UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Cuaderno de trabajo:

Recursive Best First Search (RBFS)[1]

Albert Sanchis

DSIC

Departamento de Sistemas Informáticos y Computación

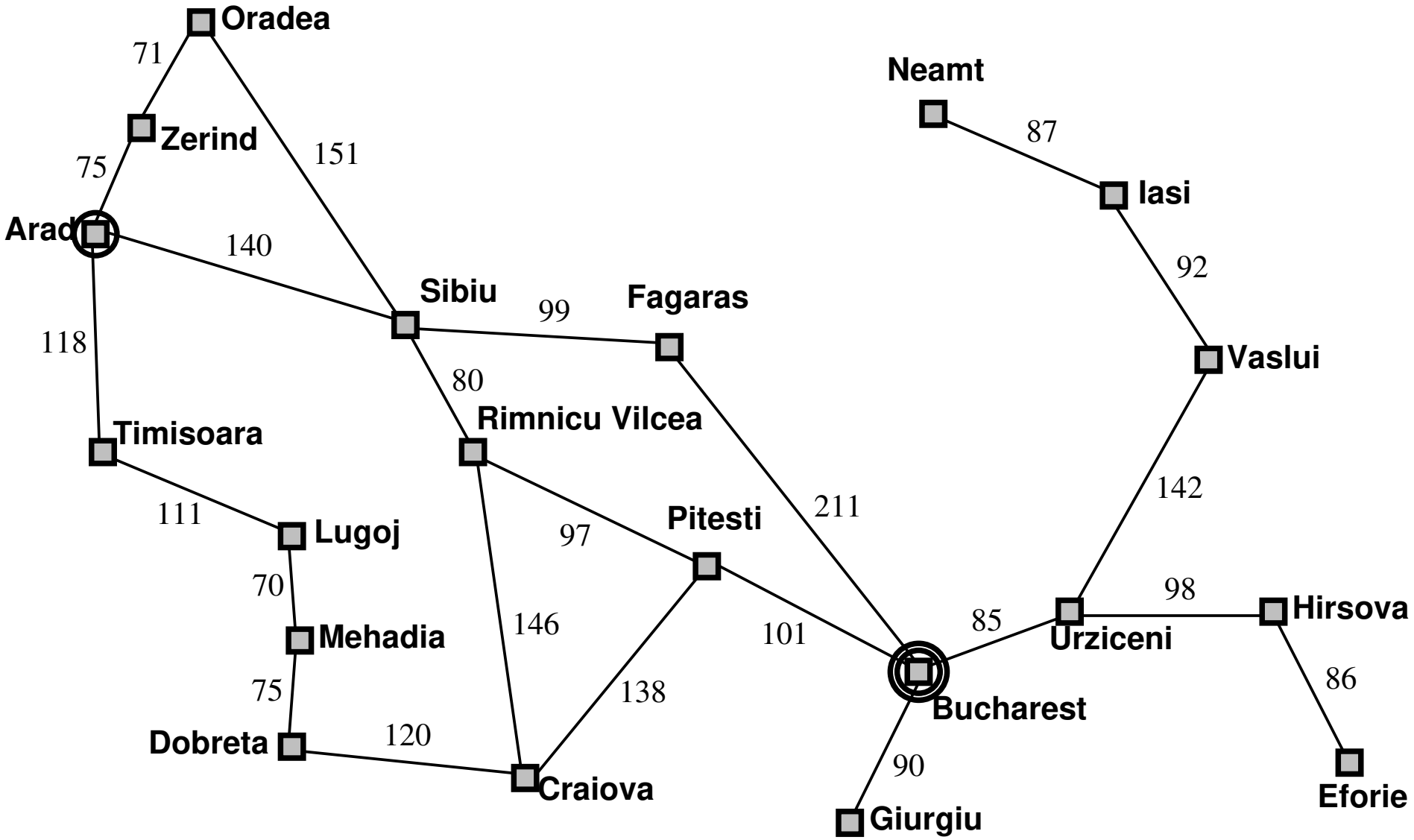
[1] Para una correcta visualización, se requiere Acrobat Reader v. 7.0 o superior

Objetivos formativos

- Caracterizar la búsqueda convencional en un grafo de estados.
- Describir búsqueda *Recursive Best First Search* (*RBFS*).
- Construir el árbol de búsqueda *RBFS*.
- Aplicar búsqueda *RBFS* a un problema clásico.
- Analizar la calidad de búsqueda *RBFS*.

Problema: La ruta más corta entre dos puntos

Búsqueda de una ruta más corta desde Arad a Bucarest [1]:



Acciones(Arad) = {Ir(Sibiu), Ir(Timisoara), Ir(Zerind)}.

Problema: La ruta más corta entre dos puntos

Distancias en línea recta a Bucharest

	Bucharest		Bucharest
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

El algoritmo RBFS (main) [2]

RBFS(G, s', f**)** // G grafo ponderado, s' comienzo, f ***func. eval.***

$P = InitStack(s')$ // Inicializa ***Path*** con el nodo raíz

$b = \infty$ // Cota inicial

$F_{s'} = f_{s'}$ // El valor almacenado es inicializado al valor f

$(F_r, r) =$ **BT**$(G, P, F_{s'}, f, b)$ // Devuelve v. almacenado y est. obj.

if $r \neq$ NULL: **return** P // Si solución, devuelve ***Path*** al objetivo

El algoritmo RBFS (backtracking) [2]

BT(G, P, F_s, f, b) // G grafo, P ***Path***, Valor almacenado $F_{s'}$, f, b cota

$s = Top(P)$ // ***Path***: extraer cima de la pila

if $Goal(s)$: **return** (f_s, s) // Solución encontrada!

$O = InitQueue()$ // ***Open***: cola de prioridad para nodos hijo

for all $(s, n) \in Adjacents(G, s)$ **and** $n \notin P$: // Generando hijos n no en ***Path***

if $f_s < F_s : F_n = max(f_n, F_s)$ // Si s visitado, el hijo hereda el v. almacenado

else: $F_n = f_n$ // En otro caso, el v. almacenado es ***f***

$Push(O, n, F_n)$ // Hijos ordenados en cola de prioridad por v. almacenado

if $EmptyQueue(O)$: **return** (∞, NULL) // No hijos, cota = ∞

while True:

$(n, F_n) = Top(O)$ // Mejor hijo en función del valor almacenado F

if $F_n > b$: **return** (F_n, NULL) // Se excede la cota, backtracking

$(n', F_{n'}) = Top2(O)$ // 2-mejor F o si no existe, entonces $F_{n'} = \infty$

$Push(P, n)$ // Añadir hijo al ***Path*** explorado

$(F_n, r) =$ **BT**$(G, P, F_n, f, min(b, F_{n'}))$ // Recursión con posible nueva cota

if $r \neq NULL$: **return** (F_n, r) // Si solución, fin recursión sin actualización

$Update(O, n, F_n)$ // Actualizar nodo n en O

$Pop(P)$ // Descartar último hijo del ***Path***

- ***Cuestión 1***: Construye el árbol de búsqueda resultante de aplicar el algoritmo ***RBFS*** al problema de búsqueda de una ruta más corta desde Arad a Bucarest.

- ***Cuestión 2***: ¿El algoritmo encuentra solución? Si la respuesta es "Sí":
  - ¿Cuál ha sido la solución encontrada?
  - ¿Cuál es el coste de esta solución?
  - ¿Se trata de la solución óptima?

Referencias

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

[2] Richard E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1993.