Bloc 1 – Representació de Coneixement i Cerca

Tema 5:
Cerca Heurística

Bloc 1, Tema 5 - Índex

1. Cerca heurística
2. Cerca voraç
3. Cerca A*
4. Heurístiques per al problema del 8-puzle

Bibliografia

- S. Russell, P. Norvig. ***Artificial Intelligence. A modern approach.*** Prentice Hall, 3rd edició, 2010 (Capítol 3) http://aima.cs.berkeley.edu/

Alternativament:

- S. Russell, P. Norvig. ***Intel·ligència artificial . Una aproximació moderna.*** Prentice Hall, 2ª edició, 2004 (Capítols 3 i 4) http://aima.cs.berkeley.edu/2nd-ed/

1. Cerca heurística

Cerca heurística o informada: utilitza coneixement específic del problema per a guiar la cerca.

Pot trobar solucions més eficientment que una cerca no informada . És especialment útil en problemes complexos d'explosió combinatòria (p. ex.: problema de viatjant).

Per què utilitzar heurístiques? En ocasions no és viable utilitzar una cerca sistemàtica que garantisca optimalidad. La utilització d'algunes heurístiques permeten obtenir una *bona* solució encara que no siga l'òptima.

Guia intel·ligent del procés de cerca que permet podar grans parts de l'arbre de cerca.

Per què utilitzar heurístiques és apropiat?

1. Normalment, en problemes complexos, no necessitem solucions òptimes, una *bona* solució és suficient.
2. La solució de l'heurística per al cas pitjor podria no ser molt bona, però en el món real el cas pitjor és poc freqüent.
3. Comprendre el per què (per què no) funciona una heurística ajuda a aprofundir en la compressió del problema

1. Cerca heurística

Aproximació general cerca primer-el-millor:

- Procés de cerca en arbre o graf (TREE-SEARCH o GRAPH-SEARCH) en el qual un node se selecciona per a ser expandit sobre la base d'una funció d'avaluació $f(n)$
- $f(n)$ és una estimació de cost de manera que el node amb el cost més baix s'expandeix primer de la cua de prioritats
- Totes les estratègies de cerca es poden implementar usant $f(n)$; l'elecció de f determina l'estratègia de cerca
- Dos casos especials de cerca primer el millor: *cerca voraç* i *cerca A**.

3. Cerca A*

A*és l'algorisme més conegut de cerca primer el millor

Avalua els nodes combinat *g*(n), el cost d'aconseguir el node **n**, i h(n), el valor heurístic:
f(n)=g(n)+h(n)

f(n) és el cost total estimat de la solució òptima a través del node n

Els algorismes que utilitzen una funció d'avaluació de la forma **f(n)=g(n)+h(n)** es denominen algorismes de tipus A.

3. Cerca A*

La cerca A* utilitza una funció heurística admissible

Una heurística és admissible si mai sobreestima el cost per a aconseguir l'objectiu.

Formalment:

- Una heurística h(n) és admissible si $\forall n$, $\mathbf{h(n) \leq h^*(n)}$, on $h^*(n)$ és el cost real d'aconseguir l'objectiu des de l'estat **n**.

- En utilitzar un heurístic admissible, la cerca A* retorna la solució òptima

- h(n) >= 0 així que h(*G*)=0 per a qualsevol objectiu *G*

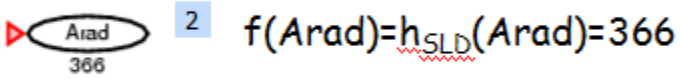
P. ex. $h_{SLD}(n)$ mai sobreestima la distància en carretera real entre dues ciutats

3. Cerca A*: l'exemple de Romania

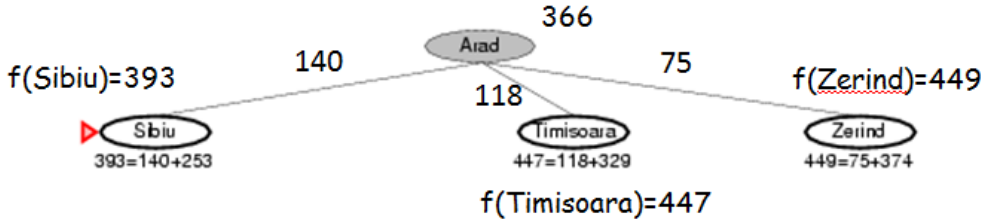
$f(n)=g(n)+h(n)$

- $h(n)=h_{SLD}(n)$
- expandeix node amb el menor cost estimat total
- Cerca A*

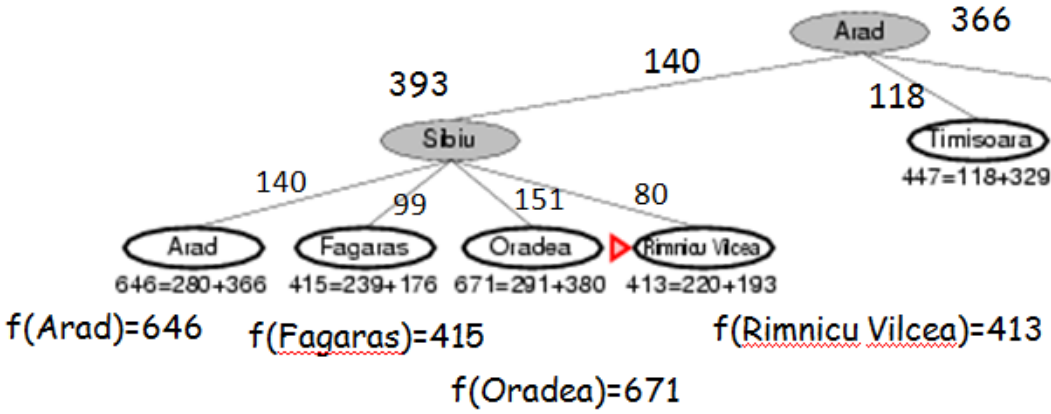
Iteració 1:



Iteració 2:



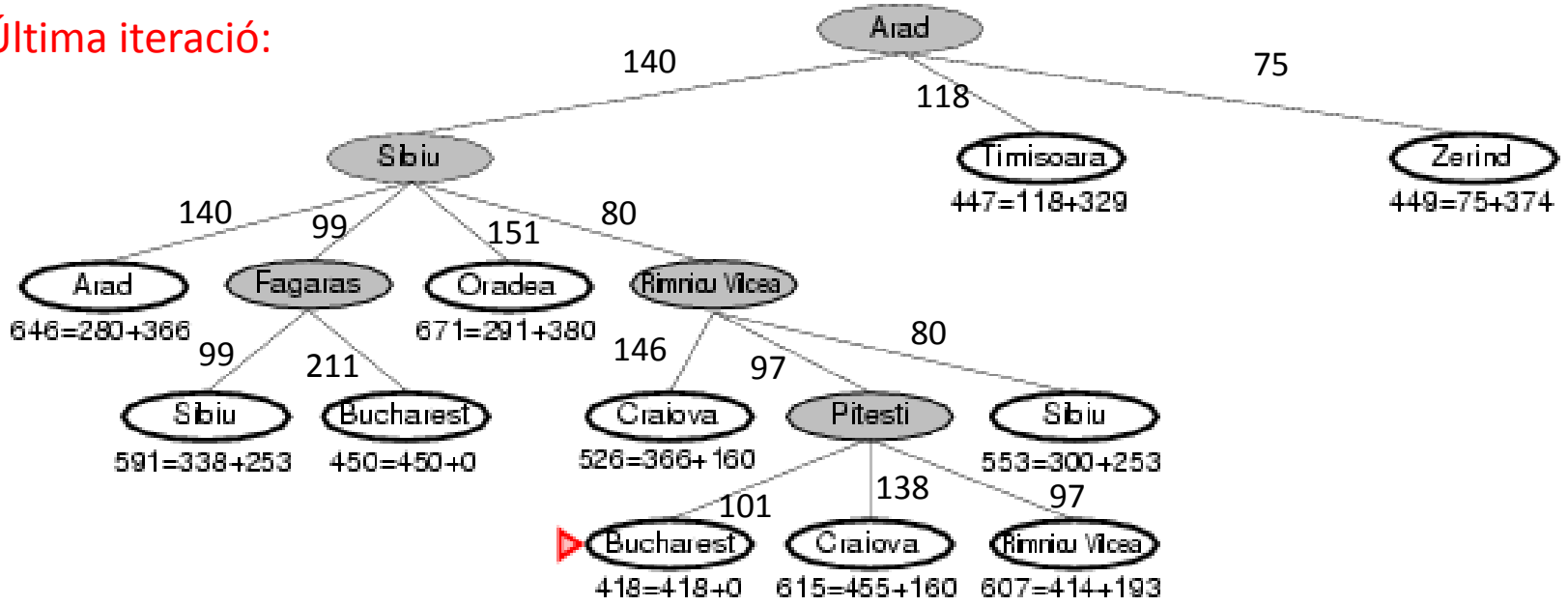
Iteració 3:



En la versió GRAPH-SEARCH: Arad és un estat repetit; el node Arad de la llista CLOSED té un cost millor.

3. Cerca A*: l'exemple de Romania

Última iteració:

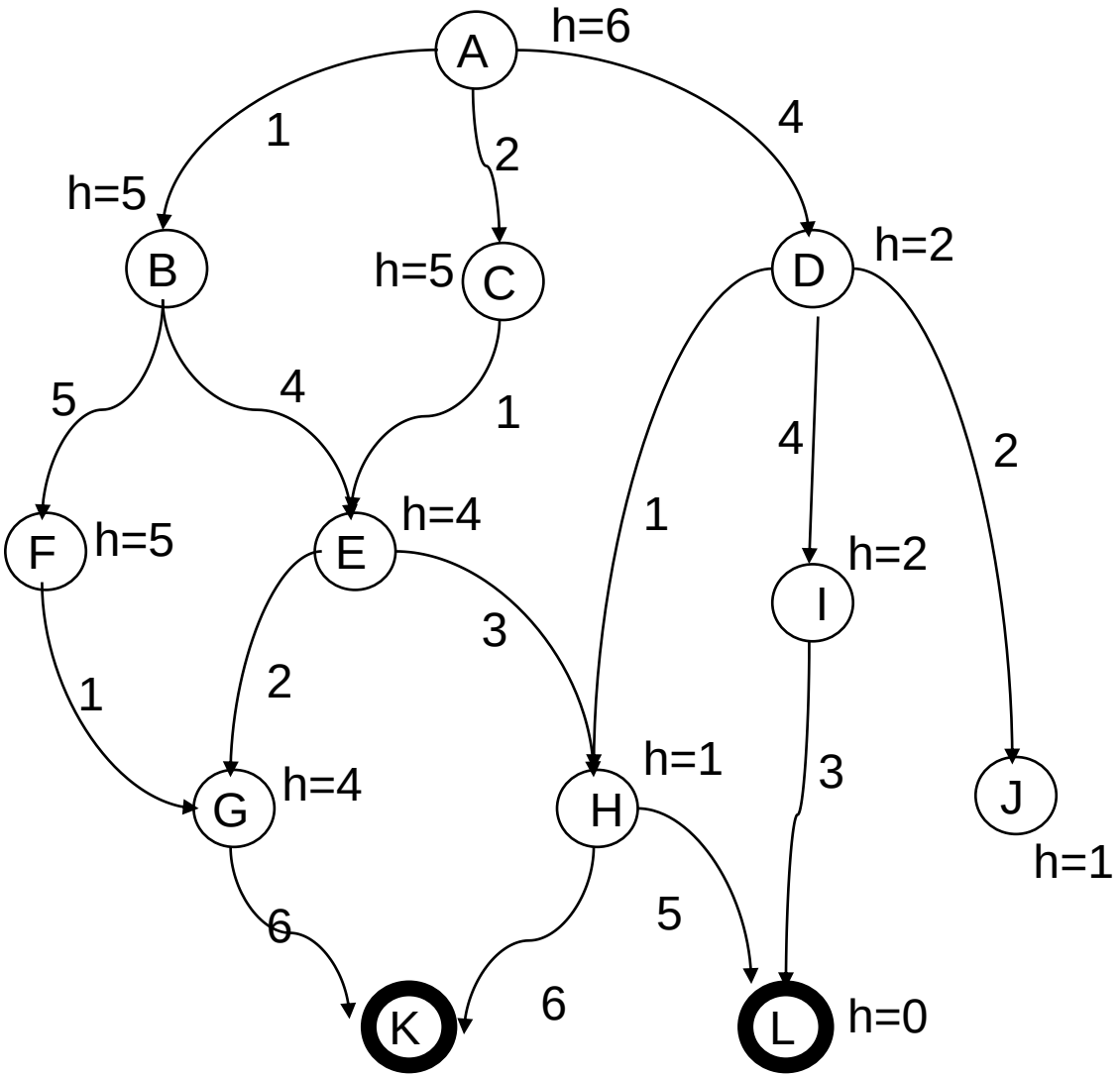


llista OPEN= {Bucarest(418), Timisoara(447), Zerind(449), Craiova(526), Oradea (671)}

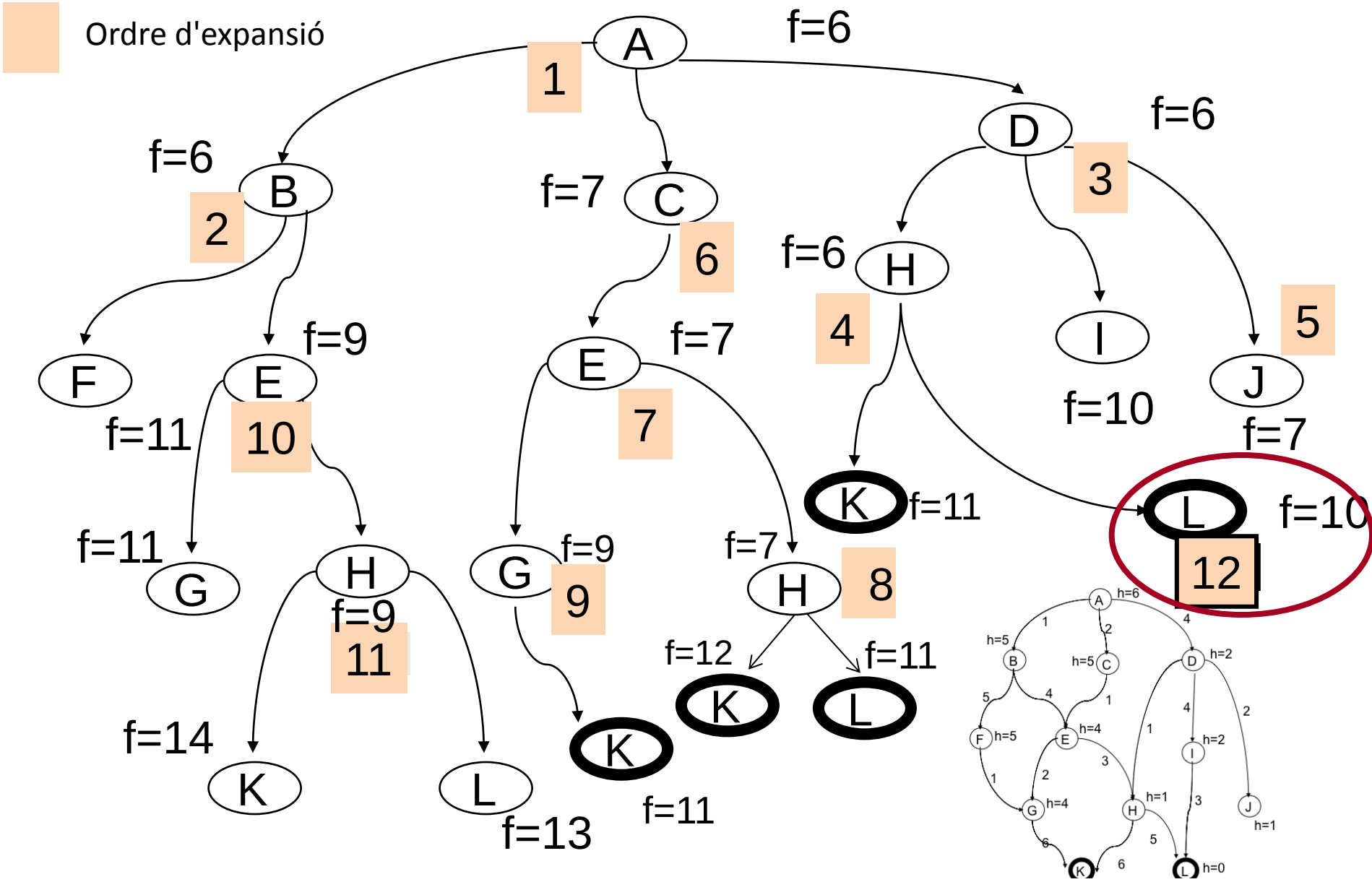
llista CLOSED = {Arad, Sibiu, Rimnicu Vilcea, Fagaras, Pitesti}

El node Bucarest ja està en OPEN amb cost=450. El nou node té un cost estimat menor (cost=418) que el node que està en OPEN. Reemplacem el node de Bucarest en OPEN amb el nou node trobat.

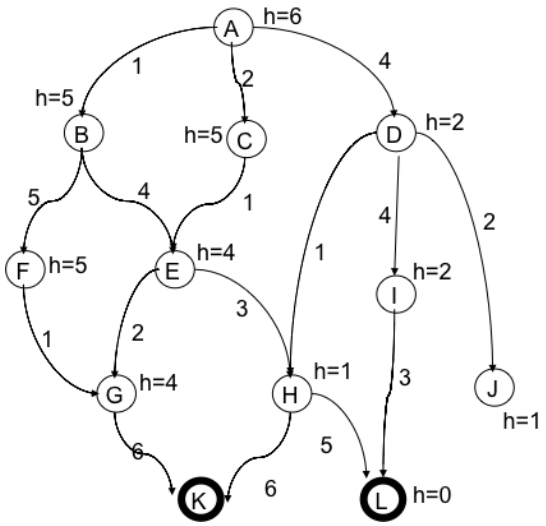
3. Cerca A*: un altre exemple



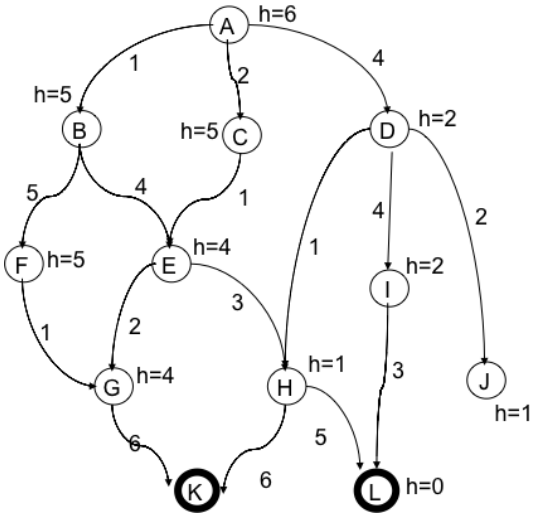
3. Cerca A*: versió TREE-SEARCH sense control de nodes repetits



3. Cerca A*: versió TREE-SEARCH amb control de nodes repetits



3. Cerca A*: versió GRAPH-SEARCH



3. Cerca A*: comparació i anàlisi

- Comparació amb altres estratègies de cerca:
  - Cerca en amplària (òptima si tots els operadors tenen el mateix cost). Equivalent a f(n)=nivell(n)+0, on $h(n)=0 < h^*(n)$
  - Cost uniforme (òptima). Equivalent a f(n)=g(n)+0 on $h(n)=0 < h^*(n)$
  - Profunditat (no òptima). No comparable a A*

- Coneixement heurístic:
  - h(n)=0, absència de coneixement
  - $h(n)=h^*(n)$, coneixement màxim
  - Si $h2(n) \geq h1(n)$ $\forall$n (tots dos admissible) llavors h2 domina a h1 (h2 és més informat que h1); h2 mai expandirà més nodes que h1

3. Cerca A*: comparació i anàlisi

h(n) = 0: cost computacional d'h(n) nul. (Cerca lenta. Admissible.)
h(n) = h*(n): cost computacional gran d'h(n) . (Cerca ràpida. Admissible).
h(n) > h*(n): cost computacional d'h(n) molt alt. (Cerca molt ràpida. No admissible)

En general, h* no és conegut però és possible establir si h és una cota inferior d'h* o no.

Per a problemes molt complexos es recomana reduir l'espai de cerca. En aquests casos, paga la pena utilitzar h(n) > h*(n) amb l'objectiu de trobar una solució en un cost raonable fins i tot encara que aquesta no siga la solució òptima.

Per a cada problema, trobar un equilibri entre el cost de cerca i el cost del camí solució.

3. Cerca A*: optimalitat

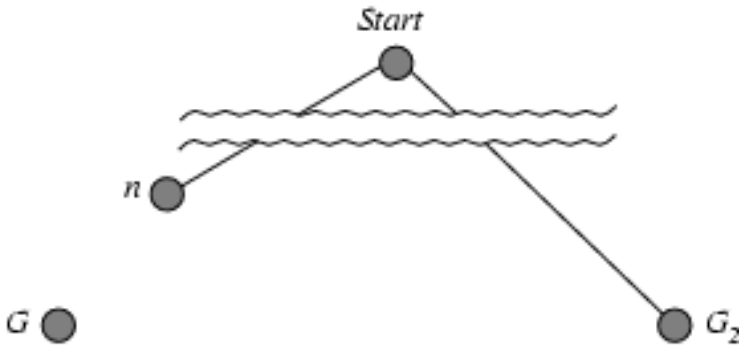
Condicions per a l'optimalitat de A*:

1. h(n) ha de ser una heurística admissible
2. h(n) ha de ser una heurística consistent (només necessari per a la versió GRAPH-SEARCH si no es reexpandeix el node en CLOSED)

Admissibilitat:

- h(n) és admissible si $\forall n, h(n) \leq h^*(n)$
- Siga *G* un estat objectiu òptim, i n un node en el camí a *G*. f(n) mai sobreestima el cost del camí a través de n.
  - $f(n)= g(n)+h(n) \leq g(n)+h^*(n) = g(G) = f(G) \Rightarrow f(n) \leq g(G)$ (1)

Provar que si h(n) és admissible, llavors A * és òptim:



G és un estat objectiu òptim: $f(G) = g(G)$

G2 és un estat subòptim en la llista OPEN amb cost
$f(G2) = g(G2) > g(G)$ (2)

n és un node de la llista OPEN en el camí òptim a G .

Si no s'escull n per a expansió llavors $f(n) >= f(G2)$ (3)

Si combinem (1) i (3) :
$f(G2) <= f(n) <= g(G) \Rightarrow g(G2) \leq g(G)$
Açò contradiu (2) així que s'escull n

3. Cerca A*: optimalitat

Consistència (també anomenada *monotonicidad*) és una condició lleugerament més forta que l'admissibilitat.

Consistència: $h(n)$ és consistent si, per a cada node n i cada successor n' de n generat amb una acció a es compleix $h(n) <= h(n') + c(n,a,n')$

En el cas de:

- aplicació de l'algorisme GRAPH-SEARCH amb $f(n)=g(n)+h(n)$, i
- es troba un node n repetit que ja està en CLOSED i el valor del qual $f(n)$ és menor que el valor $f(n)$ del node en CLOSED i s'opta per no re-expandir el node, i
- es compleix que $h(n)$ és monòtona

llavors l'algorisme GRAPH-SEARCH retorna la solució òptima perquè

- la seqüència de nodes expandits per GRAPH-SEARCH és una funció no decreixent de $f(n)$
- es garanteix que quan s'expandeix un node n, s'ha trobat la solució òptima a aquest node, per la qual cosa no és necessari tornar a re-expandir n en cas que siga un node repetit (el cost del node repetit mai serà millor que el cost del node ja expandit).

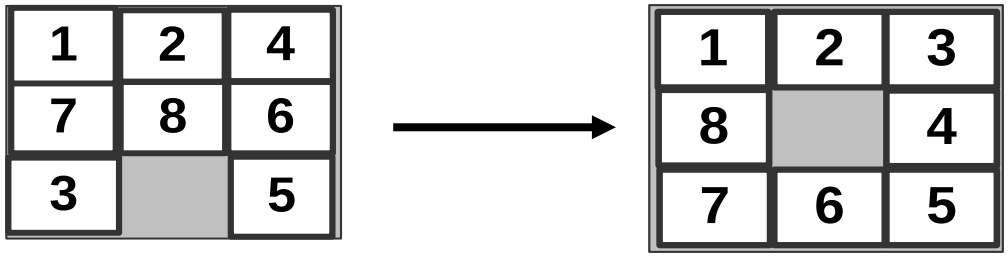
3. Cerca A*: avaluació

- Complet:
  - Sí, llevat que existisquen infinits nodes **n** tal que $f(n) < f(G)$
- Òptima:
  - Sí, si es compleix la condició de consistència (per a la versió GRAPH-SEARCH)
  - Sempre existeix almenys un node **n** en OPEN que pertany al camí òptim de la solució
  - Si C^* és el cost de la solució òptima:
    - A* expandeix tots els nodes amb $f(n) < C^*$
    - A* podria expandir alguns nodes amb $f(n) = C^*$
    - A* no expandeix nodes amb $f(n) > C^*$

- Complexitat temporal:
  - $O(b^{C^*/\text{min\_cost\_acció}})$; exponencial amb la longitud de camí
  - El nombre de nodes expandits és exponencial amb la longitud de la solució

- Complexitat espacial:
  - Manté tots els nodes en memòria (com totes les versions GRAPH-SEARCH)
  - A* normalment es queda sense espai molt abans que s'esgote el temps
  - Per tant, el major problema és l'espai, no el temps

4. Heurísticas per al problema de 8-puzle



Estat inicial

Estat objectiu

- El problema de 8-puzle:
  - El cost mitjà d'una solució és aproximadament 22 passos (factor de ramificació ≈ 3)
  - Cerca exhaustiva a profunditat 22: $3^{22} \approx 3.1 \times 10^{10}$ estats
  - Una bona funció heurística pot reduir el procés de cerca

- **Heurístic 1:**
  - h1(n): nombre de fitxes descol·locades
  - h1(n)=5 per a l'exemple de l'estat inicial

- **Heurístic 2:**
  - h2(n): distàncies de Manhattan (sumia de les distàncies de cada fitxa a la seua posició objectiu)
  - h2(n)=1+2+4+1+1=9 per a l'exemple de l'estat inicial

Distàncies de Manhattan domina a fitxes descol·locades (h2(n)>= h1(n), $\forall$n)

4. Heurístiques per al problema del 8-puzle

Cerca A* amb l'heurística
fitxes descol·locades