Búsqueda No Informada

Sistemas Inteligentes

Dr. Víctor de la Cueva
vcueva@itesm.mx

Búsqueda no informada

- También llamada búsqueda ciega.
- El término significa que la estrategia no tiene información adicional acerca del problema más allá de la proporcionada por su definición.
- Las estrategias son distinguidas por el orden en el cual los nodos son expandidos.
- Las estrategias que saben si un nodo es "más prometedor" que otro se llaman búsqueda informada o búsqueda heurística.

Búsqueda a lo ancho

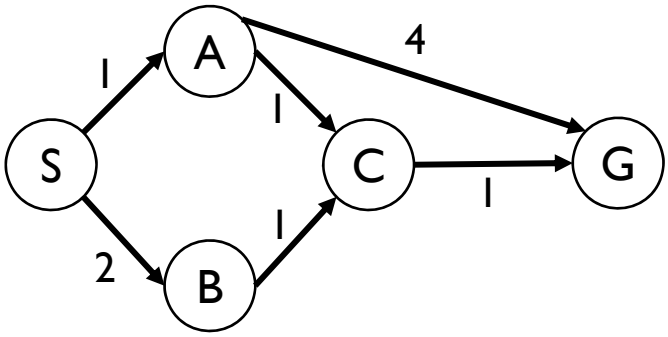
- *Breadth-first search* (BFS).
- Es una estrategia simple en la cual el nodo raíz se expande primero, luego todos su sucesores, luego los sucesores de los sucesores y así sucesivamente.
- En general, todos los nodos de una misma profundidad se expanden antes que los nodos de la profundidad siguiente.
- Se implementa fácilmente usando una cola (estructura FIFO) para la frontera.

Algoritmo BFS

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier) /* chooses the shallowest node in frontier */
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node , action)
      if child.STATE is not in explored or frontier then
        if problem GOAL-TEST(child.STATE) then return SOLUTION(child)
        frontier ← INSERT(child, frontier)
```

Ejemplo de BFS

Grafo de Espacio de Estados



Fuente: MOOC AI, EdX, Dan Klein. UC Berkeley (2013)

Análisis BFS

- Completo
- No necesariamente óptimo. Sólo es óptimo si el costo del camino es una función no decreciente de la profundidad.
- $O(b^{d+1})$
- $O(b^d) = O(b^{d-1}) + O(b^d)$ (explorados + frontera)

Búsqueda de costo uniforme (UCS)

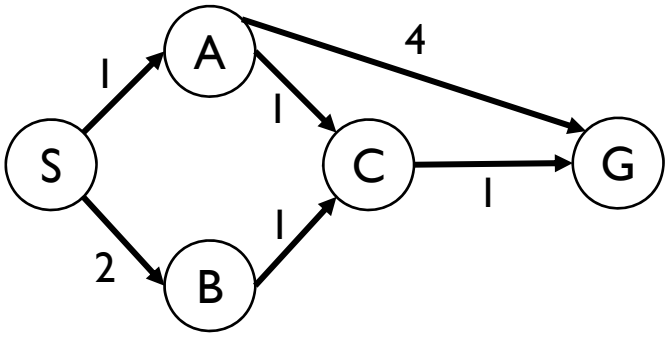
- Cuando todos los costos de paso son iguales, BFD es óptimo.
- Como una simple extensión, podemos encontrar un algoritmo que es óptimo con cualquier función de costo de paso.
- En lugar de expandir el nodo más superficial, UCS expande el nodo n con el menor costo de camino g(n).
- Esto se hace almacenando la frontera en una cola con prioridad ordenada por g.

Algoritmo UCS

```
function UNIFORM- COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier) /* chooses the lowest-cost node in frontier */
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        frontier ← INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child
```

Ejemplo de UCS

Grafo de Espacio de Estados



Fuente: MOOC AI, EdX, Dan Klein. UC Berkeley (2013)

Variante de BFS

- Búsqueda Bidireccional
  - Se ejecutan dos búsquedas simultáneamente, una hacia adelante a partir de EI y otra hacia atrás a partir del EM.

    $b^{d/2}+b^{d/2} << b^{d}$
  - Comprueba si un nodo está en la frontera del otro.
  - Requiere de una función reversible:

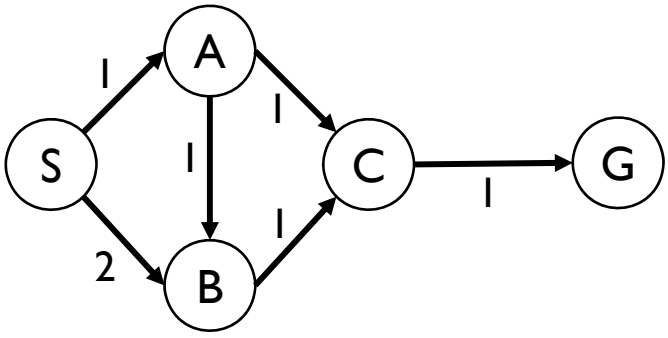
    Predecesor(n) = Sucesor(n-1)

Búsqueda en Profundidad (DFS)

- Depth-first search (DFS).
- Siempre expande el nodo más profundo en la frontera actual del árbol de búsqueda.
- Si un nodo no tiene sucesores, el proceso "regresa hacia arriba" y sigue con el siguiente nodo más profundo.
- Se implementa fácilmente usando una estructura LIFO (pila o *stack*), lo cual significa que el nodo más recientemente generado se selecciona para la expansión.

Ejemplo de DFS

Grafo de Espacio de Estados



Fuente: MOOC AI, EdX, Dan Klein. UC Berkeley (2013)

Análisis de DFS

Ventajas

- Ahorra mucha memoria porque en un momento solamente guarda un camino completo de la raíz a la hoja.
- Se programa fácilmente en forma recursiva.

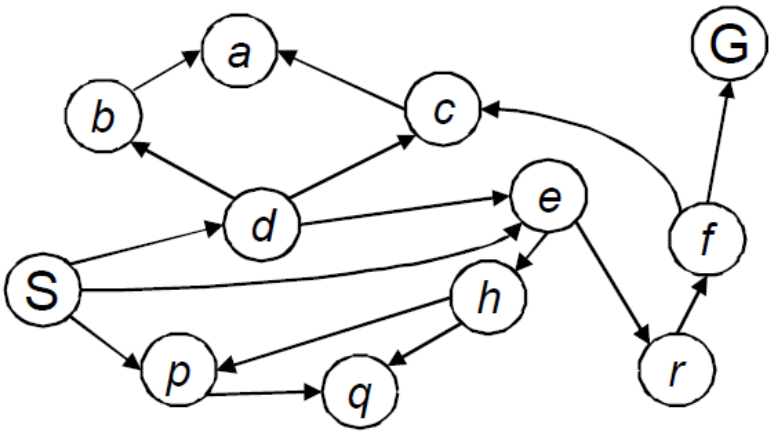
Desventajas

- Puede seleccionar un camino muy largo
- Puede ciclarse

Variantes de DFS

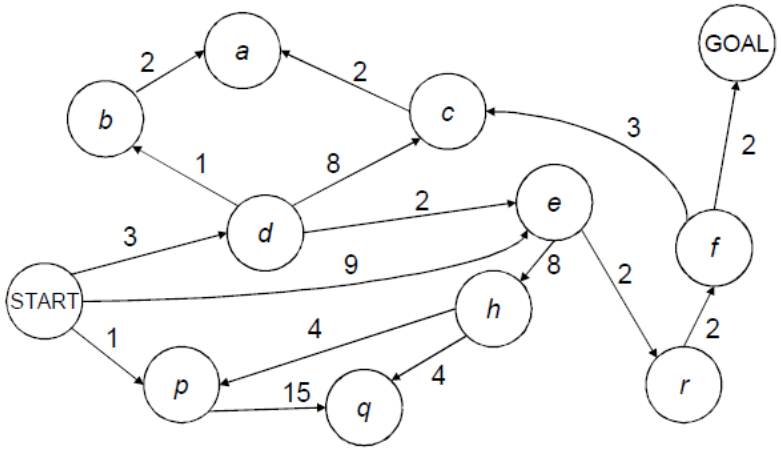
- DFS con profundidad limitada
  - Se establece un límite de profundidad l
    - Si l < d es "muy" no completo
    - Si l > d es "muy" no óptimo
  - A veces se puede establecer un límite bueno si se conoce el problema.
- DFS con profundidad iterativa
  - El límite l va aumentando de 1 a n.
  - Puede parecer muy costosa porque los estados se generan muchas veces pero esto no es costoso.

Ejemplo DFS y BFS



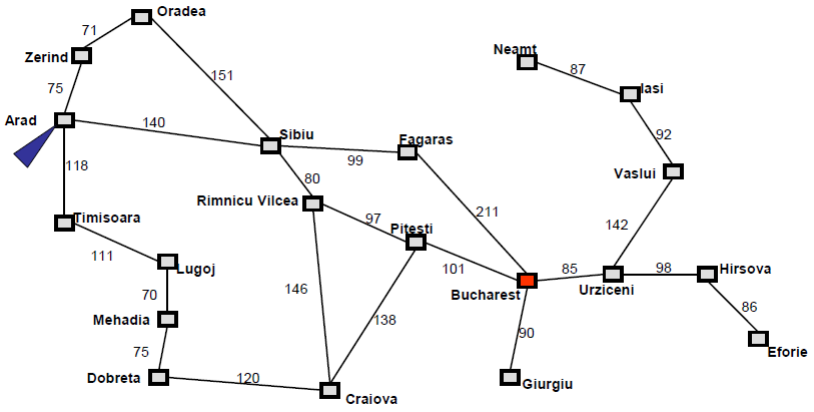
Fuente: MOOC AI, EdX, Dan Klein. UC Berkeley (2013)

Para UCS



Fuente: MOOC AI, EdX, Dan Klein. UC Berkeley (2013)

Un tour por Rumania



Referencia

- S. Russel and P. Norvig. Inteligencia Artificial un enfoque moderno. 2ª edición, Pearson, España (2004).