Universidad de Granada

Método de Búsqueda con Adversario (Juegos)

Nota:

En la 'practica3.zip' están contenidas dos carpetas:

- Carpeta 'Memoria':_ la cual contiene la memoria de prácticas.
- Carpeta 'Ficheros':_ contiene:
player.cpp y player.h
player76656695.cpp y player76656695.h
Fichero proyecto CodeBlocks aspiradoras2.cbp

1. Análisis del problema.

Nos encontramos con un entorno **multi-agente**, donde dos aspiradoras intervienen simultáneamente.

Tenemos que las acciones que realiza una aspiradora influyen en la percepción y toma de decisiones de la otra. El entorno es **competitivo**, es decir, las aspiradoras tienen metas contrapuestas.
Las aspiradoras perciben el estado actual, determinan la acción a realizar que les permita el mayor beneficio, entendiendo por beneficio el número de unidades de basura conseguida.

Las aspiradoras siguen un proceso deliberativo, basado en una búsqueda con adversario en un espacio de estados representado por un árbol de juego.

Nos encontramos con el problema de la eficiencia en la limpieza de dos aspiradoras que compiten entre sí en las cuales se implementó un agente reactivo.

El entorno es el mapa, el cual está compuesto por 'paredes' que son las casillas ocupadas, los obstáculos; y los 'pasillos' en los cuales se encuentra la suciedad. Los cuales formarán parte del problema.
En nuestra práctica tenemos que diseñar e implementar las técnicas de búsqueda con adversario que se narran en Nilsson o Russell.

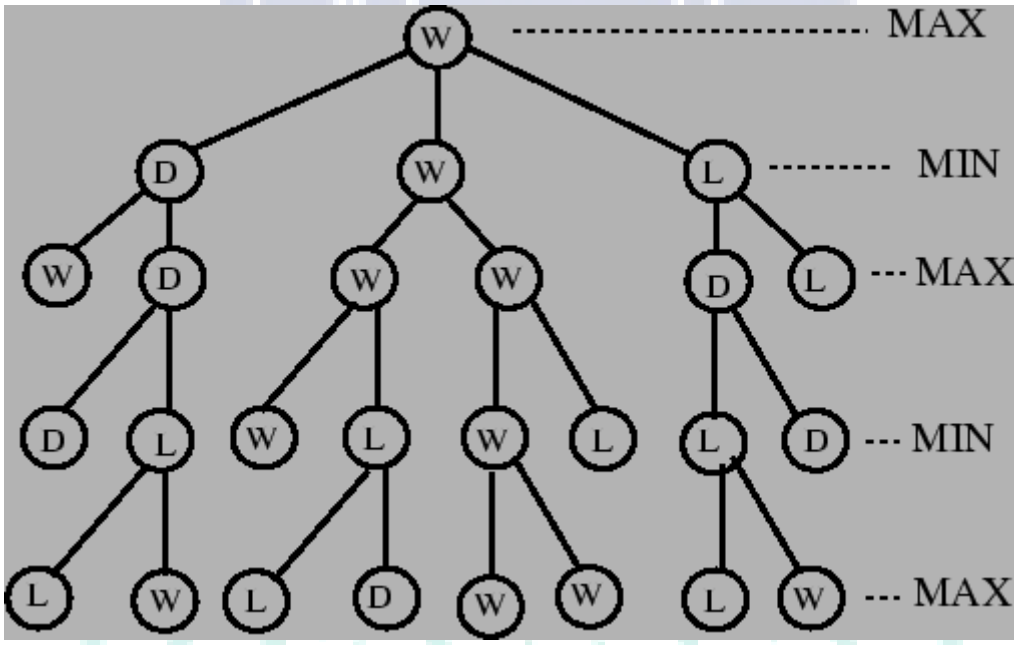
Tenemos que elaborar un programa ejecutable que represente dos aspiradoras en un entorno dado (que se corresponde con mapa de tamaño determinado con sus correspondientes bordes) Ésta cuadrícula contiene una serie de obstáculos y casillas libres en su interior donde se encuentra la suciedad a limpiar.

La aspiradora debe de ganar a su contrincante.
Como restricción tenemos que la aspiradora no puede volver a una casilla ya visitada anteriormente, por la que ya ha pasado y está limpia.
Por lo tanto, teniendo en cuenta la restricción anterior, la aspiradora puede quedar encerrada en un espacio.
Debe de quedar el menor número de suciedad posible.
La suciedad aspirada debe de ser mayor que la de la aspiradora contrincante.

También se ha de tener en cuenta que la suciedad no vuelve a aparecer en el sitio que se ha visitado.
El comportamiento del robot debe intentar optimizar la limpieza de todas las casillas de la habitación.

2. Descripción de la solución propuesta.

ESTRATEGIA MINIMAX Y HEURÍSTICA:



Consiste en la expansión del nodo padre en nodos hijos, generando un árbol de juegos, la estrategia óptima se determinaría examinando el valor minimax de cada nodo.

MAX: Este valor en un nodo es la utilidad de estar en el estado correspondiente, asumiendo que ambos jugadores juegan en mejor de sus casos posibles, el más óptimo desde ahí hasta el final del juego. Esta opción óptima lleva al valor minimax más alto al sucesor.

Debe encontrar el mejor movimiento para *MAX el cual* realiza el primer movimiento, después los jugadores van realizando movimientos de forma alternativa en sus correspondientes turnos.

Ambas aspiradoras tienen el mismo número de movimientos en cada turno.

MAX se mueve a un estado de valor máximo.

MIN es el contrario a MAX, MIN son los nodos con profundidad impar.

MIN a su vez juega también óptimamente, se mueve a un estado de valor mínimo.

Implementamos el algoritmo minimax, en el cual usamos unos cálculos simples recurrentes de los valores minimax de cada uno de los estados sucesores.

Esta recursión va avanzando hacia los extremos del árbol donde los valores minimax retroceden por el árbol cuando la recursión se deshace poco a poco.

Va realizando exploraciones de profundidad completa del árbol generado.

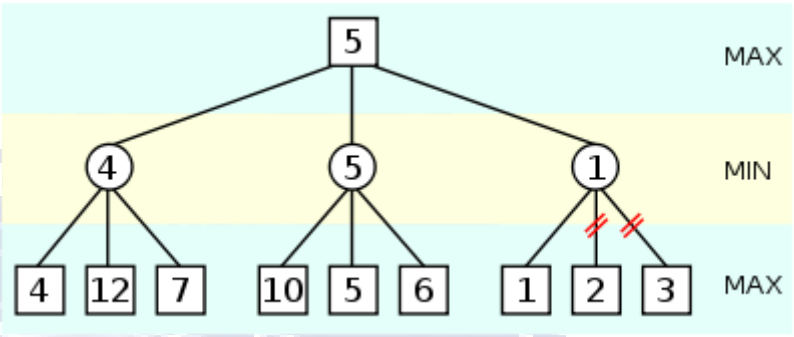
Las primeras fases del juego son las más sencillas ya que el uso de las simetrías nos proporciona una ramificación baja.

Su funcionamiento resumidamente se basa en ir expandiendo nodos desde el nodo padre hasta los hijos, el mejor movimiento que hace MAX es el que evita que perdamos. Cuando MAX haya realizado este movimiento, MIN prevé que en la próxima jugada MAX será el ganador por lo que abandona inmediatamente.

Tomamos el valor final de la heurística con la cual me quedo y a partir de eso elegimos a unos de los hijos sucesores o bien ir cambiando de mejor acción. Una vez que nos quedamos con el mejor valor que obtenemos de aplicar valorMin valorMax recursivamente recorriendo el árbol de estados.

El algoritmo minimax consiste en ir recorriendo un árbol de estados y cada estado tiene un valor pero para obtener ese valor tiene que ser el de un nodo frontera. Entonces sabemos que el valor al que le aplicamos la heurística siempre es un nodo frontera y ese valor lo va devolviendo y nos quedamos con el mejor.

POSIBLE PODA:



Respecto a la heurística, para mejorar la eficiencia de nuestro algoritmo, podríamos utilizar una poda alfa-beta para el árbol que se genera, ya que esta puede aplicarse a árboles de cualquier profundidad.

Al ser la búsqueda minimax primero en profundidad, implica que solo tendríamos que tener en cuenta a la hora de podar los nodos a lo largo.

El parámetro alfa nos da el límite, el valor de la mejor opción en el MAX, el más alto; mientras que el parámetro beta nos da el valor de la mejor opción que tiene el MIN, es decir, el valor más bajo.

Esta búsqueda actualizaría el valor de alfa y beta a medida que vamos recorriendo nuestro árbol y va podando las ramas no útiles-necesarias-restantes en un vértice cuando el valor del vértice actual es peor que el valor nuevo/actual de alfa y beta para el MAX-MIN.