Inteligencia Artificial

E.T.S. De Ingenierías Informática y de Telecomunicación

Practica 2

Métodos de Búsqueda

Curso 2012-2013

Nombre: José Arcos Aneas
D.N.I: 74740565-H

1. Análisis del problema.

Tenemos un mapa conocido que tiene un tamaño variable (no más de 80 por 80), el cual podemos leer para sacar toda la información (paredes, suciedad y grado de suciedad, espacios libres). El mapa está sucio en algunos puntos y no se ensucia mas con el tiempo.

Nuestro objetivo es implementar una serie de métodos para hacer una búsqueda con la información que disponemos para hacer el recorrido más inteligente posible, en un tiempo razonable. Tenemos un método de búsqueda en profundidad en el que nos podemos basar. Lo vamos a ver desde tres perspectivas diferentes, la búsqueda en anchura, búsqueda de costo uniforme y un método de escalada.

2.Descripción de la solución del problema.

Para observar los cambios con mayor claridad incluyo los datos para la búsqueda en profundidad.
Valores búsqueda en profundidad (*Valores para 100 pasos***):**
Este método es mas rápido.
Longitud del plan = 102
Nodos expandidos = 246
Nodos Explorados = 130
Energía Consumida =111

Métodos implementados.

Búsqueda en Anchura.

La búsqueda en anchura asegura encontrar el óptimo aunque por el contrario es mas lento y requiere mayor coste en memoria que la búsqueda en profundidad.

En lugar de utilizar una lista para la implementación se ha procedido a cambiar esa estructura de datos por una cola. El procedimiento consiste en que mientras no se encuentre una solución se generan nuevos estados y si insertamos a sigActions[i] en la lista y a padre p, colocamos p en la cola. Actualizamos q con el frente de la cola y hacemos que actual sea un puntero a q, procedemos a sacar un elemento de la cola. al final pasamos como plan el camino actual.

Valores búsqueda en anchura:

Longitud del plan = 61
Nodos expandidos = 33919
Nodos Explorados = 1421
Energía Consumida =63

Búsqueda de costo uniforme.

Para realizar este método previamente hemos de hacer implementado un método, clase , struct,,, para compara dos elementos de la cola.

Para este método solo hemos cambiado la cola por una cola con prioridad afectada por el comparador. El procedimiento es el mismo que el descrito para la búsqueda en anchura, la única diferencia es que ahora nuestra lista esta actualizada según el comparador que devuelve el mayor de los elementos que le pasamos. Con esto aseguramos que el mejor en coste de energía sea elegido.

Valores para búsqueda costo uniforme:

Longitud del plan = 61
Nodos expandidos = 30825
Nodos Explorados = 12734
Energía Consumida = 63

Búsqueda Escalada

El método de escalada evalúa las posibilidades desde el estado actual (descendientes) dándole un valor numérico, que es generado por la función heurística, a cada posible acción, eligiendo entre ellas la que más le convenga, en este caso he usado el método de escalada por la máxima pendiente, es decir, al generar todos los posibles acciones y asignarle un valor numérico, se ve cual de las acciones es la mejor (tiene un valor numérico menor), escoger esa acción y generar los descendientes después de coger la solución. El bucle continuará hasta que la suciedad del mapa es cero. El método de heurística que he utilizado se basa en una distancia a la suciedad más cercana, mas ver si esta cerca de una pared.

El método heurística es básicamente $heurística = h + g$ dónde g = suciedad pendiente, h = Get_h

La Suciedad_mas_cercana, es un método que mira su posición en el mapa, y ve casilla por casilla si tiene suciedad, de tenerla calcula la distancia con la posición actual. La distancia se guarda en un double y se calcula como la raíz cuadrada de las suma al cuadrado de las coordenadas del robot menos las coordenadas de la suciedad. Es decir sean (Rx, Ry) las coordenadas del robot en el mapa y (Sx, Sy) las coordenadas de la suciedad en el mapa, la distancia del robot a esa suciedad es: El método devuelve 3 veces está distancia, para marcar más la diferencia entre una y suciedad y otra.

Get_h, nos devuelve h, que es una variable auxiliar, nos ayuda que camino elegir, teniendo en cuenta que normalmente vale 1. Si la acción que puede hacer es limpiar esto vale -1000 (para que siempre limpie).