

SEXTA PRÁCTICA

PLANIFICACIÓN DE CAMINOS II

(A*)

MIGUEL IAN GARCÍA POZO

GIERM
ESCUELA DE INGENIERÍAS INDUSTRIALES



UNIVERSIDAD DE MÁLAGA

Índice

Algoritmo A*	3
- Desarrollo.....	3
- Simulación.....	5
Prueba de la Heurística	6
Heurística Admisible	6
Enlace a Github	7

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:

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

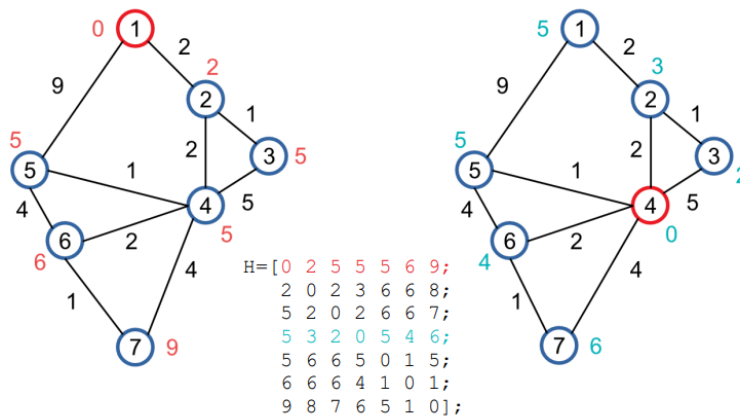
Esta fórmula, define el coste total como la suma del coste acumulado desde la salida al nodo más el coste desde el nodo al nodo destino. Como en un problema real no conocemos con precisión cuál es el coste entre el nodo y el nodo destino, la matriz de heurística va a ser una matriz estimada.

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

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 siguiente grafo usando la matriz H de la heurística



Veamos cómo desarrollamos 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(camino(:,1))
            if (camino(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 "camino"
        if (g(i,j) ~= 0 )
            if (g(i,j)+camino(1,2) < camino(nodo,2))
                camino(nodo,2) = g(i,j) + camino(1,2);
                % Guardamos el coste con la heurística
                camino(nodo,3) = g(i,j) + camino(1,2) + h(x,j);
                camino(nodo,4) = i;
            end
        end
        % Ordenamos la matriz, guardamos en "revisado" el nodo y eliminamos
        % la primera fila para no volver a comprobarlo.
        g(:,camino(1,1)) = 0;
        revisado(p,:) = camino(1,:);
        p = p+1;
        camino(1,:) = [];
        % Reordenamos la matriz en función del coste con heurística menor
        camino = sortrows(camino,3);
        i = camino(1,1);
    end
end
```

```

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

```

Mirando el código 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:

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

```

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:

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

```
coste =
```

```
4
```

```
ruta =
```

```
7 4
```

Vemos que en este caso, el algoritmo ha creído que el camino óptimo era $7 \rightarrow 4$ cuando realmente el óptimo es $7 \rightarrow 6 \rightarrow 4$. 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 $7 \rightarrow 4$. 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:

H =

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

H1 =

0	2	2	3	4	5	6
1	0	1	2	3	4	4
3	1	0	2	3	4	5
3	2	3	0	1	1	3
5	3	4	1	0	3	4
5	4	4	2	3	0	1
7	4	5	3	3	1	0

Esta es la matriz de heurística que yo he creado, probemos que tal funciona.

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

coste =

    3

ruta =

    7    6    4
```

Vemos que ahora si encuentra el camino óptimo en el que antes fallaba así que esta heurística es algo mejor que la anterior, sigamos probando.

```
>> [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
```

Vemos que en todos estos casos, 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/MiguelIan/AmpliacionRobotica/tree/main/Rob%C3%B3tica%20m%C3%B3vil/PlanCaminosA>