

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

CERCA LOCAL

Idea: Cercar la solució a un problema guardant només l'estat actual del problema i no els camins, i movent cap a estats veïns de l'estat actual.

Avantatges:

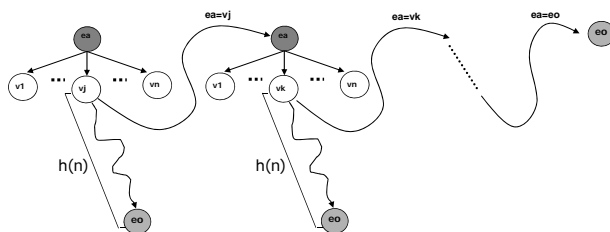
- Complexitat en memòria molt baixa quan no es busquen camins sinó simplement estats solució.
- Permeten treballar en entorns dinàmics (canviants).
- Funcionen molt bé en entorns continus (no discrets).
- Funcionen bé quan no es coneix l'estat objectiu i simplement es busquen màxims/mínims de la funció heurística.

Inconvenients:

- No es pot garantir la completesa.
- No es pot garantir l'optimalitat.

CERCA LOCAL

- **Cerca Hill-Climbing (Escalada):** expandeix només en el node que estima que és més proper a la solució segons la funció $h(n)$.



ALGORISME DE CERCA HILL-CLIMBING

(cas en que l'estat objectiu és conegut)

Funció CERCA_HC (EstatInicial, EstatObjectiu)

1. EstatActual=EstatInicial;
2. Repetir
 - a) $E = \text{Expandir_v}(\text{EstatActual})$;
 - b) $\text{MillorVei} = \text{Minim}(E, h)$;
 - c) $\text{EstatActual} = \text{MillorVei}$;
3. Fins que (EstatActual=EstatObjectiu);
4. Retornar(EstatActual);

Ffunció

Expandir_v(E): Retorna la llista de fills del node E

Minim(E,h): Retorna l'element de la llista E que té $h(E)$ mínim.

ALGORISME DE CERCA HILL-CLIMBING AMB MEMÒRIA

(cas en que l'estat objectiu és conegut)

Funció CERCA_HCMEMORIA (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) E=Ordenar(E,h)
 - e) Llista=Insertar_davant(E,Cua(Llista));
3. Fins que;
4. Si (Llista <> NIL) Retornar(Cap(Llista));
5. Sinó Retornar("No existeix Solució");

Funció

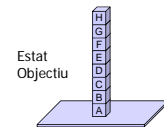
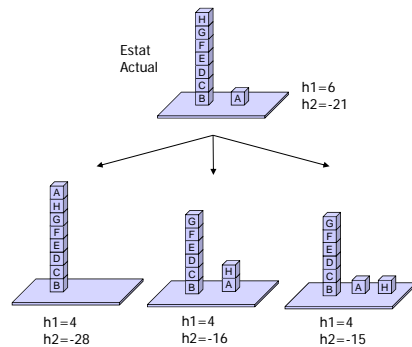
Ordenar(L,h): Retorna la llista L, després d'ordenar-la segons l'heurística h.

Insertar_davant(X,L): Retorna la llista L, després d'insertar-hi els elements de la llista X al davant.

Problemes de la cerca HC: existeixen situacions que fan que l'algorisme no pugui arribar a una solució

- **Màxims locals**, estat del problema que és millor que els seus veïns o fills, però que no és la solució.
- **Crestes**, seqüència d'estats que fan que s'encadenin molts de màxims locals de manera continua.
- **Planures**, estat del problema que és idèntic als seus veïns i que dificulta la presa de decisió.

Exemple d'un màxim local:



Heurística 1: (provoca màxim local)

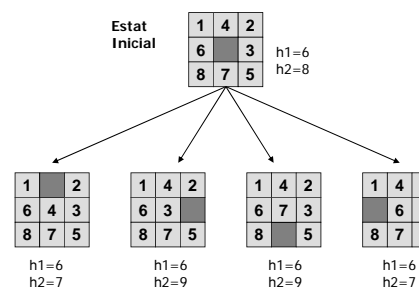
- Sumar un per a cada bloc que està sobre el bloc que li correspon a l'estat final.
- Restar un per a cada bloc que no està sobre el bloc que li correspon a l'estat final.

Heurística 2: (evita màxim local)

Direm que un bloc té la seva base correcta, si tot el que té a sota és correcta.

- Sumar el número de blocs de la base, per a cada bloc que té la base correcta.
- Restar el número de blocs de la base, per a cada bloc que té la base no correcta.

Exemple d'una planura:



1	2	3
8	6	4
7	6	5

Heurística 1: (provoca planura)

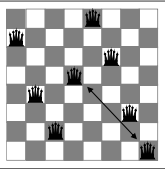
Número de rajoles que són diferents a l'estat final.

Heurística 2: (evita planura)

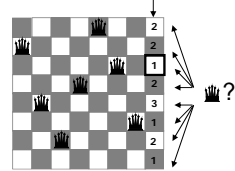
Suma de les distàncies (verticals i horitzontals) entre la posició de cada rajola i la seva posició a l'estat final.

Exemple: el problema de les 8 reines com a cerca local.

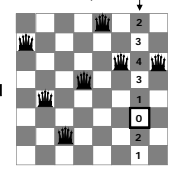
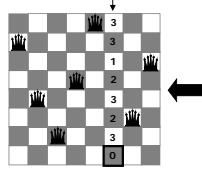
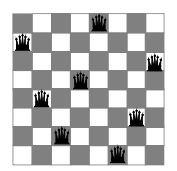
Estat Inicial: Totes les reines en files i columnes diferents però no es una solució.



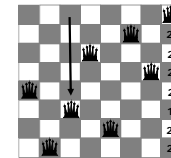
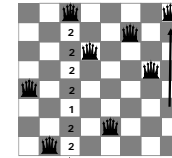
Heurística: Número de reines atacades des d'una casella concreta.



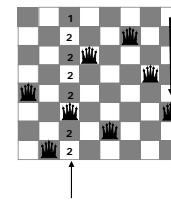
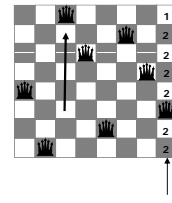
Estat Objectiu:



Exemple:



Heurística: Número de reines atacades des d'una casella concreta.

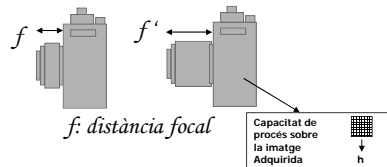


Problema: En alguns casos **no es coneix l'estat objectiu**

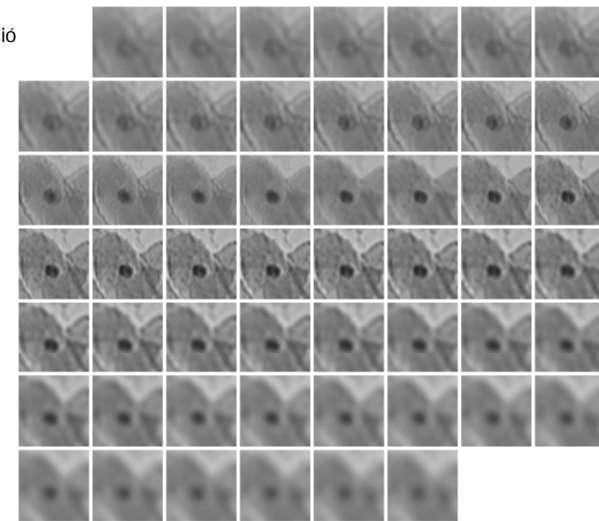
Una altra versió de HILL-CLIMBING = **STEEPEST ASCENT**

Idea: L'objectiu és l'estat que fa que la funció heurística tingui valor màxim, aleshores es tracta de fer escalada cap al màxim de la funció heurística. Aplicació a problemes d'optimització o de selecció de paràmetres.

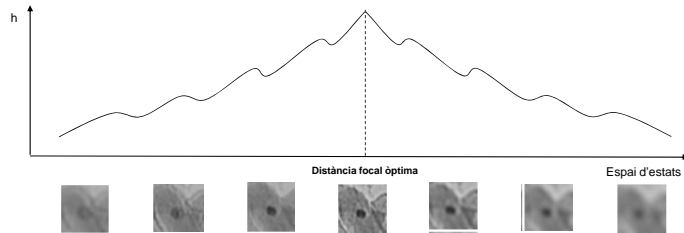
Exemple: cas d'un enfocament automàtic en una càmera digital



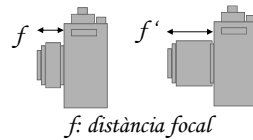
Com és la funció heurística?



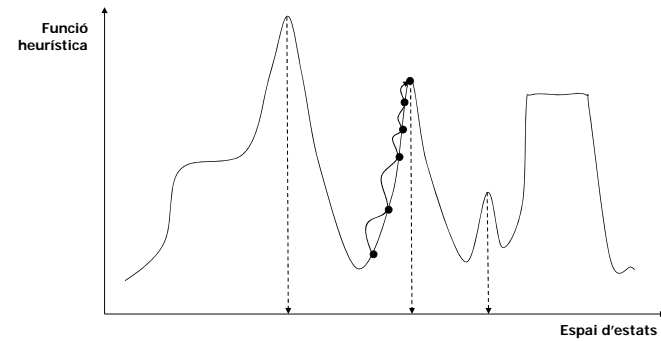
Exemple: cas d'un enfocament automàtic en una càmera digital



Heurística: estima el grau d'enfocament de la càmera a partir de l'aparença de la imatge digital adquirida.



REINTERPRETACIÓ DE LA CERCA LOCAL com a cerca de màxims en el paisatge definit per la funció heurística sobre tot l'espai de possibles estats.



ALGORISME DE CERCA STEEPEST ASCENT

(Cas en que es desconeix l'estat objectiu)

Funció CERCA_SA (EstatInicial)

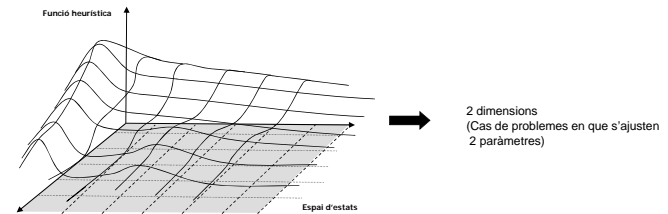
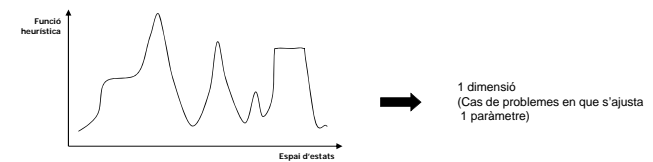
1. Successor=EstatInicial;
2. Repetir
 - a) EstatActual=Successor;
 - b) E=Expandir_v(EstatActual);
 - c) Successor=Màxim(E,h);
3. Fins que (h(EstatActual)≥h(Successor)); /* Estat Actual és màxim */
4. Retornar(EstatActual);

Ffunció

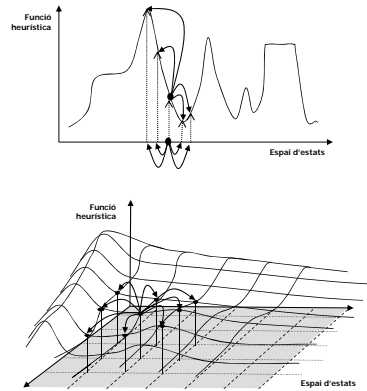
Expandir_v(E): Retorna la llista de fills del node E

Màxim(E,h): Retorna l'element de la llista E que té h(E) màxim.

Definició de l'espai d'estats: l'espai d'estats es pot estendre en espais de diferents dimensions, aquesta dimensió depèn del nombre de paràmetres que intervenen en la definició de l'heurística.



Selecció dels veïns d'un estat quan hi ha infinites possibilitats de direccions possibles i distància a l'estat actual.



Versió de Hill-Climbing/Steepest Ascent amb tractament de màxims locals

Funció Cerca_SATML (Max_iteracions, Millor_cas)

1. NumIteracions=0;
 2. Repetir
 1. Maxim_local=FALS;
 2. EstatActual=Estat_Aleatori();
 3. Repetir
 1. E=Expandir_v(EstatActual);
 2. Successor=Màxim(E,h);
 3. Si h(Successor)>h(EstatActual) llavors
 - EstatActual=Successor;
 4. Sinó
 - Maxim_Local=CERT;
 5. Fsi;
 4. Fins que (Maxim_Local=CERT);
 5. NumIteracions=NumIteracions+1;
 6. Si h(Estat_actual)>Millor_cas Llavors
 1. Millor_cas=h(EstatActual);
 2. EstatSolució=EstatActual;
 7. Fsi;
 3. Fins que (NumIteracions=Max_Iteracions);
 4. Retornar(Millor_cas,EstatSolucio);
- Ffunció

Estat_Aleatori(): Retorna un estat qualsevol de l'espai d'estats.

Max_iteracions: Inicialització que atura l'execució de l'algorisme.
 Millor_cas: Inicialització que estableix una cota inferior de les solucions que volem trobar.