UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Búsqueda no informada [1]

Alfons Juan
Albert Sanchis
Jorge Civera

DSIC

Departamento de Sistemas Informáticos y Computación

[1] Para una correcta visualización, se requiere Acrobat Reader v. 7.0 o superior

1. Problemas de búsqueda

Caracterización de los ***problemas de búsqueda:***

- ***Espacio de estados:*** conjunto de posibles estados.
- ***Estado inicial*** s_0***:*** estado donde empieza la búsqueda.
- ***Acciones(***s***):*** acciones aplicables en el estado s.
- ***Resultado(***s, a***):*** estado sucesor al aplicar acción a a s
- ***Objetivo(***s***):*** indica si el estado s es solución o no.
- ***Coste(***c***):*** coste del camino c (secuencia de estados).

Objetivo: encontrar un estado solución (óptimo si es posible)

Ejemplo: 8-puzzle

- *Espacio de estados:* posibles configuraciones de fichas y vacío.
- *Estado inicial* s_0*:* configuración inicial de fichas y espacio vacío.
- *Acciones(*s*):* movimientos del espacio vacío.
- *Resultado(*s, a*):* evidente.
- *Objetivo(*s*):* configuración final de fichas y espacio vacío.
- *Coste(*c*):* coste unitario por cada paso.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

3. Búsqueda en árbol

EnÁrbol(n_0, L**)** // nodo inicial y límite de profundidad (e.g. ∞)

$OPEN = \{n_0\}$ // inicialización de la frontera

bucle

si $OPEN = \emptyset$ **devuelve** NULL // solución no encontrada

$s = \underset{n \in OPEN}{\arg\min}\ f(n)$ // selecciona un nodo de mínimo f

si $Objetivo(s)$ **devuelve** s // ¡solución encontrada!

$OPEN = OPEN - \{s\}$ // elimínalo de la frontera

si $Profundidad(s) < L$ **para todo** $n \in Hijos(s)$:

si $n \notin OPEN$: $OPEN = OPEN \cup \{n\}$

si no deja en $OPEN$ el de menor f

4. Búsqueda en grafo

EnGrafo(n_0, L**)** // nodo inicial y límite de profundidad (e.g. ∞)

$OPEN = \{n_0\}$ // inicialización de la frontera

$CLOSED = \emptyset$ // inicialización del conjunto explorado

bucle

si $OPEN = \emptyset$ **devuelve** NULL // solución no encontrada

$s = \underset{n \in OPEN}{\arg\min}\ f(n)$ // selecciona un nodo de mínima. f

si $Objetivo(s)$ **devuelve** s // ¡solución encontrada!

$OPEN = OPEN - \{s\}$ // elimínalo de la frontera

$CLOSED = CLOSED \cup \{s\}$ // añade al conjunto explorado

si $Profundidad(s) < L$ **para todo** $n \in Hijos(s)$:

si $n \notin CLOSED$

si $n \notin OPEN$: $OPEN = OPEN \cup \{n\}$

si no deja en $OPEN$ el de menor f

si no si tiene menor f que el de CLOSED:

borra el de CLOSED e inserta n en OPEN

5. Funciones de evaluación

- ***Anchura:*** la frontera es una ***cola de prioridad (heap)***

$$f(n) = Profundidad(n)$$

- ***Uniforme:*** la frontera es una ***cola de prioridad (heap)***

$$f(n) = g(n)$$

- ***Profundidad (limitada):*** la frontera es una ***pila***

$$f(n) = -Profundidad(n)$$

  - ***Profundización iterativa:*** profundidad limitada repetida con límite creciente hasta encontrar una solución

6. Propiedades

Asumiendo acciones de coste positivo y $L = \infty$:

- ***Completitud:*** ¿siempre encuentra solución (si la hay)?
  - Anchura, uniforme y profundización iterativa.
  - Profundidad con búsqueda en grafo (control estados repetidos).
- ***Optimalidad:*** ¿siempre encuentra una solución óptima?
  - Uniforme.
  - Anchura y prof. iterativa con acciones de coste idéntico.
- ***Complejidad:*** factor de ramificación b y hasta profundidad d
  - ***Temporal:*** $O(b^d)$
  - ***Espacial:*** Anchura y uniforme: $O(b^d)$
    Profundidad (limitada): $O(b \cdot d)$ en árbol; $O(b^d)$ en grafo

7. Búsqueda en árbol con backtracking

EnArbolConBacktracking(n_0, L**)** // nodo inicial y límite de prof.

$OPEN = \{n_0\}$ $PATH = \emptyset$ // inicialización

bucle

si $OPEN = \emptyset$ **devuelve** NULL // solución no encontrada

$s = \underset{n \in OPEN}{\arg\min}\ f(n)$ // selecciona un nodo de mínima. f

si $Objetivo(s)$ **devuelve** s // ¡solución encontrada!

$OPEN = OPEN - \{s\}$ // elimínalo de la frontera

$PATH = PATH \cup \{s\}$ // añade al conjunto explorado

si $Profundidad(s) < L$ **para todo** $n \in Hijos(s)$:

si $n \notin OPEN$: $OPEN = OPEN \cup \{n\}$

si no deja en $OPEN$ el de menor f

si $Profundidad(s) = L$ **o** $Hijos(s) = \emptyset$

hacer

$PATH = PATH - \{s\}$

$s = \underset{n \in PATH}{\arg\min}\ f(n)$

mientras $Hijos(s) \cap OPEN = \emptyset$ **y** $PATH \neq \emptyset$

8. Conclusiones

Hemos visto:

- Como caracterizar los problemas de búsqueda y como distinguir entre espacio de estados y árbol de búsqueda.
- Las estrategias de búsqueda en árbol y grafo, así como algunas técnicas usuales de búsqueda no informada que se derivan.
- Consultad [1, Cap. 3] para más detalles.

Referencias

[1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.