargados con un ***doble clic*** en la celda deseada. La cantidad de agentes auxiliares disponibles (Agente_02, Agente_03, etc.), dependera de la cantidad de nodos auxiliares seleccionados en el campo "***Nodos Aux***" (Área de selección principal).

Selector de objetos:

Usted puede seleccionar cualquiera de los objetos mostrados a la izquierda.

Una vez seleccionado, aparecerá justo en la punta de flecha señaladora del cursor. Ubíque el objeto dentro del área del escenario y haga ***doble clic*** en el punto donde desee descargar el objeto.

A.6 Corriendo la simulación

- Una vez Usted haya definido el escenario y ubicado el origen (Softbot) y la meta puede proceder a ejecutar la simulación deseada: **SIN_SENSORES** o **CON_SENSORES**.

- Para ejecutar la búsqueda **MULTIDIRECCIONAL**, aparte de definir el escenario, ubicar origen (Softbot) y meta, deberá seleccionar el número de agentes auxiliares en el área de selección principal (Nodos Aux).

Terminada cualquier simulación se debe salir al menú principal y reingresar al simulador si es que se desea realizar una nueva simulación.

B. GLOSARIO DE TÉRMINOS

Agente de software: Aplicación informática con capacidad de *decidir* cómo debe actuar para alcanzar sus objetivos.

Agente Inteligente: Agente de software que puede funcionar fiablemente en un entorno rápidamente cambiante e impredecible.

Algoritmo: Un algoritmo es un procedimiento computacional bien definido que transforma una entrada de datos en una salida.

Archivo: Espacio que se reserva en el dispositivo de memoria de un computador para almacenar porciones de información que tienen la misma estructura y que pueden manejarse mediante una instrucción única.

Brecha Digital: Hace referencia a la diferencia socioeconómica entre aquellas comunidades que tienen accesibilidad a Internet y aquellas que no, aunque tales desigualdades también se pueden referir a todas las nuevas tecnologías de la información y la comunicación (TIC), como el computador personal, la telefonía móvil, la banda ancha y otros dispositivos.

Carácter: Estilo o forma de los signos de la escritura o de los tipos de la imprenta.

Digitalizar: Expresar datos en forma digital.

Display: Dispositivo de ciertos aparatos electrónicos, como los teléfonos y las calculadoras, destinado a la representación visual de información.

Drivers: Son programas informáticos que permiten a un sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz (posiblemente estandarizada) para usarlo.

Emular: Imitar las acciones de otro procurando igualarlas e incluso excederlas.

Escenario: Lugar en que ocurre o se desarrolla un suceso.

Hardware: Conjunto de los componentes que integran la parte material de una computadora.

Heurística: En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

Huella: Rastro, seña, vestigio que deja alguien o algo.

Inteligencia Artificial (IA): Es una rama de la Informática que pretende desarrollar programas en los que el ordenador desarrolle conductas típicas de los seres inteligentes.

Interfaz: Conexión física y funcional entre dos aparatos o sistemas independientes.

Percibir: Recibir por uno de los sentidos las imágenes, impresiones o sensaciones externas.

Redes neuronales: Consisten en una simulación de las propiedades observadas en los sistemas neuronales biológicos a través de modelos matemáticos recreados mediante mecanismos artificiales (como un circuito integrado, un ordenador o un conjunto de válvulas).

Sensor: Dispositivo que detecta una determinada acción externa, temperatura, presión, etc., y la transmite adecuadamente.

Simular: Representar algo, fingiendo o imitando lo que no es.

Softbots: (Robots de Software) Son agentes inteligentes que hacen uso de herramientas y utilidades para cumplir una tarea específica.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Tabla de datos: Es un cuadro que consiste en la disposición conjunta, ordenada y normalmente totalizada, de las sumas o frecuencias totales obtenidos en la tabulación de los datos, referentes a las categorías o dimensiones de una variable o de varias variables relacionadas entre sí.

C. ARTÍCULOS PUBLICADOS

C.1 Publicación No_1

González, A., Cano, H. B. y Chaves J. A., "*Integración de componentes de hardware y software en la implementación de prototipos para telemedicina".* Scientia et Technica. Año XIV, No 38, (2008). Universidad Tecnológica de Pereira.

C.2 Publicación No_2

González, A., Cano, H. B. y Ardila, W., "*Desarrollo de un simulador de escenarios para la búsqueda de rutas en robótica móvil"*. Scientia et Technica. Año XV, No 41, (2009). Universidad Tecnológica de Pereira.

C.3 Publicación No_3

González, A., Cano, H. B. y Castro, O. E., "*Adaptación de sensores virtuales para distancia en un simulador de escenarios para la búsqueda de rutas en robótica móvil".* Scientia et Technica Año XV, No 47, (2011). Universidad Tecnológica de Pereira.