INTELIGENCIA ARTIFICIAL

Trabajo Práctico Nº 1: Búsqueda

WATCH DRONE

Construcción de un agente inteligente que utiliza búsqueda.

Grupo X

José Ignacio Gómez

Pablo Nicolás Hechim

Sebastián Federico Paciuk

Índice

Trabajo práctico Nº1 – "WATCH DRONE"
Grupo X

José Ignacio Gómez – gomez.ignacio31@gmail.com
Pablo Nicolás Hechim – nicolashechim@gmail.com
Sebastián Federico Paciuk – sebapaciuk@gmail.com

Resumen. El presente trabajo de la cátedra Inteligencia Artificial, consiste en construir un agente inteligente, para comprender como éste se relaciona con el mundo en donde se desenvuelve y cómo utiliza las técnicas vistas en clases para tomar las decisiones sobre las acciones que puede emprender.
"WATCH DRONE" es un programa de computadora diseñado para implementar un agente "drone" que utiliza búsqueda para encontrar personas que cometieron un hecho ilícito. El objetivo del "drone" es obtener la ubicación exacta de dichas personas antes que logren escapar de su alcance.
La presente es la primera entrega del trabajo práctico, la cual constituye una porción de la entrega total.

1. Introducción

En este trabajo se considerará una ciudad donde un *DroneAgent* percibirá señales que serán disparadas sobre personas por un sistema de alertas en un ambiente nano-radio. Éstas alertas serán accionadas como consecuencia de un hecho ilícito y pondrán al *DroneAgent* en búsqueda, donde éste deberá distinguir entre víctimas y victimarios para así encontrar al victimario en cuestión. Se utilizarán cuatro tipos de búsquedas distintas (Anchura, Profundidad, Costo uniforme y A*).

En la próxima sección: "Solución", se detallará la solución implementada, y las aclaraciones pertinentes. A continuación, en "Resultados", mostraremos algunos de los resultados obtenidos al interactuar con el *DroneAgent*. Luego, en "Conclusiones", indicaremos qué conclusión se puede derivar luego de hacer funcionar el *DroneAgent* con los cuatro tipos de búsquedas.

2. Solución

2.1 Consideraciones previas

Procederemos a documentar la solución conceptual del problema.
En primer lugar, presentaremos al agente, sin olvidar que:

> "Un agente es un sistema que está situado en algún ambiente, y que es capaz de realizar **acciones** autónomas en ese ambiente, para cumplir sus **objetivos** de diseño. Para que un agente sea inteligente de ser *proactivo*, *reactivo* y tener *habilidad social.*"

Nuestro drone, entonces, será DroneAgent y percibirá el entorno a través de su cámara, GPS y antena y decidirá qué acciones pertinentes debe llevar a cabo para distinguir y encontrar un victimario.

El ambiente de nuestro agente, será lo que el DroneAgent percibe y a quien responde, es decir, el ambiente le proveerá señales, y será a quien el DroneAgent le devuelva una respuesta o frase. Se ilustra lo expresado en la Figura 2.

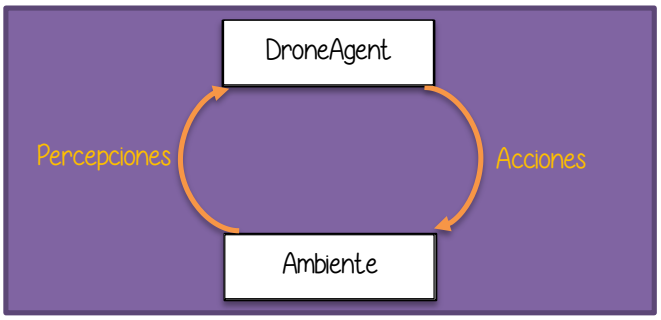


Figura 2 – Interacción Agente-Ambiente

Se necesita que el *DroneAgent* realice un proceso de búsqueda cada vez que perciba señales del ambiente. El proceso se representaría como lo muestra la Figura 3.

Figura 3 – Proceso de decisión de acción

En la Figura 4 se muestra un diagrama IDEM-IA que modela el estado del agente, las operaciones que este puede realizar, el estado del ambiente, las percepciones del agente y la función objetivo definida.

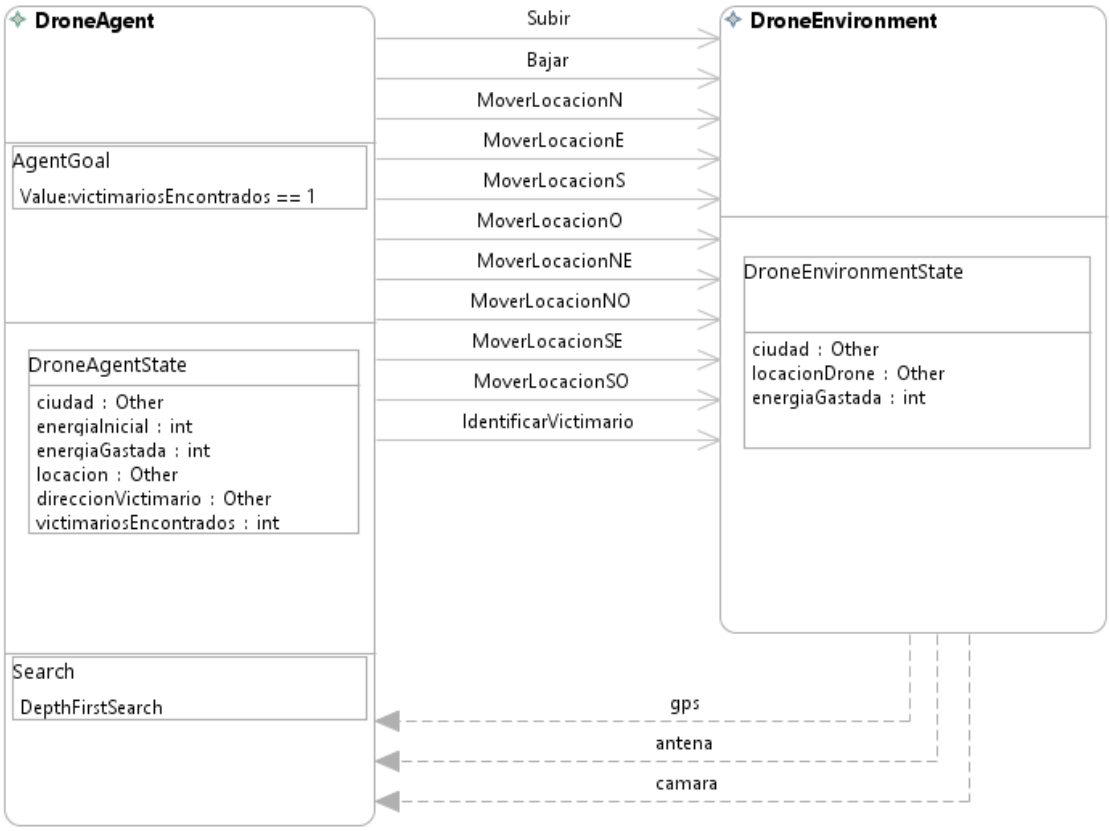


Figura 4 – Diagrama IDEM-IA

2.2 Definición de representación de estados

Estado del agente:

El estado del agente está representado por medio de seis parámetros o variables como se muestra en la Figura 4.

Gómez, José Ignacio – Hechim, Pablo Nicolás – Paciuk, Sebastián Federico

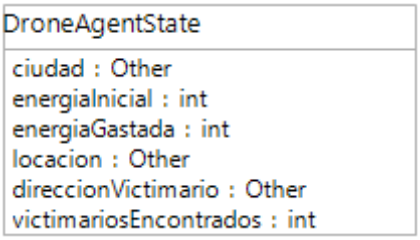


Figura 4 – Representación del estado del agente

Donde:

- ✓ **ciudad**: Estructura de datos que representa el mapa donde *DroneAgent* puede moverse, y sólo posee los datos que éste conoce.
- ✓ **energíaInicial**: Representa las unidades de energía inicial que dispone *DroneAgent*.
- ✓ **energíaGastada**: Representa el consumo total acumulado de energía que realiza *DroneAgent* al ejecutar acciones.
- ✓ **locación**: Representa la posición actual de *DroneAgent* dentro del mapa (**ciudad)**.
- ✓ **direcciónVictimario**: Indica si hay un victimario en cada una de las posiciones cardinales respecto a la posición actual de *DroneAgent*. *Aclaración: Sólo indicará si DroneAgent se encuentra en el nivel "bajo".*
- ✓ **victimariosEncontrados**: Indica la cantidad de victimarios encontrados por *DroneAgent*.

Estado inicial del agente:

- ✓ **ciudad**: Contiene cargadas la estructura de cuadrante – subcuadrante – esquina. No contempla información sobre ubicaciones de víctimas y victimarios.
- ✓ **energíaInicial**: Se inicializa en 1000 unidades de energía.
- ✓ **energíaGastada**: Se inicializa en 0 unidades de energía.
- ✓ **locación**: Se inicializa en el cuadrante "A1" correspondiente al nivel "alto" del mapa **(ciudad)**.
- ✓ **direcciónVictimario**: Se inicializan todas las posiciones cardinales en false debido a que *DroneAgent* no se encuentra en el nivel "bajo".
- ✓ **victimariosEncontrados**: Se inicializa en 0.

Estado final del agente:

- ✓ **ciudad**: Contiene cargadas la estructura de cuadrante – subcuadrante – esquina. Contempla la ubicación de víctimas y victimarios percibidos por *DroneAgent* durante la búsqueda del objetivo.
- ✓ **energíaInicial**: 1000 unidades de energía.
- ✓ **energíaGastada**: Indica la cantidad de energía consumida por *DroneAgent* hasta llegar al objetivo.
- ✓ **locación**: Indica la última ubicación de *DroneAgent*. Es decir, donde encontró al victimario.
- ✓ **direcciónVictimario**: Contiene la dirección en la que se encuentra el victimario respecto a la posición de *DroneAgent*. Ésta corresponde a la última percepción que *DroneAgent* obtuvo con su cámara cuando alcanzó el objetivo.
- ✓ **victimariosEncontrados**: Indica la cantidad de victimarios encontrados, en este caso, será un único victimario.

Estado del ambiente

- ✓ **ciudad**: Estructura de datos que representa el mapa donde *DroneAgent* puede moverse. Contempla las intensidades, cantidades y ubicaciones de víctimas y victimarios.
- ✓ **locaciónDrone**: Indica la ubicación actual de *DroneAgent*.
- ✓ **energía**: Indica la energía gastada hasta el momento por *DroneAgent*.

2.3 Definición de percepciones

- ✓ **GPS**: Permite a *DroneAgent* saber cuál es su ubicación actual.
- ✓ **Antena**: Permite a *DroneAgent* identificar unívocamente las señales de los nano – radios. Dichas señales son percibidas con distinta intensidad según el nivel de altura en el que se encuentre *DroneAgent*.
- ✓ **Cámara**: Permite a *DroneAgent* identificar a un victimario en el nivel de altitud "bajo". Le muestra en línea recta todas las calles desde la posición actual hasta el límite del cuadrante en el cual se encuentra, pudiendo así decidir en qué dirección moverse para encontrar un victimario.

2.4 Prueba de meta

Se considera que se alcanzó el objetivo si se encuentra un victimario o se recorren todas las esquinas de la ciudad:

```
victimariosEncontrados == 1 || recorrerTodasLasEsquinas == TRUE
```

2.5 Definición de operadores

- ✓ **Subir:** Acción que permite a *DroneAgent* desplazarse desde el nivel de altitud "bajo" a "medio" y desde "medio" a "alto".
- ✓ **Bajar:** Acción que permite a *DroneAgent* desplazarse desde el nivel de altitud "alto" a "medio" y desde "medio" a "bajo".
- ✓ **MoverLocación<DirecciónCardinal>:** Acción que permite a *DroneAgent* desplazarse hacia un cuadrante, subcuadrante o esquina adyacente.
- ✓ **Identificar:** Acción que permite a *DroneAgent* diferenciar víctimas de victimarios cuando se encuentra en una esquina.

2.6 Heurística

La heurística que se plantea en el trabajo práctico es el cociente entre la energía disponible de *DroneAgent* y la intensidad percibida en su locación actual. Si *DroneAgent* se encuentra en el nivel de altitud "bajo" y no hay victimario en ninguna de las direcciones posibles a tomar en línea recta, se sumará un "plus condicional" a la relación definida.

$$h(n) = \frac{energía\ disponible}{1+intensidad\ percibida} + plus\ condicional \quad (1)$$

2.7 Estrategia seleccionada

Búsqueda A*

En este método la decisión de qué nodo expandir se toma en base a una función heurística que mientras menor sea su resultado, mejor va a ser el nodo en cuestión.

Búsqueda Primero en Profundidad

En este método los nuevos nodos generados van a la cabeza de la lista de nodos a expandir (LIFO). El *método* sólo necesita mantener un único camino desde el nodo raíz. Falla en espacios infinitos. Problemas con loops, se controlan estados repetidos. En espacios finitos es completa.
Como en nuestro árbol no existirán los estados repetidos, se termina la búsqueda al recaer la misma en un estado repetido.

Búsqueda en Anchura

En este método los nuevos nodos generados se colocan dentro de una lista de tipo FIFO, por lo cual primero se expande el nodo raíz, se generan sus sucesores y luego se expande cada uno de los sucesores, etc. Se va expandiendo por nivel hasta que en algún alcanza el objetivo planteado.

Búsqueda de Costo Uniforme

En este método se trabaja de igual modo que en la búsqueda en anchura siempre y cuando los costos empleados sean iguales porque expande el nodo no expandido más superficial. En caso de diferir, expande el nodo *n* con el camino de costo más pequeño. No se preocupa por el número de pasos que tiene un camino, pero sí sobre su coste total.

Hay posibilidades de entrar en bucle infinito. Se puede garantizar completitud si el costo de cada paso es mayor o igual a alguna constante positiva pequeña *k*.

3. Resultados

En esta sección se presentan los test realizados que comprueban el correcto funcionamiento de las distintas estrategias de búsqueda además de compararlas entre sí, en los casos de que el *DroneAgent* posee información de su entorno así como también cuando no la posee.

Una aclaración importante es que, dado que el Drone no conoce originalmente en qué esquinas hay gente y en cuál de ellas se encuentra el victimario, la etapa de búsqueda no conducirá por si sola a encontrar el victimario (ya que para percibirlo, es necesario encontrarse en la misma posición que éste). Es por ello que una segunda condición de parada es haber recorrido todas las esquinas y no haber encontrado un victimario.

El escenario utilizado para estos test es presentado en la Figura 5. La estructura es muy sencilla: un cuadrante de nivel alto que contiene a un subcuadrante, de nivel medio, el cual contiene cuatro esquinas interconectadas según lo indican las líneas. La esquina de bajada del subcuadrante es A1M1B2. El victimario se encuentra en A1M1B7, y los operadores tienen el siguiente orden de precedencia: IdentificarVictimario, MoverLocacionN, MoverLocacionE, MoverLocacionS, MoverLocacionO, Bajar, Subir.

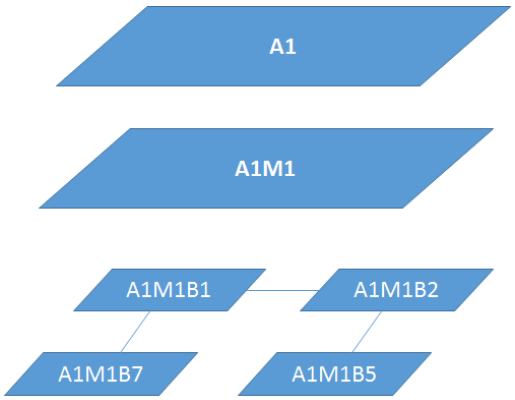


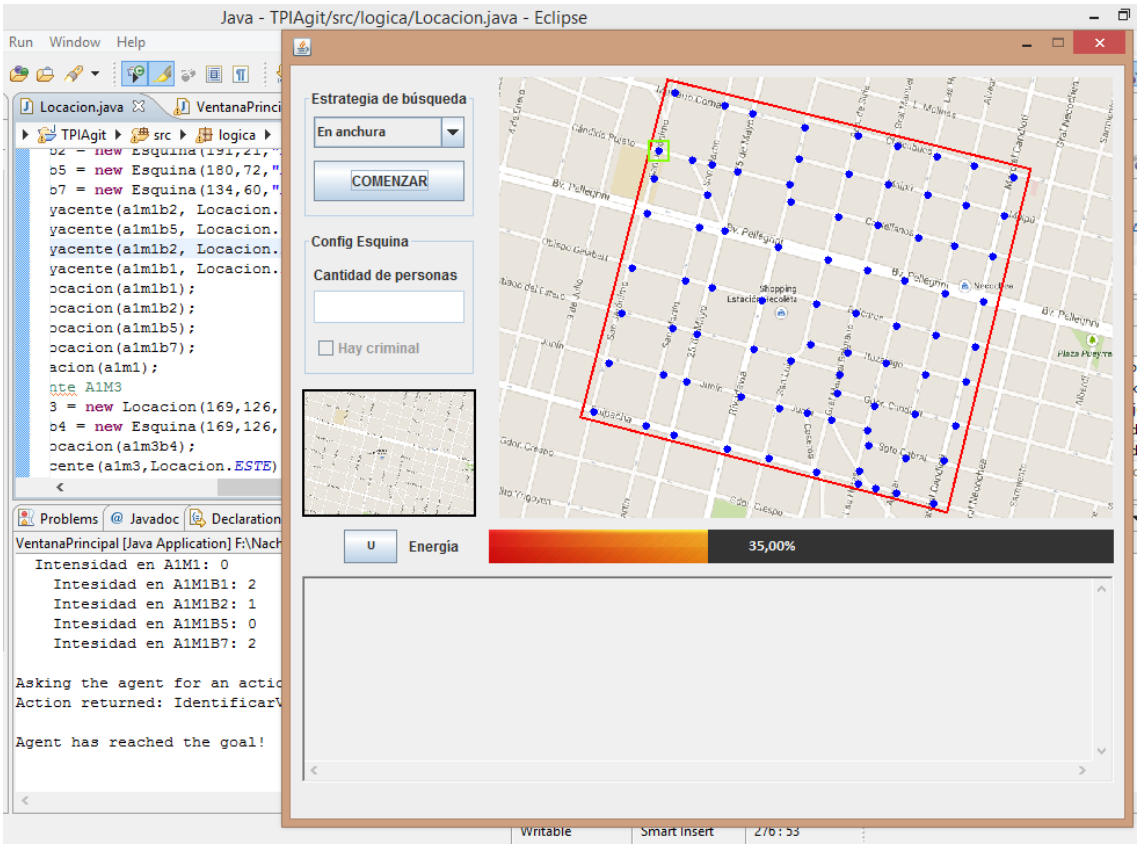
Figura 5 – Escenario de prueba de estrategias

3.1 Prueba "Búsqueda en Anchura" (entorno desconocido)

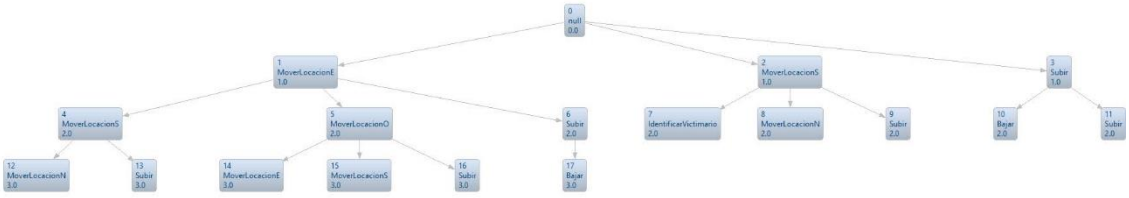
```
Los pasos que el drone siguió en esta prueba fueron:

1 - Action returned: Bajar
2 - Action returned: Bajar
3 - Action returned: MoverLocacionS
4 - Action returned: IdentificarVictimario
5 - Action returned: MoverLocacionN
6 - Action returned: MoverLocacionO
7 - Action returned: IdentificarVictimario
8 - Action returned: MoverLocacionS
9 - Action returned: IdentificarVictimario

Agent has reached the goal!
```

Aquí puede observarse claramente el comportamiento de un ¡a estrategia por anchura: en un prinipio sólo se ejecuta la acción de bajar (la única posible). Luego, en orden de precedencia se evalúan los operadores. El primero que brinda el camino a cumplir el objetivo el drone es la acción "Mover al Sur", que no necesariamente es la opción más óptima. Del mismo modo, el mecanismo de búsqueda continúa expandiendo los nodos en el orden de precedencia siempre que se cumplan las precondiciones para ejecutarlo.

El estado final de la simulación es el siguiente:



A continuación analizamos el árbol de búsqueda de esta estrategia, cuando el drone se encuentra en el estado posterior a ejecutar el paso 6 previamente mencionado:



En él podemos observar cómo las acciones se evalúan en orden de precedencia; la primera acción posible es "MoverLocacionE" (dado que la esquina A1M1B1 no tiene esquinas al norte); luego

Gómez, José Ignacio – Hechim, Pablo Nicolás – Paciuk, Sebastián Federico

le sigue "MoverLocacionS"; no hay ningún otro operador que pueda ejecutarse en esta etapa, más que subir.

3.2 Prueba "Búsqueda en Profundidad" (entorno desconocido)

El resultado que obtenemos es el mismo que con la prueba 3.1, con la diferencia en el comportamiento del drone. En este caso, como la estrategia de profundidad siempre expande el nodo de más a la izquierda, el drone comenzará a ejecutar una acción cíclica, hasta que vea que no tiene otra opción más que reanudar su camino hacia su objetivo.

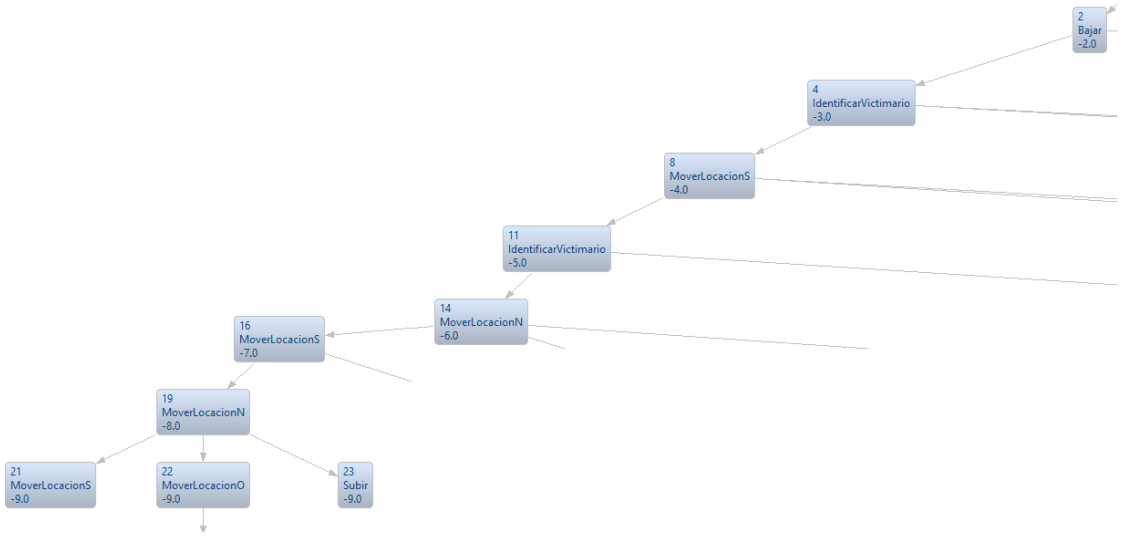
Estas son las acciones ejecutadas con la estrategia de búsqueda en profundidad:

```
1 - Action returned: Bajar
2 - Action returned: Bajar
3 - Action returned: IdentificarVictimario
4 - Action returned: MoverLocacionS
5 - Action returned: IdentificarVictimario
6 - Action returned: MoverLocacionN
7 - Action returned: MoverLocacionS
8 - Action returned: MoverLocacionN
9 - Action returned: MoverLocacionO
10 - Action returned: IdentificarVictimario
11 - Action returned: MoverLocacionS
12 - Action returned: IdentificarVictimario

Agent has reached the goal!
```

Ciclo Mover N-S; se da por la precedencia de estos operadores sobre el resto.

Sale del ciclo porque no le queda mucha energía, y si sigue en el ciclo no llegará a su objetivo.

Este comportamiento cíclico se puede observar mejor en el árbol de búsqueda:



Aún así, logra llegar al objetivo.

3.3 Prueba "Búsqueda por Costo Uniforme" (entorno desconocido)

Esta estrategia busca el camino más corto en cuanto a un costo definido que, en nuestro caso, es la energía del Drone. Como todas las esquinas de nivel inferior poseen personas, la diferencia de costo en este escenario viene dado por el número de movimientos que el drone realiza. Veamos cuáles son los pasos que éste ejecuta para llegar al objetivo:

```
1 - Action returned: Bajar
2 - Action returned: Bajar
3 - Action returned: IdentificarVictimario
4 - Action returned: MoverLocacionS
5 - Action returned: IdentificarVictimario
6 - Action returned: MoverLocacionN
7 - Action returned: MoverLocacionO
8 - Action returned: IdentificarVictimario
9 - Action returned: MoverLocacionS
10 - Action returned: IdentificarVictimario

Agent has reached the goal!
```

Interpretando lo anterior, el Drone descendió al nivel bajo y, en vez de ir en dirección Oeste por el camino más corto (y menos costoso) hacia el victimario, decide ir primero hacia el sur. ¿Por qué sucede esto?

A esto nos referíamos cuando decíamos que en la etapa de búsqueda el Drone no conoce que en esa esquina hay un victimario; según su percepción al descender al nivel inferior, sólo conoce la cantidad de personas en cada esquina, pero noen cuál se halla el victimario. Es por ello que la meta que busca cumplir es la de recorrer todas las esquinas y, dado el orden de precedencia de los operadores, decide ir al sur primero.

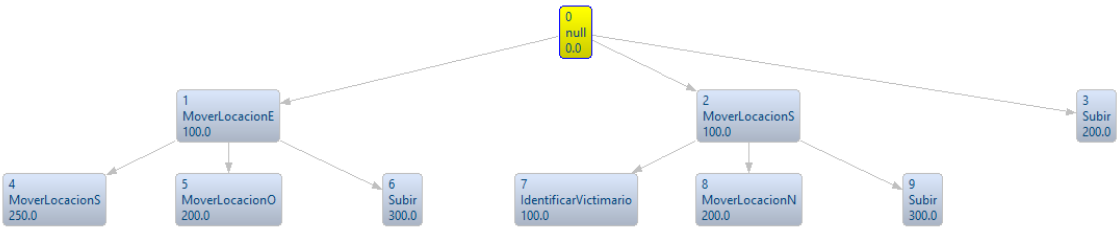
A continuación analizamos qué sucede cuando el Drone conoce dónde se halla el victimario.

3.4 Prueba "Búsqueda por Costo Uniforme" (entorno conocido)

Ahora que el drone conoce el ambiente, este es el proceso que sigue para llegar al victimario:

```
1 - Action returned: Bajar
2 - Action returned: Bajar
3 - Action returned: MoverLocacionO
4 - Action returned: MoverLocacionS
5 - Action returned: IdentificarVictimario

Agent has reached the goal!
```

Se puede observar como rápidamente el Drone converge a su objetivo.



Del árbol podemos ver también como la estrategia va a cumulando el costo de cada movimiento.

3.5 Prueba "Búsqueda A" (entorno desconocido)*

Esta última es una estrategia informada; la misma se vale de una "heurística", una estimación del costo restante hasta alcanzar el objetivo, para conducir el drone por un camino que estima es el mejor para que éste alcance su meta. También se vale de la función de costo, para poder medir cuan perjudicial son las decisiones que toma.

Esta es la secuencia de pasos que ejecuta el drone utilizando esta estrategia:

```
1 - Action returned: Bajar
2 - Action returned: Bajar
3 - Action returned: MoverLocacionO
4 - Action returned: MoverLocacionS
5 - Action returned: IdentificarVictimario

Agent has reached the goal!
```

Vemos que a pesar que el drone no conoce en qué posición exacta se encuentra el victimario, puede estimar dicha posición. Basándose en las potencias de la señal recibida por la antena (siempre busca la mayor potencia) y considerando también la energía disponible, el drone consigue hacer ciertas asunciones o estimaciones que le permiten llegar hasta el objetivo.

4. Conclusiones

La realización del presente trabajo nos permitió llevar la teoría al ámbito práctico para la construcción de un agente inteligente. Resultando muy fructífero desde el punto de vista que asentó bases de conocimiento, demandó análisis intensos y mucha interrelación de los contenidos vistos en clases. Desde la práctica logramos asimilar de forma más directa cómo funcionan las estrategias de búsquedas en un caso real.

Consideramos que el modelo propuesto para resolver problema fue bueno y determinante para el desarrollo del trabajo práctico. Nos resultó fácil de usar, intuitivo y de mucha ayuda. Es una buena manera de modelar un sistema, ya que, busca explotar los rasgos característicos de un agente y su relación con el ambiente.

Cabe destacar que como el agente se desenvuelve en un ambiente que inicialmente no conoce, las estrategias de búsquedas propuestas no siempre encuentran la solución óptima del problema.

Positivamente sentimos haber alcanzado el objetivo planteado: *"comprender cómo se relaciona el agente inteligente con el mundo en el que desenvuelve y cómo utiliza las técnicas aprendidas para la toma de decisiones sobre las acciones que puede emprender."*

Dejamos a modo de resumen, de la experiencia en el práctico, el siguiente cuadro de ventajas y desventajas sobre las estrategias de búsquedas:

Búsqueda	Ventajas	Desventajas
A*	✓ Precisa. ✓ El costo se estima por una función heurística que definimos, sencilla y efectiva. ✓ La idea de minimizar el costo, permite que se elija siempre el mejor camino desde el origen hacia el objetivo.	✓ No es óptima. ✓ No es completa si existen bucles. ✓ Mantiene todos los nodos en memoria.
Primero en Profundidad	✓ Efectiva. ✓ Recorre los nodos de manera ordenada pero no uniforme.	✓ No es completa. Falla en espacios infinitos. Se controlan los estados repetidos. ✓ No es óptima, recorre todos los caminos posibles en el árbol para intentar hallar el estado final.
Anchura	✓ Efectiva si el costo del camino es una función no decreciente de la profundidad del nodo. ✓ Es completa.	✓ Los requisitos de memoria son más grandes que los de tiempo de ejecución. ✓ Los problemas de búsqueda de complejidad exponencial no pueden resolverse por métodos sin información, salvo en casos pequeños.
Costo uniforme	✓ Es completa y óptima si se aseguran costos mayores o iguales a una constante *k* positiva.	✓ Su complejidad no puede ser fácilmente caracterizada porque está dirigida por los costos de los caminos más que por las profundidades. Explora un árbol grande en pequeños pasos antes de explorar caminos que implican pasos grandes y quizás útiles.

A pesar de las ventajas y desventajas de cada tipo de búsqueda, las cuatro permitieron resolver el problema propuesto.

5. Bibliografía

1. S. Russell, P. Norvig; **Inteligencia Artificial. Un enfoque moderno.** Prentice Hall (1996).
2. Ejemplos de Agentes Inteligentes implementados, ofrecidos por la cátedra Inteligencia Artificial, FRSF, UTN.