Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Sistemas Inteligentes

Tema 2: Estrategias de búsqueda (cont.)

Curso 2024-25

Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Búsqueda A*. Búsqueda óptima

- Una función heurística h(n) se dice que es **admisible** (garantiza la obtención de un camino de coste mínimo hasta un objetivo) cuando se cumple:

$$h(n) \leq h^*(n) \ \forall n$$

- Decimos entonces que un algoritmo A que utiliza una función heurística admisible es un **algoritmo A***
- Cuanto más correctamente estimemos h(n) menos nodos de búsqueda generaremos
- Problema: **si nuestra función heurística nos devuelve un valor superior a h*, para algún nodo, no se puede garantizar que encontremos la solución óptima**

Algoritmo A*

Algoritmo Algoritmo A*

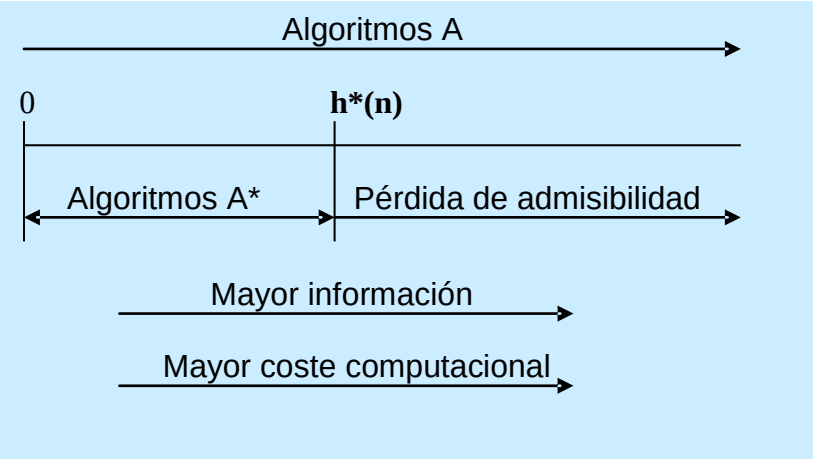
```
1:  listaInterior ← ∅
2:  listaFrontera ← inicio
3:  while listaFrontera ≠ ∅ do
4:      n ← obtener nodo de listaFrontera con menor f(n) = g(n) + h(n)
5:      listaFrontera.del(n)
6:      listaInterior.add(n)
7:      if n es meta then
8:          devolver reconstruir camino desde la meta al inicio (punteros)
9:      for all hijo m de n que no esté en listaInterior do
10:         g'(m) ← n.g + c(n, m)                ▷ g del nodo a explorar m
11:         if m no está en listaFrontera then
12:             almacenar la f, g y h del nodo en (m.f, m.g, m.h)
13:             m.padre ← n
14:             listaFrontera.add(m)
15:         else if g'(m) es mejor que m.g then   ▷ ¿nuevo camino mejor?
16:             m.padre ← n
17:             recalcular f y g del nodo m
18: devolver no hay solución
```

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Nivel de información heurístico

- En general el **nivel de información** de las heurísticas permite **encontrar antes la solución**, pero tiene la desventaja de requerir un **mayor coste computacional** para su cálculo. La figura muestra los *límites de la admisibilidad* en los algoritmos tipo A:

- h(n) = 0
- h(n) ≤ h*(n)
- h(n) > h*(n)

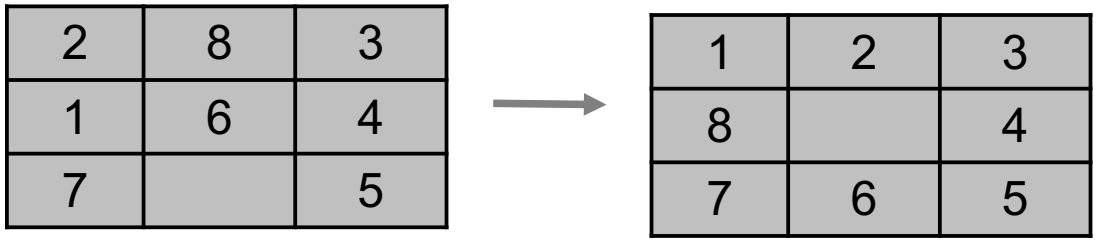


Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Ejemplo de heurísticas

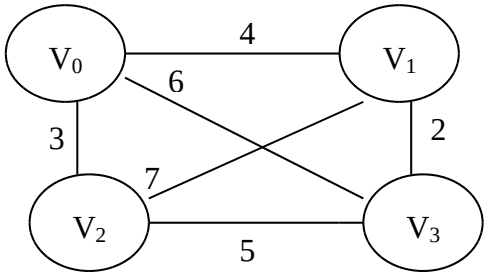
- 8-puzzle:



h1 = número de piezas mal colocadas

h2 = suma de las distancias de las piezas a sus posiciones en el objetivo (distancia de Manhattan)

- Viajante de comercio:



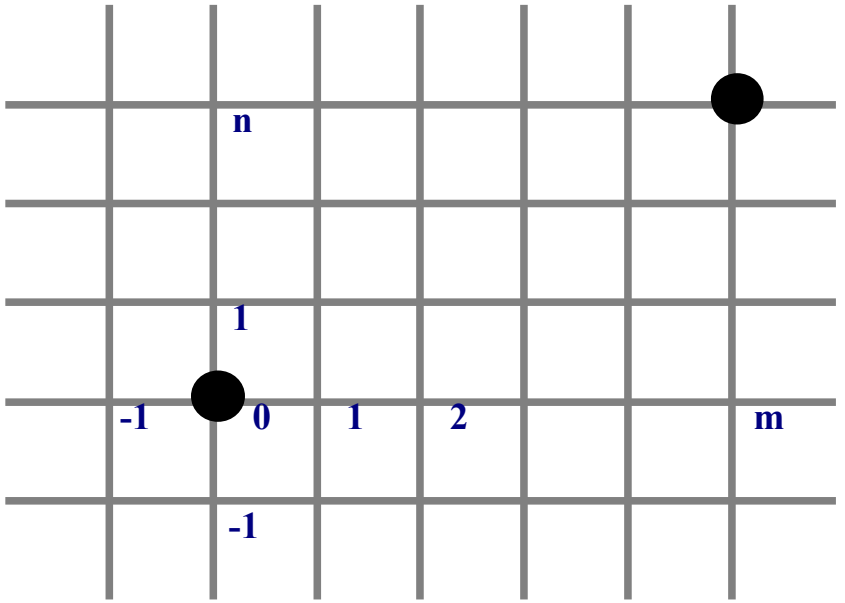
h= suma de las distancias de las ciudades aun no visitadas a sus vecinos más cercanos

Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

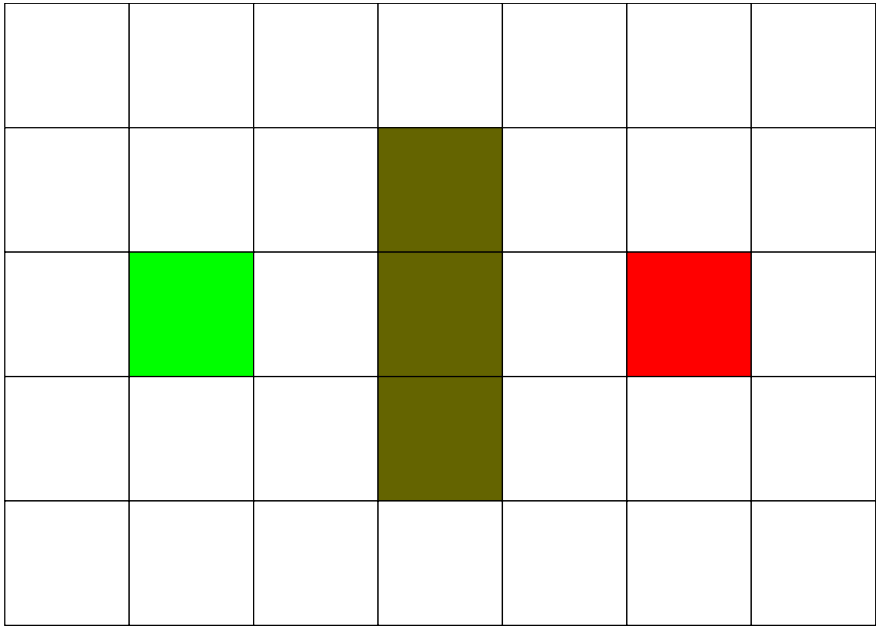
Problemas de camino mínimo

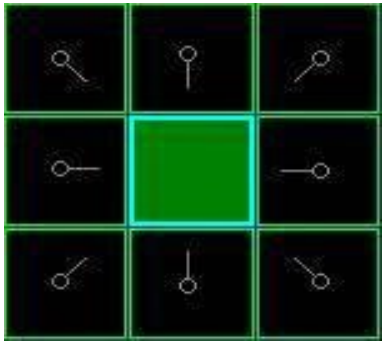
- **Coste actual óptimo**
  - g*((x, y)) = |x| + |y|
- **Heurística admisible**
  - h1((x,y)) = sqrt((m-x)^2 + (n-y)^2)
- **Heurística óptima**
  - h*((x,y)) = |m-x| + |n-y|



Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Ejemplo detallado A*



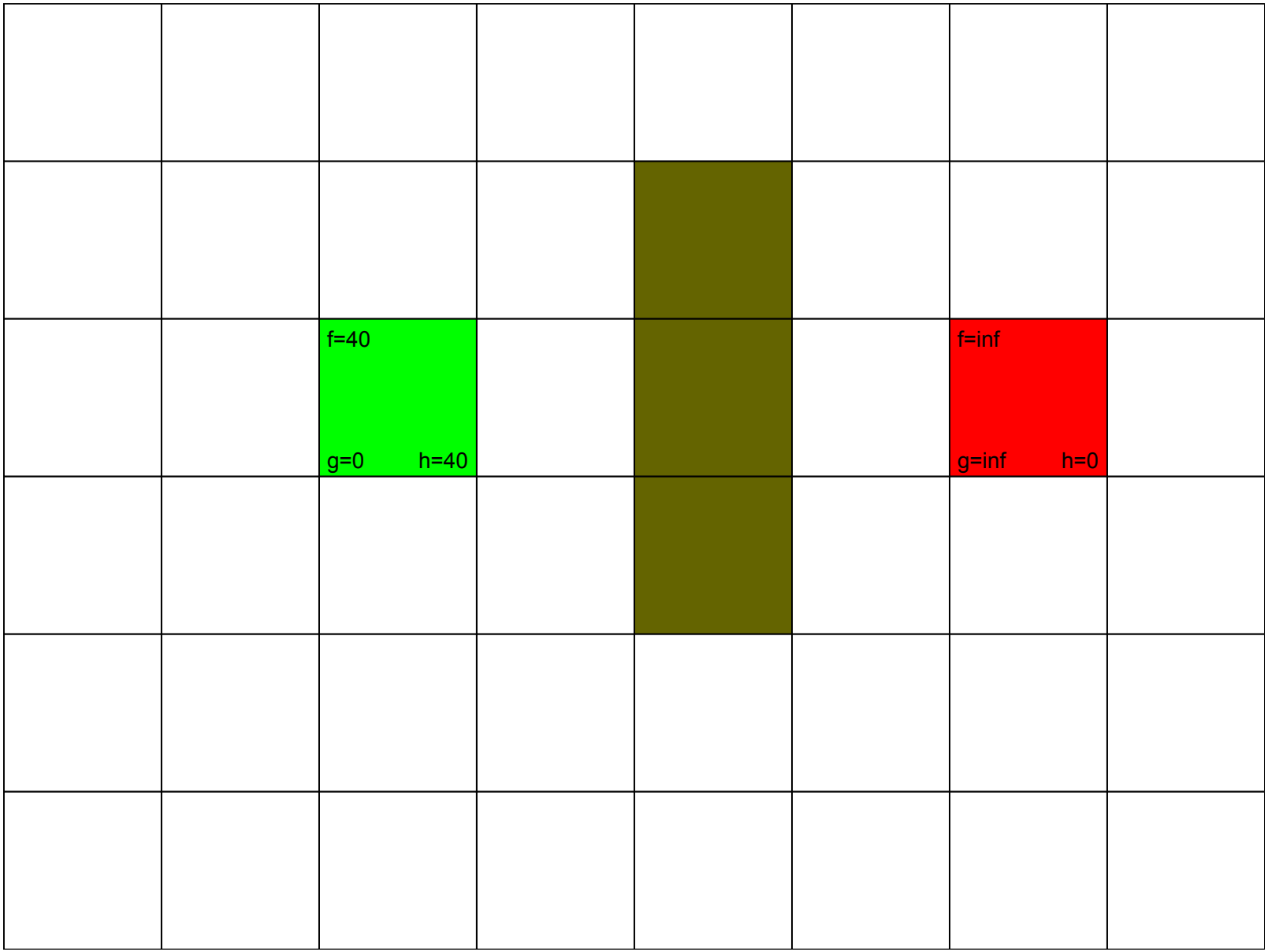


- g(n) : coste de moverse entre nodos (recto 10, diagonal 14)
- h(n) : distancia de Manhattan: |m-x| + |n-y|
  Atención, no admisible para 8-con. utilizada por simplicidad…

Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado



Recordemos...

Algoritmo Algoritmo A*

```
1:  listaInterior ← ∅
2:  listaFrontera ← inicio
3:  while listaFrontera ≠ ∅ do
4:      n ← obtener nodo de listaFrontera con menor f(n) = g(n) + h(n)
5:      listaFrontera.del(n)
6:      listaInterior.add(n)
7:      if n es meta then
8:          devolver reconstruir camino desde la meta al inicio (punteros)
9:      for all hijo m de n que no esté en listaInterior do
10:         g'(m) ← n.g + c(n, m)                ▷ g del nodo a explorar m
11:         if m no está en listaFrontera then
12:             almacenar la f, g y h del nodo en (m.f, m.g, m.h)
13:             m.padre ← n
14:             listaFrontera.add(m)
15:         else if g'(m) es mejor que m.g then   ▷ ¿nuevo camino mejor?
16:             m.padre ← n
17:             recalcular f y g del nodo m
18: devolver no hay solución
```

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

	f=74 g=14 h=60	f=60 g=10 h=50	f=54 g=14 h=40				
	f=60 g=10 h=50	f=40 1 g=0 h=40	f=40 g=10 h=30			f=inf g=inf h=0	
	f=74 g=14 h=60	f=60 g=10 h=50	f=54 g=14 h=40				

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

	f=74 g=14 h=60	f=60 g=10 h=50	f=54 g=14 h=40				
	f=60 g=10 h=50	f=40 1 g=0 h=40	f=40 2 g=10 h=30			f=inf g=inf h=0	
	f=74 g=14 h=60	f=60 g=10 h=50	f=54 g=14 h=40				

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

		f=88 g=28 h=60	f=74 g=24 h=50				
	f=74 g=14 h=60	f=60 g=10 h=50	f=54 3 g=14 h=40				
	f=60 g=10 h=50	f=40 1 g=0 h=40	f=40 2 g=10 h=30			f=inf g=inf h=0	
	f=74 g=14 h=60	f=60 g=10 h=50	f=54 g=14 h=40				

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

		f=88 g=28 h=60	f=74 g=24 h=50				
f=94 g=24 h=70	f=74 g=14 h=60	f=60 g=10 h=50	f=54 3 g=14 h=40				
f=80 g=20 h=60	f=60 6 g=10 h=50	f=40 1 g=0 h=40	f=40 2 g=10 h=30			f=inf g=inf h=0	
f=94 g=24 h=70	f=74 g=14 h=60	f=60 5 g=10 h=50	f=54 4 g=14 h=40				
	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 g=24 h=50				

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 9 g=24 h=50	f=74 11 g=34 h=40	f=74 g=44 h=30		
f=94 g=24 h=70	f=74 g=14 h=60	f=60 7 g=10 h=50	f=54 3 g=14 h=40				
f=80 g=20 h=60	f=60 6 g=10 h=50	f=40 1 g=0 h=40	f=40 2 g=10 h=30			f=inf g=inf h=0	
f=94 g=24 h=70	f=74 10 g=14 h=60	f=60 5 g=10 h=50	f=54 4 g=14 h=40				
f=108 g=28 h=80	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 8 g=24 h=50	f=74 g=34 h=40			
		f=108 g=38 h=70	f=94 g=34 h=60	f=88 g=38 h=50			

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 9 g=24 h=50	f=74 11 g=34 h=40	f=74 12 g=44 h=30	f=74 g=54 h=20	f=102 g=72 h=30
f=94 g=24 h=70	f=74 g=14 h=60	f=60 7 g=10 h=50	f=54 3 g=14 h=40		f=74 g=54 h=20	f=68 13 g=58 h=10	f=88 g=68 h=20
f=80 g=20 h=60	f=60 6 g=10 h=50	f=40 1 g=0 h=40	f=40 2 g=10 h=30		f=82 g=72 h=10	f=68 g=68 h=0	f=82 g=72 h=10
f=94 g=24 h=70	f=74 10 g=14 h=60	f=60 5 g=10 h=50	f=54 4 g=14 h=40				
f=108 g=28 h=80	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 8 g=24 h=50	f=74 g=34 h=40			
		f=108 g=38 h=70	f=94 g=34 h=60	f=88 g=38 h=50			

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante

Ejemplo detallado

	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 9 g=24 h=50	f=74 11 g=34 h=40	f=74 12 g=44 h=30	f=74 g=54 h=20	f=102 g=72 h=30
f=94 g=24 h=70	f=74 g=14 h=60	f=60 7 g=10 h=50	f=54 3 g=14 h=40		f=74 g=54 h=20	f=68 13 g=58 h=10	f=88 g=68 h=20
f=80 g=20 h=60	f=60 6 g=10 h=50	f=40 1 g=0 h=40	f=40 2 g=10 h=30		f=82 g=72 h=10	f=68 14 g=68 h=0	f=82 g=72 h=10
f=94 g=24 h=70	f=74 10 g=14 h=60	f=60 5 g=10 h=50	f=54 4 g=14 h=40				
f=108 g=28 h=80	f=94 g=24 h=70	f=80 g=28 g=20 h=60	f=74 8 g=24 h=50	f=74 g=34 h=40			
		f=108 g=38 h=70	f=94 g=34 h=60	f=88 g=38 h=50			

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Inconvenientes de mantener la admisibilidad

- El mantenimiento de la admisibilidad fuerza al algoritmo a **consumir mucho tiempo** en discriminar caminos cuyos costes no varían muy significativamente
- Principal desventaja: se queda sin espacio debido a que mantiene todos los nodos generados en memoria
- No es práctico para problemas grandes
- Dos soluciones:
  - Algoritmos que mejoran el coste espacial: A*PI (A* por profundización iterativa), A* SRM (A* acotada por memoria simplificada), búsqueda primero el mejor recursiva
  - Aumentar la velocidad a costa de una pérdida acotada de calidad → **técnicas de relajación de la restricción de optimalidad**

Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Relajación de la restricción de optimalidad

- **1.Técnica de ajuste de pesos**
  - El objetivo de esta técnica es definir una función f() ponderada, $f_w()$, como alternativa a la utilizada en A*

    $$f_w(n) = (1-w)g(n)+w\ h(n)$$

  - **g(n):** Proporciona la componente en anchura de la búsqueda.
  - **h(n):** Nos indica la proximidad al objetivo.
  - Variando de forma continua w dentro del rango $0 \leq w \leq 1$ obtenemos **estrategias mixtas intermedias**.
- Si h(n) es admisible tenemos que:
  - En el rango $0 \leq w \leq 1/2$, A* con $f_w(n)$ también es admisible.
  - Dependiendo de la diferencia existente entre h(n) y h*(n), A* con $f_w(n)$ puede perder la admisibilidad en el rango $1/2 < w \leq 1$

Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Relajación de la restricción de optimalidad

- **2.Técnica de la admisibilidad-ε**
  - Objetivo: aumentar la velocidad de búsqueda a costa de obtener una solución subóptima
  - Un algoritmo es **admisible-ε** cuando para cualquier grafo termina siempre dando como resultado una solución cuyo coste no excede del coste óptimo, C* , por un factor 1+ε:

    $$f(sol) \leq (1+\varepsilon)C^*$$

- Técnicas de admisibilidad-ε
  - **2.1. Ponderación Dinámica**
  - **2.2. Estimación de coste de búsqueda**

Universitat d'Alacant
Universidad de Alicante

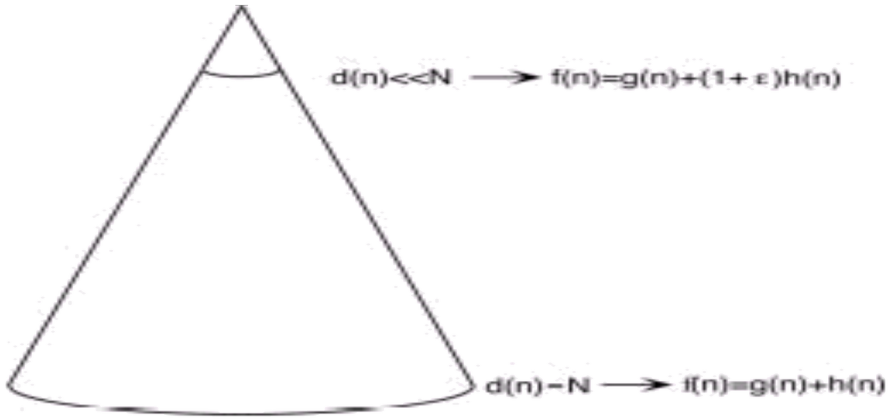
Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Relajación de la restricción de optimalidad

- **2.1.Técnica de ponderación dinámica o APD**

  $f(n) = g(n)+h(n)+\varepsilon[\ 1- d(n)/N\ ]\ h(n)$

- Donde:
  - d(n) es la profundidad del nodo n y N nos proporciona la profundidad de un nodo solución (se supone conocida).
- ¿Qué ocurre en los niveles iniciales?
- ¿Y en los cercanos a la solución?



Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Relajación de la restricción de optimalidad

- **2.2. Alg. de estimación de coste de búsqueda A_ε^***
  - Utiliza una lista adicional denominada **Lista_focal** (**Lf**)
  - Esta lista es una sublista de Lista_Frontera (LF) que contiene únicamente aquellos nodos cuya f(n) no excede del mejor valor de cualquier f(n) dentro de la Lista_Frontera por un factor (1+ε):

    Lf = {n: f(n) ≤ (1+ε) min(f(m))}, m ∈ LF
  - A_ε^* opera de forma idéntica al algoritmo A* **salvo que selecciona aquel nodo de Lista_focal con menor valor de Hf(n), una segunda heurística**, además de h(n), que estima el coste computacional requerido para completar la búsqueda a partir del nodo n

Universitat d'Alacant
Universidad de Alicante

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Relajación de la restricción de optimalidad

- **Comparación de técnicas de admisibilidad-ε**
  - El algoritmo de ponderación dinámica es más sencillo, pero únicamente es aplicable a problemas donde se conoce la profundidad en la cual nos va a aparecer la solución, o disponemos de una cota superior de dicha profundidad. Sólo en estos casos se garantiza la admisibilidad-ε
  - En cuanto al algoritmo Aε* la separación en dos heurísticas permite incorporar estimaciones de coste no integradas con las funciones g(n) y h(n) (por ejemplo, proximidad a la solución)

Universitat d'Alacant
Universidad de Alicante

Bibliografía

- Stuart Russell, Peter Noving. “Inteligencia Artificial. Un enfoque Moderno” Ed. Pearson. Prentice Hall.

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universitat d'Alacant
Universidad de Alicante