Agente de Búsqueda: Tron

Rojas Fredini Emmanuel
Bedrij Walter

11 Mayo, 2011

Resumen

Con el objeto de estudiar la inteligencia artificial se propone el diseño de un agente de tipo búsqueda. Se analiza la representación del mundo y la toma de decisiones.

1. Enunciado

Suponga que se desea emular el juego Tron, en el cual Kevin Flynn debe guiar su "light cycle" azul en un recuadro de 10x10 celdas, mientras debe evitar chocarse los muros y los rastros (muros de luz) creados por él mismo y por su oponente. Kevin debe maniobrar rápidamente para forzar a su oponente a chocar contra los muros.

Cada vez que Kevin realiza un movimiento, su enemigo también lo hace. Kevin y su enemigo solo pueden moverse vertical u horizontalmente. ¿Puedes ayudar a Kevin a vencer a su oponente utilizando búsqueda?

2. Representación del mundo

El mundo es una grilla cuadrada de 10×10 casillas. Cada casilla tiene información acerca del tipo de objeto que la ocupa: *estela*, *moto*, *vacío* o *borde*. Los competidores se ubican en esquinas opuestas y el movimiento es alternado.

2.1. Caracterización del ambiente

Cada agente sólo puede observar una porción limitada del mundo, un entorno de 3×3 casillas centrada en el agente.

Las acciones del agente son determinísticas. Pudiendo elegir una de las casillas adyacentes. Si la casilla de destino se encuentre *vacía*, el agente se mueve a esa posición, caso contrario muere.

El objetivo es sobrevivir más tiempo que el contrincante.

2.2. Comunicación entre el ambiente y el agente

El ambiente impacta los sentidos del agente enviándole una copia fiel de su entorno próximo, además de sus coordenadas. Como respuesta a esto el agente devuelve una acción posible. El ambiente entonces ejecuta la acción, modificando el mundo.

3. Planificación

Como el objetivo del juego es sobrevivir, el objetivo de la búsqueda es recorrer la mayor cantidad de casillas. Considerando que sólo una porción del mundo es observable en cada momento, se realizan algunas suposiciones.

- A las casillas se les asigna un estado inicial.
- La toma de decisiones se realiza en base a ese estado.
- El estado de una casilla sólo cambia mediante observación directa.
- No se toma en cuenta al rival.

3.1. Búsqueda

Según el estado inicial que se les asigne a las casillas desconocidas serán los algoritmos de búsqueda.

Con el fin de evitar un comportamiento previsible (dentro de lo posible) se intenta la aplicación de los operadores en orden aleatorio.

3.1.1. Estado interno

El estado interno se compone de la posición del agente en ese momento y de una representación interna del mundo, como una matriz de celdas libres u ocupadas.

3.1.2. Prueba de meta

Alcanzar un estado donde ya no sea posible moverse (casillas adyacentes ocupadas). Además se desea llegar a ese estado en la mayor cantidad de movimientos posible.

4. Implementación

4.1. Hipótesis Todo bloqueado

Se asume que el mundo desconocido presenta un estado de bloqueado. Esto reduce la cantidad de casillas a considerar en la búsqueda, lo suficiente como para implementar *búsqueda a lo ancho*.

A medida que el agente explora el mundo, el estado de las casillas se modifican y la búsqueda se ralentiza al tener que ser cada vez más profunda.

Debido a que en un principio la cantidad de opciones es ínfima, el comportamiento se vuelve previsible (comparable a un *agente reflejo*). Sin embargo cuando se encuentra con los obstáculos introducidos por el agente enemigo, la configuración regular del mundo se rompe.

4.2. Hipótesis Todo libre

Se asume que el mundo desconocido presenta un estado de libre. Para realizar la búsqueda en tiempos aceptables, como en el caso general(por ejemplo el comienzo) la ramificación del árbol de búsqueda es considerablemente frondoso, es decir tiene a lo sumo 4 operadores aplicables donde en la mayoría de las casillas se aplican los 4, para acelerar la búsqueda se utilizo un método de búsqueda en profundidad, además lo único que determina que tan bueno es un camino es su longitud.

Por otro lado la grilla es considerablemente grande, de 10x10, por eso para evitar calcular los arboles en profundidad a un nivel tan grande se opto por usar una búsqueda en profundidad limitada. El limite de la búsqueda se establece ante cada ejecución de las acciones por parte del agente utilizando una heurística. Además hay casos donde no es posible realizar un camino establecido por el limite de profundidad, en caso de que la búsqueda a un nivel de profundidad falle, como lo que queremos es recorrer un camino 'largo' entonces lo que hacemos es quedarnos con el camino mas largo que hallamos encontrado.

Definiremos un par de valores para luego explicar la heurística de limite de profundidad:

Sean

- α = profundidad de búsqueda
- *casillas 'recorribles'* un parámetro de la heurística

La heurística es como sigue:

- Si *casillas 'recorribles'* $>$ α entonces uso búsqueda con profundidad α
- Si *casillas 'recorribles'* $\leq$ α entonces uso búsqueda con profundidad *casillas 'recorribles'*

Las *casillas 'recorribles'* es una estimación(no necesariamente correcta) de las casillas que la moto podrá recorrer desde su posición actual y según su visión del mundo. Para calcular esta estimación usamos el algoritmo *flood fill*, este sirve para determinar el área(en nuestro caso *casillas 'recorribles'*) conectada a un nodo determinado (en nuestro caso la posición de la moto) en un arreglo. Cabe señalar que por mas que por medio de este método se calcule un conjunto de casillas, eso no significa que esas casillas sean verdaderamente recorribles ya que la moto va dejando la estela, la cual imposibilita el volver a circular por el mismo camino, por eso mismo que es una estimación.

Ese cambio de la profundidad de búsqueda tiene como fin acelerar la búsqueda. El problema se da en un escenario como el siguiente:

- Una grilla de 6×6
- Un $\alpha = 40$

Vemos que la profundidad de búsqueda es mayor que la cantidad de casillas que se pueden recorrer, incluso desde el principio. Entonces según nuestro método, si no existiese la heurística, se buscarían los caminos de longitud 40, pero como no existe

en un ambiente tan pequeño un camino tan largo la búsqueda fallara, luego usaremos el camino mas largo del árbol de búsqueda creado. El problema esta en que solo necesitamos un camino de 35 niveles, en el comienzo, y crear todos los posibles caminos de 40 niveles para luego desecharlos es un desperdicio de tiempo.

En cambio la estimación de *casillas 'recorribles'* nos dice en un comienzo que las grillas recorribles son 35, luego estableciendo esta como α y la búsqueda no fallara (en este caso) y dará un resultado mas rápidamente.

4.2.1. Un ejemplo sencillo

En la figura 1 se muestra una moto, como un circulo azul, y varias posiciones ocupadas por estelas, como rectángulos rojos, y un par de caminos, como rectángulos negros. Podemos ver que al agente se le presentan 2 posibles caminos, uno mas largo que el otro, aunque con la visibilidad del agente no es posible decir cual es el largo a priori.

La profundidad de búsqueda fue establecida a 3. En la figura 2 se muestra el árbol en la primer iteración, el camino de los nodos con contorno azul es el camino que seleccionó la búsqueda:

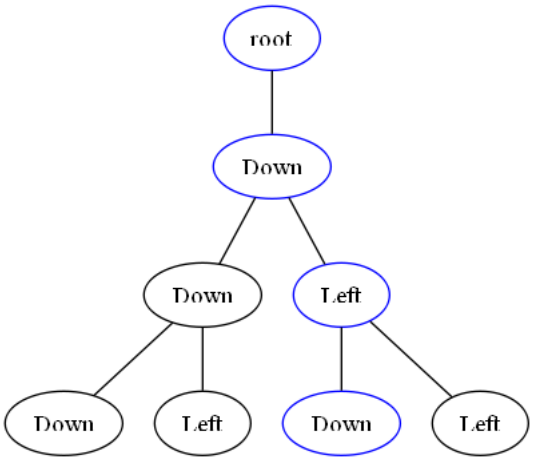

Figura 2: Árbol de búsqueda

5. Figuras

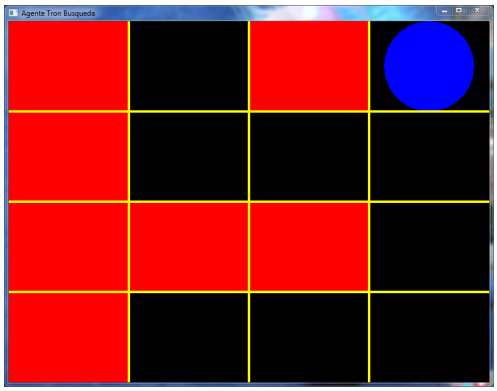

Figura 1: Mapa del ejemplo

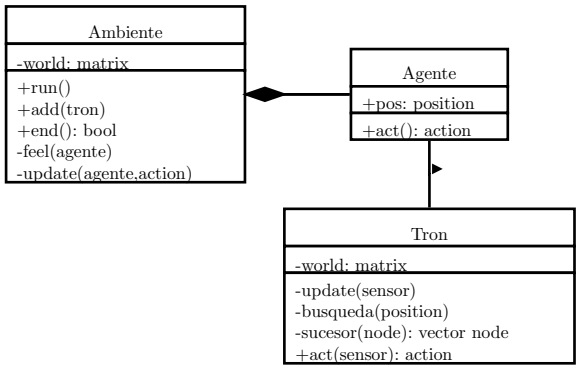

Figura 3: Diagrama de clases