TEMA 5: PROCESOS DE BÚSQUEDA

- Introducción
- Búsqueda a ciegas
  - Búsqueda en anchura
  - Búsqueda de coste uniforme
  - Búsqueda en profundidad
  - Búsqueda bidireccional
- Búsqueda informada
  - Concepto de heurística
  - Método de ascensión de colinas
  - Búsqueda por el mejor nodo
  - Algoritmo A y A*

INTRODUCCIÓN

- Un problema puede definirse por cuatro componentes:
  - Estado inicial
  - Descripción de las acciones u operadores disponibles. Función sucesor
  - Test objetivo
  - En algunos casos, una función coste del camino
- El estado inicial y la función sucesor definen el espacio de estados del problema.
- La solución es el camino del estado inicial a un estado objetivo y la optima será la de coste menor.
- Problemas de juguete vs. problemas del mundo real.
- Ejemplos de problemas:
  - El 8-puzzle (juguete)
  - Las 8-reinas (juguete)
  - Problemas del viajante de comercio; distribución VLSI; navegación robot; secuencia ensamblaje automático; búsqueda en Internet

Búsqueda de soluciones: Árbol de búsquedas

- Árbol de búsquedas generado por:
  estado inicial + función sucesor → definen espacio de estados
- Si existe más de un camino para llegar a un estado: GRAFO
- Raíz del árbol ≡ nodo de búsqueda (estado inicial)
- nodo ≠ estado
  - Estado: una configuración del mundo
  - Nodo: estructura de datos del árbol
    - Estado actual correspondiente al nodo
    - Nodo padre
    - Acción
    - Costo del camino
    - Profundidad
- Frontera: conjunto de nodos no expandidos
- Estrategia: selección del nodo a expandir

Árbol de búsquedas

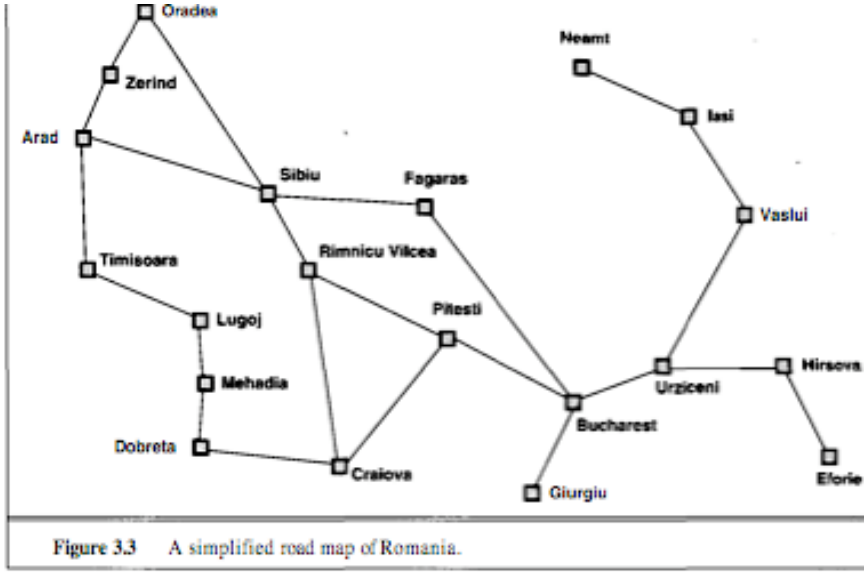
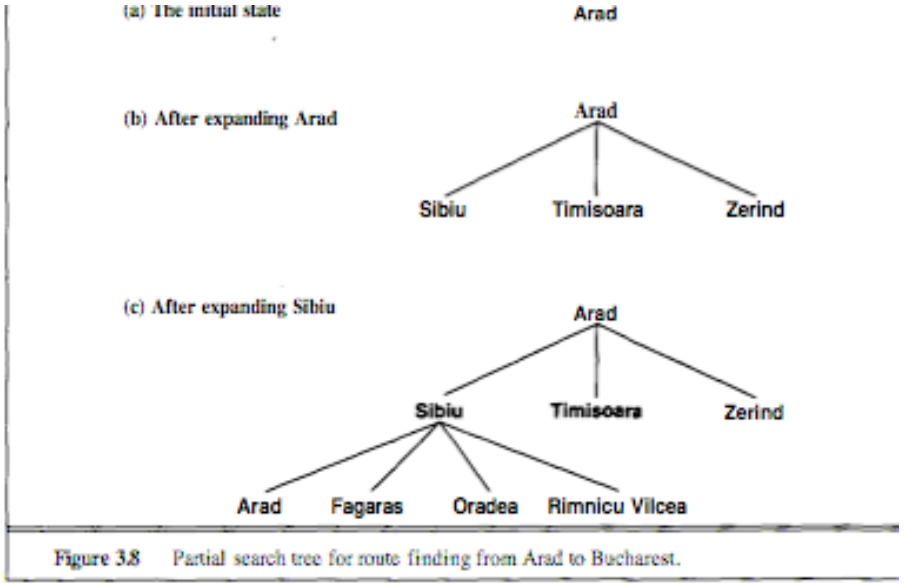


Figura 1: mapa de Rumania simplificado.

Figura 2: árbol de búsquedas.



Rendimiento de la resolución del problema

- Completitud: ¿el algoritmo encuentra la solución?
- Optimización: ¿la solución encontrada es óptima (coste menor)?
- Complejidad:
  - En tiempo: ¿cuánta “tarda”*?
  - En espacio: ¿cuánta memoria?
- En un árbol de búsqueda:
  - tiempo ≡ nº nodos
  - espacio ≡ nº nodos
- Así, la complejidad será función de:
  - b ≡ factor de ramificación del árbol (nº máximo de sucesores)
  - d ≡ profundidad del nodo objetivo
  - m ≡ longitud máxima de cualquier camino en el espacio de estados
- Costo de la búsqueda es función de la complejidad en tiempo.
- Coste total = costo de la búsqueda + costo del camino.

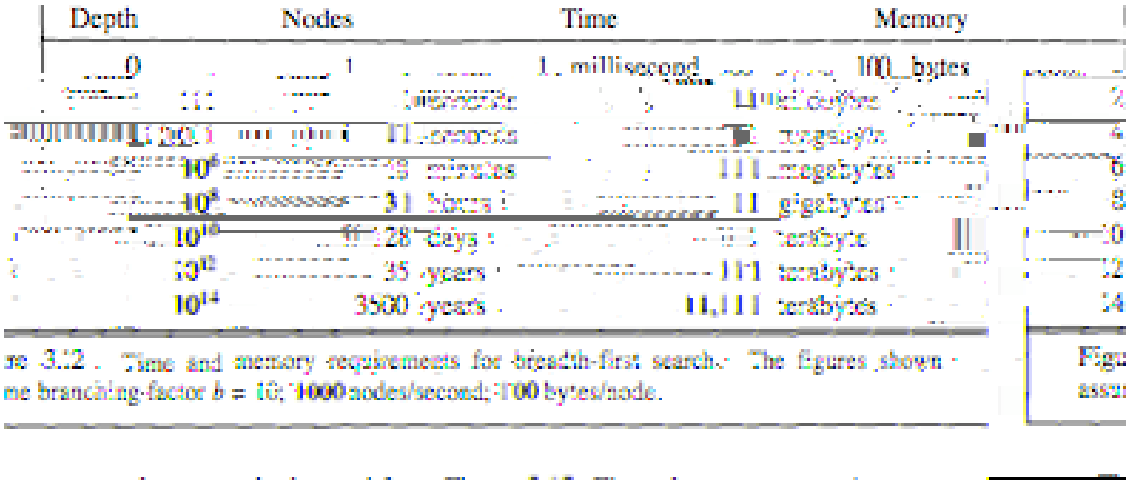
*”tarda”: parámetro n

Búsquedas no informadas

- “Búsqueda a ciegas”
- No disponen de información adicional acerca de los estados (no conocen lo prometedor que resulta un estado)
- Sólo distinguen estado objetivo del que no
- Son sistemáticas
  - No dejan a priori ningún nodo sin examinar
  - No exploran más de una vez el mismo nodo
- Técnica de búsqueda que emplea un árbol de búsqueda
- Se caracterizan por el orden de expansión de los nodos frontera

Búsqueda en anchura

- Características:
  - Es completa, si nodo solución se encuentra a profundidad d y b es finito.
  - Complejidad en tiempo: $O(b^{d+1})$
  - Complejidad en espacio = tiempo, pues los nodos explorados pertenecen a la solución.
- Conclusión (ver gráfico):
  -
  -



re 3.12. Time and memory requirements for breadth-first search. The figures shown ne branching factor $b = 10$; 1000 nodes/second; 100 bytes/node.

7

BÚSQUEDA EN ANCHURA: ALGORITMO

- FRONTERA: cola FIFO (1º en entrar 1º en salir)
- ABIERTOS (frontera) contiene los nodos no expandidos (implementado como Cola)
- CERRADOS contiene los nodos ya expandidos

1. Nodo inicial en ABIERTOS
2. Crear la lista CERRADOS inicialmente vacía
3. (3) Si ABIERTOS está vacía, no existe solución. TERMINAR
4. Suprimir el primer nodo n de ABIERTOS y colocarlo en CERRADOS
5. Si n es el objetivo → Solución
6. Expandir el nodo n, si no tiene sucesores ir a (3).
7. Colocar los sucesores de n al final de ABIERTOS
8. Ir a (3)

BÚSQUEDA DE COSTE UNIFORME

- Se tienen costes sobre los arcos. Se desea encontrar el camino de coste mínimo
- es la suma de los costes hasta el nodo en el que nos encontramos
- Es una generalización del anterior: si costo es cte genera búsqueda anchura
- ABIERTOS puede implementarse como una cola con prioridad (FIFO)

1. Nodo inicial, s, en ABIERTOS. Si s no es objetivo g(s) = 0
2. Crear la lista CERRADOS inicialmente vacía
3. (3) Si ABIERTOS está vacía, no existe solución. TERMINAR
4. Suprimir de ABIERTOS el nodo i tal que g(i) sea mínima y colocarlo en CERRADOS
5. Si i es el objetivo → Solución
6. Expandir el nodo i, si no tiene sucesores ir a 3.
7. Para cada nodo sucesor del nodo i, j, calcular y colocar los sucesores en ABIERTOS
8. Ir a 3

BÚSQUEDA EN PROFUNDIDAD

Descripción

- Se expanden los sucesores del último nodo generado (el nodo más profundo en la frontera actual del árbol de búsqueda).
- Los nodos explorados que no tienen descendientes en la frontera se quitan de la memoria:
  - se almacena sólo un camino del nodo raíz al nodo actual
  - más los nodos de la frontera (no explorados)
- Elección equivocada supone camino muy largo o infinito.
- ¿Y si el árbol de búsquedas es ilimitado? Solución: añadir límite de profundidad (l): búsqueda de profundidad limitada. Evitamos caminos infinitos.
- Problema si l Tipos de fracaso:
  - Valor de fracaso estándar ⇔ no existe solución
  - Valor de corte ⇔ no existe solución dentro del límite de profundidad

BÚSQUEDA EN PROFUNDIDAD

- Características:
  - Requisitos modestos de memoria: nodos (longitud máxima)
  - No es completa, si se elije camino equivocado:
    - = ∞ árbol ilimitado
    - puede ser mucho mayor que
  - No siempre encuentra la solución óptima.
  - En ocasiones se llega antes a la solución (a no ser que entre en una rama equivocada)
  - En el caso peor, complejidad en tiempo:
    - $O(b^m)$, se generan todos los nodos del árbol de búsqueda.
    - $O(b^l)$, si profundidad limitada.
- Diámetro del espacio de estados: elección de l

BÚSQUEDA EN PROFUNDIDAD: ALGORITMO

- ABIERTOS (frontera) puede implementarse como una pila (LIFO)

1. Nodo inicial, s, en ABIERTOS.
2. Crear la lista CERRADOS inicialmente vacía
3. (3) Si ABIERTOS está vacía, no existe solución. TERMINAR
4. Suprimir el primer nodo de ABIERTOS y colocarlo en CERRADOS
5. Si n es el objetivo → Solución
6. Si la profundidad del nodo es igual a la máxima profundidad ir a (3).
7. Expandir el nodo n, si no tiene sucesores ir a (3).
8. Colocar los sucesores de n al principio de la lista ABIERTOS
9. Ir a (3)

BÚSQUEDA BIDIRECCIONAL

- Descripción
  - Se realiza la expansión en ambas direcciones ⇒ debe ser posible calcular el sucesor y el predecesor de un nodo
  - Motivación: $b^{d/2} + b^{d/2} < b^d$
  - La búsqueda finaliza cuando encontramos un nodo común en las fronteras de los dos árboles de búsqueda.
  - s es el estado inicial y t el estado objetivo
  - S-ABIERTOS y S-CERRADOS son listas de nodos frontera y explorados desde el estado inicial
  - T-ABIERTOS y T-CERRADOS son listas de nodos frontera y explorados desde el estado objetivo
  - Costes
    - asociado al arco que une n con x:
    - para x, generado a partir del nodo inicial s, es el coste del camino más corto encontrado hasta ahora de s a x.
    - Para x, generado a partir del nodo objetivo t, es el coste del camino más corto encontrado hasta ahora de x a t.
- Ventajas
  - Coste en espacio y en tiempo: $O(b^{d/2})$
  - Es completo y óptimo (para costos uniformes) si la búsqueda es en anchura.

BÚSQUEDA BIDIRECCIONAL: ALGORITMO

1. Colocar s en S-CERRADOS, con gs(s)=0. Expandir s. Para cada sucesor x, poner x en S-ABIERTOS y hacer gs(x) =c(s,x)
   Idem para t en T-CERRADOS
2. Decidir si se va hacia delante (paso 3) o hacia atrás (paso 4)
3. Seleccionar de S-ABIERTOS el nodo n gs(n) mínimo. Mover n a S-CERRADOS. Si n está en T-ABIERTOS ir a paso 5. Si no para cada sucesor x de n hacer:
   a. Si x no está ni en S-ABIERTOS ni en S-CERRADOS añadirlo a S-ABIERTOS, añadir puntero al padre y calcular gs(x)
   b. Si x estaba en S-ABIERTOS, comparar el coste del camino anterior con el costo del nuevo, guardar el camino más corto
   c. Si x estaba en S-CERRADOS no hacer nada
   d. Volver al paso 2
4. Seleccionar de T-ABIERTOS el nodo n gt(n) mínimo. Mover n a T-CERRADOS. Si n está en S-ABIERTOS ir a paso 5. Si no para cada predecesor x de n hacer:
   a. Si x no está ni en T-ABIERTOS ni en T-CERRADOS añadirlo a T-ABIERTOS, añadir puntero al padre y calcular gt(x)
   b. Si x estaba en T-ABIERTOS, comparar el coste del camino anterior con el costo del nuevo, guardar el camino más corto
   c. Si x estaba en T-CERRADOS no hacer nada
   d. Volver al paso 2
5. Considerar el conjunto de nodos de la intersección de S-CERRADOS y la unión de T-CERRADOS y T-ABIERTOS, seleccionar el nodo n de coste mínimo para construir una solución que contenga este estado.

Heurística

- Puede usarse para:
  - Decidir qué nodo expandir a continuación
  - Decidir qué sucesores se generan
  - Decidir no tener en cuenta nunca determinados nodos
- Es una medida (a menudo tomada de la experiencia) de lo prometedor que resulta un nodo (¿cómo de próximo está el estado que encierra respecto al estado objetivo?)
- Se determina por una función de valoración heurística
- El proceso de elegir el nodo más prometedor puede tener una visión global o local

Ascensión de colinas

- Problemas donde no importa el camino, sino maximizar una función objetivo (ej: ocho reinas): problemas de optimización.
- Busca un estado solución.
- Búsqueda local: trabajan con el estado actual (no con caminos); se mueven a los vecinos.
- En cada momento se toma el nodo que maximiza la función objetivo.
- Ventajas
  - Usan poca memoria.
  - Pueden encontrar soluciones razonables en espacios de estados grandes o infinitos donde no son adecuados los algoritmos sistemáticos.
- Desventajas
  - No siempre obtiene el mejor resultado (no óptimo).
  - No puede salir de los máximos locales. Para evitar este problema es necesario implementar un modo de volver a situaciones anteriores.
  - Problema semejante son las mesetas.
- Se puede combinar con otros métodos para acercarse primero a la solución.
- Otros algoritmos: temple (recocido) simulado, algoritmos genéticos, VNS.

Ascensión de colinas

- Algoritmo
  1. Empezar en estado inicial. Si solución → TERMINAR
  2. Expandir el nodo
  3. Para cada sucesor
     a. Comprobar si es solución, si lo es → TERMINAR
     b. Si no lo es, comprobar si está más cerca de la solución que los testeados hasta el momento. Si está recordar dicho nodo, si no olvidarlo.
  4. Tomar el mejor elemento del paso anterior y usarlo como siguiente nodo a expandir
  5. Regresar al paso 2

Búsqueda por el mejor nodo

- Problemas donde la solución es el camino desde un estado inicial al objetivo.
- En cada iteración se escoge el nodo más prometedor según una **función evaluación *f(n)*** y se expande.
- Existe una **función heurística *h(n)*** = costo estimado del camino más barato desde el nodo n a un nodo objetivo (ej: línea recta entre ciudades)
- transmite el conocimiento del problema al algoritmo de búsqueda.
- Se establece =
- Problemas:
  - No es óptimo.
  - No es completo (Ej: figura 1 p. 4 Camino Isai a Fagaras: búsqueda atrapada en Neamt)

Búsqueda por el mejor nodo

- Algoritmo

1. Nodo inicial, s, en ABIERTOS.
2. Crear la lista CERRADOS inicialmente vacía
3. Si ABIERTOS está vacía, no existe solución. TERMINAR
4. Suprimir en ABIERTOS un nodo i con f mínimo y colocarlo en CERRADOS
5. Si i es el objetivo → Solución (devolver el camino por medio de los punteros de i a s). TERMINAR
6. Expandir el nodo i, generando todos los sucesores
7. Para cada sucesor j de i
   a. Calcular f(j)
   b. Si j no estaba en ABIERTOS ni en CERRADOS, añadirlo a ABIERTOS y establecer un puntero de j a i.
   c. Si j ya estaba en ABIERTOS decidir si j apunta a i o a su antiguo predecesor en función del valor de f(j)
   d. Si j ya estaba en CERRADOS, decidir si j apunta a i o a su antiguo predecesor como en el caso anterior. Además propagar la mejora a los suceso

Algoritmos A y A*

- Intentan mejorar el anterior dando garantía de que la solución es óptima.
- Descubre un camino mínimo entre el estado inicial y el final, explorando un grafo de espacio mínimo de estados
- Es de tipo mejor nodo
- La función evaluación tiene dos componeno

Algoritmos A y A*

1. ABIERTOS contiene sólo el nodo inicial, poner los valores de g para ese nodo a 0 y su valor f a h+0. Inicializar CERRADOS vacío.
2. Repetir hasta que se encuentre el nodo meta.
    1. Si no hay nodos en ABIERTOS → FALLO
    2. Tomar el nodo de ABIERTOS con mejor f' (MEJORNODO). Eliminarlo de ABIERTOS y ponerlo en CERRADOS.
    3. Mirar si MEJORNODO es objetivo → Solución y TERMINAR. Si no generar sucesores de MEJORNODO, para cada sucesor:
        1. Poner SUCESOR apuntando a MEJORNODO
        2. Calcular g(SUCESOR) = g(MEJOR NODO) + coste (MEJORNODOS, SUCESOR)
        3. Mirar si SUCESOR está en ABIERTOS y actualizar f si es necesario (1)
        4. Si no estaba en ABIERTOS ver si estaba en CERRADOS (2)
        5. Si SUCESOR no estaba ni en ABIERTOS ni en CERRADOS, ponerlo en ABIERTOS y añadirlo a la lista de sucesores de MEJORNODO. Calcular f(SUCESOR) = g (SUCESOR) + h (SUCESOR)

Algoritmos A y A*

(1). Mirar si SUCESOR está en ABIERTOS y actualizar f si es necesario.

1. En este caso el nodo ha sido generado pero no procesado
2. Llamamos al nod58 1274.4931599 T83- 0 0 58 1515 Tm( ) Tj.4972a0 0 58 227j 58 0

Algoritmos A y A*

- Propiedades de A
  - Si todo nodo tiene un número finito de descendientes es completo.
    - Es decir, garantiza alcanzar una solución.
    - La componente de costo de la heurística es la que lleva a la solución (no hay posibilidad de profundizar indefinidamente en una rama)
  - Es no admisible:
    - Admisible significa que encuentra la solución óptima
    - Sería admisible si la heurística no sobreestima el coste real, es decir, la heurística es optimista
- Cuando es admisible, el algoritmo de denomina A*

Bibliografía

- Stuart Russell, PeterNorving. Inteligencia Artificial. Un enfoque moderno. PrenticeHall. 2004