

CERCA INFORMADA
Resolució de problemes de presa de
decisió per exploració d'alternatives

CERCA INFORMADA

Idea: Solucionar la ineficiència de la cerca no informada, fent servir informació específica i heurística sobre el problema.

Heurística: Criteri, mètode o principi per decidir quina entre varies alternatives és la que sembla ser més efectiva per arribar a un determinat objectiu.

Funció heurística: funció que estableix una correspondència entre un estat d'un problema i un número que estima l'efectivitat de passar per aquest estat per a poder arribar a la solució.

$$h : \{\text{Estats del problema}\} \rightarrow \mathbb{R}$$

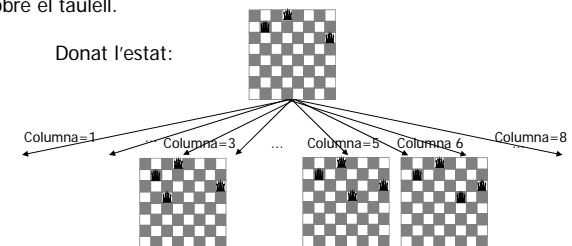
$h(n)$: Cost estimat per anar des de n fins a l' estat objectiu.

Propietats de les HEURÍSTIQUES:

- Permeten evitar el problema de les explosions combinatòries, quasi sempre.
- Permeten accelerar el procés de trobar una solució, quasi sempre.
- No permeten assegurar una complexitat baixa.
- Permeten aprofundir en la comprensió del problema, ja que el comportament d'una heurística ens pot informar sobre la forma de les solucions.

Exemples d'HEURÍSTIQUES:

- **Problema de les 8 Reines**, suposem que anem col.locant una reina en cada fila de manera que no es mati amb les que ja estan col.locades sobre el taulell.



On és millor col.locar la següent reina, per arribar abans a una solució ?
... a la columna 3, a la 5 o a la 6 ?



Heurística: Número de caselles que no són atacades per cap altra reina.

Columna=1 ... Columna=3 ... Columna=5 Columna=6 ... Columna=8

h=7 h=9 h=10

En cada cas:

No arriba a cap solució Solució Solució

Exemples d'HEURÍSTIQUES:

- Problema del puzzle de rajola,**

Estat Inicial

1	4	2
6	3	3
8	7	5

Estat Objectiu

1	2	3
8	4	4
7	6	5

Heurística 1:
Número de rajoles que són diferents a l'estat final.

Heurística 2:
Suma de les distàncies (verticals i horitzontals) entre la posició de cada rajola i la seva posició a l'estat final.

1	2
6	4
8	7

h1=6
h2=7

1	4	2
6	3	3
8	7	5

h1=6
h2=9

1	4	2
6	7	3
8	5	5

h1=6
h2=9

1	4	2
6	3	3
8	7	5

h1=6
h2=7

Exemples d'HEURÍSTIQUES:

- Problema del mapa de carreteres,** trobar una trajectòria des d'un lloc a un altre quan es desconeix el kilometratge real.

Exemples d'HEURÍSTIQUES:

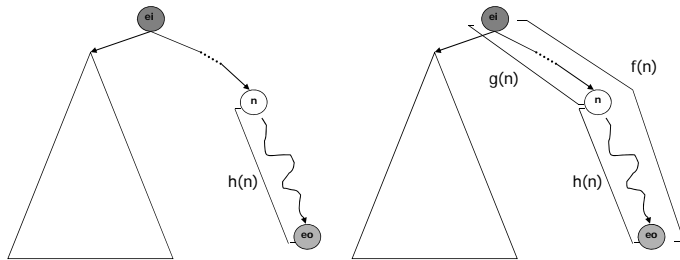
- Problema del mapa de carreteres,** trobar una trajectòria des d'un lloc a un altre.

Estat Inicial: Fraga
Estat Objectiu: Barcelona

Heurística:
Distància en línia recta des de la posició actual fins a l'objectiu

CERCA INFORMADA

- **Greedy best-first search (cerca GBFS)**, expandeix primer els camins que estima que són més propers a la solució segons la funció $h(n)$.
- **Cerca A***: expandeix primer els camins que estima com a més òptims segons la funció $f(n)=g(n)+h(n)$.



Funció CERCA_GBFS (NodeArrel, NodeObjectiu)

1. Llista= [[NodeArrel]];
 2. Fins que (Cap(Cap(Llista))=NodeObjectiu O bé (Llista=NIL) fer
 - a) C=Cap(Llista);
 - b) E=Expandir(C);
 - c) E=EliminarCicles(E);
 - d) Llista=Inserció_ordenada_h(E,Cua(Llista),h);
 3. Fins que;
 4. Si (Llista<>NIL) Retornar(Cap(Llista));
 5. Sinó Retornar("No existeix Solucio");
- Ffuncio

Inserció_ordenada_h(E,L,h): Inserta els elements de la llista E a la llista L, la inserció es fa de manera que els camins quedin ordenats de menor a major segons el valor de la funció h.

Funció CERCA_A* (NodeArrel, NodeObjectiu)

1. Llista= [[NodeArrel]];
 2. Fins que (Cap(Cap(Llista))=NodeObjectiu O bé (Llista=NIL) fer
 - a) C=Cap(Llista);
 - b) E=Expandir(C);
 - c) E=EliminarCicles(E);
 - d) Llista=Inserció_ordenada_f(E,Cua(Llista),f);
 3. Fins que;
 4. Si (Llista<>NIL) Retornar(Cap(Llista));
 5. Sinó Retornar("No existeix Solucio");
- Ffuncio

Inserció_ordenada_f(E,L,f): Inserta els elements de la llista E a la llista L, la inserció es fa de manera que els camins quedin ordenats de menor a major segons el valor de la funció f.

Exercici: Donades les següents taules, dibuixeu l'arbre que construirà el GBFS i l'A* i digueu quines solucions donaran per anar de E1 a E8.

Taula de Costos

	E1	E2	E3	E4	E5	E6	E7	E8
E1	-	3	4	-	-	-	-	-
E2	3	-	5	4	-	-	-	-
E3	4	5	-	-	2	-	-	-
E4	-	4	-	-	5	4	-	-
E5	-	-	2	5	-	-	4	-
E6	-	-	-	4	-	-	-	-
E7	-	-	-	-	4	-	-	3
E8	-	-	-	-	-	-	3	-

F. Heurística:

	E8
E1	11.0
E2	10.4
E3	8.9
E4	6.7
E5	6.9
E6	4.0
E7	3.0

Exercici: Donades les següents taules, dibuixeu l'arbre que construirà l'A* en funció de la funció heurística que faci servir per anar de E1 a E8.

Taula de Costos

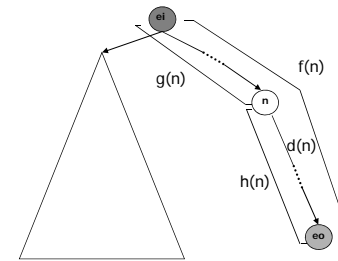
	E1	E2	E3	E4	E5	E6	E7	E8
E1	-	1	1	-	-	-	-	-
E2	1	-	1	1	-	-	-	-
E3	1	1	-	-	1	-	-	-
E4	-	1	-	-	1	1	-	-
E5	-	-	1	1	-	-	1	-
E6	-	-	-	1	-	-	-	-
E7	-	-	-	-	1	-	-	1
E8	-	-	-	-	-	-	1	-

F. Heurístiques:

	h1	h2	Cost real òptim
E1	3.5	6.0	4
E2	2.7	4.3	4
E3	2.9	5.0	3
E4	1.5	3.2	3
E5	1.5	2.5	2
E6	0.5	2.0	4
E7	0.5	1.5	1

Optimalitat: per assegurar l'optimalitat de l'A* cal que l'heurística sigui admissible.

Admissibilitat: una heurística $h(n)$ és admissible si ens assegura que mai sobre-estima el cost que falta per arribar a la solució.



$d(n)$: Cost òptim per arribar a la solució.

Si h és admissible, llavors

$$h(n) \leq d(n) \quad \forall n: \text{node}$$

Anàlisi dels algorismes:

Criteri	Cerca GBFS	Cerca A*
Complet	Sí (si eliminem cicles)	Sí (si eliminem cicles)
Òptim	No	Sí (si h admissible)
Temps	$O(b^m)$	$O(b^m)$
Espai	$O(b^m)$	$O(b^m)$

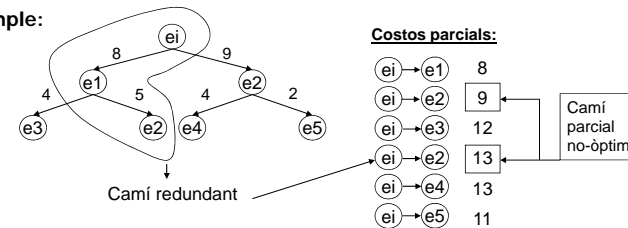
Problema: Quan es consideren els costos acumulats, existeixen camins redundants que disminueixen l'eficiència dels algorismes de cerca.

Camí redundat: es aquell camí que conté camins parcials no òptims.

Principi: qualsevol camí òptim està format per camins parcials òptims (principi bàsic en tècniques de programació dinàmica)

Conclusió: la solució òptima no és un camí redundat.

Exemple:



ELIMINACIÓ DE CAMINS REDUNDANTS

Funció CERCA_A* (NodeArrel, NodeObjectiu)

1. Llista = [[NodeArrel]];
2. Fins que (Cap(Cap(Llista))=NodeObjectiu O bé (Llista=NIL) fer
 - a) C=Cap(Llista);
 - b) E=Expandir(C);
 - c) E=EliminarCicles(E);
 - d) EliminarCaminsRedundants(E,Llista,TCP);
 - e) Llista=Inserció_ordenada_f(E,Llista,f);
3. Ffinsque;
4. Si (Llista<>NIL) Retornar(Cap(Llista));
5. Sinó Retornar("No existeix Solució");

Ffuncio

Taula de Costos Parcial:

TCP = [[A, C1] , [B, C2] , ... [Z, Cn]]

Funcio EliminarCaminsRedundants(var E, var Llista)

1. Per a (cada camí de E) Fer
 1. CP=Associar(Cap(camí),TCP);
 2. Si (Cost(Cap(camí)) < CP) llavors
 1. Actualitzar TCP amb Cap(camí) i Esborrar de E el camí amb cost CP
 3. Sinó
 1. Esborrar camí de E
 4. Fsi
2. Fper

Ffuncio

Demostració: Optimalitat de A* quan l'heurística és admissible

Suposem dos camins C1, C2, que són solució d'un problema

I suposem que tenen costos: $g(C1)=f^*$ i $g(C2)>f^*$

on f^* és el cost òptim de la solució del problema

Suposem que A* troba com a solució C2, que és no òptim

1. Si A* ha trobat C2 és perquè en algun moment $f(C2)=g(C2)+h(C2)=g(C2)\leq f(n)$ on n: node del camí C1
2. Si h admissible, llavors per qualsevol n, node del camí C1, es compleix que $f(n)=g(n)+h(n)\leq g(n)+d(n)\leq f^*$

Per tant, del punt 1 deduïm $g(C2)\leq f(n)$

i del punt 2 deduïm que $f(n)\leq f^*$, per tant $g(C2)\leq f^*$

que és una Contradicció amb el que havíem assumit inicialment.

Propietats de les heurístiques:

Factor de ramificació efectiu, suposem que un problema particular que es resol amb l'algorisme A* a una profunditat d i amb una heurística concreta, h, requereix expandir en promig N nodes.

Aleshores el factor de ramificació efectiu, b^* , d'aquest problema amb aquesta heurística es defineix com el factor de ramificació que hauria de tenir un arbre equilibrat de profunditat d que tingués N nodes, això és

$$N = 1 + b^* + (b^*)^2 + (b^*)^3 + \dots + (b^*)^d$$

Heurístiques que dominen, donades dues heurístiques, h_1 i h_2 , admissibles, direm que h_1 domina h_2 si $h_1 > h_2$. En aquest cas es pot assegurar que h_1 és millor que h_2 quan s'aplica A*.

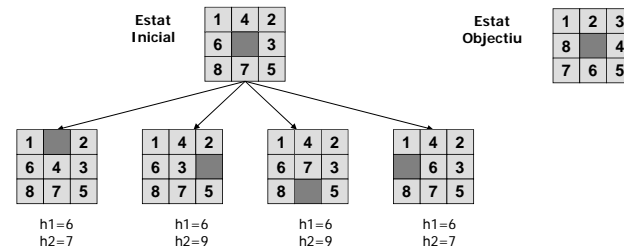
Construcció d'heurístiques dominants, donat un conjunt d'heurístiques admissibles, h_1, h_2, \dots, h_n , tals que no existeix cap heurística que domini sobre les altres, podem construir l'heurística dominant com: $h(n) = \max_i \{h_i(n)\}$

Heurístiques que aprenen: les heurístiques amb la forma:

$$h(n) = c_1 * h_1(x) + \dots + c_n * h_n(x)$$

permeten introduir l'experiència o aprenentatge modificant els valors dels c_i

Exemple: El problema del puzzle de rajola, per solucionar el problema del puzzle de rajola hem vist dues heurístiques diferents:



Heurística 1: Número de rajoles que són diferents a l'estat final.

Heurística 2: Suma de les distàncies (verticals i horitzontals) entre la posició de cada rajola i la seva posició a l'estat final.

Treball 3: Contesteu a les següents preguntes:

1. Quin és el factor de ramificació promig d'aquest problema?
2. Té sentit aplicar l'algorisme A* per solucionar aquest problema?
3. L'algorisme A* serà òptim si fem servir les heurístiques h1 i h2?
4. Quina heurística és millor?, per què?
5. Anàlisi de les dades experimentals d'aquest problema i factor de ramificació efectiu.

d	Search Cost			Effective Branching Factor		
	IDS	A*(h ₁)	A*(h ₂)	IDS	A*(h ₁)	A*(h ₂)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	-	1301	211	-	1.45	1.25
18	-	3056	363	-	1.46	1.26
20	-	7276	676	-	1.47	1.27
22	-	18094	1219	-	1.48	1.28
24	-	39135	1641	-	1.48	1.26

Figure 4.8 Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and A* algorithms with h₁, h₂. Data are averaged over 100 instances of the 8-puzzle, for various solution lengths.

Dades extretes del llibre AIMA (Russell i Norvig)