Tema 2. Búsqueda en Espacios de Estados

Algoritmos de Búsqueda Heuristica: A* descripción y propiedades

Objetivos

1. Conocer los fundamentos de los algoritmos de búsqueda y el papel que juegan en la Inteligencia Artificial
2. Conocer el paradigma de Búsqueda en Espacios de Estados y los **algoritmos básicos** de búsqueda a ciegas y sobre todo **de búsqueda inteligente o heurística**
3. Saber cómo modelar problemas para resolverlos con Búsqueda en Espacios de Estados, en particular cómo introducir conocimiento específico del dominio del problema

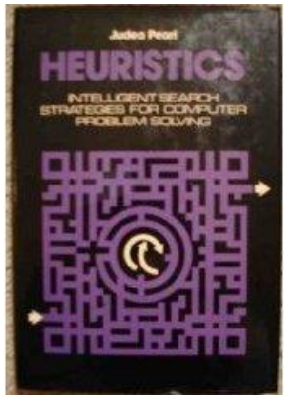
Contenidos

1. Introducción
2. Espacios de búsqueda
3. Algoritmos de búsqueda no informada
4. **Algoritmos de búsqueda informada o heurística**
    1. **Introducción**
    2. **El algoritmo A*: descripción y propiedades formales**
    3. Otros algoritmos
5. Técnicas de diseño de funciones heurísticas

2.4.1. Búsqueda Heurística
Introducción

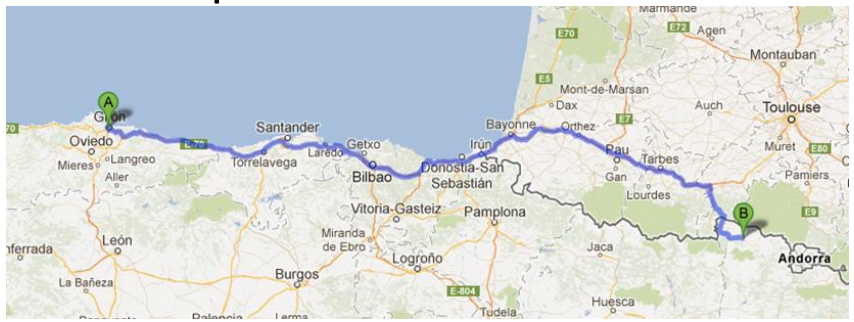
- En *Inteligencia Artificial*, un HEURISTICO es una estrategia de resolución de un problema que utiliza conocimiento específico del dominio
- El origen del término aun no ha sido determinado, hay varias opiniones
  - Heuriskein (palabra griega que significa encontrar)
  - EUREKA!! (Arquímedes en el baño)
  - Otros . . .

- Según Judea Pearl *[Pearl, 1988],* "*Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal.*"



Búsqueda Heurística
Introducción (II)

- El papel de los heurísticos en la búsqueda
  - Ayudan a decidir qué nodo expandir (algoritmos iterativos) o qué regla aplicar (backtracking) utilizando conocimiento sobre el problema
  - Se espera que mejoren el proceso de búsqueda en media, pero no en el peor caso, cuando se utilizan en un método exacto (admisible)
  - Si se usan en un método no exacto, se espera que produzcan soluciones razonables (mejores que las aleatorias) en un tiempo razonable
  - Tienen asociado un coste, luego son útiles si la mejora compensa al coste
- Ejemplo: cálculo de rutas en mapas

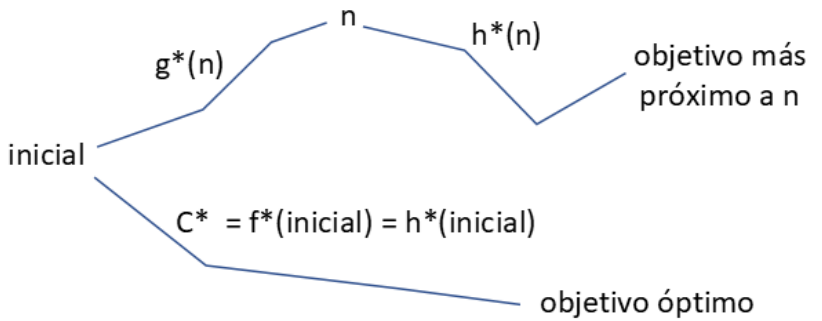


  - Con búsqueda heurística no se intenta buscar un camino a través de Ciudad Real, por ejemplo
  - Con búsqueda a ciegas se analizaría la posibilidad de pasar por todas las ciudades más cercanas que el objetivo

2.4.2 El Algoritmo A*

[Hart, P., Nilsson, N., Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Syst. Science and Cybernetics, SSC-4(2):100-107, 1968.]

- Dado un espacio de búsqueda, definimos
  - $f^*(n) = g^*(n) + h^*(n)$
  - $g^*(n)$ es el coste mínimo del inicial a n
  - $h^*(n)$ es el coste mínimo de n a los objetivos, luego
  - $f^*(n)$ es el coste mínimo del inicial a los objetivos condicionado a pasar por n
  - $C^* = f^*(inicial) = h^*(inicial)$, coste de la solución óptima



- A* = BF con $f(n) = g(n) + h(n)$
  - $f(n)$ es una estimación de $f^*(n)$
  - $g(n)$ = mejor coste de inicial a n hasta el momento, es una variable de A* (n.PATH-COST)
  - $h(n)$ = estimación positiva de $h^*(n)$, con $h(n) = 0$, si n es objetivo. Es el HEURÍSTICO y se debe definir utilizando información sobre el problema, en particular sobre el estado n

Ejemplos de heurísticos

- El problema del 8-puzzle
  - En un estado *n* podemos estimar el coste de llegar a la solución como el número de fichas que no están en la posición que les corresponde en el objetivo

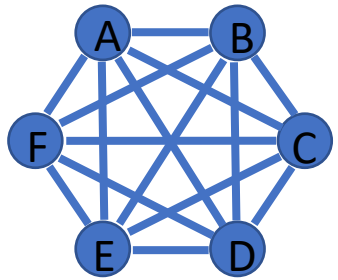
$h($ | 2 | 8 | 3 |
|---|---|---|
| 1 | | 4 |
| 7 | 6 | 5 | $) = 3$

Objetivo

1	2	3
8		4
7	6	5

- El problema TSP
  - En este caso, podemos tener en cuenta el número de ciudades que quedan por abandonar, y para cada una de ellas contar la media del coste de los arcos incidentes

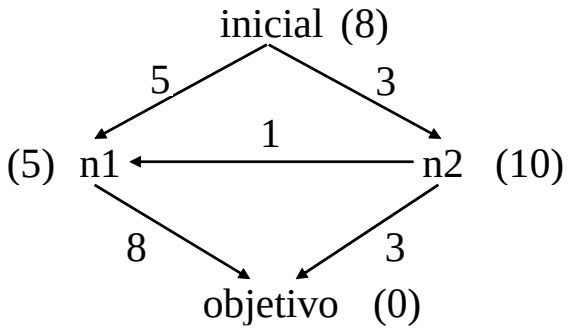
h(**(A{BCD}E)**) =
(113+25+18+180) / 4 +
(92+9+20+39+17) / 5 = 130,55



	A	B	C	D	E	F
A		21	12	15	113	92
B			7	32	25	9
C				5	18	20
D					180	39
E						17

Ejercicio abstracto de A y heurísticos*

- Dados el espacio de búsqueda y el heurístico h() (valores entre paréntesis) siguientes



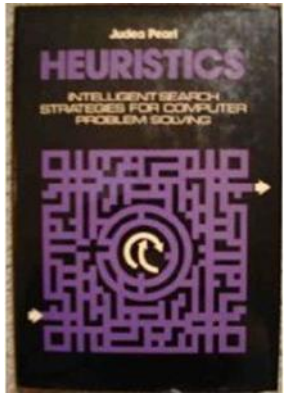
- Responder razonadamente a estas cuestiones:
  - ¿Está bien definido el heurístico h()?
  - ¿Cuáles son los valores de h*() para los cuatro nodos? ¿y los de g*()?
  - ¿Cuál es el valor de C*?
  - ¿Cuál es el valor de g(n2) y de g(n1)?
  - Hacer una traza del algoritmo A*: indicar los nodos que se expanden y los contenidos de *frontier, reached* y el árbol de búsqueda en cada iteración.
  - ¿Encuentra A* la solución óptima con este heurístico? ¿Qué estados se expanden y con qué valor de f()?
  - ¿La encontraría si se conociese h*() para todos los estados? ¿Qué estados se expandirían en este caso?

*Propiedades formales del Algoritmo A**

¿Cuándo podemos decir que un heurístico h es bueno o al menos mejor que otro?

- **Admisibilidad:** es una propiedad de los heurísticos que garantiza la admisibilidad de A*, es decir que A* siempre encuentra una solución óptima, por supuesto si le damos espacio y tiempo suficiente para terminar.
- **Dominancia:** es una relación entre dos heurísticos para el mismo problema que garantiza (o casi) que un heurístico es mejor que otro; concretamente que los dos llegan a la solución óptima, pero uno lo hace sin expandir más nodos que el otro.
- **Consistencia o Monotonía:** es una propiedad de los heurísticos que garantiza la admisibilidad, pero además facilita la gestión de los estados repetidos, concretamente si expande un estado entonces no es necesario volver a expandirlo.

Hart, P., Nilsson, N., Raphael, B., *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Trans. Syst. Science and Cybernetics, SSC-4(2):100-107, 1968.

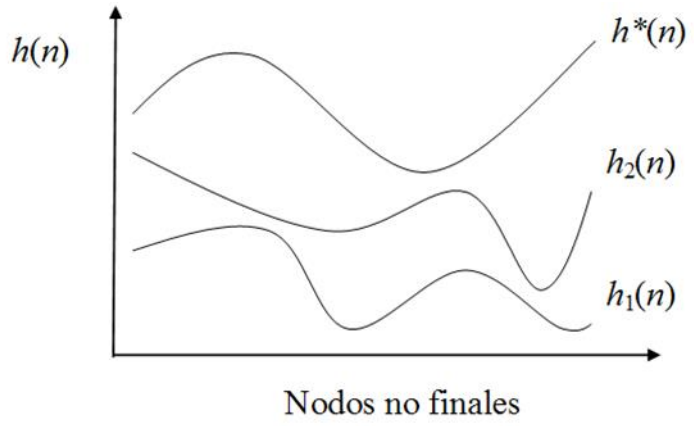


*Propiedades formales del Algoritmo A**
Admisibilidad

- **Definición** [*Heurístico bien definido*]. *h* está bien definido si $h(n) \geq 0$ para todo estado *n* no objetivo y $h(n) = 0$ si *n* es objetivo.
- **Definición** [*Heurístico Admisible*]. Si $h(n) \leq h^*(n)$, para todo *n*, entonces *h* es un heurístico admisible.
- **Teorema** [*Admisibilidad de A**]. Si *h* es admisible, entonces A*(*h*) es admisible, es decir siempre encuentra la solución óptima.
- **Corolario**. A*($h=0$) y A*(h^*) están bien definidos y son admisibles.

*Propiedades formales del Algoritmo A**
Dominancia

- **Definición** [*Dominancia entre Heurísticos*]. Si $h_1(n) < h_2(n) \leq h^*(n)$, para todo n no final, entonces h_2 está mejor informado que h_1 (h_2 domina a h_1).
- **Teorema** [*Dominancia de Algoritmos A**]. Si h_2 está mejor informado que h_1, entonces todo nodo expandido por A*(h_2) es expandido por A*(h_1), es decir, A*(h_2) domina a A*(h_1).



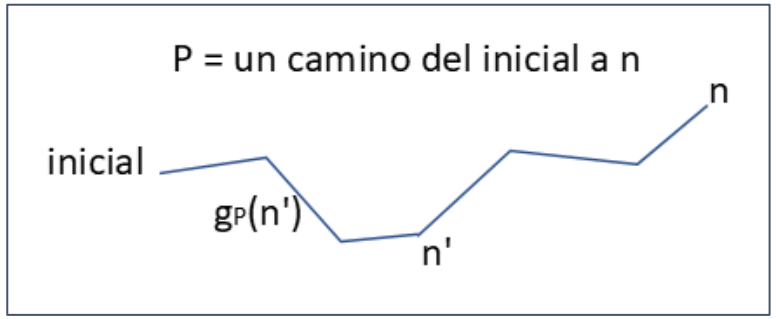
Dos heurísticos admisibles con distinto grado de información

*Propiedades formales del Algoritmo A**
Admisibilidad (Condiciones de expansión)

- **Teorema**. [*Condiciones dinámicas*]
  - Condición Necesaria: $f(n) \leq C^*$ (si n se expande $\Rightarrow f(n) \leq C^*$).
  - Condición Suficiente: $f(n) < C^*$ (si n está en **frontier** y $f(n) < C^* \Rightarrow$ se expandirá antes de que A* termine).

- **Teorema**. [*Condiciones estáticas*]
  - Condición Necesaria: Existe un camino $P_{inic\text{-}n}$ t.q. $g_P(n')+h(n') \leq C^*$ para todo n' de $P_{inic\text{-}n.}$
  - Condición Suficiente: Existe un camino $P_{inic\text{-}n}$ t.q. $g_P(n')+h(n') < C^*$ para todo n' de $P_{inic\text{-}n.}$

  (siendo $g_P(n')$ el coste del *inicial* a n' a través de $P_{inic\text{-}n}$)



*Propiedades formales del Algoritmo A**
Consistencia y Monotonía

- **Definición** [*Monotonía*]. h es monótono si para todo par de nodos n_1, n_2 se cumple $h(n_1) \leq h(n_2) + c(n_1, n_2)$
- **Definición** [*Consistencia*]. h es consistente si para todo par de nodos n_1, n_2 se cumple $h(n_1) \leq h(n_2) + K(n_1, n_2)$, siendo $K(n_1, n_2)$ el coste del camino de coste mínimo entre n_1 y n_2
- **Teorema**. h monótono $\Leftrightarrow$ h consistente

*Propiedades formales del Algoritmo A**
Consistencia y Monotonía (Consecuencias)

- **Teorema**. *h* monótono $\Rightarrow$ *h* admisible.
- **Teorema**. Si *h* es monótono y A* elige *n* para expandir, entonces $g(n)=g^*(n)$, luego no hay que reexpandir estados.
- **Teorema** [*Dominancia Amplia*]. Si

  h_1 y h_2 son monótonos,

  $h_1(n) \leq h_2(n)$ para todo *n*, y

  A*(h_2) expande *n'* y A*(h_1) no lo expande

  entonces

  $h_1(n') = h_2(n') = C^* - g^*(n')$.
- **Teorema.** Si *h* es monótono, la secuencia de valores *f(n)* de los nodos expandidos es no decreciente.
- **Teorema**. *[Condiciones de expansión (estáticas)]* Si h es monótono
  - Condición Necesaria: $g^*(n)+h(n) \leq C^*$.
  - Condición Suficiente: $g^*(n)+h(n) < C^*$.

*Ejercicios de propiedades formales del Algoritmo A**

- En cada uno de los tres espacios de búsqueda siguientes, teniendo en cuenta el heurístico correspondiente, responder de forma razonada a las siguientes cuestiones
  - ¿Qué nodos se expandirán con seguridad?
  - ¿Cuáles es seguro que no se expandirán?
  - ¿Cuáles pueden expandirse o no?
  - ¿Qué ocurriría con h=0?