SEXTA PRÁCTICA

PLANIFICACIÓN DE CAMINOS II (A*)

MIGUEL IAN GARCÍA POZO

GIERM

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

Índice

Algoritmo A*

- Desarrollo

Este algoritmo de planificación de caminos es, en esencia, muy parecido al de Dijkstra. La diferencia está en que Dijkstra busca la solución recorriendo todos los nodos del grafo hasta encontrar el camino óptimo mientras que A* busca el camino óptimo directamente, sin llegar a "visitar" todos los nodos.

El funcionamiento del algoritmo A* se basa en el uso de la heurística, que consiste en el coste estimado entre el nodo actual y el nodo destino. La fórmula con la que se lleva a cabo este algoritmo es la siguiente:

$$f(n) = g(n) + h(n)$$

Esta fórmula, define el coste total como la suma del coste acumulado desde la salida al nodo más el coste desde el nodo al nodo destino. Como en un problema real no conocemos con precisión cuál es el coste entre el nodo y el nodo destino, la matriz de heurística va a ser una matriz estimada.

$$f^*(n) = g(n) + h^*(n)$$

Donde el asterisco indica que es un coste estimado, no el real.

Como el algoritmo A* es muy parecido al de Dijkstra, podemos usar el mismo código cambiando lo necesario.

El problema nos indica que debemos recorrer el grafo de la Figura 1 usando la matriz H de la heurística

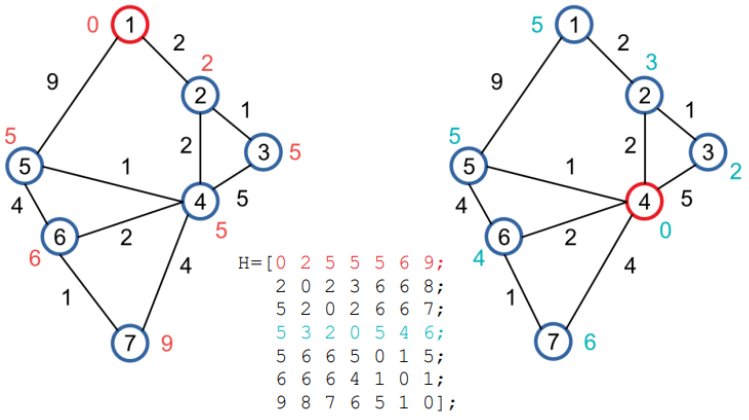

Figura 1.- Grafo con los valores de la heurística

Veamos en las Figuras 2 y 3 cómo es el código:

```
function [coste,ruta] = Aestrella(g,h,s,x)


    [m,n] = size(g); % Obtenemos el tamaño del grafo de entrada

    caminos = inf(2,m); % Definimos un vector de infinitos
    % Creamos una matriz con: columna de nodos, columna de costes, columna de costes con la heurística
    % y columna de nodo anterior
    caminos = [uint32(1):uint32(m);caminos;zeros(1,m)]';
    % Cambiamos la primera fila por la del nodo inicio y cambiamos su coste
    % a cero
    caminos([1 s],:) = caminos([s 1],:);
    caminos(1,2) = 0;
    caminos(1,3) = h(x,s);

    % Creamos la matriz donde vamos a guardar los nodos que ya hemos
    % revisado y la matriz donde pondremos la ruta solución
    revisado = zeros(m,4);
    solucion = zeros(m,4);

    % i es el nodo que estamos revisando y p es la posición en la matriz de
    % revisados donde vamos a poner el nodo que acabemos de ver
    i = s;
    p = 1;
```

```
    while i ~= x
        % Creamos un bucle para recorrer las columnas de la matriz del
        % grafo
        for j = 1:length(g(1,:))
            % Este bucle nos devolverá la fila donde está ubicado el nodo
            % que buscamos en la matriz "caminos"
            for l = 1:length(caminos(:,1))
               if (caminos(l,1) == j)
                    nodo = l;
               end
            end
            % Comprobamos si el elemento que estamos viendo es distinto de
            % cero y si el coste hasta el nodo es menor que el coste
            % anterior y guardamos la información en la matriz "caminos"
            if (g(i,j) ~= 0 )
                if (g(i,j)+caminos(1,2) < caminos(nodo,2))
                    caminos(nodo,2) = g(i,j) + caminos(1,2);
                    % Guardamos el coste con la heurística
                    caminos(nodo,3) = g(i,j) + caminos(1,2) + h(x,j);
                    caminos(nodo,4) = i;
                end
            end
        end
        % Ordenamos la matriz, guardamos en "revisado" el nodo y eliminamos
        % la primera fila para no volver a comprobarlo.
        g(:,caminos(1,1)) = 0;
        revisado(p,:) = caminos(1,:);
        p = p+1;
        caminos(1,:) = [];
        % Reordenamos la matriz en función del coste con heurística menor
        caminos = sortrows(caminos,3);
        i = caminos(1,1);
    end
```

Figura 2.- Primera parte del código del algoritmo A*

```
    revisado(m,:) = caminos(1,:);

    solucion(1,:) = revisado(m,:);
    ant = m;
    % Este bucle se va a encargar de recorrer la matriz "revisado" y
    % encontrar cuál es la ruta a seguir
    for j = 2:length(revisado)
        for l = 1:length(revisado(:,1))
           if (revisado(l,1) == revisado(ant,4) && revisado(ant,4)~= 0)
                ant = l;
                solucion(j,:) = revisado(l,:);
           end
        end
    end

    % Elimino las filas de la matriz que siguen valiendo 0
    solucion(sum(abs(solucion),2)==0,:)=[ ];
    solucion(:,sum(abs(solucion),1)==0)=[ ];

    % Defino el coste y la ruta, siendo esta la columna de los nodos de la
    % solución invertida, para poder ver el recorrido desde la salida a la
    % llegada
    coste = solucion(1,2);
    ruta = fliplr(solucion(:,1)');
end
```

Figura 3.- Segunda parte del código

Mirando el código en las Figuras 2 y 3 vemos que es muy parecido al de Dijkstra, con la diferencia de que calculamos la heurística y cuando ordenamos las filas de la matriz es en función de esta.

- Simulación

Simulemos con el ejemplo del enunciado en la Figura 4.

La función debe dar como resultado: **coste = 7**
ruta = [1 2 4 6 7]

```
>> [coste,ruta] = Aestrella(G,H,1,7)

coste =

     7


ruta =

     1     2     4     6     7
```

Figura 4.- Simulación del ejemplo del enunciado

Vemos que la simulación ha dado como resultado el camino óptimo. En este caso podríamos considerar que la heurística es buena, pero más adelante veremos que no es así.

Prueba de la Heurística

Ahora queremos comprobar si la heurística es buena, para ello probemos el caso que nos pide el enunciado en la Figura 5.

```
>> [coste,ruta] = Aestrella(G,H,7,4)

coste =

     4


ruta =

     7     4
```

Figura 5.- Ejemplo con resultado erróneo del algoritmo A*

Vemos que en este caso, el algoritmo ha creído que el camino óptimo era 7 → 4 cuando realmente el óptimo es 7 → 6 → 4. Este error ha ocurrido debido a que la heurística estimada no cumple la condición de ser menor o igual a la heurística real. Esto lo vemos en que el nodo seis tiene un coste estimado hasta el nodo 4 de valor 4, cuando el coste real es 2. Esto hace que el algoritmo crea cómo óptimo el camino 7 → 4. Por lo tanto, podemos concluir que la heurística del ejercicio no es buena.

Heurística Admisible

Hemos visto que la heurística que nos ha dado el enunciado no es admisible, por esa razón vamos a cambiar la matriz anterior por una nueva que cumpla la desigualdad:

$$h^{*}(n) \leq h(n)$$

De la matriz del enunciado, pasamos a la matriz nueva en la Figura 6.

```
H =                                                  H1 =

     0     2     5     5     5     6     9              0     2     2     3     4     5     6
     2     0     2     3     6     6     8              1     0     1     2     3     4     4
     5     2     0     2     6     6     7              3     1     0     2     3     4     5
     5     3     2     0     5     4     6              3     2     3     0     1     1     3
     5     6     6     5     0     1     5              5     3     4     1     0     3     4
     6     6     6     4     1     0     1              5     4     4     2     3     0     1
     9     8     7     6     5     1     0              7     4     5     3     3     1     0
```

Figura 6.- Matriz del enunciado y propuesta de la heurística del grafo

Probemos que tal funciona la matriz de heurística nueva de la Figura 6 en la Figura 7.

```
>> [coste,ruta] = Aestrella(G,H1,7,4)

coste =

     3


ruta =

     7     6     4
```

Figura 7.- Resultado correcto del caso revisado anteriormente

Vemos que ahora sí encuentra el camino óptimo en el que antes fallaba así que esta heurística es, por ahora, mejor que la anterior, sigamos probando en la Figura 8 para confirmar que es buena.

```
>> [coste,ruta] = Aestrella(G,H1,7,3)

coste =

     6


ruta =

     7     6     4     2     3
```

```
>> [coste,ruta] = Aestrella(G,H1,5,3)

coste =

     4


ruta =

     5     4     2     3
```

```
>> [coste,ruta] = Aestrella(G,H1,1,5)

coste =

     5


ruta =

     1     2     4     5
```

Figura 8.- Tres casos de búsqueda del camino óptimo con los resultados esperados

Vemos que en todos los casos de la Figura 8, el algoritmo A* está encontrando el camino óptimo, por lo que podemos concluir que A* funciona y es más eficiente computacionalmente que Dijkstra.

Enlace a Github

https://github.com/MiguelIlan/AmpliacionRobotica/tree/main/Rob%C3%B3tica%20m%C3%B3vil/PlanCaminosA