Búsqueda Heurística IV

Pedro Meseguer
IIIA-CSIC
Bellaterra, Spain
pedro@iiia.csic.es

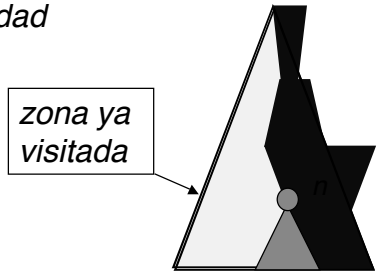
Búsqueda heurística con memoria acotada

- Motivación: alto coste en memoria de A*
- Aproximaciones:
  - *Depth-first branch-and-bound*
  - Profundización iterativa A* (IDA*)
  - DFS*
  - MREC
  - SMA*
  - RBFS

Depth-first Branch-and-Bound

búsqueda en profundidad

h admisible



UB = coste de la mejor solución encontrada

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

si UB ≤ LB, sabemos que debajo de *n* no habrá solución de menor coste que la ya encontrada → podamos subárbol

DFBnB

- Algoritmo para árboles
- Solución: si *h* admisible, solución óptima
- Memoria: lineal

Complejidad espacial: $O(bd)$

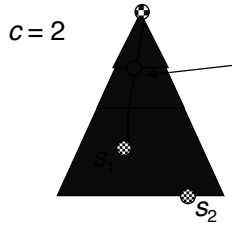
- Tiempo: exponencial

Complejidad temporal: $O(b^d)$

IDA* = ID + A*

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

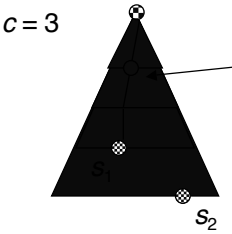
$h(n) \leq h^*(n)$ (admisible)



$f(n) > c$ ¿qué significa?

$f(n) \leq f(s_1)$
$c < f(n)$
} $c < f(s_1)$

ID no encuentra s_1 en esta iteración → se poda n



$f(n) \leq c$

n no se poda

ID encuentra s_1

IDA*

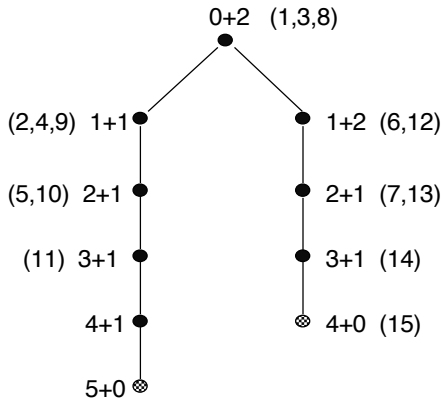
iteración con cota creciente

búsqueda en profundidad

sucesores sólo dentro de la cota

1. Cota profundidad $c \leftarrow f$(raíz)
2. $L \leftarrow$ nodo raíz, siguiente cota $c' \leftarrow \infty$.
3. Si L vacía, y $c' = \infty$, fallo, stop.
   Si L vacía, y $c' \neq \infty$, $c \leftarrow c'$, ir a 2.
   Sino, $n \leftarrow$ extrae-primero(L).
4. Si n objetivo, éxito, stop. Sino, generar sucesores de n.
5. Para cada n' sucesor de n :
   Si f(n') < c, añadir n' al comienzo de L .
   Sino, $c' \leftarrow \min(c', f(n'))$.
   Ir al paso 2.

IDA*: ejemplo



IDA*: admisibilidad

- Si *h* admisible, IDA* encuentra solución óptima

Suponemos *h* monótona (sino la dem es más compleja)

1. Primer $c = h$(raíz); como *h* admisible no hay solución con coste menor.

2. Siguiente *c* : menor $f(n)$ de los que exceden *c* anterior. No puede haber solución entre dos cotas consecutivas.

3. Como *h* monótona, las cotas son crecientes.

4. IDA* expande nodos con coste creciente: por tanto la primera solución es la de menor coste.

IDA*: complejidad

- Calidad: si *h* admisible, IDA* encuentra la solución óptima
- Espacio: lineal en *d*

Complejidad espacial: O(bd)

- Tiempo: comparamos con A*

A* requiere *k* nodos

IDA* (caso peor): los *k* nodos tienen distinta *f*

en cada iteración se visita 1 nuevo nodo

$1 + 2 + 3 + ... + k\text{-}1 + k = k(k - 1) / 2 \qquad k \approx b^d$

Complejidad temporal: O(b^{2d})

IDA* requiere visitar O(b^{2d}) nodos → mucho más lento que A*

En la práctica:

pocos valores distintos de *f*: IDA* bien

muchos valores distintos de *f*: IDA* muy lento

IDA*: Memoria y tiempo

- IDA* utiliza muy poca memoria
  - el camino actual
  - las cotas de coste *c* y *c'*
- Sucesivas iteraciones:
  - reexpanden nodos: repiten trabajo
  - la última iteración no domina a las anteriores (como ID)
  - puede llegar a ser muy lento
- ¿Algoritmo intermedio entre A* e IDA*?
  - Memoria / Tiempo

DFS*

- Objetivo: reducir reexpansión de nodos en IDA*

1. Antes de encontrar una solución, IDA* con
   c' > menor coste de los nodos podados en la iteración actual;
   la primera solución no es necesariamente óptima

2. Tras encontrar la primera solución, DFBnB
   (búsqueda de la solución óptima)

   DFS en todo el árbol, con cota = coste solución
   Si encuentra una solución con menor coste,
   continua DFS con cota = coste nueva solución

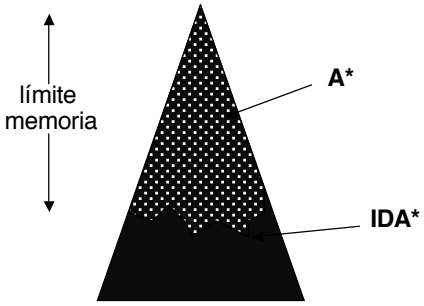
DFS*: Complejidad

- Calidad: si *h* admisible, encuentra la solución óptima

- Espacio: lineal en *d*

- Tiempo: exponencial en *d*
  solución subóptima, OK
  solución óptima: DFBB es costoso en tiempo

MREC



IDA* sobre nodos frontera:

1. $n \leftarrow$ extrae-mínima-f(*OPEN*), $c \leftarrow f(n)$
2. Iteración IDA*(n ,c). Si solución, final.
3. Sino, $c' \leftarrow$ menor $f(n')$, n' podado en IDA*(n ,c).
4. $f(n) \leftarrow c'$, añadir n a *OPEN*. Ir a 2.

MREC: Complejidad

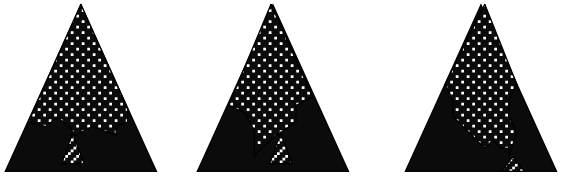
- Calidad: si *h* admisible, encuentra la solución óptima
- Espacio: bloque memoria + lineal en *d*
  (A*) (IDA*)
- Tiempo: exponencial en *d*

Simplified Memory-bounded A*: SMA*

MREC

MREC: memoria A* estática, tras búsquedas IDA* la evaluación de los nodos frontera cambia. ¿Foco en los más prometedores?

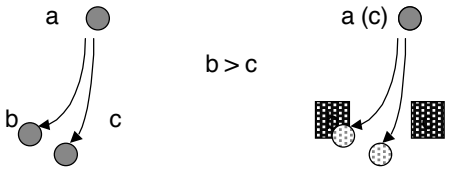
SMA*



SMA* (I)

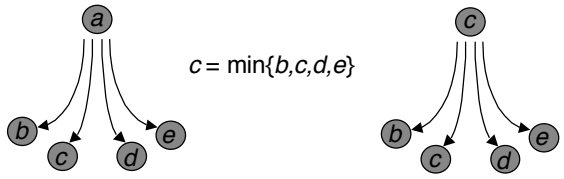
- Memoria acotada: MAX
- Generación individual de sucesores
- Si nodos generados < MAX: A*

  = MAX: elimina nodos menos prometedores
- Mantiene información de nodos eliminados:
  - El nodo padre mantiene el mejor $f(n)$ de los sucesores podados.



SMA* (II)

- Cuando se generan todos los sucesores, el coste del padre se actualiza al coste del mejor sucesor.



Reexpande un nodo: cuando los otros caminos son peores que el mejor sucesor de este nodo

SMA*

1. $L \leftarrow$ nodo raíz.
2. Si L vacía, fallo, stop. Sino, $n \leftarrow$ nodo-mínimo-$f(L)$.
3. Si n objetivo, éxito, stop. Sino, $s \leftarrow$ sig-sucesor(n).
4. Si s no solución y a profundidad máxima, entonces $f(s) \leftarrow \infty$

   Sino, $f(s) \leftarrow \max\{f(n), g(s) + h(s)\}$
5. Si todos sucesores de n generados, $f(n) \leftarrow \min\{f(s) | s \text{ sucesor}\}$; prop ant($n$)
6. Si todos sucesores de n en memoria, elimina n de L
7. Si memoria llena,

   $m \leftarrow$ extrae-máximo-$f(L)$.

   actualiza coste sucesores podados padre(m)

   Si padre(m) no esta en L, $L \leftarrow$ añade(padre(m), L)
8. $L \leftarrow$ añade(s, L)
9. Ir a 2.

SMA*: Complejidad

- Calidad: si *h* admisible, monótona y el camino a la solución cabe en memoria, encuentra la solución óptima
- Espacio: lineal en *d*
- Tiempo: exponencial en *d*

Resumen

- Diversas propuestas para acotar la memoria de A*
- DFBnB, IDA*, DFS*:
  - memoria lineal
  - pueden tardar mucho tiempo
- MREC, SMA*:
  - bloque de memoria
  - pueden ser más eficientes en tiempo