# EL LABERINTO EN LA METODOLOGIA DE LA DIFERENCIA TEMPORAL[1] Carrera de Ingeniera en Sistemas Computacionales[2] Universidad De Guayaquil[3]

Martillo Rosales[1] , Grcia Carmen[2] and Pesantez Daniela[3]

***Abstract*— La solucion a un problema dado consiste en encontrar un camino desde un nodo representando el estado inicial, hasta el estado final deseado.**

## I. INTRODUCCION

La siguiente memoria pretende aclarar y exponer el trabajo llevado a cabo para la realizacin de un estudio relacionado en el contexto de la Inteligencia Artificial, este estudio se llevar a cabo mediante el uso del famoso juego "Comecocos", de ah su nombre "Comecocos vs Laberintos con tcnicas de bsqueda". Para ello, se intentar explicar con la mayor claridad y brevedad posible el trabajo llevado a cabo en el periodo de realizacin del mismo.

En este contexto, se describirn los mtodos de bsqueda utilizados, tanto informados como no informados, siendo esta ltima una metodologa que realiza un recorrido en el espacio de bsqueda de una forma sistemtica, pero sin tener en cuenta ningn tipo de informacin sobre el dominio del problema que se est resolviendo.

## II. TRABAJOS RELACIONADOS

### A. Algoritmos de Backtracking

El backtracking o vuelta atrs es una tcnica algortmica de resolucin general de problemas mediante una bsqueda sistemtica de soluciones. Se descompone la tarea a realizar en tareas parciales y se prueba sistemticamente cada una de estas, que a su vez se descompondrn en subtareas. Cuando al elegir una tarea se comprueba que no lleva a una solucin, se debe volver atrs, y probar con otra

### B. Utilizar recursividad

En general, las soluciones recursivas son menos eficientes que las iterativas (coste mayor en tiempo y memoria). Consejos: Los algoritmos que por naturaleza son recursivos y donde la solucin iterativa es complicada y debe manejarse explcitamente una pila para emular las llamadas recursivas, deben resolverse por mtodos recursivos. Cuando haya una solucin obvia al problema por iteracin, debe evitarse la recursividad

## III. DATOS

lienzo1.hilo.start(); jButton1.setEnabled(false); jMenuItem1.setEnabled(false); lienzo1.f=2; hilo.start();

Este trabajo es realizado bajo los estandares Aprendizaje Reforzado del lenguaje netbeans basado al juego de laberinto con la metodologia difernecial temporal

### A. Explicacion de La Variable

La funcin load permite incorporar grficos a partir de archivos BMP, PNG, JPEG etc load genera un objeto Surface que representar a la imagen en la memoria del equipo el retorno de setmode tambin es una superficie, pero esta representa lo que veremos en pantalla. Se utiliza (generalmente) para dibujar en pantalla. blit recibe la superficie a imprimir y su posicin. La posicin consiste en una coordenada (x, y). Los juegos generalmente utilizan un bucle de repeticin (llamado main loop).Ejecuta pequeas operaciones muy rpidamente. Agrupa todo lo relacionado con el personaje, atributos, comportamiento. El mtodo update contiene el comportamiento del personaje.

### B. Breve descripcin de la estructura de la memoria

A continuacin se detalla la estructura que seguir la presente memoria, para facilitar en la medida de lo posible su comprensin.

Basndonos en que el objetivo de este estudio consiste en la resolucin de tres problemas usando en cada uno un tipo de algoritmo en concreto, la explicacin de cada uno de estos se dividir en tres bloques. Y a su vez, en cada uno de estos, se detallan las siguientes secciones:

Objetivo: Se comenta brevemente los que se quiere conseguir, y cul es el algoritmo a desarrollar.

Solucin al problema propuesto: Es decir, cmo ha sido resuelto el problema desde un punto de vista conceptual y sin entrar en detalles de implementacin.

Ventajas e inconvenientes detectados: Antes y despus del proceso de implementacin.

Pseudocdigo: Detalles sobre la implementacin del algoritmo.

Pruebas y resultados: Testeo del funcionamiento del algoritmo con el mayor detalle posible.

Adems, en la descripcin de los algoritmos se pretender utilizar una notacin y un nivel de detalle que faciliten la comprensin de los mismos.

### C. Explicacion de La Variable

La tarea es encontrar la ruta mas corta entre una ciudad inicial s y una ciudad meta t. Los crculos son estados a visitar: Las flechas representan sucesores.Las etiquetas en las flechas son distancias. Los cuadros son distancias lineales a la meta.

## IV. Metodologia

A groso modo el algoritmo implementado se divide en tres funciones: chooseMove, CapturaMovimiento y BusquedaProfundidad. Siendo esta ltima donde se desarrolla verdaderamente la bsqueda en profundidad. La Funcin chooseMove, es la encargada de pasar el movimiento que tiene que realizar el pacman al programa general, en esta inicialmente se procede a guardar el estado actual del pacman, se inicializan las estructuras de datos usadas para almacenar los nodos visitados y para almacenar los movimientos y se llama a la funcin BusquedaProfundidad. La funcin CapturaMovimiento, es usada dentro de la funcin BusquedaProfundidad para transformar todos los estados por los que ha pasado pacman para llegar el objetivo, a los movimientos necesarios para llegar a este. A continuacin se presenta el pseudocdigo de esta funcin: La funcin recibe como argumentos 2 estados, el origen y donde queremos ir.

CapturaMovimiento( Estado sOrigen, Estado sFinal)

Creamos lista legalMoves, con los movimientos legales de sOrigen.

Generamos un estado auxiliar sAux, vaco;

Para ( todos los legalMoves ) hacer

Asignamos a sAux, el movimiento legalMove, actual.

Si ( El estado sAux, es igual a sFinal, con el legalMove actual) Retornamos ese legalMove; Finpara

// No se puede acceder al estado sFinal,con ningn movimiento Retornamos null;

Por ltimo, se procede a comentar la funcin BusquedaProfundidad. Como se ha comentado anteriormente esta es una funcin recursiva, que recibe como argumento un Estado y el nmero de puntos que hay actualmente en el tablero. Este segundo argumento ayudar a descubrir si en el estado actual pacman llega al objetivo, es decir, se ha comido un punto.

En esta situacin, se retornar una lista con todos los estados que han llevado al objetivo, y se devolver el control a la funcin chooseMove. La primera vez que se llama a la funcin el estado actual ser el correspondiente a la situacin inicial del juego y el nmero de dots ser 0. El pseudocdigo ser el siguiente:

ArrayList de estados: BusquedaProfundidad ( State sActual, int NumeroDots)

Si ( el estado actual sActual es objetivo) entonces

Capturamos el historial de estados hasta esta situacin. Retornamos la lista con los estados. Sino

Si ( La posicin de sActual no ha sido visitada ) Se guarda en la estructura de datos de visitados.

legalMoves = Capturamos los movimientos legales de sActual.

Para ( Todos los legalMoves ) hacer

sNext = estado siguiente con el movimiento actual.

Si ( sNext no ha sido visitado, y no es el padre del estado actual sState ) entonces

ListaEstados (lista de estados a el Objetivo) = BusquedaProfundidad(sNext, Nmero de dots con sActual);

Si (ListaEstados no es vacia ) entonces

Comprobamos si el nmero de estados que tiene es mejor, que el camino encontrado anteriormente. Y si es mejor, seleccionamos esta.

Finsi Finsi Finpara Finsi

Si llegamos hasta este punto, no existe ningn camino al objetivo, desde el hijo actual. Entonces retornamos null;

### A. Formalmente, un MDP

Es una tupla M =¡ S,A, , R ¿ formada por: Un conjunto finito de estados S(1, ..., n). Un conjunto finito de acciones A, que pueden depender de cada estado: Una funcin de recompensa ( R), que define la meta y mapea cada estadoaccin a un nmero (recompensa), indicando lo deseable del estado; Un modelo del ambiente o funcin de transicin de estados : A S S que nos dice la probabilidad de alcanzar el estado s S al realizar la accin a A en el estado s S, que se puede denotar como (s —s, a). Utilizan la experiencia para encontrar la funcin de valor. Calcula una poltica pseudoptima. Se basan en la actualizacin por el nuevo estado:

$$V_t(s_t) \leftarrow V_t(s_t) + \alpha\left[V_{t+1}(s_t) - V_t(s_t)\right]$$
$$V_t(s_t) \leftarrow V_t(s_t) + \alpha\left[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)\right],$$

donde $\alpha \in [0, 1]$ es el factor de aprendizaje.

Fig. 1. A schematic illustration of dissociative recombination. The direct mechanism, $4m_\pi^2$ is initiated when the molecular ion $S_L$ captures an electron with kinetic energy.

$$sV(s)V(s) + [r + V(s) - V(s)]s$$

### B. Elementos Adicionales

- Poltica (): define como se comporta el sistema en cierto tiempo. Es un mapeo (a veces estocstico) de los estados a las acciones.
- Funcin de valor ( V): indica lo que es bueno a largo plazo. Es la recompensa total que un agente puede esperar acumular empezando en un estado s (V(s)) o en un estado haciendo una accin a (Q(s, a)) Las recompensas estn dadas por el ambiente, pero los valores se deben de estimar (aprender) en base a las observaciones. Aprendizaje por refuerzo aprende las funciones de valor mientras interacta con el ambiente.
- La aproximacin lineal de la funcin no es la nica opcin. Ms recientemente, los mtodos basados en las ideas de estadstica no paramtrica se han explorado (qu e se pueden ver para construir sus propias caractersticas).
- Mtodo SARSA

$$Q(st, at) = Q(st, at) + rt+1 + Q(st+1, at+1)Q(st, at)A$$

## V. Resultados

Al intentar resolver este tipo de problemas, es necesario establecer una serie de pautas que ayudarn a determinar un orden en su resolucin. Para ello, se establecern una serie de simplificaciones, desde un punto de vista conceptual, que

sern tiles para dejar las ideas un poco ms claras, dejando as, de forma transitoria de lado la parte de implementacin del algoritmo, para centrarnos exclusivamente en la idea de resolucin del problema.

En primer lugar, consideraremos en principio que el espacio de bsqueda es un grafo, lo que simplifica bastante las operaciones, ya que de este modo ser posible generar un algoritmo que no recaiga en bucles infinitos, permitiendo as la posibilidad de ir almacenando los nodos visitados y cerrados para no repetir caminos, cosa que no se puede realizar usando bsqueda en rboles.

Para simular este grafo, se decide usar una funcin recursiva, de tal manera que se establece como nodo raz la posicin en la que se encuentra el pacman a la hora de realizar una nueva bsqueda, y tras comprobar si este es objetivo se expande generando los hijos a los que puede acceder. En el siguiente ejemplo se detalla grficamente, siendo el nodo (1,0) el nodo raz, y los nodos (1,1) y (1,2) los hijos de este. (Ver Figuras 1 y 2).

En este punto, es correcto mencionar que se realizar una nueva bsqueda cada vez que el pacman pretende comerse un nuevo punto. As conseguimos reducir considerablemente el coste computacional necesario para realizar una bsqueda de estas caractersticas, ya que se alcanzarn profundidades ms manejables. Con todo lo anterior, es fcil comprender que dentro de la funcin recursiva, primero se comprobar si el estado actual es objetivo, y posteriormente se comprueba si existe alguna rama hija que permita alcanzar el objetivo establecido.

Para realizar la comprobacin anterior, se selecciona el primer hijo que se encuentra y se llama a la funcin recursiva pasndole este como argumento. En cada llamada a esta funcin, se realiza un cambio de contexto en el estado del algoritmo, es decir, el nodo que antes era hijo, pasa a ser un nodo raz, para posteriormente poder expandir sus hijos y seguir con la bsqueda.

Con esta transformacin se consigue ir recorriendo paso a paso todos los nodos del grafo basndonos en los estados virtuales[1], para as estudiar el camino a seguir de la forma ms sencilla posible. Una vez encontrado el objetivo, se transforman los estados por los que se ha ido pasando, en los movimientos que tiene que realizar pacman para llegar.

Si por esta rama no se encuentra el objetivo, se retornar false, y se realizar la misma comprobacin con todos los hijos. As, se tiene conciencia de que estamos usando un algoritmo en profundidad.

Para que la realizacin del algoritmo sea posible, es necesario utilizar una serie de estructuras de datos. En concreto, se utiliza una lista para almacenar los nodos visitados, y una pila donde se almacenar los movimientos que tiene que realizar pacman hasta llegar al objetivo.

Adems, y como extra al algoritmo, cada vez que se retorna un camino vlido hasta el objetivo, se comprueba que tenga el menor nmero de pasos posibles. Con esto se tiene la certeza de que el camino encontrado en cada bsqueda ser ptimo.

sectioniLUSTRACION DEL PROGRAMA

Como ya se ha comentado anteriormente, una de las ventajas de usar un algoritmo de bsqueda en profundidad basado en grafos, es que se pueden almacenar los nodos visitados para no recaer en bucles infinitos.

Al principio del desarrollo de la prctica, en vez de usar este tipo de estructuras de datos, pens en resolver el problema mediante rboles, pero tras analizar el problema en profundidad y realizar algunas pruebas de implementacin, pude observar que el pacman realizaba muchos movimientos absurdos y en ocasiones incluso no pasaba del primer nivel.


Fig. 2. A schematic illustration of dissociative recombination. The direct mechanism, $4m_{\pi}^{2}$ is initiated when the molecular ion $S_L$ captures an electron with kinetic energy.

Se observa adems que el algoritmo consume mucho coste computacional, sobre todo en niveles elevados, ya que como los puntos estn ms separados, es necesario expandir ms nodos y la profundidad de la bsqueda se hace realmente compleja.

### A. *Corridas iniciales*

- Se llama a expandir

$$r([], l(s, 0/0), 9999, , si, Solucion)$$

- La primer llamada no satisface meta(Nodo).
- El primer caso es expandir la hoja

$$l(s, 0/0).$$

- La primera parte del trabajo, lo hace bagof /3 que computa los sucesores de s con sus distancias asociadas

$$Succ/[a/2, e/2].$$

- El predicado listaSuccs/3 construye la lista de posibles arboles sucesores

$$As/[l(a, 7/2), l(e, 9/2)].$$

- Con esa lista mejor

$$rF/2$$

encuentra que el mejor

$$rF1/7y$$

- Se llama recursivamente a expandir con el arbol expandido a

$$t(s, 7/0, [l(a, 7/2), l(e, 9/2)]).$$

El resto no ha cambiado.

### B. Contiunacion de la corrida

- La tercer llamada es expandir

$$([], t(s, 7/0, [l(a, 7/2), l(e, 9/2)]), 9999, , si, Solucion).$$

- Como

$$F/99999$$

la evaluacion continua en el tercer caso.
- Se actualiza el umbral con el mejor F del resto de la expansion

$$[l(a, 7/2), l(e, 9/2)](elvalordeFparaayaestaguardadoenvariable, F/$$

- se llama recursivamente a expandir el camino recorrido aumentado con s Camino,

$$Arbol/l(a, 7/2), elUmbral1/9$$

y el resto sigue igual.
- continuar

$$/7$$

decide como procede la busqueda de acuerdo al arbol expandido. Si una solucion Sol se ha encontrado, se regresa este valor. En cualquier otro caso, la expansion continua dependiendo del valor de Solucionado (no o nunca).

## VI. TABLA DE RESULTADO

Una vez analizados los diferentes algoritmos propuesto el siguiente paso es verificar la efectividad instalndolo en un mecanismo autnomo, para esta parte del experimento usaremos, el lenguaje de programacin

### A. EXPLICACION DE LA TABLA

Basados en las necesidades de construir un software flexible, que le permita guiarse en diferentes escenarios hasta encontrar la salida y que aprenda como encontrar el camino. Estas necesidades se traducen en un conjunto de parmetros e ndices descritos a continuacin. El sistema Laberinto Virtual est diseado para crear escenarios virtuales y recorrerlos de una forma artificial

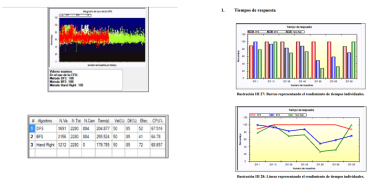

Fig. 3. La figura nos visualizara lo que se basa en el software de las estadisticas de cuanto es su renmiento de tiempo como lo podemos apreciar en ambas graficas.

TABLE I
AN EXAMPLE OF A TABLE

| s | T |
|---|---|
| A | M |

### B. FIGURA DE LA TABLA

Una decoracin alrededor del escenario. Un laberinto de ladrillo. Enemigos con autonoma y movimientos en bloque.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity Magnetization, or Magnetization, M, not just M. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write Magnetization (A/m) or Magnetization A[m(1)], not just A/m. Do not label axes with a ratio of quantities and units. For example, write Temperature (K), not Temperature/K.

## VII. CONCLUSION

La resolucin de problemas en IA requiere, normalmente, determinar una secuencia de acciones o decisiones. Esta secuencia ser ejecutada posteriormente por un agente con el fin de alcanzar un objetivo a partir de una situacin inicial dada. Dependiendo del problema concreto, la ejecucin de la secuencia de acciones o decisiones tiene asociado un coste que se tratar de minimizar.

Con el desarrollo de esta prctica, queda ms claro que no todos los mtodos a utilizar para la resolucin de un problema tienen el mismo coste y en consecuencia la misma complejidad. Siendo los mtodos de bsqueda informados ms idneos para el tipo de problema expuesto en esta ocasin.

## PRUEBAS Y RESULTADOS

Los datos de las pruebas se capturan al final del nivel al que llega AGENTE, aunque se realiza la bsqueda para encontrar cada punto, los datos aqu expuestos correspondern a la suma total del ltimo nivel completado, siendo este el ms complejo computacionalmente.

Fig. 4. La figuranos representa la coneccion de los nodos mediante los resultados de la tabla asignada

El tiempo efectivo es el tiempo que sistema encuentra el camino sin tomar en cuenta la bsqueda, ya que el sistema lo sigue aprovechando el camino aprendido..

Fig. 5. Mostraremos los resultados indicados

## AGRADECIMIENTO

Agradezco a la informacion brindada del ingeniero y mis companeros y asi poder obtener nuevo conocimientos; a su vez tener la oportunidad de exponerlo

## REFERENCES

[1] F. H. MARTINEZ SARMIENTO, Projection, design and construction of robotic platform for artificial intelligence research. in Tecnura [online]., 2010.

[2] S. Russell, Un enfoque moderno. pearson prentice hall, in Segunda Edicin, May 2004.

[3] M. Villares Souto, Entorno de juegos para la aplicacin de tcnicas de i.a, in Proyecto Fin de Carrera, Facultad de Informtica de la Universidad de A Corua, 2004, pp. 145155.

[4] S. D. y Eduardo Zalama Casanova, A mobile robot with enhanced gestual abilities.

[5] J. M. C. y J. M. Molina, Introduccin a la teora de agentes y sistemas multiagente.

[6] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In Proceedings of the Twelfth International Conference on Machine Learning, pages 362370. Morgan Kaufmann, 1995.

[7] Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone. Half field offense in robocup soccer: A multiagent reinforcement learning case study. In RoboCup 2006: Robot Soccer World Cup X, pages 7285. Springer-Verlag, 2006.

[8] Alexander A.Sherstov and Peter Stone. Function approximation via tile coding: Automating parameter choice. In Lecture Notes in Computer Science: Abstraction, Reformulation and Approximation, pages 194205. Springer Berlin / Heidelberg, 2005.

[9] Jing Peng and Ronald J. Williams. Incremental multi-step q-learning. In Machine Learning, pages 226232. Morgan Kaufmann, 1994.

[10] R. E. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, 1957.

[11] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup soccer keepaway. Adaptive Behavior, 13(3):165188, 2005.