UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Cuaderno de trabajo:

Iterative Deepening A* (IDA*)[1]

Albert Sanchis

DSIC

Departamento de Sistemas Informáticos y Computación

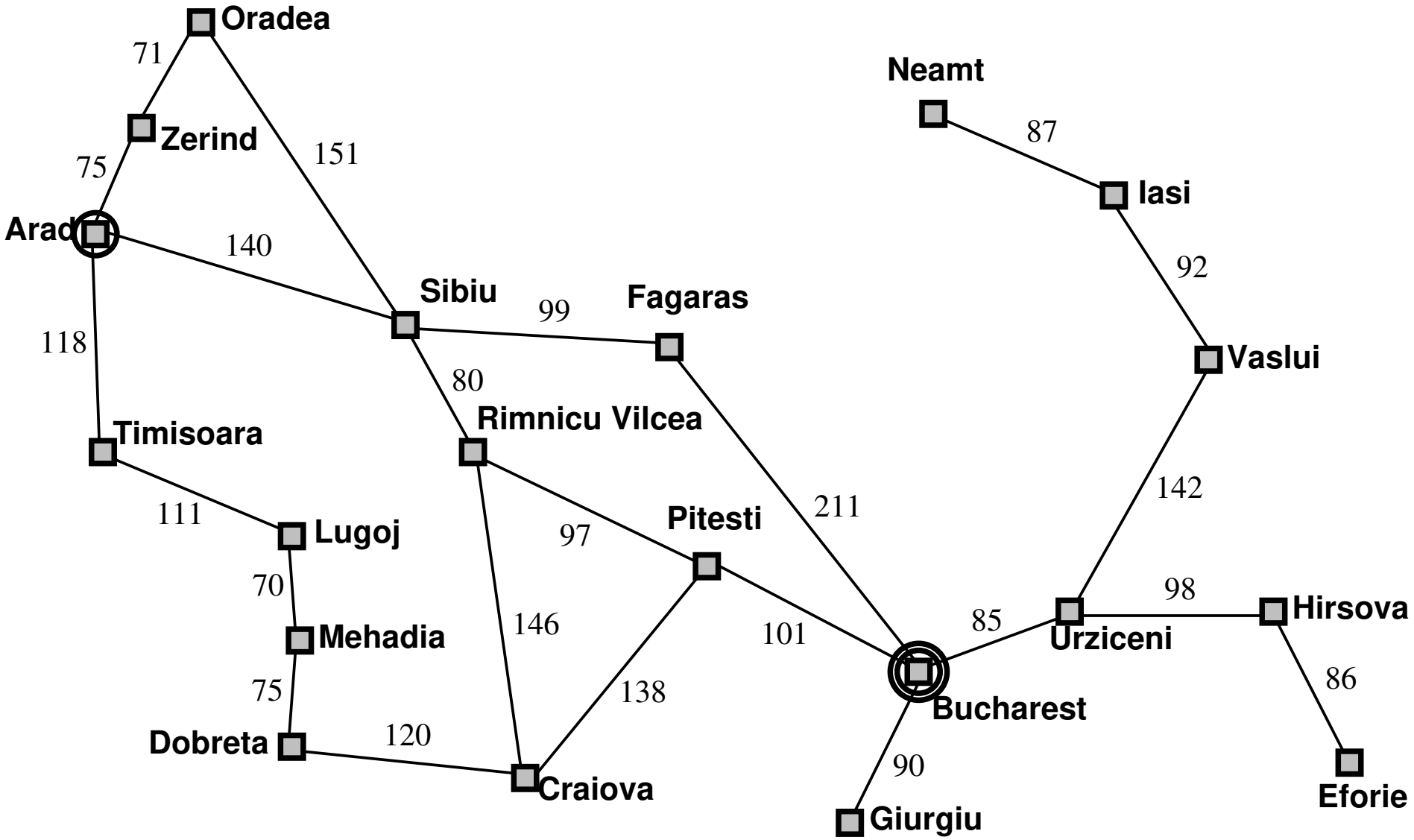
[1] Para una correcta visualización, se requiere Acrobat Reader v. 7.0 o superior

Objetivos formativos

- Caracterizar la búsqueda convencional en un grafo de estados.
- Describir búsqueda IDA*.
- Construir el árbol de búsqueda IDA*.
- Aplicar búsqueda IDA* a un problema clásico.
- Analizar la calidad de búsqueda IDA*.

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 IDA* (main) [2]

IDA(G, s', h**)** // G grafo ponderado, s' comienzo, h ***heuristica***

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

$b = h(s')$ // Inicializa la cota con $f_{s'} = h(s')$

while True:

$(nextb, r) =$ **BT**(G, P, h, b) // *nextb* cota siguiente; *r* estado obj.

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

if $nextb = \infty$: **return** NULL // no hijos para calcular la sig. cota

$b = nextb$ // actualización de la cota para la iteración siguiente

El algoritmo IDA* (backtracking) [2]

```
BT(G, P, h, b)                          // G grafo ponderado, Path P, h, cota b
  s = Top(P)                            // Path: extrae cima de la pila
  f_s = g_s + h(s)                      // f valor del nodo a explorar
  if f_s > b: return (f_s,NULL)         // b excedida fin para calcular nextb
  if Goal(s): return (f_s, s)           // solución encontrada!
  min = ∞                               // mínimo valor de un hijo f
  n = FirstAdjacent(G, s)               // generación: n primer hijo de s
  while n ≠ NULL:                       // hijo izquierdo y no en Path
    if n ∉ P:                           // n no en Path para evitar ciclos
      Push(P, n)                        // añadir hijo al Path explorado
      (nextb, r) = BT(G, P, h, b)       // hijo devuelve mín. f y estado sol.
      if r ≠ NULL: return (nextb, r)    // si r solución, fin recursión
      if nextb < min: min = nextb       // actualiza valor mín. f
      Pop(P)                            // Descarta último hijo de Path
    n = NextAdjacent(G, s, n)           // generación: n siguiente hijo de s
  return (min, NULL)                    // sol. no encontrada, devuelve mínimo f
```

- ***Cuestión 1***: Construye el árbol de búsqueda resultante de aplicar el algoritmo ***IDA**** 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] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.