NOMBRE: Esteban Vivanco

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2613 — Inteligencia Artificial — 1' 2021

Tarea 2 – Respuesta Pregunta 1

Consideramos una implementación de *Weighted* A^* de la siguiente manera

$$f(n) \ = \ g(n) \ + \ w \cdot h(n)$$

Siendo $f(n)$ el costo total de llegar a un nodo n desde el inicio del camino (s), $g(n)$ el costo del camino, $h(n)$ una funcion heuristica que estima el costo del camino optimo desde n hasta el *goal* y w una penalizacion arbitraria que cumple con $w \geq 1$.

1 Desempate

Como se menciono anteriormente, $h(n)$ una funcion heuristica que estima el costo del camino optimo desde n hasta el *goal*, de esta manera, podemos pensar que al elegir un menor $h(n)$ estamos apostando por un mejor "futuro" en el camino. Esto considerando que h es una **heuristica admisible**, es decir, que nunca sobre-estima el costo de una solucion optima desde s hasta el *goal*.

2 Video

En la demostracion se define $\delta(s,t)$ como el costo de un camino optimo entre s y t. De esta manera

A^* tiene optimo y este esta acotado superiormente por $\delta(s_{start}, t_{goal})$
Weighted A^* tiene optimo y este esta acotado superiormente por $w \cdot \delta(s_{start}, t_{goal})$

Notamos que a traves del parametro w estamos modificando que tan suboptimo pueda ser *Weighted* A^*. Luego se propone la siguiente secuencia de inecuaciones. Esta secuencia considera a s^* como un estado intermedio **dentro del camino optimo**, que cumple con $g(s^*) \ = \ \delta(s_{start}, s^*)$.

$$\delta(s_{start}, s^*) \ + \ \delta(s^*, t_{goal}) \ = \ \delta(s_{start}, t_{goal})$$
$$g(s^*) \ + \ \delta(s^*, t_{goal}) \ \leq \ \delta(s_{start}, t_{goal}) \text{ (ya que } s^* \text{ se encuentra en el camino optimo)}$$
$$g(s^*) \ + \ h(s^*) \ \leq \ \delta(s_{start}, t_{goal})$$

Ademas, por la cota superior de optimalidad antes mencionada se cumple que

$$f(s) \ = \ min_{t \in Open} \{f(t)\}$$
$$f(s) \ \leq \ f(s^*)$$
$$f(s) \ \leq \ g(s^*) \ + \ w \cdot h(s^*)$$
$$f(s) \ \leq \ w \cdot (g(s^*) \ + \ h(s^*))$$
$$f(s) \ \leq \ w \cdot \delta(s_{start}, t_{goal}) \text{ (reemplazamos por la definicion de nuestra cota)}$$

Ademas, sabemos que nuestra heuristica h es admisible, por ello cumple con

$$h(s_{goal}) \ = \ 0$$

Por ello, cuando s_{goal} es extraido tenemos que

$$\begin{aligned} g(s^*) \ + \ w \cdot h(s^*) \ &\leq \ w \cdot \delta(s_{start}, t_{goal}) \\ g(s^*) \ + \ 0 \ &\leq \ w \cdot \delta(s_{start}, t_{goal}) \\ g(s^*) \ &\leq \ w \cdot \delta(s_{start}, t_{goal}) \end{aligned}$$

De manera que queda demostrado que la suboptimalidad de *Weighted* A^* esta acotada por w veces el costo optimo que acota a A^* (sin *Weighted*).

En simples palabras, sabiendo que A^* tiene un optimo, se demostro que *Weighted* A^* tambien lo tiene y que este optimo es, en el peor de los casos, w veces el optimo "original" (sin la penalizacion w), a pesar de que en la practica no se llega a ese caso borde.

3 Ineficiente

En esta seccion se propone un *"anti-weighted"* A^* donde, ante un empate, se elige el camino con mayor $h(n)$ o, de manera equivalente, con menor $g(n)$. Se propone el siguiente valor de $f(n)$ (con fines ilustrativos).

$$f(n) \ = \ g(n) \ + \ w \cdot h(n) \ , \quad 0 < w < 1$$

A partir del *hint* del enunciado, es natural pensar en un *Cubo Rubik*. Quienes hemos intentado armarlo sabemos lo "facil" que es no llegar a ninguna solucion si no utilizamos alguno de los algoritmos que nos ensenaron o aprendimos viendo algun video de youtube. En concreto, considerando que el cubo tiene **43 252 003 274 489 856 000 permutaciones posibles**, un algoritmo de resolucion que se base en la aleatoriedad de movimientos (probablemente nuestro primer acercamiento a resolver el cubo), es exponencialmente mas ineficiente que un algoritmo de resolucion que me guie por el camino optimo, como por ejemplo, el utilizado por Max Park en 2020 para resolver el cubo en tan solo 5,90 segundos.

Si bien nuestro *"anti-weighted"* A^* no es directamente aleatorio, al realizar un movimiento en el cubo, nuestro problema a resolver es uno nuevo, distinto al anterior. Consideramos tambien que el grafo de resolucion del cubo tiene ciclos, en la practica, al realizar 4 movimientos identicos de manera consecutiva, volvemos a la posicion anter de realizar el primer movimiento de la secuencia.

Considerando lo anterior y que nuestro *"anti-weighted"* A^* siempre elige como estado proximo a aquel con mayor $h(n)$, notamos que *"anti-weighted"* A^* es exponencialmente mas ineficiente que *Weighted* A^*, ya que, elegir el estado con mayor $h(n)$, por definicion, implica elegir el estado que se encuentra mas alejado del *goal* dentro de los movimientos posibles.

4 astar.py

values/casos	caso base	$w = 5$	$w = 8$	**w = 10**	$w = 12$	$w = 15$
Tiempo Total	479.18	9.08	5.87	**4.60**	7.14	7.58
Expansiones Totales	3, 247, 710	54, 764	33, 238	**23,946**	45, 621	46, 814
Costo Total	452	834	966	**1,070**	1, 162	1, 166

(1)

Notamos que el mejor rendimiento, en base a "Expansiones Totales" y "Tiempo Total" lo obtenemos con $w = 10$. (Ver figuras 1-6 en el Anexo)

5 estimate_suboptimality

Ver figura 7 en el Anexo

6 Anexo

```
→  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1    151817    290844       45     24.29      0.00
    2    170564    320461       42     26.10      0.00
    3    198327    384013       42     29.38      0.00
    4    191259    362370       41     25.77      0.00
    5    620168   1162970       46     88.28      0.00
    6    440524    829358       47     65.20      0.00
    7    410133    786993       44     59.24      0.00
    8    148509    287219       53     20.33      0.00
    9    301944    575488       46     45.48      0.00
   10    614465   1177611       46     95.11      0.00
Tiempo total:         479.18
Expansiones totales: 3247710
Costo total:         452
```

Figura 1: A^*, w = 0
Fuente: Elaboración propia

```
→  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1      1552      3156       95      0.41      0.00
    2     10803     17060       92      1.52      0.00
    3      6674     12126       68      0.97      0.00
    4      3091      6063       77      0.55      0.00
    5      3759      6843       88      0.57      0.00
    6      3300      5893       99      0.52      0.00
    7      1456      2513       72      0.30      0.00
    8      5632     10254       91      0.89      0.00
    9     16944     24814       76      2.87      0.00
   10      1551      3047       76      0.48      0.00
Tiempo total:         9.08
Expansiones totales: 54762
Costo total:         834
```

Figura 2: *Weighted* A^*, w = 5
Fuente: Elaboración propia

```
→  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1       976      1955       87      0.35      0.00
    2      7839     12326      110      1.44      0.00
    3       585      1218       82      0.20      0.00
    4       935      1806       81      0.29      0.00
    5      5716     10167       98      0.86      0.00
    6      2195      4345      113      0.39      0.00
    7      1506      2828       76      0.30      0.00
    8      2853      4449      111      0.46      0.00
    9      4123      7312      104      0.62      0.00
   10      6510     10873      104      0.97      0.00
Tiempo total:         5.87
Expansiones totales: 33238
Costo total:         966
```

Figura 3: *Weighted* A^*, w = 8
Fuente: Elaboración propia

```
➜  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1      3174      5665      107      0.67      0.00
    2      2934      5688      136      0.52      0.00
    3       643      1375      100      0.16      0.00
    4      1102      2126       81      0.34      0.00
    5      1721      3369      116      0.46      0.00
    6      1559      3157      113      0.33      0.00
    7      1451      2902       76      0.24      0.00
    8      2689      5103      121      0.47      0.00
    9      3678      6786      108      0.63      0.00
   10      4995      7872      112      0.78      0.00
Tiempo total:         4.60
Expansiones totales: 23946
Costo total:         1070
```

Figura 4: *Weighted* A^*, w = 10
Fuente: Elaboración propia

```
➜  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1      3251      5965      107      0.68      0.00
    2      3018      5828      136      0.52      0.00
    3      1799      3516      116      0.35      0.00
    4      2364      3803      109      0.47      0.00
    5      3186      5581      102      0.52      0.00
    6      9505     15071      143      1.29      0.00
    7      2128      4144       84      0.38      0.00
    8      9528     14881      147      1.35      0.00
    9      4752      8440      108      0.70      0.00
   10      6090      8395      110      0.89      0.00
Tiempo total:         7.14
Expansiones totales: 45621
Costo total:         1162
```

Figura 5: *Weighted* A^*, w = 12
Fuente: Elaboración propia

```
➜  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1      3993      7229      105      0.77      0.00
    2      2759      5116      132      0.52      0.00
    3       508      1099      100      0.19      0.00
    4      2798      4724      111      0.57      0.00
    5      4335      7386      102      0.73      0.00
    6      7740     13275      161      1.26      0.00
    7      2900      5613       84      0.49      0.00
    8      7254     12546      157      1.05      0.00
    9      3712      6602      104      0.58      0.00
   10     10815     13744      110      1.44      0.00
Tiempo total:         7.58
Expansiones totales: 46814
Costo total:         1166
```

Figura 6: *Weighted* A^*, w = 15
Fuente: Elaboración propia

```
➜  Parte1 /usr/bin/python3 /Users/estebanvivanco/Desktop/0_Inteligencia_Artificial/T2/Parte1/test_astar.py
#prob      #exp      #gen     |sol|    tiempo maxsubopt
    1      3174      5665      107      0.59      3.06
    2      2934      5688      136      0.60      4.25
    3       643      1375      100      0.22      3.12
    4      1102      2126       81      0.36      2.61
    5      1721      3369      116      0.44      3.22
    6      1559      3157      113      0.41      3.23
    7      1451      2902       76      0.41      2.24
    8      2689      5103      121      0.53      2.57
    9      3678      6786      108      0.67      3.18
   10      4995      7872      112      0.73      3.11
Tiempo total:         4.95
Expansiones totales: 23946
Costo total:         1070
```

Figura 7: *Weighted* A^*, w = 10. Suboptimalidad agregada.
Fuente: Elaboración propia