UNIVERSIDAD CATÓLICA ANDRÉS BELLO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Simulación del mecanismo de búsqueda y recolección de alimento de las colonias de hormigas utilizando Algoritmos Evolutivos y Bioinspirados con enfoque multiagente

TRABAJO ESPECIAL DE GRADO
presentado ante la
UNIVERSIDAD CATÓLICA ANDRÉS BELLO
Como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR Juan Carlos Arvelo Flores
Ricardo Casanova Nébola-Kopak

PROFESOR GUIA Dr. Wilmer Pereira

FECHA Caracas, 29 de Noviembre de 2005

INDICE DE CONTENIDO

SINOPSIS 1
CAPÍTULO I 2
INTRODUCCIÓN 2
Planteamiento del Problema 2
Objetivo General 3
Objetivos Específicos 3
Justificación 4
Limitaciones y Alcances 5
CAPÍTULO II 7
MARCO TEÓRICO 7
ROBÓTICA 7
LEGO y la Inteligencia Artificial 10
RIS (Robotics Invention System) 11
RCX (Robotic Command eXplorer) 12
INTELIGENCIA ARTIFICIAL 15
Objetivos por alcanzar en la IA 17
HORMIGAS 18
Comunicación y Comportamiento 20
COMPUTACIÓN EVOLUTIVA 21
Origen 21
Paradigmas de la Computación Evolutiva 23
La Computación Evolutiva y las tendencias actuales 24
SISTEMA MONOAGENTE 25
SISTEMA MULTIAGENTE 26

CAPÍTULO III 28
MARCO REFERENCIAL 28
Aprendizaje 28
Refuerzo 28
Aprendizaje por Refuerzo 28
Programación Dinámica 31
Algoritmo Value Iteration 33
Algoritmo Q-Learning 34
Algoritmo Ant Learning 35
CAPÍTULO IV 36
MARCO METODOLÓGICO 36
Descripción detallada de las etapas 37
¿Por qué la metodología de Cascada con Retorno? 40
CAPÍTULO V 41
DESARROLLO 41
Diseño y construcción de los robots y el ambiente controlado 41
AMBIENTE CONTROLADO (MALLA DE NAVEGACIÓN) Y ROBOTS .. 41
Iteración 1 41
Iteración 2 51
Iteración 3 53
SIMULADOR 54
Iteración 1 54
Iteración 2 58
CAPÍTULO VI 61
RESULTADOS 61
CONCLUSIONES 73
RECOMENDACIONES 77
BIBIOGRAFÍA 78

INDICE DE FIGURAS

Figura 1. Imagen de robot. 8
Figura 2. Robot de LEGO. 10
Figura 3. Kit del RIS. 11
Figura 4. Modelo de robot construido con el RIS. 12
Figura 5. Arquitectura interna del RCX. 13
Figura 6. El RCX y sus accesorios. 14
Figura 7. La torre infrarroja. 14
Figura 8. Imagen de una hormiga. 19
Figura 9. Esquema de un Sistema Monoagente. 25
Figura 10. Modelo de Cascada con retorno. 37
Figura 11. Modelo de malla de navegación. 43
Figura 12. Transmisión RCX->Torre infrarroja. 47
Figura 13. Opción para la configuración del máximo rango de transmisón infrarroja (en el PC). 47
Figura 14. Propiedades del Proyecto. 59
Figura 15. Comparación de las velocidades de aprendizaje con respecto al tiempo. . 63
Figura 16. Comparación de las velocidades de aprendizaje con respecto a la cantidad de movimientos. 64

INDICE DE TABLAS

Tabla 1. Definiciones de Inteligencia Artificial. 16
Tabla 2. Comparación entre los paradigmas de la Computación Evolutiva. 23
Tabla 3. Relación entre aprendizaje a nivel Psicológico e Inteligencia Artificial. 28
Tabla 4. Especificaciones del paquete que transmite / recibe cada RCX-Torre IR 49
Tabla 5. Simulación de una hormiga con un nivel de aprendizaje rápido. 63
Tabla 6. Simulación de una hormiga con un nivel de aprendizaje medio. 63
Tabla 7. Simulación de una hormiga con un nivel de aprendizaje lento. 63
Tabla 8. Simulación de tres hormigas reduciendo 1 unidad de feromona. 65
Tabla 9. Simulación de tres hormigas reduciendo 2 unidades de feromona. 65
Tabla 10. Simulación de tres hormigas reduciendo 5 unidades de feromona. 66
Tabla 11. Resultados de la simulación de tres hormigas. 66
Tabla 12. Simulación del proceso de búsquela de alimento con 5 hormigas 67
Tabla 13. Comparación de la simulación empleando el tiempo como medida de referencia. 68
Tabla 14. Resultados de la comparación de la simulación empleando el tiempo como medida de referencia. 68
Tabla 15. Comparación de la simulación empleando la cantidad de movimientos como medida de referencia. 69
Tabla 16. Resultados de la comparación de la simulación empleando la cantidad de movimientos como medida de referencia. 70
Tabla 17. Resultados de las simulaciones para determinar cuan cerca están los movimientos de la ruta óptima. 71
Tabla 18. Simulación del proceso de búsquela de alimento con 7 hormigas 71
Tabla 19. Resultados de la simulación del proceso de búsquela de alimento con 7 hormigas. 72

SINOPSIS

Este trabajo se centra en la simulación del mecanismo de búsqueda y recolección de alimento de las colonias de hormigas. Para dicha simulación se construyó un ambiente real utilizando robots y un software de PC. Para la construcción de los robots se utilizó el kit LEGO MINDSTORMS, el cual resultó ser de gran ayuda, ya que permite de manera rápida y sencilla elaborar un robot capaz de adaptarse al problema que se desea resolver. El software de PC fue desarrollado en Visual Basic, en el cual es posible modificar parámetros como el nivel de aprendizaje, la tasa de evaporación (tiempo y cantidad) y el número de hormigas que interactúan en el ambiente.

En ambos simuladores, los entes debían aprender a explorar su ambiente en búsqueda de alimento, tal como lo hacen las hormigas en el mundo real. Para dicho aprendizaje se seleccionó el algoritmo Ant-Learning, el cual es de elaboración propia.

A lo largo del desarrollo de este proyecto se estudiaron distintas ciencias, las cuales se encuentran muy relacionadas. Particularmente, la robótica y la inteligencia artificial encuentran bastante aplicación en el campo de la biología; específicamente en el comportamiento animal.

En este Trabajo Especial de Grado se pudo demostrar que desarrollar actividades en forma grupal puede ser mucho más efectivo que trabajar de manera individual. A nivel organizacional esto resulta de gran ayuda, pues permite alcanzar los objetivos planteados obteniendo el mayor provecho de cada uno de los recursos de los que se disponga.

CAPÍTULO I
INTRODUCCIÓN

Planteamiento del Problema

Las hormigas siempre han sido objeto de diferentes estudios científicos, ya que resulta fascinante que criaturas que individualmente pueden resultar poco inteligentes sean capaces de desarrollar comportamientos grupales tan sofisticados y complejos. Uno de los aspectos más sorprendentes de estos insectos es su forma de comunicación, que les permite actuar coordinadamente ante una enorme variedad de sucesos y necesidades.

En el área de la Computación Evolutiva se hace uso de modelos computacionales de procesos naturales de evolución como base en el diseño e implementación de algoritmos para la resolución de problemas. Dichos algoritmos reciben el nombre de Algoritmos Evolutivos y tienen como característica común su inspiración en la simulación de la evolución de poblaciones de individuos a través de procesos de selección y reproducción.

Recientemente ha surgido otro conjunto de propuestas inspiradas en modelos biológicos, tales como Algoritmos de Optimización basados en colonias de hormigas y otros insectos, que se encuadran en lo que se ha denominado Algoritmos Bioinspirados.

La idea de este proyecto es explorar diferentes modelos computacionales a fin de hallar la mejor técnica que permita emular el mecanismo de búsqueda y recolección de alimento de las colonias de hormigas utilizando algoritmos de aprendizaje.

Objetivo General

Hallar la mejor técnica que permita emular el mecanismo de búsqueda y recolección de alimento de las colonias de hormigas utilizando algoritmos de aprendizaje.

Objetivos Específicos

- Estudiar el funcionamiento de los Algoritmos Evolutivos y Bioinspirados.
- Emular el mecanismo de comunicación, búsqueda y recolección de alimento de las hormigas.
- Diseñar e implementar un algoritmo en lenguaje Java que utilice información del comportamiento de las hormigas al momento de buscar y recolectar alimento para emularlo con robots LEGO MINDSTORMS Robotics Invention System 2.0.
- Diseñar un software con una sencilla interfaz gráfica para PC, a fin de simular el comportamiento de los robots.
- Hacer pruebas preliminares del funcionamiento del algoritmo.
- Realizar estudios comparativos entre los algoritmos de aprendizaje y el mecanismo de búsqueda y recolección de alimento de las hormigas, a fin de hallar el algoritmo que garantice la mayor probabilidad de éxito y supervivencia de la colonia.

Justificación

La utilización de la Robótica y la Inteligencia Artificial (IA) para emular comportamientos animales resulta de suma importancia, ya que permite realizar estudios y avances en diferentes áreas de la ciencia.

¿Pero por qué centrarse en colonias de hormigas?. La hormiga como ente individual parece poco inteligente, pero es capaz de desarrollar comportamientos grupales realmente complejos. Trasladándonos al área computacional cada "hormiga" (robot) puede ser programada con relativa facilidad y en conjunto estos seres deberían ser capaces de desarrollar tareas complejas. En principio este trabajo se basa en el mecanismo de búsqueda y recolección de alimentos de las colonias de hormigas, pero puede ser utilizado por ejemplo para el desarrollo de proyectos de exploración con ayuda de robots a lugares nunca antes visitados por el hombre. Estos robots no sólo podrían buscar alimento, sino cualquier tipo de material y además evitarían poner en riesgo vidas humanas.

Los robots son utilizados en una diversidad de aplicaciones, desde robots soldadores en la industria automotriz, hasta brazos teleoperados en el transbordador espacial.

Cada robot lleva consigo su problemática y sus soluciones afines; no obstante mucha gente considera que la automatización de procesos a través de entes mecánicos (robots) está en sus inicios. Por ejemplo, es un hecho que la introducción de la tecnología robótica en la industria, ya ha causado un gran impacto.

Es necesario hacer mención de los problemas de tipo económico, social e incluso político, que puede generar una mala orientación de la robótica. Se hace indispensable que la planificación de los recursos humanos, tecnológicos y financieros se realice de una manera inteligente.

Los robots pueden ser utilizados en una gran cantidad de procesos, entre los cuales podemos mencionar: los laboratorios, donde llevan a cabo con efectividad

tareas repetitivas como la colocación de tubos de pruebas dentro de los instrumentos de medición, aumentando la productividad, mejorando el control de calidad y reduciendo la exposición del ser humano a sustancias nocivas; en la industria nuclear, con el desarrollo de teleoperadores para manejar material radiactivo; en la exploración espacial, con el diseño, construcción y desarrollo de vehículos autónomos, capaces de llevar a bordo complejos sistemas y sofisticadas cámaras que permitan la exploración; en la educación, para la enseñanza de la programación; etc.

Así mismo, se espera que desde el punto de vista sociológico este proyecto sirva de aporte a los estudios de carácter organizacional no sólo de animales sino de cualquier sociedad a nivel general.

Por otro lado, el enfoque multiagente permite la reducción de costos, ya que en un ambiente netamente centralizado al producirse una falla se corre el riesgo de que el sistema deje de funcionar, mientras que en un sistema multiagente si una unidad (un robot en este caso) dejara de operar por alguna razón, el sistema en general no se vería afectado y seguiría funcionando.

Limitaciones y Alcances

Limitaciones:

- La programación de los robots para que realicen las actividades será hecha sólo en lenguaje Java.
- El RCX (Robotic Command eXplorer) posee una capacidad de memoria de 64 KB. Una vez cargada la máquina virtual de Java en el RCX (ocupa 32 KB), sólo se dispondrán de 32 KB para almacenar la aplicación desarrollada.
- Se trabajará con la cantidad mínima disponible de robots debido a que el grupo de robótica de la universidad no dispone de un gran número de ellos y resulta costosa su adquisición.

- El software desarrollado para la simulación de los algoritmos dispondrá de una interfaz gráfica muy simple.

Alcances:

- Construcción de al menos tres robots que permitan emular el mecanismo de búsqueda y recolección de alimento de las colonias de hormigas.
- Programación del algoritmo que garantice la mayor probabilidad de éxito y supervivencia de la colonia.
- Simulación en la computadora del algoritmo de aprendizaje programado.
- Simulación en los robots del algoritmo más eficiente obtenido del análisis comparativo, basado en la simulación hecha en la computadora.

CAPÍTULO II
MARCO TEÓRICO

ROBÓTICA

El término robot procede de la palabra checa robota, que significa "trabajador", "siervo". Fue empleado por primera vez en la obra teatral de 1920 R.U.R. (Robots Universales de Rossum) por el novelista y dramaturgo checo Karel Capek. Desde entonces se ha empleado la palabra robot para referirse a una máquina que realiza trabajos para ayudar a las personas o efectúa tareas difíciles para los humanos.

El *Robot Institute of America* define un robot como un manipulador programable capaz de realizar diversas funciones diseñado para desplazar materiales, partes, herramientas o determinados artefactos mediante movimientos programados variables y cuyo objetivo es la realización de ciertas tareas. Russell y Norvig lo definen simplemente como: "un agente artificial, activo, cuyo entorno es el mundo físico. El término activo descarta de esta definición a las piedras; el término artificial descarta a los animales; y el término físico descarta a los agentes de software puros o softbots, cuyo entorno lo constituyen los sistemas de archivos, bases de datos y redes de cómputo" [21].

Actualmente, el término se aplica a todos aquellos ingenios mecánicos, accionados y controlados de manera electrónica, capaces de llevar a cabo secuencias simples que permitan realizar operaciones básicas como: carga y descarga de materiales, operaciones de ensamblaje y soldadura, entre otras.

La robótica es, por lo tanto, la ciencia o la rama de la ciencia que se ocupa del estudio, desarrollo y aplicaciones de los robots. Otra definición de robótica es el diseño, fabricación y utilización de máquinas automáticas programables con el fin de realizar tareas repetitivas.

Básicamente, la robótica se ocupa de todo lo concerniente a los robots, lo cual incluye el control de los motores, neumáticos, sensores, sistemas de cómputos, etc. En la robótica se aúnan varias disciplinas diferentes, como la Mecánica, la Electrónica, la Informática, etc.

Figura 1. Imagen de robot.

Fuente: http://www.chem.sunysb.edu/msl/LEGO/8852b.jpg

El término robótica se le atribuye a Isaac Asimov. Los tres principios o leyes de la robótica según Asimov son:

- Un robot no puede lastimar ni permitir que sea lastimado por ningún ser humano.
- El robot debe obedecer a todas las órdenes de los humanos, excepto las que contradigan la primera ley.
- El robot debe autoprotegerse, salvo que para hacerlo entre en conflicto con la primera o segunda ley.

Hoy en día, este campo se dirige a la consecución de máquinas y dispositivos que sepan interactuar con el medio en el cual se desenvuelven; a estos robots capaces de tomar decisiones propias, tomando como guía la retroalimentación que les proporcionan sus respectivos sensores físicos, se les denomina robots autónomos.

Los robots, según su grado de complejidad pueden clasificarse en:

- **Robots de Primera Generación:** Repiten la tarea programada secuencialmente. No toman en cuenta las posibles alteraciones de su entorno.
- **Robots de Segunda Generación:** Adquieren información limitada de su entorno y actúan en consecuencia. Pueden localizar, clasificar (visión) y detectar esfuerzos y adaptar sus movimientos en consecuencia.
- **Robots de Tercera Generación:** Su programación se realiza mediante el empleo de un lenguaje natural. Poseen la capacidad para la planificación automática de sus tareas.

Desafortunadamente, el comportamiento del robot puede verse afectado por características ajenas, como por ejemplo: lecturas erróneas de los sensores, derrape de las ruedas, agotamiento de las baterías, rotura de alguna de las partes del robot, etc.

El beneficio que los robots generan es increíble para los trabajadores, industrias y países. Claro está, estos beneficios dependerán de la correcta utilización e implementación de los mismos, es decir, se deben emplear en las labores adecuadas, por ejemplo manipulando objetos muy pesados, sustancias peligrosas o bien trabajando en situaciones extremas o dañinas para el hombre. La idea es dejar más bien a los seres humanos desempeñar las labores y/o tareas de ingenieros, programadores, supervisores, etc.

¿Pero cuáles son esos beneficios?. Podemos destacar el mejoramiento en el manejo, control y productividad de las actividades laborales, todo esto asociado a una mejora significativa en cuanto a la calidad del producto final. Al ser los robots máquinas que pueden trabajar día y noche, sin perder un ápice de su desempeño, reducen los costos de producción.

LEGO y la Inteligencia Artificial

La compañía LEGO es una organización que se ha comprometido en el proyecto de la Inteligencia Artificial (IA), aplicándola en sus productos con el fin de revolucionar el juguete y las formas de entretenimiento de los niños y jóvenes de la nueva era.

Se podría decir que existen diferentes eras del juguete de LEGO. La primera consistía en la experiencia de la construcción. Luego aparecieron los neumáticos y pequeños motores que añadieron movimiento a los juguetes. En la tercera era los juegos de rol y los productos temáticos hicieron su aparición. Ahora, una cuarta etapa ha comenzado, donde la inteligencia y el comportamiento se han convertido en una parte integral de los productos de la compañía. LEGO constituyó la empresa LEGO MINDSTORMS, una división de la compañía que tiene como fin unir el mundo de las computadoras con la IA y los juguetes de construcción. Los productos de LEGO permiten a los niños diseñar y construir sus propios robots, y además, darles vida a través de una PC. A pesar de que la compañía recomienda el uso del kit para niños a partir de los 12 años de edad, el promedio de edades de las personas que lo usan está alrededor de los 30 años.

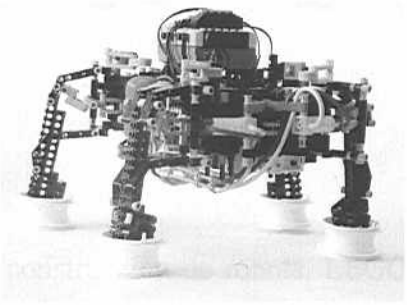


Figura 2. Robot de LEGO.

Fuente: http://www.yellow-springs.k12.oh.us/ys-mls/_borders/LEGO_robot.jpg

RIS (Robotics Invention System)

El RIS (Robotics Invention System) es un kit con 717 piezas de LEGO que permite la construcción y la programación de una variedad de robots móviles programables con una PC.

Hay quien afirma que LEGO MINDSTORMS constituye uno de los primeros intentos por difundir los robots a gran escala, equivalente a lo que fue la familia de los ZX en su tiempo para el mundo de las computadoras.

A diferencia de otros kits para la construcción de robots, el kit de LEGO es un producto de consumo, que permite construir desde un brazo hasta un robot recolector sin necesitar una sola soldadura. El RIS 2.0 incorpora una microcomputadora (RCX), un sensor de luz y tacto, dos motores y un transmisor infrarrojos, permitiéndole a los robots construidos percibir e interactuar con el ambiente que les rodea.

Figura 3. Kit del RIS.
Fuente: Elaboración propia

Para el diseño y la construcción de robots, LEGO propone una metodología de cinco fases, las cuales son:

- **Lluvia de ideas (mindstorm)**: Consiste en una etapa de diseño, de visualización de las funcionalidades y apariencia del robot.

- **Construcción**: Utilizando el RIS, se procede al ensamblaje de robot. Para ello se emplean piezas de lego, sensores de luz, de contacto, motores, etc.
- **Programación**: En esta etapa y utilizando una PC, se procede a la programación de la lógica que determinará el comportamiento del robot. Es justo aquí donde el robot es dotado de un "cerebro".
- **Descarga**: Una vez construido el robot y programada la lógica para su comportamiento, se procede a transferir dicha lógica de la PC al robot, utilizando para ello la torre infrarroja que viene junto con el kit.
- **Activación**: En este punto se verifica que la lógica diseñada corresponda con el comportamiento esperado del robot.

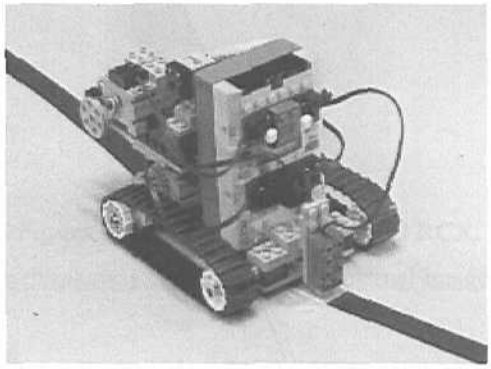


Figura 4. Modelo de robot construido con el RIS.
Fuente: http://www.vahal.cz/lego/assets/images/rcx.jpg

RCX (Robotic Command eXplorer)

El componente RCX funciona como el cerebro del robot. Lo constituyen los siguientes componentes:

- **Unidad Central de Proceso**: El RCX posee un microcontrolador de la serie HITACHI H8-3292, que trabaja a una velocidad comprendida entre los 10 y 16 MHz, suficiente para permitirle al robot interactuar con el medio que lo rodea.

- **Memoria ROM**: El microcontrolador contiene 18 KB (kilobytes) de memoria ROM y 512 bytes de memoria de acceso aleatorio (RAM). El ROM contiene algoritmos para descargar el firmware, desplegar datos en la pantalla de cristal líquido (LCD) y establecer comunicación con los motores y sensores.
- **RAM**: El RCX contiene 32 KB de memoria RAM. Dos componentes son almacenados en esta memoria: el firmware y los programas de usuario.

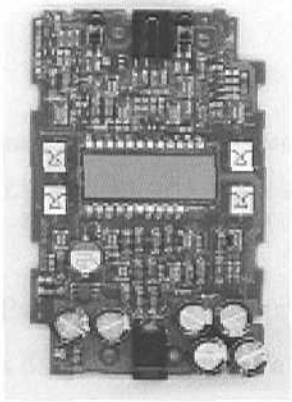


Figura 5. Arquitectura interna del RCX.

Fuente: http://www.shirakawas.com/mindstorms/internal/image/rcx_omote300.jpg

- **Puertos de entrada/salida**: El RCX contiene 3 puertos de entrada que pueden aceptar cualquier variedad de sensores LEGO como sensores de choque, de luz, de temperatura y rotación. También posee tres puertos de salida utilizados para controlar las acciones del robot frente a su medio ambiente. Por lo general los puertos de salida son utilizados para controlar motores.
- **Puerto de Comunicación Infrarrojo**: El puerto infrarrojo permite la descarga en el RCX de las aplicaciones diseñadas. El mismo permite la comunicación en ambos sentidos, es decir, el RCX tiene la capacidad de transmitir y recibir datos desde y hacia el PC. También puede ser utilizado para comunicar a dos robots entre ellos sin la intervención de otro agente.

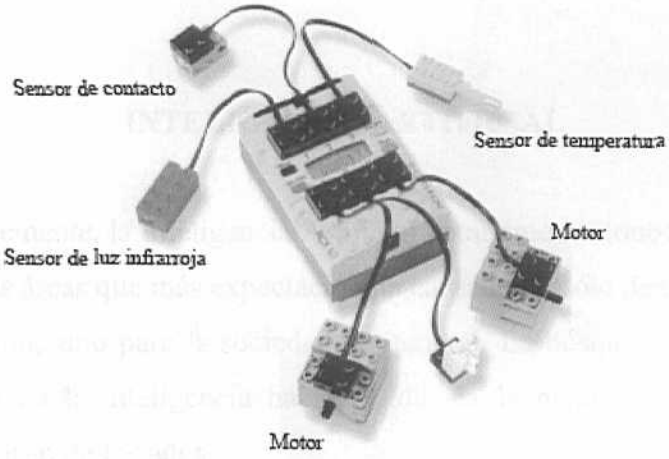


Figura 6. El RCX y sus accesorios.

Fuente: http://www.kvartercentret.nv.kk.dk/Datavaerksted/Robotvaerksted/images/RCX'en.jpg con modificaciones propias

- **Torre Infrarroja**: Para transmitir los programas desde la PC hacia el RCX y viceversa, los datos son enviados a través de una señal infrarroja.

Figura 7. La torre infrarroja.

Fuente: http://artdecom.mesh.de/images/bilder/ir-tower2.jpg

INTELIGENCIA ARTIFICIAL

Indudablemente, la Inteligencia Artificial comúnmente conocida por sus siglas IA, es una de las áreas que más expectación ha causado no sólo dentro de las ciencias de la computación, sino para la sociedad en general. La búsqueda para comprender los mecanismos de la inteligencia ha resultado ser la piedra filosofal de muchos científicos a lo largo de los años.

De las principales áreas de la Inteligencia Artificial, una de las que ha atraído más investigadores ha sido el aprendizaje de máquina. El aprendizaje puede definirse como la construcción o modificación de representaciones de lo que se experimenta. Esta definición se concentra en una representación interna que el sistema de aprendizaje construye (ya sea una persona o un programa de computadora) y modifica con base en su entorno (típicamente, lo que el sistema de aprendizaje experimenta es su entorno, o al menos cierta parte de él).

Indudablemente, resulta vital en el proceso de emular comportamientos inteligentes, que un sistema pueda mejorar su comportamiento sobre la base de la experiencia que colecte al efectuar la misma tarea en repetidas ocasiones, y que además desarrolle algún tipo de noción de lo que es un error, y cómo evitarlo.

No existe una definición exacta de lo que es la Inteligencia Artificial. De acuerdo con varios autores, estas definiciones varían en torno a las siguientes dimensiones: las que se refieren a procesos mentales y al razonamiento, las que aluden a la conducta, las que miden la condición deseable en función de eficiencia humana, y finalmente las que lo hacen de conformidad con un concepto de inteligencia humana, denominado racionalidad.

La siguiente tabla muestra definiciones de lo que es Inteligencia Artificial, de acuerdo a varios autores.

Procesos mentales y razonamiento	Sistemas que piensan como humanos	Sistemas que piensan racionalmente
	"La interesante tarea de lograr que las computadoras piensen...máquinas con mente, en su amplio sentido literal." (Haugeland, 1985) "La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades tales como toma de decisiones, resolución de problemas, aprendizaje..." (Bellman,1978)	"El estudio de las facultades mentales mediante el uso de modelos computacionales." (Charniak y McDermott, 1985) "El estudio de los cálculos que permiten percibir, razonar y actuar." (Winston,1992)
Conducta	Sistemas que actúan como humanos	Sistemas que actúan racionalmente
	"El arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia."(Kurzweil,1990) "El estudio de cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor." (Rich y Knight, 1991)	"Un campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales."(Schalkff, 1990) "La rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente." (Luger y Stubblefield, 1993)

Tabla 1. Definiciones de Inteligencia Artificial.

Fuente: Russel, Stuart. Inteligencia Artificial. Un Enfoque Moderno (1996)

La Inteligencia Artificial es un campo joven, heredera de diversas ideas, puntos de vista y técnicas de otras disciplinas. Ésta involucra fundamentos de filosofía, matemáticas, psicología, lingüística y ciencias de la computación.

En la actualidad, la Inteligencia Artificial empieza a extender sus áreas de investigación en diversas direcciones y trata de integrar diferentes métodos en sistemas a gran escala, en su afán por explotar al máximo las ventajas de cada esquema.

Adicionalmente a la Robótica, la Inteligencia Artificial puede ser aplicada a disciplinas como: Lingüística Computacional, Minería de Datos (*Data Mining*), Procesamiento del Lenguaje Natural (*Natural Language Processing*), Sistemas de Apoyo a la Decisión (*Decision Support System*), Videojuegos, Reconocimiento de Patrones, Sistemas Expertos, etc.

Objetivos por alcanzar en la IA

En la Inteligencia Artificial existen cuatro posibles objetivos por alcanzar, a continuación se describen cada uno de ellos:

- **Sistemas que actúan como humanos (Prueba de Turing):** Alan Turing (1950) intentó ofrecer una definición de Inteligencia Artificial que se pudiese evaluar. Para ello, desarrolló la famosa prueba de Turing (test de Turing), la cual sugiere que para que un ser o máquina se considere inteligente debe lograr engañar a un evaluador. La prueba era exitosa si el evaluador era incapaz de determinar si una computadora o un humano era quien había respondido en el otro extremo del terminal.
- **Sistemas que piensan como humanos (Ciencia Cognitiva):** El hombre a través de la historia se ha atribuido a sí mismo el nombre de homo sapiens (hombre sabio) como una valoración de sus habilidades mentales. Los esfuerzos en el campo de la Inteligencia Artificial se enfocan a lograr la comprensión de entidades inteligentes. Por ello, una de sus principales razones de estudio es aprender más acerca de nosotros mismos. A diferencia de otras ciencias como la filosofía y la psicología, que igualmente se ocupan del estudio de la inteligencia, los esfuerzos de la Inteligencia Artificial se encuentran encaminados tanto a la construcción de entidades inteligentes como a su comprensión.

  Para poder afirmar que un determinado programa actúa como un ser humano, es necesario que éste tenga la habilidad de transformar los datos (entrada/salida) en información, y lo más importante, que sea capaz de reflejar la conducta humana.
- **Sistemas que piensan racionalmente (Lógica):** La evolución de la lógica está intrínsecamente ligada a la evolución intelectual del ser humano, ya que como ciencia del razonamiento, su historia representa la historia misma del hombre. La lógica surge desde el primer momento en que el

hombre, al enfrentar a la naturaleza, infiere, deduce y razona, con el ánimo de entenderla y aprovecharla para su supervivencia [21].

Alrededor de 1965 ya existían programas que, con tiempo y memoria suficientes eran capaces de, a partir de una descripción de un problema, encontrarle solución (si existía).

La lógica permite representar el conocimiento y razonar. Los mayores éxitos de la Inteligencia Artificial han venido del uso y desarrollo de esta habilidad.

- **Sistemas que actúan racionalmente (Teoría de Decisiones):** Un agente racional es un ente capaz de percibir el entorno y actuar con base en sus creencias y objetivos. Todas las habilidades indicadas en la prueba de Turing son necesarias para actuar de forma racional. Por lo tanto, es de suma importancia la capacidad para representar el conocimiento y razonar, pues de esta manera se podrán tomar las decisiones correctas ante una situación determinada.

HORMIGAS

Para poder comprender el fenómeno social tanto en humanos como en animales, es necesario conocer los comportamientos sobre los cuales se basa la sociabilidad. No existe una sola forma de sociabilidad.

Un modelo para este tipo de problema lo constituyen las hormigas. Su gran variedad de especies (alrededor de 14.000), hacen de este grupo de insectos un modelo único para estudios sociobiológicos. No todas las especies de hormigas tienen el mismo tipo de organización social, por ejemplo: unas son "poligynicas" (más de una reina fértil) y otras son "monogynicas"; en unas especies las obreras son totalmente estériles, mientras que en otras las obreras ocasionalmente ponen huevos no fecundados de los cuales emergen machos, etc. Sin embargo, todas las hormigas son "eusociales", es decir:

- Existe cooperación en el cuidado de la cría entre diferentes individuos de la misma especie.
- Muestran una división del esfuerzo reproductivo: una o alguna de ellas en una colonia procrean, las demás lo hacen en menor grado o no del todo.
- Más de dos generaciones de hormigas conviven y colaboran entre sí en un mismo nido.

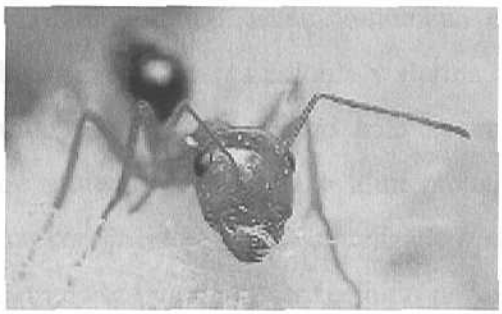


Figura 8. Imagen de una hormiga.

Fuente: http://www.alrfoto.com/gal71/gal08.htm

Las hormigas caminan por la Tierra hace aproximadamente cien millones de años. Están en casi todas partes: desiertos, selvas, montañas, etc. Juntas, pesan más que toda la Humanidad. Biológicamente hablando, forman una sociedad perfecta: conviven de todas las edades, desde huevos hasta adultos, se dividen el trabajo para llevar adelante la colonia y cuidan a sus crías. Algunas son granjeras, otras son agricultoras, otras guerreras, y hasta esclavizadoras. La mayoría son hembras.

Las hormigas son insectos exitosos desde la perspectiva de la evolución. Las diferentes especies construyen sus nidos bajo tierra, en árboles, entre hojas unidas. Incluso, algunas son nómadas, y forman nidos temporales. Su alimentación también es variada: las hay carnívoras y herbívoras; unas consumen hongos cultivados por ellas mismas y otras prefieren las excreciones azucaradas de ciertos pulgones.

Comunicación y Comportamiento

Uno de los aspectos más sorprendentes de las hormigas, y que les permite actuar coordinadamente ante una enorme variedad de sucesos y necesidades, es su forma de comunicación.

De los sistemas de comunicación conocidos en hormigas, la comunicación química parece ser la más importante. Ellas comunican calidad y localidad del alimento, territorialidad, pertenencia a una colonia y alarma, entre otras, con señales químicas llamadas feromonas. En este trabajo sólo se emplearon dos de ellas: feromona de búsqueda de alimento y feromona de alimento encontrado.

La feromona es un compuesto químico sintetizado y secretado por un animal, que éste difunde por la atmósfera y que al ser recibido por otro animal de la misma especie, a veces a kilómetros de distancia, provoca respuestas específicas que pueden ser fisiológicas o de comportamiento.

Son varios los sistemas de reclutamiento químico conocidos en hormigas. En los sistemas más primitivos, las hormigas reclutan a la compañera con señales táctiles y la guían físicamente a la fuente del alimento. En los sistemas más avanzados, la hormiga reclutadora, al encontrar el alimento, regresa al nido depositando feromonas sobre la tierra.

Resulta fascinante entender como animales casi ciegos, moviéndose aproximadamente al azar, pueden encontrar el camino desde su nido hasta la fuente de alimentos y regresar. La concentración de las feromonas va a depender de la cantidad y calidad del alimento y reclutará cantidades diferentes de hormigas según la concentración específica de feromona. En principio, una hormiga aislada se mueve esencialmente al azar, pero las siguientes deciden con una buena probabilidad seguir el camino con mayor cantidad de feromonas.

Las feromonas no son mensajes escritos, ya que desaparecen por evaporación. Si una hormiga quiere transmitir un mensaje por más tiempo, debe repetirlo, secretando constantemente la sustancia correspondiente.

COMPUTACIÓN EVOLUTIVA

Origen

En el mundo natural, los seres vivos constantemente se enfrentan a diversos problemas que deben resolver con éxito, desde situaciones tan simples como cazar un insecto hasta huir de un depredador para garantizar su supervivencia. La Computación Evolutiva se centra en visualizar la naturaleza como un gran ente, extrayendo de ella su capacidad de resolver problemas y enfocando dicha capacidad al mundo de la programación. ¿Pero de dónde proviene esta capacidad? Proviene de la evolución, producida por la selección natural [(1)], que favorece la supervivencia de los individuos más adaptados a su entorno.

Aunque desde 1930 se especuló acerca del potencial de la selección natural para resolver problemas de optimización y aprendizaje, no fue hasta 1960 que se establecieron las bases de los tres paradigmas que conforman la denominada Computación Evolutiva: *las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos*.

La Computación Evolutiva, como disciplina de la Inteligencia Artificial (IA), ha atraído la atención de un gran número de investigadores, tomando auge en los últimos veinte años del siglo XX. Ella ha dado pie a ciertos paradigmas, como técnicas híbridas que incrementan el poder de los algoritmos evolutivos o representan alternativas viables para resolver problemas de optimización o aprendizaje en ciertos dominios.

(1) Selección Natural: Teoría de la Selección Natural de Charles Darwin.

Bajo el término de Computación Evolutiva se engloba un amplio conjunto de técnicas de resolución de problemas complejos basados en la emulación de los procesos naturales de evolución. No es la primera vez que se hace uso de la emulación de procesos naturales para resolver problemas científicos y técnicos, entre los más conocidos podemos mencionar: las *Redes Neuronales* para la resolución de problemas de aprendizaje artificial y el *Simulated Annealing* para resolver problemas de optimización que se plantean en las ciencias físicas.

La principal aportación de la Computación Evolutiva a la metodología de resolución de problemas consiste en el uso de mecanismos de selección de soluciones potenciales y de construcción de nuevos candidatos por recombinación de características de otros ya presentes, de modo parecido a como ocurre en la evolución de los organismos naturales.

La Computación Evolutiva hace uso de modelos computacionales de procesos naturales de evolución como base en el diseño e implementación de algoritmos para la resolución de problemas. Existe una gran variedad de propuestas y estudio sobre estos modelos, los cuales reciben el nombre genérico de Algoritmos Evolutivos. Estos tienen como característica común su inspiración en la simulación de la evolución de poblaciones de individuos a través de procesos de selección, reproducción y mutación.

Recientemente han surgido otro conjunto de propuestas inspiradas en modelos biológicos, tales como Algoritmos de Optimización basados en colonias de hormigas, enjambres, etc., que se encuadran en lo que se ha denominado Algoritmos Bioinspirados.

Los Algoritmos Evolutivos y Bioinspirados (AEB) es una definición a un tipo particular de algoritmos, ellos permiten abordar la resolución de una gran variedad de problemas planteados como los relacionados con optimización y búsqueda en espacios complejos.

Paradigmas de la Computación Evolutiva

- **Estrategias Evolutivas:** Consisten en una serie de técnicas computacionales para resolver de forma eficiente determinados problemas continuos. Fueron desarrolladas por Ingo Rechenberg y Hans-Paul Schwefel en la década de los años setenta. Con el tiempo, se han convertido en uno de los algoritmos evolutivos más poderosos.
- **Programación Evolutiva:** Creada en la década los sesenta por L. J. Fogel, tiene como objetivo la creación de sistemas de aprendizaje basándose en el proceso de la evolución por selección natural. Involucra tres fases básicas: *selección*, *evaluación* y *recombinación*.
- **Algoritmos Genéticos:** Fueron desarrollados por John H. Holland en los años sesenta y su motivación inicial fue la de proponer un modelo general de proceso adaptable.

	Estrategia Evolutiva	Programación Evolutiva (PE)	Algoritmos Genéticos
Representación	Números reales	Números reales	Números binarios y reales
Auto adaptación	Desviaciones estándares	No (PE Estándar)	No
Mutación	Gaussiana, Operador principal	Gaussiana, Operador único	Inversión de bit, operador secundario
Selección	Determinística, extintiva o basada en preservación	Probabilística, extintiva	Probabilística, basada en preservación
Recombinación	Discreta (azar) Intermedia (promedio) Sexual (2 padres)	No	Uniforme, simple y múltiple Operador principal Sexual (2 Padres)

Tabla 2. Comparación entre los paradigmas de la Computación Evolutiva.

Fuente: Elaboración propia

La Computación Evolutiva y las tendencias actuales

Los tres paradigmas emergentes más recientes que suelen asociarse con la computación evolutiva son los siguientes:

- **Algoritmos Culturales**: Una teoría propuesta por los sociólogos sugiere que la cultura podría ser codificada de forma simbólica y transmitida dentro y entre poblaciones de individuos. Basándose en esta idea, a principios de los noventa se desarrolló un modelo computacional en el que la evolución de la cultura es vista como un proceso de herencia que opera a dos niveles: *microevolutivo* y *macroevolutivo*. En el nivel microevolutivo, los individuos pueden definirse en términos de *"características de comportamiento"*. Estas características se transmiten de generación en generación usando varios operadores inspirados en modelos sociológicos. A nivel macroevolutivo, los individuos pueden generar "mapas" o descripciones generalizadas de sus experiencias. Los mapas individuales pueden mezclarse y modificarse para formar "mapas de grupo", usando una serie de operadores especializados. Ambos niveles comparten un enlace de comunicación.

  La idea principal de los algoritmos culturales es preservar las creencias que son aceptables socialmente y desechar las que no lo son.

- **Colonias de Hormigas:** Este paradigma es sin lugar a dudas el más popular en estos últimos años. Consiste en el estudio del comportamiento de las hormigas sociales. El algoritmo principal es en realidad un sistema multiagente en el que las interacciones a bajo nivel entre agentes simples (denominados "hormigas") producen, en su conjunto, un comportamiento complejo, correspondiente a toda la colonia. Desde el punto de vista de la Inteligencia Artificial, las colonias de hormigas se basan en técnicas de búsqueda local con registro histórico de las rutas recorridas más y/o

menos promisorias. En combinación con algoritmos evolutivos y aprendizaje por refuerzo, esta técnica ha sido aplicada recientemente con gran éxito a problemas de diseño en ingeniería, a optimización combinatoria y a optimización no lineal en general.

- **Optimización de Enjambre de Partículas:** Esta técnica, al igual que las colonias de hormigas, está basada en los comportamientos sociales. Desarrollada a mediados de los noventa, es similar al algoritmo genético, ya que utiliza una población inicial de soluciones aleatorias. Sin embargo, a diferencia de éste, se le asigna una velocidad aleatoria a cada solución, y la población de soluciones posibles, a las que se denomina "partículas", las cuales se hacen "fluir" a través del espacio de búsqueda. En lugar de emplear los operadores genéticos tradicionales (selección, cruza sexual y mutación), cada partícula ajusta su trayectoria, de acuerdo a su propia experiencia y a la de sus compañeros.

SISTEMA MONOAGENTE

Es el sistema clásico de la programación, basado en un único ente que recibe un conjunto de entradas, las procesa, y genera una serie de salidas, como se muestra en la figura 9.

Figura 9. Esquema de un Sistema Monoagente.

Fuente: Elaboración propia

SISTEMA MULTIAGENTE

Los Sistemas Multiagentes (SMA) se han afianzado como una nueva aproximación para el desarrollo de sistemas que trascienden el ámbito de la Inteligencia Artificial. Se conciben actualmente como sistemas computacionales en los cuales varios agentes autónomos interactúan entre sí para colaborar en la solución de un conjunto de problemas o en la consecución de una serie de objetivos colectivos.

Estos agentes pueden ser dotados de una capacidad cognoscitiva, de conocimientos, de mecanismos de raciocinio, entre otros. Cada agente tiene un comportamiento puramente reactivo y aunque está desprovisto de inteligencia, emerge un comportamiento inteligente que resulta de la pertinencia de los intercambios entre agentes. Estos sistemas permiten, por ejemplo, modelar la construcción de un nido de termitas o comportamientos distribuidos para lograr un fin de modo colectivo.

Los agentes son sistemas computacionales que principalmente tienen dos capacidades importantes. La primera, es que tienen al menos alguna acción autómata para decidir ellos mismos lo que requieren hacer para cumplir sus objetivos. La segunda, es la capacidad de interactuar con otros agentes no simplemente para intercambiar datos, sino que se tienen analogías de tipo social, actividades que nosotros los humanos utilizamos a diario. La cooperación depende de las interacciones entre los agentes e incorpora tres elementos: la colaboración, la coordinación y la resolución de conflictos.

Los Sistemas Multiagentes encuentran aplicación en muchas áreas dentro y fuera de la informática, como por ejemplo: supervisión hospitalaria, asistencia financiera, manipulación y filtrado de información aplicados a diferentes dominios, agentes bancarios, difusión de noticias y publicidad, realidad virtual, control de procesos y manufactura, telecomunicaciones, ventas y comercio electrónico, etc.

El paradigma "multiagente" representa una nueva forma de ver los sistemas distribuidos y provee un camino para el desarrollo de aplicaciones inteligentes y más robustas.

CAPÍTULO III
MARCO REFERENCIAL

Aprendizaje

Conjunto de procesos que permiten cualquier cambio permanente del comportamiento o conducta de un animal, hombre o máquina como consecuencia de una experiencia o práctica previa.

Refuerzo

Estímulo que modifica un reflejo condicionado o una conducta aprendida, aumentando la probabilidad de que se produzca una cierta clase de respuesta (refuerzo positivo) o disminuyéndola (refuerzo negativo).

Psicología	Inteligencia Artificial
Condicionamiento Clásico	Aprendizaje basado en Ejemplos
Condicionamiento Operante	Aprendizaje por Refuerzo

Tabla 3. Relación entre aprendizaje a nivel Psicológico e Inteligencia Artificial.

Fuente: http://www.cs.us.es/~delia/sia/html98-99/pag-alumnos/web10/indice.html

Aprendizaje por Refuerzo

El Aprendizaje por Refuerzo existe desde hace varios años. Sus inicios se remontan a los primeros días de la Cibernética. Forma parte de otras disciplinas como: la psicología y la computación.

El objetivo del Aprendizaje por Refuerzo es usar el premio/castigo para aprender una función de agente satisfactoria, la cual permite tomar decisiones a partir de una percepción del entorno. Es complicado obtener una satisfactoria función de agente, ya que dicho agente nunca sabe las acciones que debe tomar para lograr ciertas recompensas.

Definimos al agente como el aprendiz encargado de observar su entorno para recoger información que le permita modificar su comportamiento para así aprender a resolver un determinado problema. Como se mencionó anteriormente, el objetivo fundamental del aprendizaje por refuerzo es la utilización de las recompensas para la obtención de una función de agente. Por tanto nuestro agente será una función que, recibiendo como entrada una percepción del entorno, devolverá la acción siguiente a realizar.

La función de agente utiliza la información almacenada para realizar la toma de decisiones, de ahí el nombre de Aprendizaje por Refuerzo. La principal característica de este tipo de aprendizaje es que ocurre en tiempo real, es decir, no existe separación entre la fase de aprendizaje y la fase de utilización.

Distintos autores han formulado el "Aprendizaje por Refuerzo" como una estrategia de aprendizaje donde no se necesita un conjunto de eventos evaluados. El sistema establece una relación entrada/salida a través de un proceso de éxito o fracaso, produciendo una señal (*señal de refuerzo*) que mide el buen funcionamiento del mismo.

Los pesos se ajustan con base en la señal de refuerzo dado un mecanismo de probabilidades. La idea es la siguiente: si la acción tomada por el sistema es seguida de un estado satisfactorio, entonces la tendencia es a producir un reforzamiento a esa acción; de forma contraria, la tendencia es a producir disminución a dicha acción.

En el modelo estándar de Aprendizaje por Refuerzo existen tres aspectos fundamentales a tomar en cuenta, ellos son: *el ambiente, la función de refuerzo y la función valor.*

- **El ambiente**: Todo sistema de aprendizaje adquiere conocimiento dado una serie de situaciones que han ocurrido con anterioridad y, basándose en

estas, ejecuta la acción o acciones que se consideren convenientes para un ambiente dinámico.

Cuando el sistema de aprendizaje por refuerzo puede observar toda la información, suministrada por lecturas de sensores, descripciones simbólicas, situaciones "mentales", etc.; el ambiente, entonces, puede influenciar en la decisión de cual acción debe tomarse en un instante dado. De producirse este fenómeno, se dice que el sistema escoge acciones basadas en "verdaderos" estados del ambiente. Este caso ideal es la mejor base posible para que ocurra un aprendizaje por refuerzo.

- **La función de refuerzo:** Los sistemas de aprendizaje por refuerzo tienen como base fundamental asociar situaciones/hechos previos con acciones a través de interacciones de ensayo y error bajo un ambiente dinámico.

  El objetivo de este tipo de sistema es el uso de la función refuerzo, la cual nos proporciona las acciones positivas que el agente busca maximizar. En otras palabras, existe una relación entre estado/acción y los refuerzos; tras realizarse una acción en un estado determinado, el agente recibirá algún refuerzo (recompensa o castigo), que vendrá dado por un valor escalar.

  El agente aprende a maximizar la suma de los refuerzos recibidos cuando va de un estado inicial a un estado final.

- **La función valor:** Esta función se refiere a la manera como el agente escoge "buenas" acciones, pero, ¿Cuál acción debería ser ejecutada en cada estado?. El valor de un estado es definido como la suma de los refuerzos recibidos siguiendo la política fijada para alcanzar el estado final.

  Por tal motivo, es complicado diseñar una estrategia o algoritmo que permita encontrar de manera eficiente la función valor óptima. Esta es una pregunta que se hacen los investigadores una vez que se ha determinado emplear aprendizaje por refuerzo para resolver problemas.

Programación Dinámica

Existen básicamente dos estrategias para resolver problemas de aprendizaje. Primero, el que consiste en buscar en el conjunto de comportamientos aquel que se desempeñe de manera correcta en el ambiente, particularmente los Algoritmos Genéticos siguen esta corriente. Segundo, aquel que consiste de técnicas estadísticas y métodos dinámicos de programación para poder determinar la utilidad y/o efectividad de ciertas acciones dentro del ambiente.

Los métodos más usados para resolver problemas de aprendizaje son los algoritmos por reforzamiento, destacando dos principalmente: *Value Iteration* y *Q-Learning*.

Comúnmente una serie de problemas pueden ser resueltos recursivamente utilizando términos matemáticos. Sin embargo, el tiempo de ejecución de la solución recursiva, frecuentemente de orden exponencial, lo hace impracticable. Por lo general, la manera de resolver esos problemas es dividiéndolos en pequeños subproblemas independientes. La eficiencia de la Programación Dinámica consiste en resolver los subproblemas una sola vez, guardando los resultados en una tabla para su posterior utilización.

La Programación Dinámica tiene mayor aplicación en la resolución de problemas de optimización. La solución de problemas mediante esta técnica se basa en un principio enunciado por Bellman en 1957, que dice: "En una secuencia de decisiones óptima toda subsecuencia ha de ser también óptima".

En líneas generales, el diseño de todo algoritmo de Programación Dinámica consta de los siguientes pasos:

- Planteamiento de la solución como una sucesión de decisiones y verificación de que está con el principio de óptimo (mencionado anteriormente).
- Definición recursiva de la solución.

- Cálculo del valor de la función óptima mediante una tabla donde se almacenan soluciones a problemas parciales para reutilizar los cálculos.
- Construcción de la solución óptima haciendo uso de la información contenida en la tabla anterior.

Como hemos mencionado anteriormente, el objetivo principal del aprendizaje por refuerzo es encontrar la relación entre los estados y sus respectivos valores. Una vez que se ha completado este proceso, la política óptima puede ser obtenida con relativa facilidad.

Para ello, es necesario incorporar ciertas notaciones: $V^*(x_t)$ es la función de valor óptima, donde x_t es el vector (estructura de datos) que contiene los estados; $V(x_t)$ es la aproximación de la función valor; γ es el factor de variación (valor escalar) en el rango [0,1] originado por el refuerzo; $e(x_t)$ representa el error en la aproximación del valor del estado ocupado en el instante de tiempo t.

A continuación se muestra la ecuación (1) que expresa la función valor óptima en un estado dado.

$$V(x_t) = e(x_t) + V^*(x_t) \qquad (1)$$

De forma similar a la ecuación anterior, la aproximación en el instante de tiempo $t+1$ se expresa de la siguiente manera:

$$V(x_{t+1}) = e(x_{t+1}) + V^*(x_{t+1}) \qquad (2)$$

Como se dijo previamente, la política óptima para el valor del estado x_t viene dada por la suma de los refuerzos, que comienza en el estado x_t hasta llegar al estado terminal. Por esta razón, existe una relación entre los valores de los sucesivos estados x_t y x_{t+1}.

Esta relación es expresada en la siguiente ecuación (3).

$$V^*(x_t) = r(x_t) + \gamma V^*(x_{t+1}) \qquad (3)$$

De manera análoga ocurre en la aproximación $V(x_t)$, como se muestra en la ecuación (4).

$$V(x_t) = r(x_t) + \gamma V(x_{t+1}) \quad (4)$$

El proceso de aprendizaje consiste de encontrar una solución a la ecuación (4) para todos los estados x_t.

Algoritmo Value Iteration

Value Iteration es una de las extensiones de la programación dinámica. La idea básica es calcular la utilidad de cada posible estado, y utilizar éstas para seleccionar la acción óptima de cada estado. El método converge cuando se alcanza una diferencia mínima (error) entre los valores de la iteración t respecto a la iteración t+1.

En cada iteración (t+1), se estima la utilidad da cada estado basada en los valores de la iteración anterior (t). Cuando $t \rightarrow \infty$, los valores de la utilidad convergen a un valor estable.

$$wt = max\ u\ (r(xt,u) + \gamma V(xt+1)) - V(xt) \quad (5)$$

En la ecuación anterior, "u" representa la acción realizada en el estado "x_t", causando una transición al estado "x_{t+1}". El par "$r(x_t,u)$" significa el refuerzo recibido por el estado "letra" al instante de desempeñar la acción "letra".

Para lograr obtener una actualización de este algoritmo, se debe encontrar primero la acción que devuelva su máximo. La única forma de lograr esto es ejecutar una acción y calcular la suma de los refuerzos recibidos y el valor aproximado del estado siguiente.

Algoritmo Q-Learning

Q-Learning es otra extensión a la programación dinámica. Este algoritmo asigna un refuerzo al hecho de tomar una acción en un estado dado, a estos valores de utilidad se les llama también valores-Q. Se denota Q(a,i) para expresar el valor de tomar la acción "a" en el estado "i". La relación entre estos valores y la función de refuerzo vista anteriormente se representa con la siguiente ecuación:

$$U(i) = \max Q(a,i) \qquad (6)$$

A diferencia de Value Iteration, Q-Learning encuentra una relación de pares (situación, acción) a valores (Q-valores). En vez de tener una función valor asociada, Q-Learning hace uso de la función Q. En cada estado, hay un Q-valor asociado con cada acción. Q-valor podría ser definida como la suma de los refuerzos recibidos al ser ejecutada una acción siguiendo la política dada.

Los Q-valores juegan un papel importante en el aprendizaje por refuerzo, por dos razones: primero, son suficientes para realizar la toma de decisiones sin necesidad de tener un modelo del entorno; segundo, estos valores pueden aprenderse directamente a partir de los refuerzos.

El algoritmo usa una tabla para guardar la evaluación acumulativa, la cual se calcularía después de cada transición del estado i al j.

$$Q(a,i) = Q(a,i) + \beta(r + \max Q(a',j)) - Q(a,i) \qquad (7)$$

En este caso, "β" representa la tasa de aprendizaje y debe ser mayor que 0; y "r" simboliza los valores devueltos por el par (situación, acción) al aplicarse la función Q.

Algoritmo Ant Learning (Elaboración propia)

Se basa en la simulación del mecanismo de búsqueda y recolección de alimento de las colonias de hormigas, utilizando técnicas de aprendizaje que permiten a las mismas cooperar entre sí con el fin de alcanzar un objetivo común: la obtención de alimento y la supervivencia de la colonia.

El algoritmo se inicia con la salida de las hormigas exploradoras del nido. Cada una de ellas desconoce dónde puede estar el alimento, ya que sólo poseen información de su entorno inmediato. Cuando una hormiga llega a un punto de intersección del ambiente controlado, se produce una liberación de la feromona de búsqueda de alimento, la cual indica que esa posición ya ha sido visitada por una hormiga de la colonia. Esto evita que el resto de sus compañeras transite por el mismo camino que ella ya ha recorrido sin éxito. Dicho camino no será visitado por otra hormiga mientras la feromona se encuentre presente. Ella disminuye a medida que transcurre cada instante de tiempo, permitiendo que cuando la feromona no se encuentre presente, la posición pueda volver a ser visitada por otra hormiga.

Si una hormiga encuentra alimento, automáticamente envía una señal indicando el éxito de su búsqueda, en ese instante comienza a liberar la feromona de alimento encontrado. Si existen otras hormigas en posiciones cercanas a donde se encuentra la que ha hallado el alimento, las mismas se acercarán para prestarle ayuda en la recolección del mismo. Las hormigas también regresarán eventualmente al nido para avisar a sus compañeras que la búsqueda ha resultado exitosa. Dicho regreso puede ser hecho gracias a la feromona de búsqueda que fue liberada con anterioridad. De esta forma es posible que las hormigas vuelvan de regreso a su hogar (ver Apéndice F del Anexo).

Es así como las hormigas, utilizando mecanismos bastante simples, son capaces de generar comportamientos grupales de gran complejidad, permitiendo la supervivencia de la especie.

CAPÍTULO IV
MARCO METODOLÓGICO

Una metodología es un conjunto de pasos y procedimientos que deben seguirse para la creación y desarrollo de un proyecto, software o sistema determinado. Es una versión amplia y detallada de un ciclo de vida completo en el desarrollo de sistemas. Podemos decir que el ciclo de vida define las fases y tareas esenciales para el desarrollo de sistemas, sin importar el tipo o la envergadura del sistema que se intenta construir.

La metodología puede ser puntualizada de la siguiente manera:

- Tareas paso a paso para cada fase.
- Funciones individuales y en grupo desempeñadas en cada tarea.
- Productos resultantes y normas de calidad para cada tarea.
- Técnicas de desarrollo que se utilizan en cada tarea.

Para la realización de este proyecto se utilizará el modelo de ***Cascada con Retorno*** (*Fig. 10*). Este modelo permite regresar a niveles anteriores en caso de producirse un error en uno de ellos. Tiene como principal objetivo reducir el nivel de riesgo en caso de no disponer de ciertos requerimientos en cualquiera de las etapas. De este modo se flexibiliza el proceso de desarrollo del proyecto.

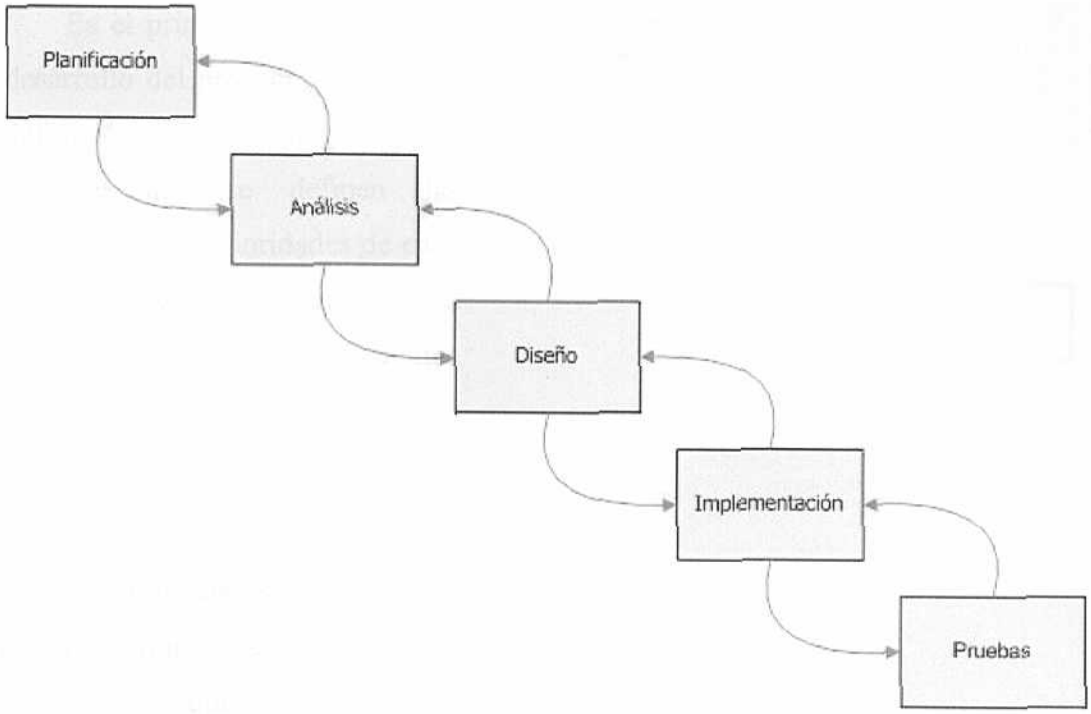
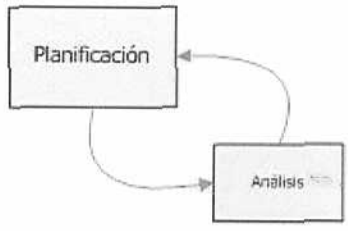


Figura 10. Modelo de Cascada con retorno.

Fuente: Elaboración propia

Descripción detallada de las etapas

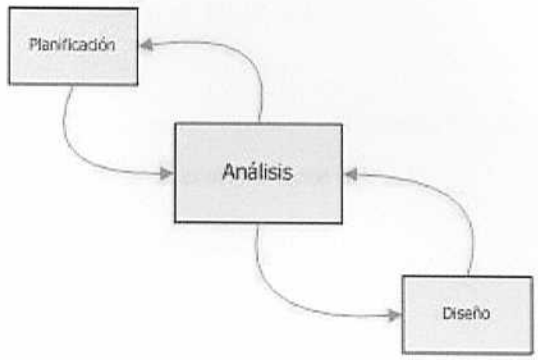
Planificación:



Su propósito es identificar y establecer las prioridades sobre las diferentes actividades. Se procede a realizar el levantamiento de información, especificación de requerimientos, familiarización con las tecnologías y el reconocimiento de las técnicas y estándares a emplear en el desarrollo del proyecto.

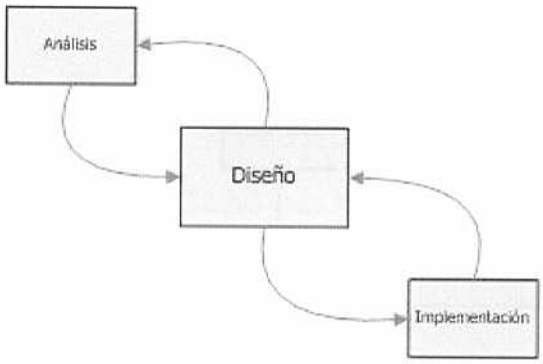
Análisis:

Es el primer paso dado hacia el desarrollo del proyecto. Se realiza la definición del problema y luego, a partir de allí se definen las necesidades y las prioridades de cada una de las tareas o actividades. Esta etapa determina el o los puntos críticos que tiene el proyecto.



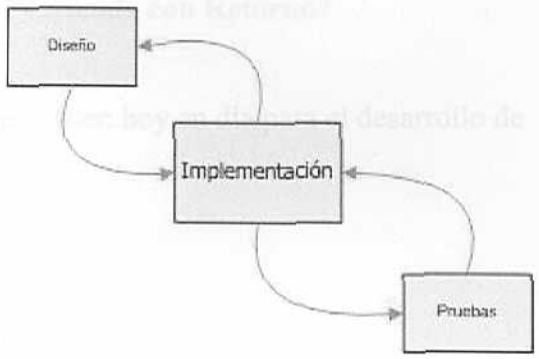
Diseño:

Su propósito es proponer una solución que satisfaga las necesidades que han sido especificadas durante la fase de análisis. Adicionalmente, avalúa las soluciones alternativas y especifica la solución detallada. Este se presenta en principio en un alto nivel de abstracción, sin embargo, a medida que se avanza se van obteniendo y estableciendo objetivos específicos tanto a nivel de datos como a nivel funcional.



Implementación:

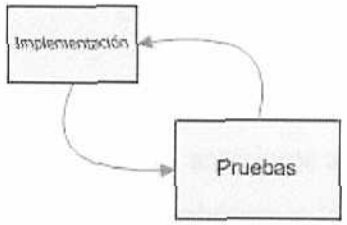
Su propósito es construir y/o ensamblar los componentes técnicos y poner en funcionamiento el sistema diseñado en la etapa anterior, de manera clara y ejecutable de acuerdo a los parámetros establecidos en la arquitectura previamente definida.



Además, existe un período o fase de construcción/prueba aplicada a cada uno de los componentes del sistema, asegurando que cada módulo ha sido probado. Se realizan evaluaciones a nivel de entrada y salida de datos para asegurar que los resultados son los que realmente se esperan.

Pruebas:

Es el último paso, el cual se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado y los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores y el brindar la seguridad de que la entrada definida produzca resultados reales de acuerdo con los requerimientos iniciales.



¿Por qué la metodología de Cascada con Retorno?

Entre las diferentes metodologías que existen hoy en día para el desarrollo de software se pueden mencionar:

- Modelo de Cascada Pura.
- Capas Solapadas o Sashimi.
- Espiral.
- Prototipo Descartable.
- Prototipo Evolutivo.
- Entrega Evolutiva.
- Cascada con Retorno.
- UML.

Se seleccionó la metodología de ***Cascada con Retorno*** ya que es un esquema que se adapta perfectamente a un ciclo de desarrollo por etapas muy bien definidas, ideal para un proyecto de esta envergadura.

Adicionalmente, esta metodología permite retomar etapas o fases anteriores a la actual. Este proyecto trae consigo la introducción al mundo de la robótica y la inteligencia artificial, la familiarización con algoritmos de aprendizaje y ambientes de programación. La metodología de ***Cascada con Retorno*** resulta de gran ayuda para descartar análisis, diseños e implementaciones que pudieran parecer ideales en principio, pero que pudieran no adaptarse a los requerimientos reales del problema de estudio, todo ello reduciendo el nivel de riesgo del proyecto.

CAPÍTULO V
DESARROLLO

Diseño y construcción de los robots y el ambiente controlado

Una vez seleccionada la metodología a emplear para el desarrollo del proyecto, es posible distinguir las siguientes fases a cumplir:

- Planificación
- Análisis
- Diseño
- Implementación
- Pruebas

A continuación se describen cada una de las fases en detalle, junto con las diferentes iteraciones que debieron ser realizadas para la construcción del ambiente controlado, de los robots que navegarían en él y del simulador.

AMBIENTE CONTROLADO (MALLA DE NAVEGACIÓN) Y ROBOTS

Iteración 1

Planificación:

Es esta fase se procede a establecer prioridades en cada una de las actividades, realizar el levantamiento de información y especificación de los requerimientos, así como la familiarización con las técnicas y tecnologías a emplear. Para ello, se

procedió a establecer con el Dr. Wilmer Pereira, Tutor guía del Trabajo Especial de Grado y Coordinador del Grupo de Investigación de Inteligencia Artificial (GIIA) de la Universidad Católica Andrés Bello (UCAB), una serie de reuniones de evaluación. En ellas se ratificó el uso del kit Robotics Invention System 2.0 de la LEGO MINDSTORM como base fundamental para el diseño y construcción de los robots a utilizar en el ambiente controlado, dada la posibilidad de emplear los kits pertenecientes al grupo de investigación que dirige el Dr. Pereira, lo cual abarataría considerablemente los costos del proyecto.

Por otro lado, se acordó la utilización de Java como lenguaje de programación de los robots; primero por la alta disponibilidad de documentación que existe en Internet para programar el RCX [1] y segundo por la intención del Dr. Pereira de establecer un estándar en el desarrollo de este tipo de software dentro de la UCAB.

Análisis:

En lo que respecta al ambiente controlado, se analizaron diferentes algoritmos para la navegación de los robots en el mismo. Entre ellos se pueden mencionar el *Método Bayesiano* [2], la teoría *Dempster-Shafer* [3] y el *Mapeo por Histogramas en Movimiento* [4] (The Histogrammic in Motion Mapping). Dado que estos algoritmos se basaban en la utilización de sonares, que no existirían obstáculos en el ambiente controlado y que el alimento no sería una estructura física sino un color, fue descartada su utilización.

(1) **RCX**: Robotic Command Explorer
(2) **Método Bayesiano**. Introduction to AI Robotics. Robin R. Murphy. Pág. 380
(3) **Teoría Dempster-Shafer**. Introduction to AI Robotics. Robin R. Murphy. Pág. 386
(4) **Mapeo por Histogramas en Movimiento**. Introduction to AI Robotics. Robin R. Murphy. Pág. 395

También se procedió a evaluar la utilización de diversos algoritmos de aprendizaje ya diseñados como el Q-Learning y el Value Iteration (ver Capítulo III, Marco Referencial), de forma de ver la posibilidad de adaptarlos al problema planteado. Estos algoritmos se basan en iteraciones, es decir, ambos en las primeras corridas del algoritmo no logran obtener rutas óptimas del origen al destino (objetivo). Sin embargo, a medida que aprenden de iteraciones previas logran converger, obteniendo una ruta óptima para solventar el problema planteado. Un ejemplo de estos algoritmos de aprendizaje puede darse en la vida real, cuando una persona viaja de una ciudad a otra. Probablemente las primeras veces se pierda o no encuentre una ruta eficiente. Sin embargo, a medida que recorre la trayectoria o camino una y otra vez, eventualmente logra encontrar la ruta (solución) óptima. En el caso de las hormigas, ellas no aprenden de iteraciones previas, pues simplemente se guían por feromonas las cuales se evaporan. Es por ello que luego de discutir este aspecto, se decidió optar por el diseño y elaboración de un algoritmo propio (Ant Learning) que permitiera simular de forma óptima el proceso que realizan las hormigas al momento de buscar y recolectar su alimento.

Luego de varias reuniones con el Dr. Pereira, se estableció que el ambiente controlado sería diseñado utilizando una malla, como la mostrada en la figura 11.

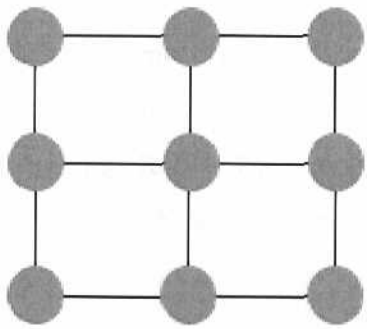


Figura 11. Modelo de malla de navegación.

Fuente: Elaboración propia

La utilización de la malla permitiría un mejor control sobre la navegación de los robots en el ambiente, además de evitar el empleo de sonares para que los robots pudieran percibir su entorno. En su lugar podrían utilizarse sensores de luz infrarroja para guiarlos a través de la malla (leyendo los colores del camino).

Por otro lado, era importante establecer la forma en que los robots se comunicarían entre sí para simular el comportamiento de las hormigas al momento de buscar y recolectar alimento. Para ello se acordó implementar un mecanismo que permitiera establecer una comunicación en red de al menos tres (3) entes en el ambiente controlado.

Diseño:

Una vez seleccionado el tipo de ambiente a construir y establecido el tipo de comunicación que debería existir entre los robots, surgieron una serie de preguntas. ¿Qué dimensiones debería tener la malla de navegación? ¿Qué forma podría tener? ¿Cuál sería la estrategia para establecer la comunicación entre los robots? ¿Se debería escoger una comunicación centralizada o distribuida? ¿Cómo manejar la liberación de feromonas en el ambiente?.

Al discutir sobre el tamaño del ambiente se llegó a la conclusión que éste dependía estrechamente de la forma que se escogería para comunicar los robots entre sí, por lo que se procedió a analizar cómo diseñar el mecanismo de comunicación. Para establecer dicha comunicación entre las hormigas se decidió utilizar una herramienta provista por el mismo RCX: El puerto IR o puerto infrarrojo. Investigando un poco más acerca del RCX, se halló que el puerto infrarrojo permite comunicaciones del tipo:

- RCX <–> RCX
- RCX <–> PC (utilizando la torre infrarroja que viene con el kit de LEGO MINDSTORM)

En el mundo real, las hormigas forman parte de un sistema distribuido, en el cual cada una de ellas es un ser autónomo. Ello permite que cada hormiga actúe de

forma independiente con un comportamiento muy simple, pero generando un comportamiento grupal sumamente complejo. Al tratar de trasladar este esquema al ambiente controlado, se generaron una serie de interrogantes, como por ejemplo:

- La comunicación simultánea: ¿Cómo sería posible permitir que todos los robots del ambiente se comunicaran a la vez?.
- ¿Cuál sería el mejor esquema para abordar la comunicación de los robots en el ambiente?.
- ¿Los robots se moverían simultáneamente en el ambiente?. De ser así, ¿Cómo se controlaría el proceso de comunicación?.

Consultando con el Dr. Pereira, nos propuso la utilización de un esquema muy popular en el mundo de los sistemas multiagentes: un sistema distribuido pero bajo el control de un ente externo (PC). Ello implicaría la introducción de un ente adicional que manejara de forma centralizada las comunicaciones. ¿Cuál pudiera ser ese ente? ¿Otro robot? ¿Un sistema externo?. Se acordó que sería un sistema externo, que correría en un PC como un servidor, estableciendo comunicación con el resto de los robots a través de sus puertos infrarrojos y la torre IR del kit LEGO MINDSTORM.

Aún así, esto no resolvía el punto de la comunicación simultánea. ¿Podrían las hormigas (robots) comunicarse de forma simultánea?. Si la comunicación se llevaría a cabo a través de rayos infrarrojos, ello implicaría que la información desde y hacia los robots estaría viajando constantemente en forma de ondas por el ambiente. Realizando pruebas previas, se encontró que no era posible una transmisión simultánea, ya que al momento de que dos o más robots transmitieran información hacia el servidor, ocurrían dos situaciones:

- Las señales, al ser transmitidas, se mezclaban, generando una sola onda. Esto producía información "basura", que era procesada por el servidor de forma incorrecta.
- Las señales no se mezclaban, pero llegaban a la vez al servidor a través de la torre IR. Al ocurrir esto el búffer de la torre colapsaba, al no ser capaz de procesar dos o más señales a la vez.

Teniendo estos aspectos en cuenta, se decidió utilizar una política mediante la cual una sola hormiga pudiera transmitir a la vez, y sería el servidor el que se encargaría de administrar qué hormiga transmitiría en qué instante de tiempo. Esta política es conocida popularmente en el mundo computacional como **token ring**. Ello implica que un ente central controla de forma secuencial el acceso a un recurso compartido entre varios procesos que tratan de tener acceso a él de forma recurrente. En nuestro caso particular, ese ente central controlaría la comunicación en el ambiente, con lo cual:

- Al manejar la comunicación por tokens, sólo un robot podría transmitir a la vez.
- Los robots no se moverían de forma simultánea, sino uno a la vez.
- Debería establecerse un protocolo de comunicación entre cada robot y el sistema central.

Ahora bien, ¿Qué tipo de información podrían intercambiar los robots entre sí y con el ente central?. Si bien es cierto que en el mundo real cada hormiga sólo tiene conocimiento de su entorno inmediato, sería necesario que el sistema que correría en la PC tuviera una visión global de todo cuanto ocurriría en el ambiente controlado, de forma de brindarle a cada robot sólo la información de su entorno inmediato. Para ello se estableció que al momento de que una hormiga llegase a un punto de cruce de la malla y tuviera el token en su poder, dicha hormiga transmitiría hacia el ente central la siguiente información:

- La coordenada del punto donde se encontrara.
- Un identificador de la hormiga (único para cada uno de los robots del ambiente).
- Una señal de activación para la segregación de la feromona (búsqueda de alimento o alimento encontrado).
- En caso de encontrar alimento, una señal especial para que el ente central comunique a las hormigas que se encontran aledañas, la ubicación del alimento y así estas (hormigas) puedan acercarse al mismo.

- Para cada paquete enviado, un valor de compendio o check sum, de forma de garantizar que el paquete llegara a su destino y correctamente.

Ya teniendo estas premisas establecidas, se procedió a estudiar el tamaño y las características del ambiente que se construiría. Haciendo pruebas de comunicación con una aplicación demo para la transmisión de data del RCX hacia un PC (como se muestra en la figura 12), se estableció que el rango de transmisión de la señal infrarroja tenía como máximo 1,80 metros, dependiendo si se configuraba o no el máximo rango de transmisión infrarroja (ver Figura 13).

Figura 12. Transmisión RCX->Torre infrarroja.

Fuente: Elaboración propia

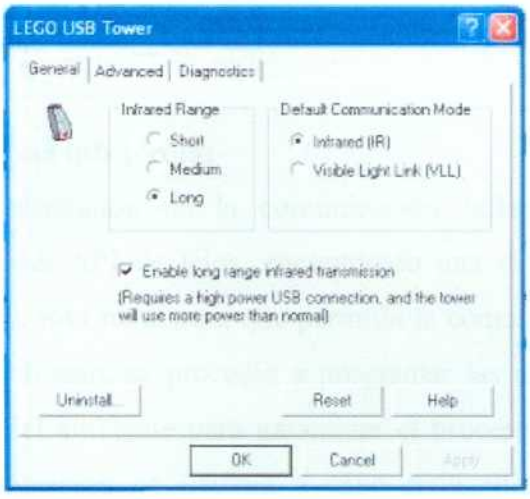


Figura 13. Opción para la configuración del máximo rango de transmisión infrarroja (en el PC).

Fuente: Elaboración propia

Tomando en cuenta esto, se decidió que el ambiente tuviera las siguientes dimensiones:

- 1,50 metros de lado.
- Cinco (5) cuadrículas o sectores por lado, de un tamaño de 30 centímetros cada una.

Se acordó que cada cuadrícula de la malla tuviera treinta centímetros de lado para permitir el tránsito de los robots de forma cómoda y libre por el ambiente. Por otro lado, la comunicación RCX->PC fue realizada colocando al RCX y la torre infrarroja directamente de frente, sin mayores ángulos de inclinación. En el caso de los robots, sería posible que el RCX y la torre estuvieran posicionados formando un ángulo cualquiera. ¿Cómo podría manejarse ese caso?. El Dr. Pereira nos comentó que en los casos de competencias de fútbol de robots, la comunicación entre los "jugadores" era realizada colocando el punto de transmisión de forma centrada y por encima del ambiente, con los infrarrojos de cada robot apuntando hacia arriba. Por ello se decidió diseñar cada robot con el RCX apuntando hacia arriba, y colocando la torre IR por encima del ambiente, justo en el punto central del mismo.

Implementación:

Comunicación vía infrarroja:

Para la implementación de la comunicación infrarroja en el ambiente controlado se consultó el API de lejos, encontrando una clase llamada RCXPort, perteneciente al paquete josx.rcxcomm, que permitía la comunicación entre un RCX y la PC. Una vez hecho esto, se procedió a programar las estructuras que debería contener cada agente del ambiente para garantizar el proceso de comunicación. Se decidió que la comunicación se llevaría a cabo transmitiendo paquetes con la siguiente forma:

0	1	2	3	4	5
Pos X	Pos Y	Buscando alimento / Alimento encontrado	Pos. Libre / Ocupada	ACK / !ACK	Check Sum

Posición	Contenido
0	La posición X que ocupa el robot dentro de la malla al momento de transmitir.
1	La posición Y que ocupa el robot dentro de la malla al momento de transmitir.
2	Bit que indica si se esta buscando alimento o si el alimento ha sido encontrado.
3	Bit que activado pregunta al servidor si la posición X,Y (en las posiciones 0 y 1 del paquete) esta libre u ocupada (si se ha liberado feromona en esa posición o no).
4	El paquete 000011 indica ACK, mientras que el paquete 000000 indica ¡ACK (la última transmisión fue exitosa o no).
5	Contiene el check sum o valor de compendio del paquete. Sirve para que al momento de rearmar el paquete en el receptor, el mismo pueda validar que la información llegó y lo hizo de forma correcta.

Tabla 4. Especificaciones del paquete que transmite / recibe cada RCX-Torre IR

Fuente: Elaboración propia

Construcción de la malla de navegación

Ya que se había decidido trabajar con sensores de luz infrarroja, se procedió a diseñar la malla de navegación de la siguiente manera:

- Los puntos de cruce serían de un color en particular.
- El alimento sería de un color distinto a los puntos de cruce.
- El nido tendría su propio color (distinto al del alimento y al punto de cruce).
- Se establecerían canales que unieran los puntos de cruce, para que cada robot pudiera mantenerse en el camino utilizando los sensores infrarrojos.

Se decidió construir la base del ambiente utilizando láminas de anime comprimido, y los puntos de intersección (alimento, nido y puntos de cruce) con cartulinas (en forma de círculos).

Pruebas:

Comunicación vía infrarroja utilizando la clase RCXPort

Cuando se procedió a probar la comunicación entre un RCX y la torre infrarroja, la misma pudo establecerse sin ningún tipo de problema, pero al momento de introducir otro RCX para probar la comunicación entre varios robots, surgieron problemas en la comunicación. Cuando dos o más RCX trataban de comunicarse (aunque fuera a diferentes intervalos de tiempo) con la torre infrarroja, la misma no era capaz de recibir los paquetes enviados. Buscando información exhaustivamente en Internet y libros especializados, no fue posible determinar el origen de esta falla. Luego de un considerable número de pruebas, fue posible descubrir por qué fallaba la comunicación. Cuando se utiliza la comunicación infrarroja, cada RCX posee un protocolo interno de comunicación, el cual hace por definición que se responda cualquier mensaje llegado a través de dicha vía, aún cuando dicho mensaje no tenga como destinatario al RCX que responde. Cuando la torre transmitía algún mensaje para el inicio del envío de la comunicación de un determinado RCX, este mensaje llegaba a todos los dispositivos (no se debe olvidar que la señal infrarroja viaja en forma de onda a través del aire). Por el protocolo interno de los RCX, todos respondían, lo cual hacía que varios mensajes llegaran a la vez hacia la torre infrarroja, colapsando el buffer de entrada del dispositivo.

Dado los resultados obtenidos, se optó por descartar la utilización de la clase RCXPort como base para la comunicación en red.

Navegación en la malla diseñada

Cuando se realizaron pruebas para verificar si los robots eran capaces de navegar correctamente a través de la malla, se encontró que los mismos presentaban ciertas fallas al momento de realizar los cruces hacia la izquierda o derecha, ya que perdían fácilmente la ruta de la malla y por ende la orientación dentro del ambiente.

Iteración 2

Implementación:

Comunicación vía infrarroja

Consultando nuevamente el API de lejos, se encontró otra clase llamada RCXLNPAddresingPort, perteneciente al paquete josx.rcxcomm. Esta clase permitía la comunicación en red de hasta 15 dispositivos a través de señales infrarrojas. Cada dispositivo, al definir un objeto de dicha clase, lo inicializaba pasando como parámetros al constructor la dirección que tendría el dispositivo y la dirección a la cual debía de transmitir. Las direcciones debían pasarse en forma hexadecimal. Se estableció entonces una dirección para el servidor y para cada una de las hormigas, haciendo que todas y cada una de ellas transmitiera información sólo hacia el servidor y que el mismo enviara información a un robot específico, dependiendo de cual tuviera en ese momento el token para la transmisión / recepción.

Ajustes en la navegación de los robots

Viendo las fallas presentadas en la primera iteración de pruebas, se verificó que cada robot leyera correctamente los colores del ambiente, para evitar que

pudieran, en un momento dado, mezclarlos. Así mismo se revisó detalladamente el código, de forma de optimizarlo.

Pruebas:

Comunicación vía infrarroja utilizando la clase RCXLNPAddresingPort

Realizando pruebas con esta clase, se verificó que efectivamente podía establecerse una comunicación en red entre varios dispositivos. Se procedió a hacer una prueba exhaustiva de la comunicación, haciendo que el servidor se comunicara de forma ininterrumpida con dos (2) RCX.

La comunicación funcionó bien, pero siempre, luego de transcurridos alrededor de cinco minutos transmitiendo data, los RCX colapsaban, generando una excepción en Java y finalizando la aplicación. Revisando en Internet y consultando con el Ingeniero Gonzalo Ron (experto en lenguaje Java con varias certificaciones de la SUN Microsystem) se llegó a la conclusión de que por alguna razón la pila de excepciones que genera cada aplicación hecha en Java se desbordaba, colapsando el programa.

Navegación en la malla diseñada

Una vez hechos los ajustes apropiados en los módulos de navegación y realizando pruebas en la malla, se logró corregir el problema del cruce en los puntos de intersección.

Iteración 3

Implementación:

Comunicación vía infrarroja

Revisando concienzudamente el código fuente buscando el origen de la falla y el colapso del RCX de cada robot, se decidió que en lugar de transmitir un paquete completo por vez, se transmitiera el mismo paquete, pero byte a byte, es decir, que se desglosara el paquete a enviar, se transmitiera de forma separada y que en el receptor fuera reensamblado y procesado.

Pruebas:

Comunicación vía infrarroja utilizando la clase RCXLNPAddresingPort

Una vez realizado el cambio en la lógica de transmisión / recepción de paquetes, se realizaron pruebas para verificar el funcionamiento de las aplicaciones, logrando transmitir hasta por más de una hora continua información entre el servidor y tres (3) RCX.

SIMULADOR

Iteración 1

Planificación:

Es la primera etapa en la metodología. Aquí se definen detalladamente los aspectos funcionales y operativos de la plataforma donde se va a desarrollar el simulador para PC. Por otra parte, se delimita la estrategia a seguir al momento de resolver posibles problemas.

El objetivo por alcanzar al terminar la planificación es disponer detalladamente, en cuanto a sus fases, del cronograma de trabajo a seguir en el desarrollo del simulador.

Adicionalmente, esta fase abarca lo referente al levantamiento de información y requerimientos del sistema, donde lo más importante es conocer a nivel biológico el mecanismo de búsqueda y recolección de alimento por parte de las colonias de hormigas.

Análisis:

En lo que respecta al simulador se realizó una fase de estudio y evaluación de las posibles herramientas a ser utilizadas. Originalmente la primera pregunta fue: ¿Bajo que lenguaje de programación se desarrollaría la simulación?

El lenguaje de programación utilizado para desarrollar el simulador fue Visual Basic, herramienta desarrollada por Microsoft. Originalmente habían otras posibilidades aparte de la escogida, ellas fueron: Java y Flash.

Finalmente se seleccionó Visual Basic ya que:

- Es un lenguaje conocido y fácil de utilizar.
- El entorno gráfico de desarrollo es agradable y sencillo.
- La depuración del programa se realiza rápidamente.
- La construcción de interfaces gráficas se hace de manera sencilla.

Por tanto, este lenguaje de programación poseía las funciones y estructuras de datos necesarias para la programación del algoritmo requerido.

Diseño:

Aquí se dio comienzo a la interfaz gráfica del simulador. Al mismo tiempo, se diseñaron aquellos posibles procedimientos, funciones y estructuras de datos a utilizar.

Una vez seleccionado el lenguaje de programación a utilizar, surgieron las siguientes interrogantes: ¿Qué tamaño debería tener el ambiente o matriz de navegación?, ¿Cuántas hormigas como máximo cabrían en el ambiente?, ¿Cómo sería la manera de coordinar a las hormigas para que actuaran de forma autónoma?

Al discutir sobre el tamaño del ambiente se llegó a la conclusión, luego de realizar un estudio, que una estructura matricial sería la más adecuada. La idea era crear un ambiente que no fuese excesivamente grande, tardándose un largo tiempo en recorrer desde el inicio (nido) hasta el destino (alimento), ni muy pequeño, pues no se apreciaría el desempeño a nivel grupal que logran desarrollar las hormigas para realizar la búsqueda y recolección de alimento.

Delimitar la cantidad de hormigas con un número máximo era necesario ya que, en un ambiente reducido es imposible colocar a tantas hormigas, y más aún, que desempeñen su labor de búsqueda y recolección eficientemente. De haber una gran cantidad de entes (hormigas) en vez de trabajar en forma conjunta, lo que harían sería estorbarse, es decir, las mismas no podrían realizar su función como consecuencia de que otra compañera se encuentre trabajando sobre el mismo lugar.

Implementación:

Paralelamente al desarrollo de los robots y el ambiente, se procedió a la programación del *Ant Learning* (elaboración propia). Para ello, se utilizó la herramienta Visual Basic, el cual permite trabajar en un entorno amigable de manera fácil y cómoda.

Luego de haber recopilado suficiente información acerca de las colonias de hormigas, y de haber evaluado el aspecto técnico y físico, era el momento de comenzar a programar el algoritmo que emularía el comportamiento de dichos insectos; pero, ¿Cómo aplicar el Ant Learning?. El objetivo de esta investigación se centra en poder simular un mecanismo de búsqueda y recolección de alimento de las hormigas, a fin de hallar la mayor probabilidad de éxito y supervivencia de la colonia.

Cada vez que una hormiga se mueve desde un punto a otro debe tomar una decisión, la cual viene dada de manera aleatoria, construyendo caminos desde el nido hasta el alimento; algunas de estas decisiones acercan el robot al alimento y otras, lo alejan. A medida que el robot vaya recorriendo e inspeccionando caminos va aprendiendo de sus errores. Eventualmente, un robot encontrará el alimento; allí procederá a retornar al nido en busca de otras compañeras (hormigas) que le permitan conjuntamente realizar el proceso de recolección.

El ente autónomo debe aprender a llegar a su destino escogiendo las acciones que le ofrezcan un mayor beneficio. El Ant Learning requiere elaborar una matriz de

estados donde se almacenará para cada posición, si está en proceso de búsqueda de alimento, si está en proceso de recolección de alimento y adicionalmente, la hormiga que ha pasado por esa posición.

Cada estado simboliza las posibles situaciones que pueden ocurrir a medida que uno o varios robots interactúan con el ambiente.

Se utilizó concurrencia para coordinar a las hormigas y hacer que realicen sus tareas de forma independiente, pero ¿Visual Basic tendría la capacidad de manejar el control de concurrencia?. Para implementar esta concurrencia se recurrió a llamadas a archivos .dll (API de Windows) que permitiesen controlar cada ente de forma autónoma. Con el funcionamiento de entes de manera individual es posible que ellos interactúen entre sí, ya sea para colaborar en la solución de un problema o tratar de alcanzar objetivos en forma colectiva.

Pruebas:

Cuando se procedió a probar la aplicación, la misma presentó problemas una vez que se corría el ejecutable. Inicialmente la aplicación funcionaba bien pero al momento de hacer las llamadas a las funciones que contenían los .dll, entonces el programa terminaba abruptamente.

La primera impresión acerca de la aplicación era que los errores provenían al hacer las llamadas a los archivos.dll, pues anteriormente nunca se presentaron esa clase problemas al trabajar con Visual Basic. Por tal razón se decidió probar el programa utilizando sólo un ente, es decir, correr el simulador utilizando únicamente una hormiga y verificar que el Ant Learning funcionaba de manera correcta.

Al comprobar que el mecanismo de búsqueda y recolección de alimento realizado por una sola hormiga funcionaba perfectamente, se procedió a realizar un estudio más profundo a cerca del API de Windows.

Iteración 2

Diseño:

Una vez que el Ant Learning fue probado con éxito, se decidió indagar aún más en el estudio del las funciones que incorpora el API de Windows.

Por tal motivo, el principal objetivo era encontrar detalladamente ¿Qué es el API?, ¿Cómo son llamadas a las funciones API?, ¿Qué requieren las llamadas a las funciones API?, ¿Cómo se declara una función DLL?, ¿Cuáles son las funciones API más utilizadas?.

El API de Windows es un conjunto de funciones, tipos y mensajes que se encuentran predefinidos para poder programar sobre los sistemas operativos de 32 bits de Microsoft. A este conjunto se le denomina Win32 (ver Apéndice C del Anexo).

Teniendo ahora mucho más conocimiento acerca del API de Windows y de sus funciones más importantes, encontrar la falla que se originaba en la aplicación fue muy fácil.

Habitualmente al crear un archivo ejecutable (.exe) en Visual Basic, se origina lo que se denomina una aplicación "Standard Alone" o sin dependencias; esto significa que el programa no necesita de un archivo externo para ser ejecutado. Para nuestro caso en particular, al necesitar de un archivo externo (.dll) fue necesario entonces, compilar el proyecto bajo la opción de "P-Code"; la cual permite generar una aplicación que será ejecutada por un intérprete.

Implementación:

Para poder compilar la aplicación en modo "P-Code" fue necesario modificar las propiedades de la aplicación y adaptarla para su correcto funcionamiento, como se muestra en la figura 14.

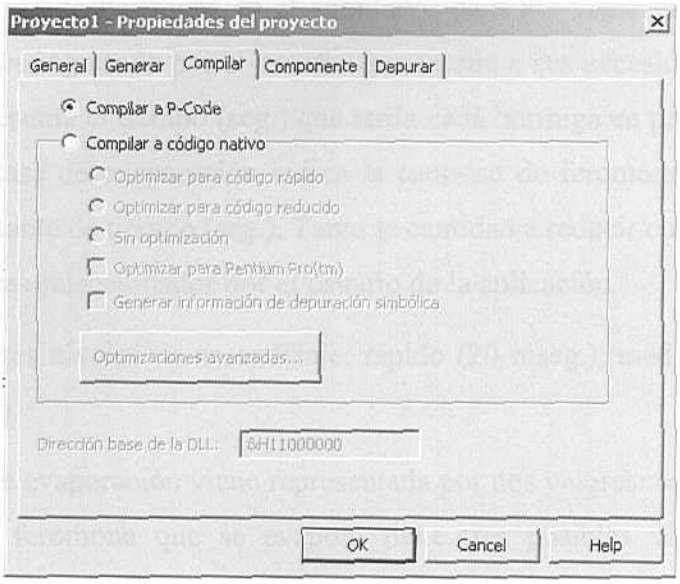


Figura 14. Propiedades del Proyecto.

Fuente: Elaboración propia.

El utilizar la forma de compilación "P-Code" proporciona como ventaja que ocupa menos espacio, en cuanto a cantidad de bytes y es mucho más rápido a la hora de ejecutarse.

Pruebas:

Ahora que finalmente la aplicación funcionaba con todos los requisitos necesarios a nivel técnico, la ejecución del programa cumplió con su objetivo principal, simular el mecanismo de búsqueda y recolección de alimento de una colonia de hormigas.

Una colonia de hormigas en el simulador para PC viene dado por un valor (entero), el cual es ingresado por el usuario de acuerdo a sus necesidades. El nivel de aprendizaje representa el tiempo (seg.) que tarda cada hormiga en procesar y ejecutar una acción. La tasa de evaporación indica la cantidad de feromona (entero) que se reduce en un instante de tiempo (seg.). Tanto la cantidad a reducir como el instante de tiempo son valores seleccionados por el usuario de la aplicación.

Existen tres niveles de aprendizaje: rápido (20 mseg.), medio (100 mseg.) y lento (500 mseg.).

La tasa de evaporación viene representada por dos valores: cantidad y tiempo. La cantidad de feromona que se evapora tiene tres posibles valores: rápido (5 unidades), medio (2 unidades) y lento (1 unidad). La cantidad de feromona asignada originalmente en una posición es 0, una vez que la hormiga pasa por una posición, ésta se eleva al valor 10. El instante de tiempo que tarda en reducirse una unidad de feromona tiene tres posibles valores: rápido (20 mseg.), medio (100 mseg.) y lento (500 mseg.).

CAPÍTULO VI
RESULTADOS

El principal objetivo de este Trabajo Especial de Grado fue hallar la mejor técnica que permitiese emular el mecanismo de búsqueda y recolección de alimento de las colonias de hormigas.

Para ello se construyeron tres robots capaces de moverse en un ambiente formado por figuras geométricas, varios cuadrados unidos sucesivamente formado una especie de malla. La idea era captar los eventos externos que ocurren alrededor de los robots y transformarlos en acciones, empleando para ello los sensores de luz. Dichas acciones lograrán que los entes actúen en forma cooperativa, alcanzando el objetivo que tienen en común; el alimento.

Los movimientos generados como consecuencia de la programación fueron exitosos; y los eventos captados por los sensores permitieron identificar y obtener los datos del medio ambiente. Esta información constituyó la base conocimiento utilizada para que los robots fuesen aprendiendo.

El algoritmo de aprendizaje *Ant Learning* (elaboración propia) mostró ser una excelente opción para resolver los problemas de navegación, colisión y posicionamiento. Se hicieron varias pruebas utilizando los robots construidos con el kit LEGO MINDSTORMS y se pudo observar y comprobar que el objetivo (alimento) la mayoría de las veces era alcanzado por un robot. Es importante destacar, que en contadas ocasiones, el desempeño de los robots se vio afectado por eventos externos al ambiente y errores en las lecturas de los sensores. Recordemos que estos dispositivos son muy sensibles a la luz y el brillo, lo que hace que sean poco precisos.

Por otro lado, a pesar de que puede hablarse de comportamiento emergente cuando dos o más entes interactúan entre sí generando acciones que no fueron programadas en un principio por el desarrollador, en este caso cada hormiga parece mostrar un comportamiento emergente a nivel individual. Cada hormiga fue

programada para navegar dentro de la malla de forma correcta, cruzando en cada intersección de acuerdo al valor aleatorio generado por ella misma y a las feromonas liberadas en el ambiente. Se había pensado que sería necesario programar adicionalmente los casos de cruce en los puntos extremos de la malla, pero al momento de realizar pruebas, cada hormiga cruzaba sin problemas en cada punto, aún cuando no fueron programadas para realizar este cruce en los mismos.

En cuanto a la simulación, se pudo observar el comportamiento de las hormigas en un ambiente ideal. Se utilizaron varios factores (nivel de aprendizaje, tasa de evaporación de la feromona en tiempo y cantidad, cantidad de hormigas) que permitieron comprobar que en realidad el trabajo en forma grupal permite hallar la solución de un problema en menor tiempo que cuando ésta (actividad) es realizada individualmente. En la simulación se evitan los inconvenientes externos que originan imprecisiones, como por ejemplo: los sensores de luz.

Una de las ventajas de la simulación es que hay la posibilidad de ver en forma rápida y sencilla el aprendizaje y comportamiento grupal logrado por las hormigas.

A continuación se muestran los resultados obtenidos de la simulación. Una iteración comprende desde que un robot parte del nido hasta que llega al alimento. El tiempo expresado en segundos representa el instante desde que una o varias hormigas salen del nido hasta que encuentran el alimento. Los movimientos representan el número de decisiones (para cada intersección) que toma una hormiga en una iteración dada. La cantidad de iteraciones indica cuantas veces fue ejecutado el algoritmo *Ant Learning* (elaboración propia).

Prueba 1:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. En este caso, el nivel de aprendizaje es el factor que varía (rápido, medio y lento), mientras que la tasa de evaporación de la feromona (tiempo y cantidad) se mantienen constantes. Se utilizó una sola hormiga para el proceso de búsqueda.

Los datos y la gráfica obtenida fue la siguiente:

Nivel del Aprendizaje: *Rápido.*										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	49	56	64	8	1	132	40	44	2	12
Movimientos	57	101	258	60	22	282	43	60	32	100

Tabla 5. Simulación de una hormiga con un nivel de aprendizaje rápido.

Fuente: Elaboración propia.

Nivel del Aprendizaje: *Medio.*										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	70	141	24	4	28	15	3	3	19	73
Movimientos	90	279	136	33	146	91	26	20	159	90

Tabla 6. Simulación de una hormiga con un nivel de aprendizaje medio.

Fuente: Elaboración propia.

Nivel del Aprendizaje: *Lento.*										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	221	56	222	199	114	40	168	50	24	33
Movimientos	133	76	386	325	223	67	274	49	46	63

Tabla 7. Simulación de una hormiga con un nivel de aprendizaje lento.

Fuente: Elaboración propia.

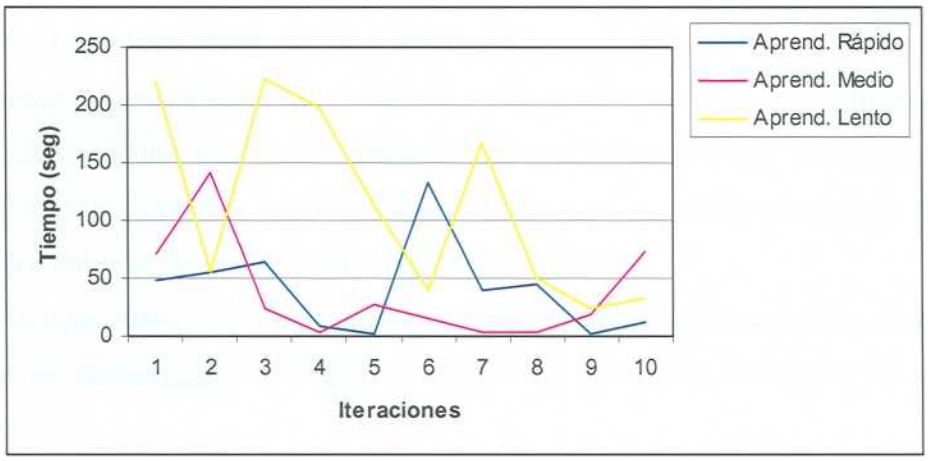


Figura 15. Comparación de las velocidades de aprendizaje con respecto al tiempo.
Fuente: Elaboración propia.

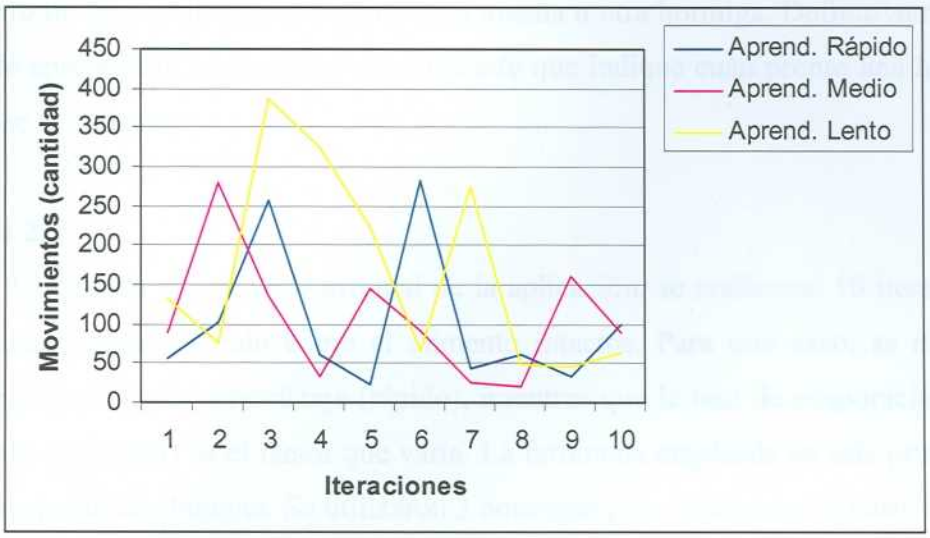


Figura 16. Comparación de las velocidades de aprendizaje con respecto a la cantidad de movimientos.
Fuente: Elaboración propia.

Es obvio que mientras más rápido es el aprendizaje de la hormiga más rápidamente encontrará el alimento. Sin embargo, esto no es una regla. Las hormigas realizan su proceso de búsqueda de alimento en forma aleatoria, por eso independientemente de la velocidad en la que se desplacen por el ambiente, éste no representa un factor decisivo.

Puede darse el caso, como efectivamente se muestra en la figura 15, que una hormiga que se desplaza muy lentamente sea capaz de conseguir el alimento más rápidamente que otra hormiga cuyo nivel de aprendizaje sea mayor. Esto ocurre simplemente porque la feromona es un factor muy influyente en el recorrido de una hormiga. Recordemos que una hormiga recorre un camino siempre y cuando éste no tenga feromona, pues ella simboliza que alguna hormiga ya ha pasado por ahí. La feromona puede ser dejada por ella o por una compañera.

Análogamente, una hormiga que se desplaza más rápidamente que otra por el ambiente no necesariamente tendrá como resultado una ruta más corta del nido al alimento.

Simplemente la ruta que toma una hormiga del nido al alimento depende de las decisiones que ésta hace a medida que va recorriendo el ambiente, aprendiendo

del rastro de feromona que va dejando ella misma u otra hormiga. Definitivamente el nivel de aprendizaje no es factor determinante que indique cuán pronto una hormiga consigue el alimento.

Prueba 2:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. Para este caso, se mantuvo constante el nivel de aprendizaje (rápido), mientras que la tasa de evaporación de la feromona (cantidad) es el factor que varía. La feromona empleada en esta prueba fue la de búsqueda de alimento. Se utilizaron 3 hormigas para el proceso de búsqueda.

Los datos obtenidos fueron los siguientes:

Cantidad de feromona reducida en un instante de tiempo = 1.										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	66	1	42	19	22	3	8	17	2	6
Mov Horm 1	**64**	18	**111**	21	23	11	16	65	**18**	**22**
Mov Horm 2	72	1	26	**13**	34	**68**	23	**85**	20	42
Mov Horm 3	25	**20**	41	88	**57**	2	**50**	90	19	15
Conseguido	*H 1*	*H 3*	*H 1*	*H 2*	*H 3*	*H 2*	*H 3*	*H 2*	*H 1*	*H 1*

NOTA: H1, H2, …, H3 representa la hormiga que encontró el alimento.

Tabla 8. Simulación de tres hormigas reduciendo 1 unidad de feromona.

Fuente: Elaboración propia.

Cantidad de feromona reducida en un instante de tiempo = 2.										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	146	8	29	22	33	3	7	4	13	4
Mov Horm 1	**138**	19	160	**137**	161	13	54	31	136	**28**
Mov Horm 2	185	**22**	**53**	87	**93**	**54**	84	33	61	45
Mov Horm 3	*160*	*15*	*46*	*72*	*188*	*33*	***43***	***33***	***157***	*36*
Conseguido	*H 1*	*H 2*	*H 2*	*H 1*	*H 2*	*H 2*	*H 3*	*H 3*	*H 3*	*H 1*

NOTA: H1, H2, …, H3 representa la hormiga que encontró el alimento.

Tabla 9. Simulación de tres hormigas reduciendo 2 unidades de feromona.

Fuente: Elaboración propia.

Cantidad de feromona reducida en un instante de tiempo = 5.										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	32	10	5	6	111	9	48	32	19	22
Mov Horm 1	93	100	45	24	72	**100**	**103**	**24**	33	49
Mov Horm 2	41	14	**80**	**91**	**85**	90	122	8	**86**	74
Mov Horm 3	**66**	**98**	34	83	112	76	91	10	119	**107**
Conseguido	*H 3*	*H 3*	*H 2*	*H 2*	*H 2*	*H 1*	*H 1*	*H 1*	*H 2*	*H 3*

NOTA: H1, H2, …, H3 representa la hormiga que encontró el alimento.

Tabla 10. Simulación de tres hormigas reduciendo 5 unidades de feromona.

Fuente: Elaboración propia.

Se calculó el promedio de las iteraciones, utilizando como referencia el tiempo (seg.) y la cantidad de movimientos empleados por las hormigas en encontrar el alimento. El resulto obtenido fue el siguiente:

Simulación de tres hormigas			
	Cant. Reducida (1)	**Cant. Reducida (2)**	**Cant. Reducida (5)**
Tiempo (seg.)	18,6	26,9	29,4
Movimientos	50,8	75,8	84

Tabla 11. Resultados de la simulación de tres hormigas.

Fuente: Elaboración propia.

La ruta que toman las hormigas del nido al alimento depende del rastro de feromona que van dejando a medida que recorren el ambiente. Se demuestra en la tabla anterior que a medida que la feromona (búsqueda de alimento) se evapora más rápidamente (reduciendo 5 unidades) el tiempo empleado para encontrar el alimento es mayor que cuando se evapora más lentamente (reduciendo 1 unidad). Esto tiene sentido ya que mientras no existe feromona en el ambiente, las hormigas tienden a moverse aleatoriamente impidiendo desarrollar su labor de conjunto al momento de realizar la búsqueda.

Prueba 3:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. Se realizó la simulación del mecanismo de búsqueda y relación de alimento empleando 5 hormigas.

Los datos obtenidos fueron los siguientes:

Simulación del ambiente con 5 hormigas										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	21	16	14	10	3	2	26	15	12	14
Mov Horm 1	1	11	**60**	1	1	1	72	45	**34**	65
Mov Horm 2	**42**	38	1	3	1	**22**	51	**96**	21	**26**
Mov Horm 3	5	**16**	5	**20**	12	1	**20**	2	1	53
Mov Horm 4	3	1	1	60	**46**	25	32	5	4	46
Mov Horm 5	36	9	20	16	1	1	6	5	42	12
Conseguido	*H 2*	*H 3*	*H 1*	*H 3*	*H 4*	*H 2*	*H 3*	*H 2*	*H 1*	*H 2*

NOTA: H1, H2, …, H5 representa la hormiga que encontró el alimento.

Tabla 12. Simulación del proceso de búsquela de alimento con 5 hormigas.

Fuente: Elaboración propia.

En este caso, cada una de las corridas (iteraciones) el tiempo empleado en conseguir el alimento bastante pequeño. Esto supone que la colaboración lograda por un mayor número de hormigas (5) es grande.

Prueba 4:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. Se realizaron comparaciones entre una hormiga, tres hormigas y cinco hormigas, empleando el tiempo total de búsqueda como medida de referencia para determinar cuanto tarda en conseguirse el alimento.

Los datos obtenidos fueron los siguientes:

Comparación entre una (1), tres (3) y cinco (5) hormigas, empleando el tiempo (seg.) como medida de referencia.										
Iteración	1	2	3	4	5	6	7	8	9	10
Horm (1)	70	141	24	4	28	15	3	3	19	73
Horm (3)	66	1	42	19	22	3	8	17	2	6
Horm (5)	21	16	14	10	3	2	26	15	12	14

Tabla 13. Comparación de la simulación empleando el tiempo como medida de referencia.

Fuente: Elaboración propia.

Se calculó el promedio de las iteraciones, y el resulto obtenido fue el siguiente:

Comparación entre una (1), tres (3) y cinco (5) hormigas, empleando el tiempo (seg.) como medida de referencia.		
	Suma	**Promedio**
Horm (1)	380	38
Horm (3)	186	18,6
Horm (5)	133	13,3

Tabla 14. Resultados de la comparación de la simulación empleando el tiempo como medida de referencia.

Fuente: Elaboración propia.

El promedio como medida de referencia nos indica que una hormiga tarda un poco más del doble de tiempo en encontrar el alimento que tres hormigas trabajando en forma conjunta. No obstante, la diferencia entre tres y cinco hormigas no es tan significativa. Esto nos indica que las hormigas cuando actúan de manera grupal son capaces de recorrer el trayecto que va del nido al alimento en menor tiempo; basándose en el desprendimiento de la feromona como medio de comunicación y aprendizaje.

Prueba 5:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. Se realizaron comparaciones entre una hormiga, tres hormigas y cinco hormigas, empleando como medida de referencia la cantidad de movimientos utilizados por ella (hormiga que logra conseguir el alimento) desde que parte del nido hasta que consigue el alimento.

Los datos obtenidos fueron los siguientes:

Comparación entre una (1), tres (3) y cinco (5) hormigas, empleando la cantidad de movimientos como medida de referencia.										
Iteración	1	2	3	4	5	6	7	8	9	10
Horm (1)	57	101	258	60	22	282	43	60	32	100
Horm (3)	64	20	111	13	57	68	50	85	18	22
Horm (5)	42	16	60	20	46	22	20	96	34	26

Tabla 15. Comparación de la simulación empleando la cantidad de movimientos como medida de referencia.

Fuente: Elaboración propia.

La tabla anterior muestra la comparación entre una, tres y cinco hormigas. En cada iteración las hormigas encuentran el alimento; y los movimientos representan la cantidad de decisiones que toma la hormiga que consigue el alimento en ir del nido hasta el alimento.

Se calculó el promedio de las iteraciones, y el resultado obtenido fue el siguiente:

Comparación entre una (1), tres (3) y cinco (5) hormigas, empleando la cantidad de movimientos como medida de referencia.		
	Suma	**Promedio**
Horm (1)	1015	101,5
Horm (3)	508	50,8
Horm (5)	382	38,2

Tabla 16. Resultados de la comparación de la simulación empleando la cantidad de movimientos como medida de referencia.

Fuente: Elaboración propia.

Análogamente a la referencia del tiempo, empleando más hormigas en el proceso de búsqueda de alimento es posible encontrar el objetivo utilizando menor cantidad de movimientos (desplazamientos). Esto puede ser traducido como un menor consumo de esfuerzo y recursos por parte de dichos entes; logrando obtener mayor eficiencia en sus actividades.

Prueba 6:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. Se realizaron comparaciones entre una hormiga, tres hormigas y cinco hormigas para determinar cuan cerca están de la ruta óptima.

Todo ambiente posee su ruta óptima la cual es calculada internamente por el sistema. La ruta óptima del ambiente predefinido para la simulación es 13 movimientos del nido al alimento.

Comparación entre una (1), tres (3) y cinco (5) hormigas, para determinar la ruta óptima.										
Iteración	1	2	3	4	5	6	7	8	9	10
Horm (1)	57	101	258	60	22	282	43	60	32	100
Horm (3)	64	20	111	13	57	68	50	85	18	22
Horm (5)	42	16	60	20	46	22	20	96	34	26

Tabla 17. Resultados de las simulaciones para determinar cuan cerca están los movimientos de la ruta óptima.

Fuente: Elaboración propia.

Nuevamente queda demostrado que trabajar en forma conjunta trae los mejores beneficios para un grupo. Aquí, una hormiga sólo logra acercarse a la ruta óptima en una ocasión, por el contrario cuando trabajan tres y cinco hormigas en el proceso de búsqueda de alimento, logran acercarse cuatro y cinco veces a la ruta óptima respectivamente. Inclusive en una oportunidad pudo hallarse la ruta óptima.

Prueba 7:

Utilizando el ambiente original de la aplicación, se realizaron 10 iteraciones, manteniendo tanto el nido como el alimento intactos. Se intentó sobrecargar el ambiente con un número mayor de hormigas para ver su funcionamiento.

Los datos obtenidos fueron los siguientes:

Simulación del ambiente con siete (7) hormigas										
Iteración	1	2	3	4	5	6	7	8	9	10
Tiempo (seg)	94	21	20	34	10	27	4	12	6	13
Mov Horm 1	84	30	31	22	1	**86**	1	4	3	4
Mov Horm 2	2	41	59	49	**90**	13	1	1	2	90
Mov Horm 3	**30**	1	1	71	1	37	14	**54**	1	23
Mov Horm 4	1	1	**40**	1	12	6	2	39	35	**46**
Mov Horm 5	11	9	1	20	14	18	1	14	**19**	1
Mov Horm 6	1	1	3	17	12	16	8	17	2	10
Mov Horm 7	1	**34**	20	**13**	10	6	**18**	25	28	34
Conseguido	*H 3*	*H 7*	*H 4*	*H 7*	*H 2*	*H 1*	*H 7*	*H 3*	*H 5*	*H 4*

Tabla 18. Simulación del proceso de búsquela de alimento con 7 hormigas.

Fuente: Elaboración propia.

La tabla anterior muestra una simulación empleando siete (7) hormigas. En cada iteración una de las hormigas encuentra el alimento; adicionalmente se muestra el tiempo empleado por ellas (hormigas) para realizar la búsqueda de alimento y la cantidad de movimientos de cada una.

Se calculó el promedio de las iteraciones, y el resultado obtenido fue el siguiente:

Simulación del ambiente con 7 hormigas		
	Suma	**Promedio**
Tiempo	241	24,1
Movimientos	541	54,1

Tabla 19. Resultados de la simulación del proceso de búsquela de alimento con 7 hormigas.

Fuente: Elaboración propia.

Al observar la tabla 17, se puede notar que en casi todas las iteraciones existe al menos una hormiga cuya cantidad de movimientos en busca del alimento es mínima. Esto significa que para este ambiente en particular, siete hormigas empiezan a estorbarse, es decir, ellas mismas comienzan a impedir el proceso de búsqueda de alimento ya que al haber tanta concentración de feromona en el ambiente, en ocasiones algunas de ellas deben permanecer en una misma ubicación esperando a que se evapore la sustancia química.

Igualmente, si se compara la simulación de siete hormigas con las hechas previamente (una, tres y cinco hormigas) nos damos cuenta de que tanto el tiempo promedio y movimientos promedio que realizan para el proceso de búsqueda de alimento es un poco peor incluso al empleado por tres hormigas.

CONCLUSIONES

Durante este trabajo de investigación hemos estudiado distintas ciencias, las cuales se encuentran muy relacionadas. Particularmente, la robótica y la inteligencia artificial encuentran bastante aplicación en el campo de la biología; específicamente en el comportamiento animal.

El kit LEGO MINDSTORMS resultó ser de gran ayuda para la construcción de los robots utilizados en la simulación del ambiente controlado, ya que permite de manera rápida y sencilla elaborar un robot capaz de adaptarse al problema que se desea resolver.

Dicho kit contiene una variedad de dispositivos, entre ellos, podemos destacar uno en particular; el sensor de luz, el cual fue utilizado en cada ente. La utilización del sensor de luz nos permite capturar toda la información que se encuentra en el ambiente, para luego procesar y ejecutar una acción. Sin embargo, la exactitud de este tipo de sensor se ve afectada por factores externos como la luz ambiental y el brillo, lo que origina ocasionalmente imperfecciones en el funcionamiento del robot.

Una desventaja notable del kit es la cantidad de entradas disponibles que tiene el RCX, pues sólo permite conectar un máximo de tres sensores y tres motores; limitando la funcionalidad y complejidad del robot. Adicionalmente, posee una escasa capacidad de almacenamiento de información; su memoria sólo permite 32 Kb.

El Firmware (Sistema Operativo) utilizado para programa el RCX fue LejOS. Fueron varias las razones por la que se decidió trabajar bajo esta plataforma, las más importantes fueron:

- LejOS sólo ocupa 16 Kb de memoria, dejando disponible 16 Kb para la programación desarrollada por el usuario; a diferencia de BrickOS, que ocupa 20 Kb de memoria y solo 12 Kb para la programación.

- LejOS es un firmware que esta basado en el lenguaje de programación Java, por lo que es posible utilizar toda la potencia y versatilidad que un lenguaje de este nivel proporciona, como: la utilización de clases, polimorfismo, herencia, control de concurrencia, etc.
- El estándar que se quiere implementar dentro del Grupo de Investigación de Inteligencia Artificial (CIIA) de trabajar con el lenguaje de programación Java, iniciativa promovida por el Dr. Pereira.

La navegación utilizada para encontrar el objetivo (alimento) se resolvió utilizando un algoritmo de aprendizaje (Ant Learning), de elaboración propia. Originalmente estaba planteado emplear Q-Learning ó Value-Iteration pero lamentablemente ninguno de los dos era capaz de adaptarse completamente al mecanismo de búsqueda y recolección de alimento.

Tanto Q-Learning como Value-Iteration se basan en el principio de programación dinámica, la cual permite realizar la tarea de aprendizaje de manera eficiente, implementando premios y castigos para cada acción. Ambos algoritmos poseen ventajas y desventajas, las cuales mencionamos a continuación.

En el caso de Q-Learning:

Ventajas:

- No necesita ser reprogramado en caso de que el ambiente sea modificado.
- Si el objetivo cambia de posición, el algoritmo es capaz de ajustarse, logrando alcanzar el objetivo.
- Tiende a converger más rápidamente que Value-Iteration.

Desventajas:

- La solución encontrada no siempre es la óptima, aunque comúnmente es bastante cercana.
- Mientras más estados se evalúen más efectivo será el aprendizaje, por lo que, habrá mayor consumo de recursos y tiempo.

En el caso de Value-Iteration:

Ventajas:

- No es necesario probar todos los posibles estados del sistema.
- La solución encontrada en la mayoría de los casos suele ser la óptima.
- No es necesario programar de nuevo todo el algoritmo en caso de modificarse el ambiente.

Desventajas:

- Sólo puede aplicarse considerando el par (estado/acción), donde la acción tomada en un estado cualquiera conduce al estado sucesor.
- Evalúa todas las posibles acciones es un estado dado, lo que origina lentitud al proceso de aprendizaje.
- Si el objetivo cambia de posición, puede suceder que el algoritmo nunca logre culminar o que tomó mucho más tiempo (respecto al promedio).

Cuando se trabaja con algoritmos basados en aprendizaje por refuerzo, como es el caso de Q-Learning y Value-Iteration, es necesario definir en forma detallada los estados, acciones, recompensas y castigos, ya que estos factores son la clave para el buen o mal desempeño del algoritmo. En caso de fallar cualquiera de los factores mencionados anteriormente, el funcionamiento del algoritmo podría verse seriamente afectado, o aún peor, podría darse el caso de no conseguir solución al problema planteado.

Por otra parte se estudió otras técnicas de aprendizaje, como la Computación Evolutiva, la cual involucra las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos; que también pueden ser utilizados para resolver problemas de navegación. Estos algoritmos son lentos y permiten encontrar una solución muy cercana a la óptima, pero sin garantía alguna de su convergencia. Necesitan de una primera fase de aprendizaje para posteriormente probar su funcionamiento, por lo

que, tardan mucho tiempo en procesar las soluciones al problema, siendo esto un factor limitante para la implementación de dichos algoritmos en el robot.

En cuanto al comportamiento de las hormigas, se comprobó lo fascinante que resulta que criaturas individualmente poco inteligentes pueden ser capaces de desarrollar en forma grupal mecanismos de supervivencia tan complejos y eficientes. Llevado al ámbito laboral podría resultar de gran interés pues podría ser utilizado en áreas de la ciencia, la exploración, la educación, a nivel organizacional, etc.

Por ejemplo, en la exploración, con la ayuda de robots es posible realizar la búsqueda de algún mineral o material en lugares nunca antes visitados por el hombre, sin poner en riesgo la vida de éste. A nivel organizacional, la manera de plantear las distintas estrategias de resolver un problema, al asignar para una actividad específica una cantidad de recursos adecuada; ni muy pocos, originando lentitud en el proyecto, ni muchos, para evitar que entre ellos puedan entorpecerse en sus labores. A nivel sociológico, el aporte a los estudios de carácter grupal no sólo de animales sino de cualquier sociedad a nivel general.

Por otro lado, la implementación de un enfoque multiagente permite la reducción de costos, ya que en un ambiente netamente centralizado al producirse una falla se corre el riesgo de que el sistema deje de funcionar, mientras que en un sistema multiagente si una unidad (un robot en este caso) dejara de operar por alguna razón, el sistema en general no se vería afectado y seguiría funcionando. Además, es posible lograr una colaboración de tal forma que al fijarnos una meta u objetivo, este sea capaz de lograrse en el menor tiempo posible, logrando eficiencia en cada una de las actividades que la integran.

RECOMENDACIONES

Como aporte a futuros trabajos de investigación, se realizan las siguientes recomendaciones:

- A pesar que el Dr. Wilmer Pereira desea establecer un estándar en el desarrollo de aplicaciones de robótica e inteligencia artificial utilizando lenguaje Java, si se desea establecer comunicaciones infrarrojas a nivel multiagente, se recomienda la utilización de algún otro firmware distinto a lejos, debido a que la implementación de las comunicaciones IR entre varios agentes resulta sumamente compleja y difícil de controlar.
- Si se desea trabajar con sensores de luz infrarroja, es deseable utilizar colores cuyo rango no coincida entre sí, ya que puede producir fallos en la lectura. Adicionalmente cada sensor puede obtener un rango distinto para un mismo color, por lo que se recomienda realizar un buen número de pruebas con distintos sensores de luz, de forma de obtener un rango estándar para un mismo color.
- Dotar al Laboratorio de Robótica de la Universidad Católica Andrés Bello de mayor cantidad de sensores, ya que la falta de ellos limita, en gran medida, la complejidad de los problemas y en especial la investigación de la robótica con enfoque multiagente.
- La Computación Evolutiva puede ser utilizada para resolver problemas de navegación, por lo que resultaría interesante que cualquiera de sus ramas (estrategias evolutivas, programación evolutiva y algoritmos genéticos) sea estudiada para resolver este tipo de problemas.
- Se desea dejar como propuesta de mejora para un futuro trabajo de grado la no utilización de una malla de navegación, haciendo que cada robot navegue por un ambiente desconocido sin ningún tipo de referencia como puntos de cruce o camino predefinido.

BIBIOGRAFÍA

- [1] Blum, M. S. y Hermann, H. R.: *"The Analysis of Olfactory Communication Among Animals"*. J. Theoret. Biol. 5: 443-469, 1978.
- [2] Dumpert, K.: *"Das Soziallebender Ameisen"*. Verlag Paul Parey, Berlin und Hamburg, 1978.
- [3] Eisner, T.: *"A Comparative Morphological Study of the Proventrículus of Ants (Hymenoptera: formidae)"*. Bulletin of the Museum of Comparative Zoology, Harvard, 116 (8):439-490, 1957.
- [4] Hangartner, W.: *"Control of Pheromone Quantity in Odour Trails of the Ant Acanthoyops Interjectus"*. Mayr. Experientia, 26: 664-665, 1970.
- [5] Hartwick, E. B., Friend, W. G. y Atwood, C. E.: *"Trail Laying Behavior of the Carpenter Ant Componotus Penssylvanicus"*. Canadian Entomologist, 109: 129-136, 1977.
- [6] Hingston, R. W. G.: *"Instinct and Intelligence"*. Mac Millan and Co. N.Y., 1929.
- [7] Holdobler, B.: *"Ethological Aspects of Chemical Communication in Ants"*. Adv. Stud. Behav., 8: 75-115, 1978.
- [8] Holdobler, B.: *"Recruitment Behavior in Camponotus Socius (Hym)"*. Z. Verge, Physiol., 75 : 123-142, 1971.
- [9] Holdobler, B., Moglich M. y Maschwitz, U.: *"Communication by Tandem Running in the Ant Camponotus Sericeus"*. J. Comp. Physiol., 90 :105-107, 1974.
- [10] Jaffe, K.: "Negentropy and the Evolution of Chemical Recruitment in Ants". J. Theor. Biology (en prensa). 1983.
- [11] Lestón, D.: *"The Ant Mosaic: Tropical Tree Crops and the Limiting of Pests and Diseases"*. Pestic. Abstr., 19:311-341,1973.
- [12] Levieux, J.: *"Food of Tropical Ants. I. Activity by Cicle and Diet of Camponotus Solon"*. Insectes Sociaux, 22 (4): 281-390, 1976.
- [13] Levieux, J. y Louis, D. : *"Food of Tropical Ants. II. Feeding Behaviors and Diet of Camponotus Vividus"*. Intrageneric Comparisons. Insectes Sociaux, 22 (4): 391-404, 1975.
- [14] Mc Cook, H. C. : *"The Honey Ants of the Garden of the Gods and the Occident Ants of American Plains"*. J. B. Lippincott & Co. Philadelphia. 208 pp., 182.
- [15] Parry K. y Morgan, E.: *"Pheromones of Ants: A review"*. Physiol. Entomol., 4: 161-189, 1979.

- [16] Stumper, R.: *"Radiobiologische Untersuchungen über den sozialen Nahrungshaushalt der Honigameise Proformica nasuta (Nyl)"*. Naturwissenschaften, 48 (24): 735-736, 1961.
- [17] Traniello, J.: *"Recruitment Behavior, Orientation and Organization of Foraging in the Carpenter Ant C. pennsylvanicus (Hym. Form.)"*. Ecology, 39: 695-704, 1977.
- [18] Wheeler, W. M.: *"Ants: Their Strucutre, Development and Behavior"*. Columbia University Press, New York, 25: 663 pp., 190.
- [19] Wilson, E. O.: *"Communication by Tandem Running in the Ant Genus Cardiocondyla"*. Psyche, 66: 29-34, 1959.
- [20] Wilson, E. O.: *"The Insects Societies"*. Belknap Press, Harvard University, Cambridge, Massachusetts, USA, 1971.
- [21] Russell, S. y Norvig, P.: "Inteligencia Artificial. Un enfoque moderno". 1era Edición. Edit. Prentice Hall. Edo. De Mexico, 1996.

Referencias web:

- http://atta.labb.usb.ve/Klaus/klaus.htm
- http://www.lania.mx/biblioteca/newsletters/1999-primavera-verano/tendencias.html
- http://neo.lcc.uma.es/radi-aeb/html/ant.html
- http://iridia.ulb.ac.be/~mdorigo/HomePageDorigo