Implementación de un agente inteligente basado en objetivos para la comparación de estrategias de resolución de problemas.

Abstract
El objetivo de este trabajo es construir un agente inteligente basado en objetivos, para comprender como éste se relaciona con el mundo en el cual se desenvuelve (el ambiente) y como utiliza las diferentes técnicas existentes (búsqueda informada/no informada y cálculo situacional) para tomar las decisiones pertinentes sobre las acciones que el mismo puede emprender. Se implementó un entorno gráfico, en donde se observa como un agente computacional (llamado Indiana Jones) utiliza diferentes estrategias de búsqueda y calculo situacional para recorrer una isla y hallar tesoros. La implementación permite comparar estas técnicas y sacar conclusiones sobre cual aplicar en este tipo de problemas.

Palabras Clave
Inteligencia Artificial, agente inteligente, ambiente, búsqueda informada, búsqueda no informada, A*, Costo Uniforme, calculo situacional.

Introducción

La inteligencia artificial (IA) es un área multidisciplinaria que, a través de ciencias como la informática, la lógica y la filosofía, estudia la creación y diseño de entidades capaces de razonar por sí mismas utilizando como paradigma la inteligencia humana [1].
Este trabajo surge en el marco de la Cátedra Inteligencia Artificial de la UTN Facultad Regional Santa Fe con el fin de aplicar los conocimientos adquiridos en un caso práctico.
El problema que se planteó consiste en desarrollar un agente inteligente capaz de desenvolverse en el escenario planteado, el cual consiste en una isla con aldeas, las cuales poseen distintos tesoros que esperan ser hallados por un agente arqueólogo.
Un agente inteligente, es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado [2].
Para el diseño de la solución, el agente utiliza dos estrategias de resolución: búsqueda en espacio de estado y cálculo situacional, a fin de poder luego comparar los resultados obtenidos.

Elementos del Trabajo y metodología

El trabajo fue desarrollado mediante las siguientes etapas:

1) **Análisis del problema.**

El problema planteado consiste en el diseño e implementación de un agente arqueólogo, al que se llamó "Indiana Jones", que utiliza distintas estrategias para encontrar tesoros escondidos que se encuentran en una isla, llamada "Emírece".
El objetivo del agente es encontrar un camino que le permita ir desde una aldea que representa el punto inicial del camino a recorrer a otra aldea considerada como el punto final de dicho camino. En el trayecto el agente debe recolectar los tesoros que considere más valiosos.
Como restricción, el agente solo puede transportar hasta 450 Kg, con lo cual debe ser cuidadoso a la hora de tomar cada uno de los tesoros que considere más valiosos.
Dentro de cada aldea, el agente se encontrará con un solo cofre que poseerán distintos tipos de tesoros. Los cofres deberán ser tomados o dejados de forma íntegra por el agente.
El agente contará con algunas herramientas que le permitirán llegar de la mejor forma a su objetivo:

- Un mapa, como el de la **Figura 1**, que le indica las aldeas, los caminos que conectan a las mismas y las correspondientes distancias.
- Un GPS que le permite ubicar donde se encuentra.
- Un acoplado que le permite transportar los cofres, con una capacidad de 450 Kg.

- Una cámara para poder analizar el cofre de la aldea donde se encuentra y determinar el valor real del mismo.
- Un satélite, que le permite determinar los tesoros de todas las aldeas de la isla en las cuales no se encuentra el agente.

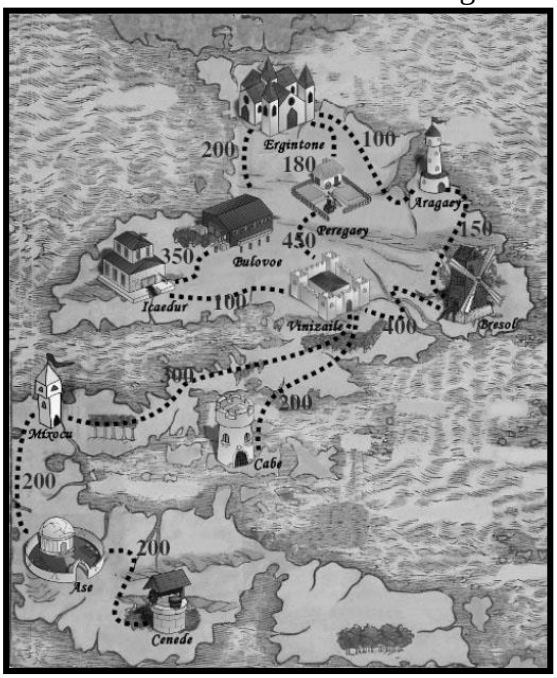

Figura 1. **Mapa de la Isla Emicere.**

Además en la isla existe un pirata que tiene un comportamiento aleatorio, roba cofres en distintas aldeas y cada cierto tiempo, por lo que obliga al agente ir verificando constantemente el camino óptimo.

2) Marco de la implementación

Como marco de implementación para la resolución del problema, se utilizó lenguaje JAVA dentro del entorno de desarrollo Eclipse. Además, contamos con un framework para simular agentes inteligentes denominado FAIA [3], y una herramienta denominada IDEM-IA [4], que permite modelar gráficamente el problema. Una vez modelado, la misma genera el código automáticamente extendiendo las clases de FAIA necesarias.

La metodología de desarrollo empleada fue del tipo ágil, con la ayuda de un versionado de código SVN-Subversion.

Para la etapa de Calculo Situacional se utilizó SWI-Prolog, para modelar la base de conocimiento del agente.

3) Modelado conceptual del problema con IDEM-IA

La figura 2 muestra la definición conceptual planteada para resolver el problema del agente explorador. En la misma se puede observar la identificación del agente Indiana_Jones y su relación con el ambiente a través de las percepciones que el agente recibe del ambiente y las acciones que ejecuta sobre el mismo para modificarlo.

Se definieron tres percepciones: (i)*ver_satelite* representa la utilización del satélite, permite al agente conocer en que aldeas hay cofres y un valor estimado de cada uno. Este valor es estimado, debido al comportamiento del pirata que puede robar cofres que el satélite observó y el agente no lo encuentra cuando llega a la aldea. (ii)*Utilizar_cámara,* representa el uso de la cámara para conocer al contenido de un cofre, esta percepción permite que el agente vea el contenido real del cofre, los distintos tesoros que en él se encuentran, y le permite calcular el peso y el valor total del mismo. Finalmente (iii) *gps* da el posicionamiento del agente. Esta percepción permite establecer donde está ubicado el agente en el mapa real. Esta percepción se podría interpretar en la interfaz, por medio de la posición del agente en la isla. Esta percepción solo fue adoptada en la resolución por Calculo Situacional.

El pirata se implementó por medio de una función random incluida en el ambiente, que elige entre las aldeas disponibles a cual robar el tesoro, si es que la misma aún lo posee. Cabe aclarar que la definición del pirata podría haberse considerado como otro agente, sin embargo solo lo modelamos como un comportamiento del ambiente.

La definición del agente en IDEM-IA consiste en identificar el estado del mismo y las acciones.

El estado del agente fue representado por medio de cuatro parámetros o variables.

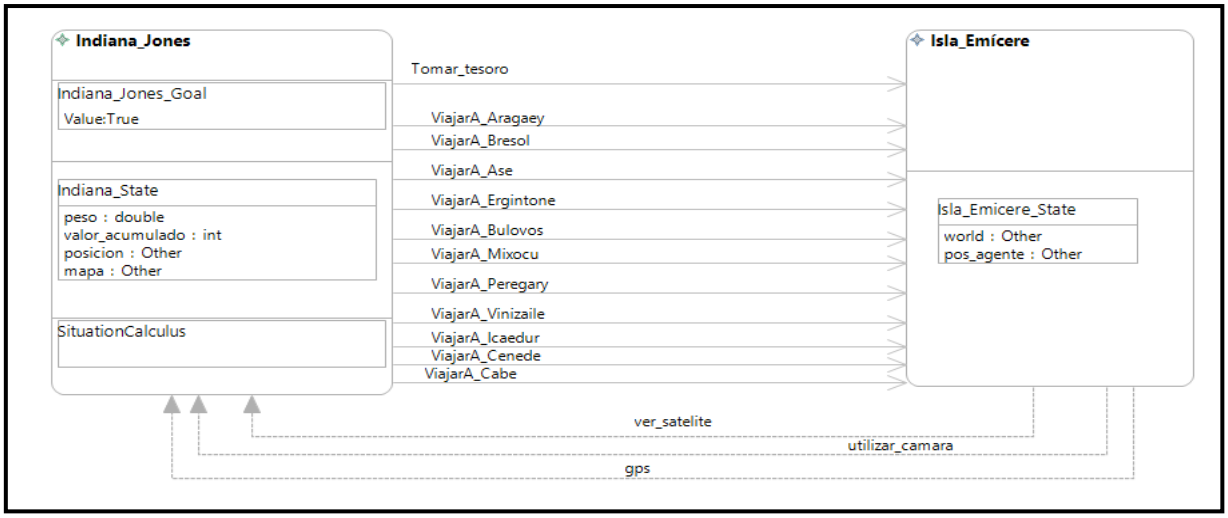

Figura 2. **Diagrama IDEM-IA implementado.**

Peso: este valor corresponde al peso acumulado que el agente está llevando en cada momento, el mismo no puede superar el valor 450 kg.
valor_acumulado: es un entero y corresponde al valor monetario que posee el total de los tesoros que transporta el agente.
Posicion: este valor hace referencia a la aldea donde el agente se encuentra en cada momento.
Mapa: es el mapa que el agente conoce.
Marca las aldeas por las que pasó y en cuales hay algún tesoro percibido a través del satélite.

Estado Inicial

El Agente comienza en la aldea inicial que plantea el escenario dado, con el peso y el valor acumulado inicialmente en cero, debido a que aún no tomo ningún cofre.
Se definieron 12 acciones que corresponden a viajar a cada una de las ciudades y a tomar el tesoro.

Tomar_tesoro: Dentro de una aldea, donde se encuentra un tesoro, el agente puede determinar si lo toma o no, en el caso de tomarlo ejecuta la acción *tomar_tesoro*, que suma a las variables de estado del agente el peso y el valor total del cofre. Para poder tomar un tesoro, el mismo debe encontrarse en la aldea y su valor distinto de cero (precondición).
ViajarA_<Aldea>: Se implementa una acción de este tipo por cada aldea del mapa. Ver figura 3. Esta acción implica un cambio de estado en la posición del agente. Para que el mismo pueda viajar a una aldea, debe encontrarse en una aldea vecina que tenga una conexión a la cual quiere viajar (precondición). Como resultado de esta acción el agente estará en la aldea <Aldea>.
Una vez realizada esta etapa, IDEM-IA genera el código correspondiente extendiendo las clases de FAIA. A partir de esta etapa, se comienza a trabajar sobre la implementación para ajustar la solución al

```
public class ViajarA_Aragaey extends SearchAction {

    /**
     * This method updates a tree node state when the search process is running.
     * It does not updates the real world state.
     */
    @Override
    public SearchBasedAgentState execute(SearchBasedAgentState s) {
        Indiana_State agState = (Indiana_State) s;

        if(agState.getMapa().getAldea("Aragaey").getVisitada()<=2 &&
                (agState.getposicion().getNombre().equals("Ergintone")||agState.getposicion().getNombre().equals("Bresol"))
        ){
            agState.getMapa().getAldea("Aragaey").setVisitada(agState.getMapa().getAldea("Aragaey").getVisitada()+1);
            agState.setposicion(agState.getMapa().getAldea("Aragaey"));
        }
        else {
            return null;
        }
        return agState;
    }
```

Figura 3. **Implentación de viajar a una de las aldeas en búsqueda.**

problema específico, codificando las acciones por un lado y definiendo estados iniciales, estado objetivo y pruebas de meta para el caso de búsqueda.

4) Implementación de búsqueda

Las estrategias de búsqueda permiten que el agente, en base a sus percepciones, evalúe las distintas posibilidades y luego de hallar la mejor opción realice la acción. Para esto se crea un árbol de nodos, en el cual, cada uno de ellos es un estado del agente según la acción que realizaría.

Para desarrollar el árbol existen dos tipos de estrategias, las informadas y las no informadas. Las primeras actúan en base a un conocimiento que posee el agente del ambiente, que le permite estimar el costo de llegar a la meta desde su posición Actual. Por otro lado, las no informadas, desarrollan su búsqueda en base a la lista de acciones implementada para el agente, trabajando con pilas o colas para la expansión del árbol.

Para la resolución de este problema con búsqueda, se utilizaron dos estrategias: A* como estrategia informada y Profundidad como estrategia no informada. Gracias a la modularidad de FAIA, el cambio de estrategias es muy simple y posibilita la comparación de los resultados obtenidos.

La estrategia A* utiliza una función que es la suma de una heurística (estimación de lo que falta para llegar a la meta) y de un costo (que calcula el costo de llegar desde el estado inicial al estado en cuestión).

La heurística para resolver este problema fue la distancia euclidea de la posición actual del agente hasta la aldea final.

Como costo en este caso fue establecer un costo a cada operador del problema. El costo de tomar un tesoro fue planteado como la inversa del valor del tesoro (1/valor_tesoro), esto posibilita que tomar un tesoro no sea muy costoso para el agente, y que ante tesoros mayores el costo de tomarlos sea menor. El costo de viajar es planteado con un valor fijo ya que no se dispone de las distancias entre aldeas, cada vez que se viaja por primera vez a una aldea el costo se estableció en 100 y si se viaja por segunda vez el costo es mayor, tomando el valor de 200, a manera de penalización.

En la implementación, debido a cómo es que se expanden los nodos, para implementar la estrategia en profundidad se ordenaron las acciones de forma tal que el primer operador sea tomar tesoro y luego viajar (teniendo en cuenta el orden de los mismos).

5) Implementación de Cálculo Situacional.

En este tipo de estrategia el agente utiliza una Base de Conocimiento (BC), por medio de ella infiere que acciones tomar según el conocimiento que dispone. A su vez a medida que se va moviendo en el ambiente, va incorporando nuevo conocimiento que agrega a la base. La BC se actualiza por medio de reglas lógicas, las que se pueden dividir en reglas diagnósticas, reglas de posibilidad y reglas de estado sucesor.

Reglas diagnósticas

Las reglas diagnosticas a través de las percepciones permiten determinar propiedades del ambiente. A través de estas reglas se agrega conocimiento a la BC.

En el problema se consideraron tres tipos de reglas diagnósticas, una que determina si hay tesoro en la aldea actual (Camara), la otra si tiene tesoro cada una de las aldeas del mapa (Satélite) y por último la posición del agente (GPS).

Las reglas de posibilidad determinan las acciones que puede llevar adelante el agente. Estas reglas establecen las precondiciones que deben cumplirse en el estado para que la acción sea posible ejecutarse.

Para la resolución se implementaron dos tipos, las que le permite tomar tesoro de una aldea, y las que permiten viajar de una aldea a otra a Indiana.

Las reglas de estado sucesor identifican cual es el nuevo estado dado la acción que se ejecutó. Estas reglas identifican las cosas que cambian y las que no cambian del estado actual.

La manera en que el agente puede optar por una acción u otra cuando hay varias acciones posibles, se da por medio de la valoración de las acciones, las mismas indican qué tan buena es una acción en una situación dada.

La valoración de acciones de nuestro problema se implementó de la siguiente forma:

- ***Excelentes***: se considera excelente que el agente tome un tesoro, si se encuentra en una isla que tiene tesoro.
- ***Muy Buenas:*** viajar a una aldea donde haya tesoro es considerada como una muy buena accion para el agente. Incluso aunque no haya tesoro ya que en el camino puede haber sido robado por el pirata.
- ***Buenas***: es bueno moverse de una aldea a otra por más que no haya tesoro.

la Figura 4 muestra la valoración de acciones mediante la implementación de las reglas logicas en prolog.

```
bestAction(X,S) :- excelent(X,S),!.
bestAction(noAction,S) :- position(cenede,S),!.
bestAction(X,S) :- very_good(X,S),!.
bestAction(X,S) :- good(X,S),!.
```

Figura 4. **Valoración de acciones en Prolog.**

Axiomas de estado sucesor.

Un Axioma de Estado Sucesor describe un predicado P que puede variar de situación en situación. Es necesario establecer uno de estos axiomas por cada uno de los predicados que cambian con el tiempo.

Cada uno de estos axiomas debe listar todas las formas en que el predicado puede ser cierto y en que el predicado puede ser falso.

En la Figura 4 se muestra los axiomas de estado sucesor definidos para la implementación, para cada flujo posible.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Successor state axioms                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

est(S) :- S is 0, asserta(position(aragaey, S)).

est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Bresol,S),not(position(_,S1)),asserta(position(bresol,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Ergintone,S),not(position(_,S1)),asserta(position(ergintone,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Bulovos,S),not(position(_,S1)),asserta(position(bulovos,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Icaedur,S),not(position(_,S1)),asserta(position(icaedur,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Vinizaile,S),not(position(_,S1)),asserta(position(vinizaile,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Mixocu,S),not(position(_,S1)),asserta(position(mixocu,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Ase,S),not(position(_,S1)),asserta(position(ase,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Cenede,S),not(position(_,S1)),asserta(position(cenede,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Cabe,S),not(position(_,S1)),asserta(position(cabe,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Aragaey,S),not(position(_,S1)),asserta(position(aragaey,S1)).
est(S1):- S1 > 0,S is S1-1,accionEjecutada(viajarA_Peregary,S),not(position(_,S1)),asserta(position(peregary,S1)).

est(S1):- S1 > 0,S is S1-1,accionEjecutada(tomar_tesoro,S),position(X,S),not(tieneTesoro(X,S1)).
```

Figura5. **Axiomas de estado sucesor.**

Resultados

Búsqueda

Con la solución implementada por medio de búsqueda informada, se logró llegar al objetivo en primera instancia, que era encontrarse en la aldea final, aunque nuestro objetivo implícito de juntar tesoros no se cumplía de manera esperada.

La expansión del árbol de la estrategia A* se realiza según el costo que posee cada nodo (heurística + costo), expandiendo así el de menor valor. En la Figura 6 podemos ver como se expande el árbol con esta estrategia.

Por otro lado, en la implementación de búsqueda no informada como amplitud o profundidad se lograron mejores resultados, ya que el orden de los operadores forzaba a que el agente tome tesoro.

La búsqueda en profundidad expandía su árbol hasta llegar a la meta planteada, expandiendo siempre el primer nodo, esto se puede ver en la Figura 7, de esta manera y teniendo un buen orden de los operadores, logramos que el agente cumpla su meta.

Calculo Situacional

En nuestra implementación, con este tipo de

Figura 6. **Árbol de búsqueda de A***

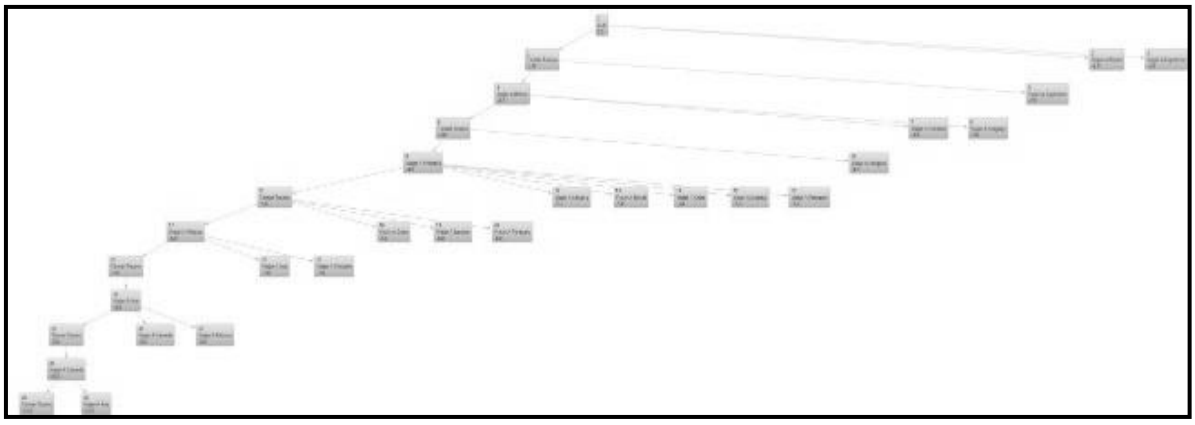

Figura 7. **Árbol de búsqueda en profundidad.**

estrategia se obtuvieron mejores resultados que con Búsqueda, ya que el proceso de unificación de predicados es más veloz que armar árboles sucesivos para tomar una decisión.

Comparación

Para poder comparar resultados entre la estrategia de búsqueda y la estrategia de cálculo situacional, optamos por evaluar cuatro parámetros que consideramos importantes, tanto en la solución planteada con pirata como sin pirata. En el caso de búsqueda consideramos la búsqueda en profundidad como punto de comparación, ya que en la primera parte fue la que mejores resultados lanzo. Los resultados obtenidos se pueden observar en la Tabla 1.

Con los resultados hallados, podemos observar que la resolución por medio de cálculo situacional obtiene mejores resultados para los parámetros de tiempo de ejecución y valor acumulado. Como ambas estrategias llegan a la meta, creemos que esos dos parámetros deben ser más importantes a la hora de evaluar la performance.

Si consideramos como parámetro el total del recorrido, vemos que calculo situacional tiene un mayor recorrido, y esto se debe a que búsqueda actúa en función del camino que lo acerca más rápido a la meta, por lo tanto búsqueda arroja un mejor resultado.

Como nuestro problema solo tenía la llegada a una Aldea Final como única meta, y no consideraba ni el valor total obtenido, ni el recorrido final, por los valores comparados consideramos que para la resolución del mismo, la mejor estrategia es Calculo Situacional. La misma cambiaria si el objetivo del problema cambiaria.

		Calculo Situacional	Profundidad
Con Pirata	Tiempo ejecución [m]	0,47	2,32
	Valor Acumulado[$]	369.050	190.400
	Peso Acumulado [Kg]	324,95	169,44
	Recorrido [Km]	1.430	1.250
Sin Pirata	Tiempo ejecución [m]	1,07	3,03
	Valor Acumulado[$]	429.700	268.850
	Peso Acumulado [Kg]	380,94	239,74
	Recorrido [Km]	2.230	1.250

Tabla 1. Comparación entre distintas estrategias.

Conclusión

Haber desarrollado un agente inteligente para resolver un problema, permitió conocer más sobre los usos de la inteligencia artificial. Además pudimos observar y comparar el uso de distintas estrategias para nuestro agente y obtener conclusiones en base a eso.

Cuando se trabaja con problemas de IA siempre debemos primero establecer una función meta u objetivo que se adecue bien al problema, y luego escoger la estrategia de búsqueda en función a esta meta. Pudimos observar en nuestro caso que la estrategia seleccionada para búsqueda informada no fue la adecuada, y por otro lado la no informada arrojo mejores resultados.

Creemos que es sumamente importante una buena definición de operadores y el orden de los mismos, como así también la función meta, para luego seleccionar una estrategia de búsqueda que se adecue de la mejor manera a nuestro planteo.

Por otro lado, el uso de cálculo situacional, a contraposición con búsqueda, dio un |Con cálculo situacional comprobamos que es imprescindible el orden de los axiomas, y una buena definición de los mismos, porque en ello está el punto más importante de la resolución. Además, en la definición de los axiomas reside la decisión de un agente, ya que a diferencia de búsqueda, calculo va resolviendo cada acción de acuerdo a las posibilidades que posee, y no en busca de la meta final.

Para finalizar podemos decir que este trabajo fue un fuerte motivador y propulsor del aprendizaje de la inteligencia artificial, una herramienta didáctica a través de la cual es posible llevar a la práctica los contenidos teóricos de la cátedra.

Referencias

[1] Nilsson, N. J. (1998) Artificial Intelligence: A New Synthesis. Morgan Kaufmann ed.

[2] Russell, S. & Norvig, P. (2010). Artficial Intelligence: A Modern Approach. Prentice-Hall, 3nd edition.

[3] Roa, J., Gutiérrez, M. y Stegmayer, G. (2009) FAIA: Framework para la enseñanza de agentes en IA. IE Comunicaciones: Revista Iberoamericana de informática educativa. Nº 7/8, ISSN 1699-4574, Pp. 43-56.

[4] Santana, W., Roa, J., Gutiérrez, M., Stegmayer, G. (2012). IDEM-IA: Un entorno de desarrollo integrado para el modelado de agentes inteligentes. IE Comunicaciones: Revista Iberoamericana de informática educativa. Nro 13, ISSN 1699-4574, Pp. 43-56.