

---

Inteligencia Artificial

# Búsqueda Informada y Exploración

---

Jorge Luis Guevara Diaz

[www.jorge.sistemasyservidores.com](http://www.jorge.sistemasyservidores.com)

---

# Que veremos?

- **Busqueda Primero el mejor**
    - Busqueda Greedy primero el mejor
    - Busqueda A\*
  - **Búsqueda Heurística con memoria limitada**
    - IDA\*
    - Busqueda recursiva primero el mejor RBFS
    - SMA\*
  - **Heurísticas**
  - **Búsqueda Local y Optimización**
  - **Busqueda Local en espacios Continuos**
-

---

# Que veremos?

- **Busqueda Primero el mejor**
    - Busqueda Greedy primero el mejor
    - Busqueda A\*
  - **Búsqueda Heurística con memoria limitada**
    - IDA\*
    - Busqueda recursiva primero el mejor RBFS
    - SMA\*
  - **Heurísticas**
  - **Búsqueda Local y Optimización**
  - **Busqueda Local en espacios Continuos**
-

---

# Que veremos?

- **Busqueda Primero el mejor**
    - Busqueda Greedy primero el mejor
    - Busqueda A\*
  - **Búsqueda Heurística con memoria limitada**
    - IDA\*
    - Busqueda recursiva primero el mejor RBFS
    - SMA\*
  - **Heurísticas**
    - Admisibilidad
    - Dominación
    - Relajando Problemas
    - Inventando heurísticas
-

---

# Búsqueda Informada y Exploración

## ■ Introducción

- ❑ Información = Usar conocimiento específico del problema
  - ❑ Estos algoritmos pueden solucionar los problemas mas eficientemente
  - ❑ Son llamados también estrategias de búsqueda heurística
-

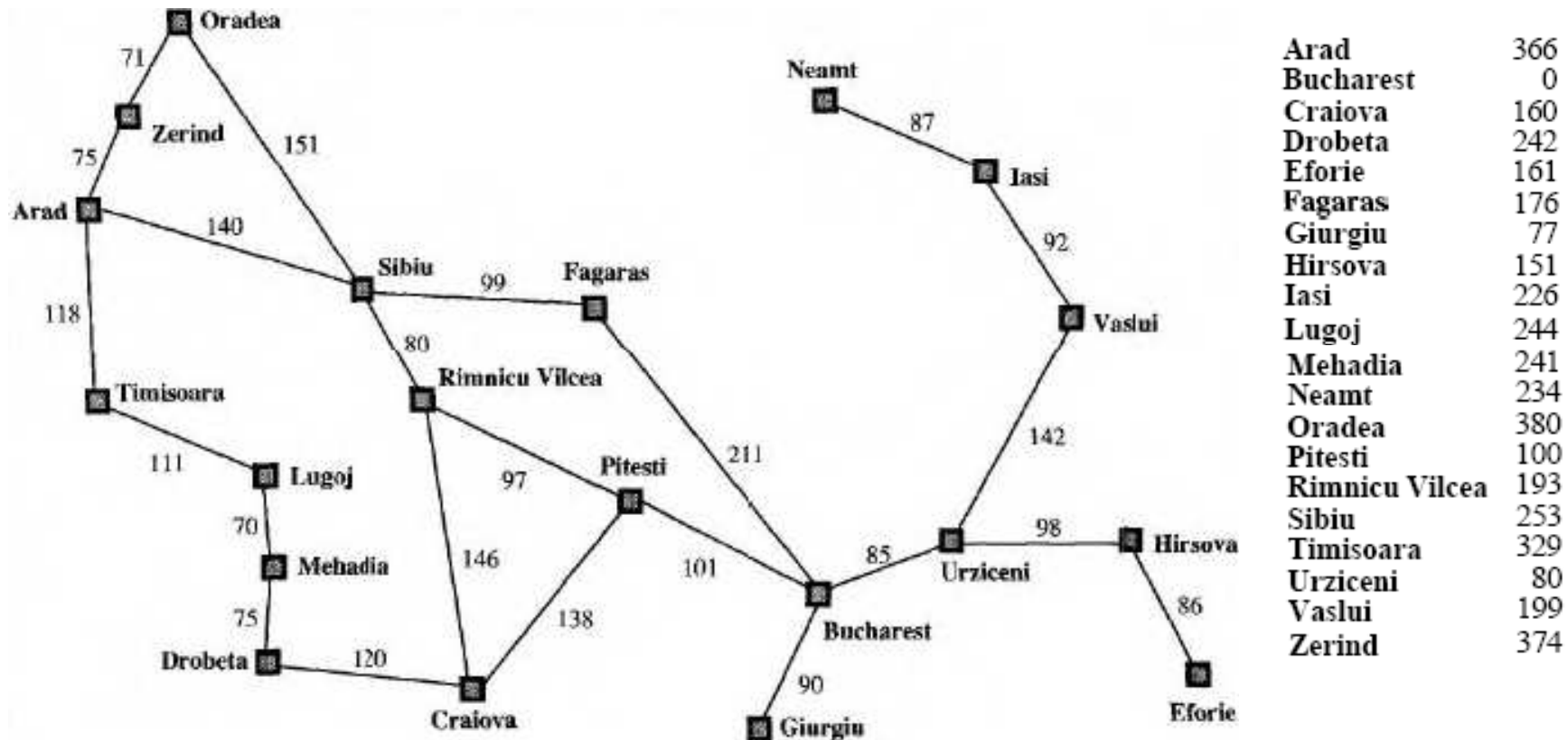
---

# Búsqueda primero el mejor

- **Idea:** usar una **función de evaluación  $f(n)$**  para cada nodo
    - Estimación de “deseable”
    - Expandir el nodo mas deseable que no ha sido expandido todavia
  - **implementacion:**

fringe: Cola ordenada en forma decreciente de “deseabilidad”  
Ordenar los nodos no expandidos en orden decreciente de deseabilidad
  - **Casos especiales:**
    - Búsqueda Greedy primero el mejor
    - Búsqueda A\*
-

# Rumania con distancia de linea recta a Bucarest en km



---

# Busqueda Greedy Primero el Mejor

## ■ Función de evaluación

$$f(n) = h(n) \text{ (heurística)}$$

- $h(n)$  = costo estimado del camino más barato del nodo  $n$  al nodo meta.
- si  $n$  es la meta entonces  $h(n)=0$
- Ejemplo.,  $h_{SLD}(n)$  = distancia de línea recta de  $n$  hacia Bucarest

## ■ Busqueda Greedy Primero el Mejor

Expande el nodo que **parece** ser el mas cercano a la meta

---

---

# Busqueda Greedy Primero el Mejor: Ejemplo



Asumamos que deseamos usar la búsqueda greedy para solucionar el problema de viajar de Arad a Bucharest.

Estado inicial=Arad

---

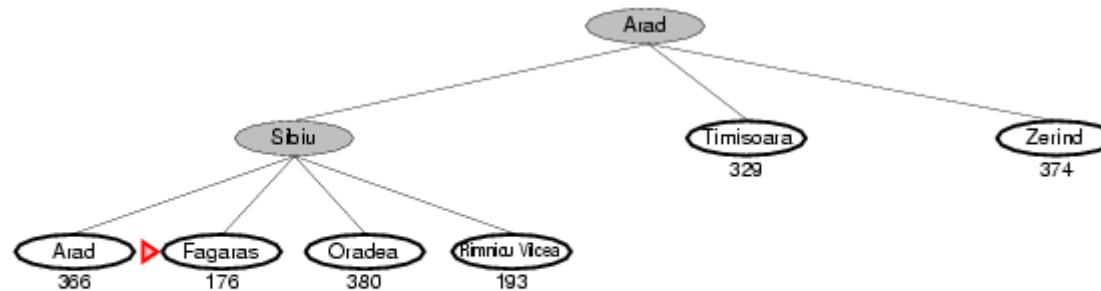
---

# Busqueda Greedy Primero el Mejor: Ejemplo



- Los primeros pasos de expansión producen:
    - Sibiu, Timisoara y Zerind
  - Busqueda Greedy primero el mejor seleccionará Sibiu.
-

# Busqueda Greedy Primero el Mejor: Ejemplo

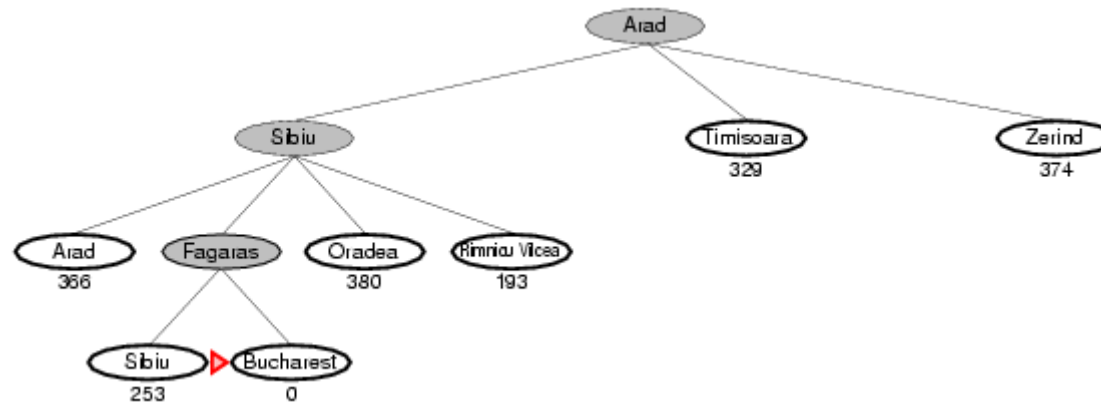


Si Sibiu es expandido obtendremos:

–Arad, Fagaras, Oradea y Rimnicu Vilcea

Busqueda Greedy primero el mejor seleccionará:  
Fagaras

# Busqueda Greedy Primero el Mejor: Ejemplo



Si Fagaras es expandido obtendremos:

–Sibiu y Bucharest

Meta alcanzada !!

–Pero no es óptima (ver Arad, Sibiu, Rimnicu Vilcea,  
Pitesti)

---

# Propiedades de la Búsqueda Greedy primero el mejor

- Completo? No – se puede caer en infinitos loops, ejemplo., lasi → Neamt → lasi → Neamt → ,  
Es completo en espacios finitos, evitando estados repetidos
  - Tiempo?  $O(b^m)$ , m es la máxima profundidad del espacio de búsqueda, una buena heurística puede dar una mejora considerable
  - Espacio?  $O(b^m)$  – guarda todos los nodos en memoria
  - Optimo? No
-

---

# Busqueda $A^*$

Idea: evitar expandir caminos que son caros

Función de evaluación  $f(n) = g(n) + h(n)$

$g(n)$  = costo del camino del nodo inicial al nodo  $n$

$h(n)$  = costo estimado del nodo  $n$  a la meta

$f(n)$  = costo estimado de la atravez de  $n$  a la meta

---

---

# Busqueda $A^*$

- Busqueda  $A^*$  usa una heurística **admisibile**
  - Una heurística es admisibile si ésta nunca sobreestima el costo para alcanzar la meta

Formalmente:

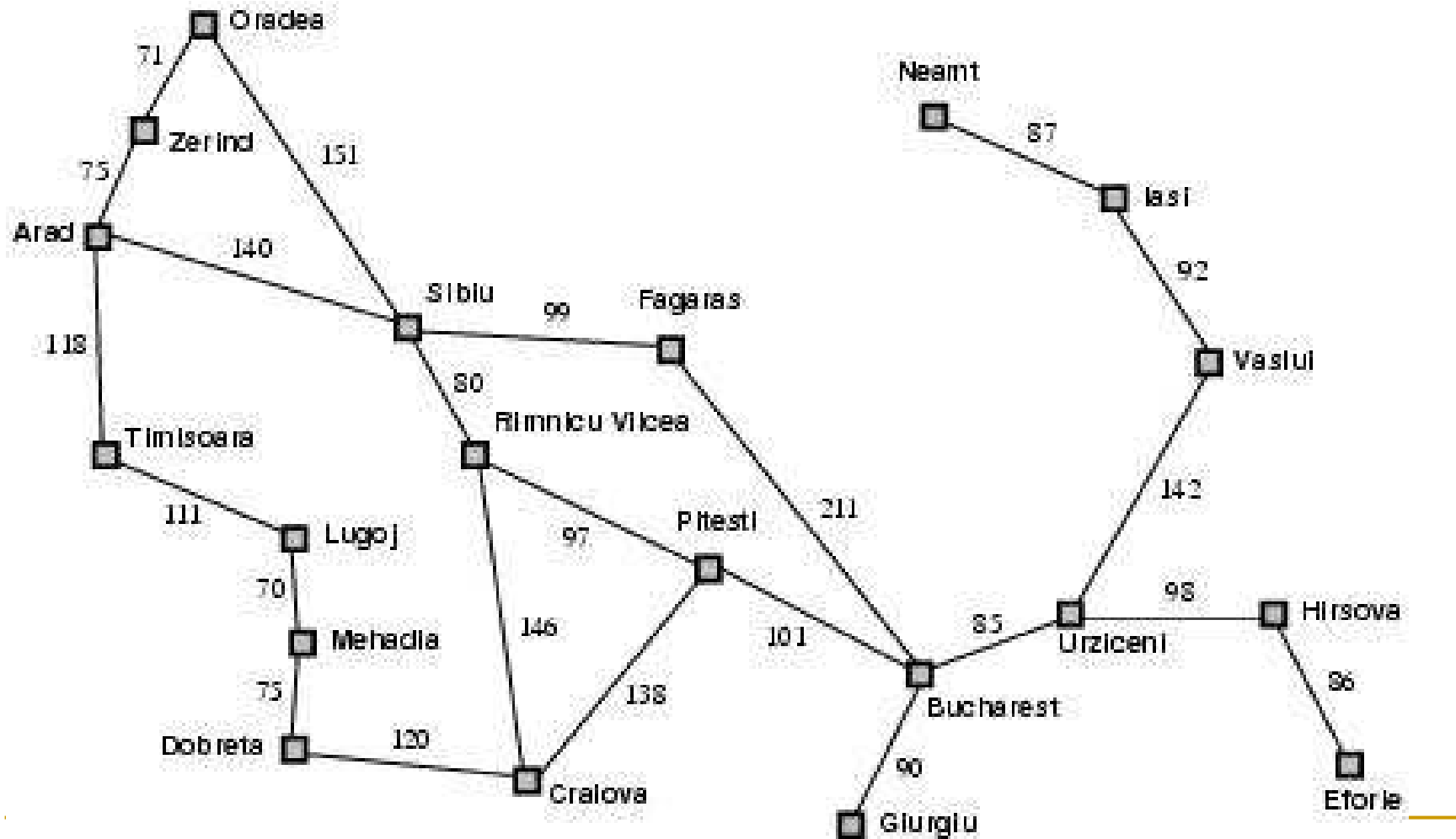
1.  $h(n) \leq h^*(n)$  donde  $h^*(n)$  es el costo **verdadero** de  $n$
2.  $h(n) \geq 0$  tal que  $h(G)=0$  para cualquier meta  $G$ .

Ejemplo.  $h_{SLD}(n)$  nunca sobreestima la actual distancia de Caminos

ˆ Busqueda  $A^*$  es óptima

---

# Busqueda A\* Ejemplo



---

# Busqueda A\* Ejemplo



- Buscar Bucarest empezando en Arad
  - $f(\text{Arad}) = c(??, \text{Arad}) + h(\text{Arad}) = 0 + 366 = 366$



# Busqueda A\* Ejemplo



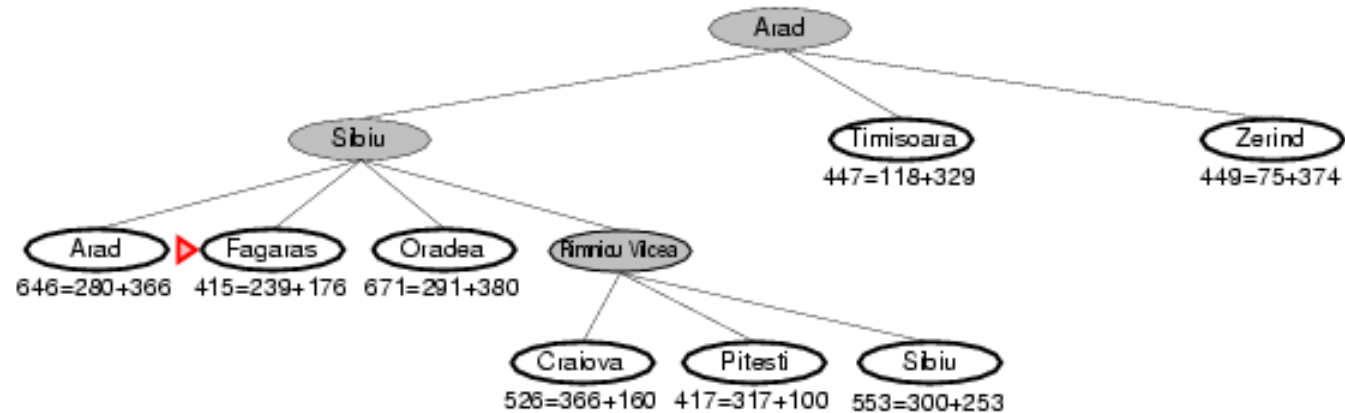
- Expandir Arad y determinar  $f(n)$  para cada nodo
  - $f(\text{Sibiu}) = c(\text{Arad}, \text{Sibiu}) + h(\text{Sibiu}) = 140 + 253 = 393$
  - $f(\text{Timisoara}) = c(\text{Arad}, \text{Timisoara}) + h(\text{Timisoara}) = 118 + 329 = 447$
  - $f(\text{Zerind}) = c(\text{Arad}, \text{Zerind}) + h(\text{Zerind}) = 75 + 374 = 449$
- Mejor opción Sibiu

# Busqueda A\* Ejemplo



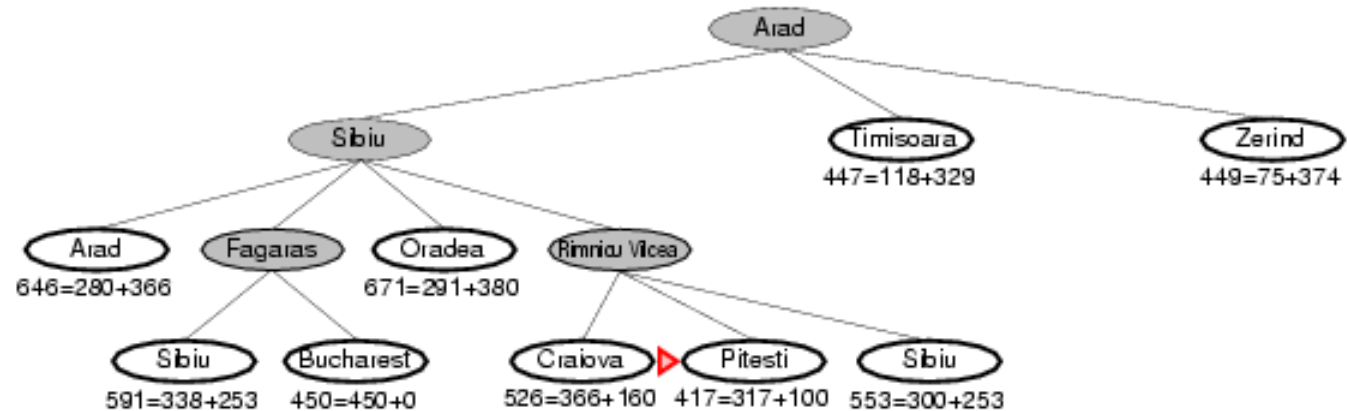
- **Expandir Sibiu y determinar  $f(n)$  para cada nodo**
  - ❑  $f(\text{Arad}) = c(\text{Sibiu}, \text{Arad}) + h(\text{Arad}) = 280 + 366 = 646$
  - ❑  $f(\text{Fagaras}) = c(\text{Sibiu}, \text{Fagaras}) + h(\text{Fagaras}) = 239 + 179 = 415$
  - ❑  $f(\text{Oradea}) = c(\text{Sibiu}, \text{Oradea}) + h(\text{Oradea}) = 291 + 380 = 671$
  - ❑  $f(\text{Rimnicu Vilcea}) = c(\text{Sibiu}, \text{Rimnicu Vilcea}) + h(\text{Rimnicu Vilcea}) = 220 + 192 = 413$
- **Mejor opción es Rimnicu Vilcea**

# Busqueda A\* Ejemplo



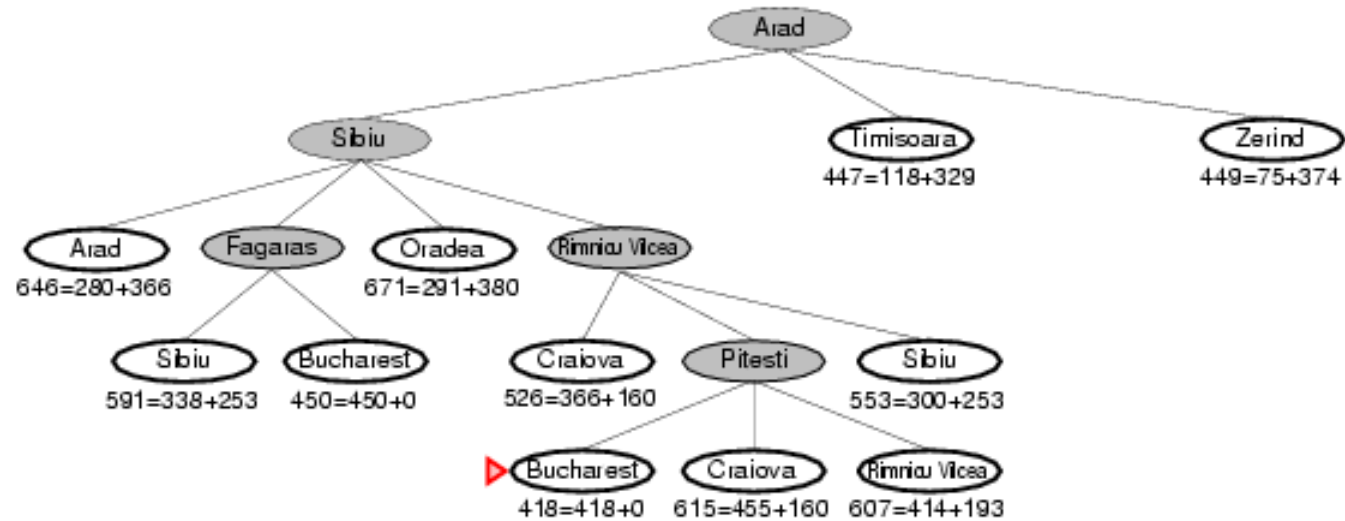
- Expandir Rimnicu Vilcea y determinar  $f(n)$  para cada nodo
  - $f(\text{Craiova}) = c(\text{Rimnicu Vilcea}, \text{Craiova}) + h(\text{Craiova}) = 360 + 160 = 526$
  - $f(\text{Pitesti}) = c(\text{Rimnicu Vilcea}, \text{Pitesti}) + h(\text{Pitesti}) = 317 + 100 = 417$
  - $f(\text{Sibiu}) = c(\text{Rimnicu Vilcea}, \text{Sibiu}) + h(\text{Sibiu}) = 300 + 253 = 553$
- Mejor opción es Fagaras

# Busqueda A\* Ejemplo



- **Expandir Fagaras y determinar  $f(n)$  para cada nodo**
  - $f(\text{Sibiu})=c(\text{Fagaras}, \text{Sibiu})+h(\text{Sibiu})=338+253=591$
  - $f(\text{Bucharest})=c(\text{Fagaras}, \text{Bucharest})+h(\text{Bucharest})=450+0=450$
- **Mejor opción es Pitesti !!!**

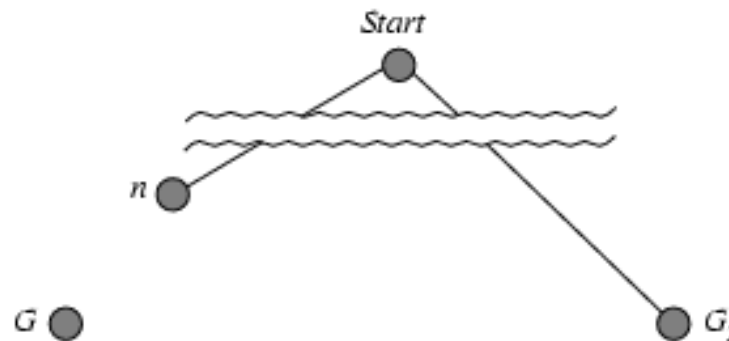
# Busqueda A\* Ejemplo



- Expandir Pitesti y determinar  $f(n)$  para cada nodo
  - $f(\text{Bucharest})=c(\text{Pitesti},\text{Bucharest})+h(\text{Bucharest})=418+0=418$
- Mejor opción es Bucharest !!!
  - Solución óptima (solo si  $h(n)$  es admisible)

# Optimalidad de $A^*$ (prueba estandar)

Suponer que alguna meta suboptima  $G_2$  ha sido generada y esta en la cola  
Sea  $n$  un nodo sin expandir en el camino mas corto a la meta óptima  $G$

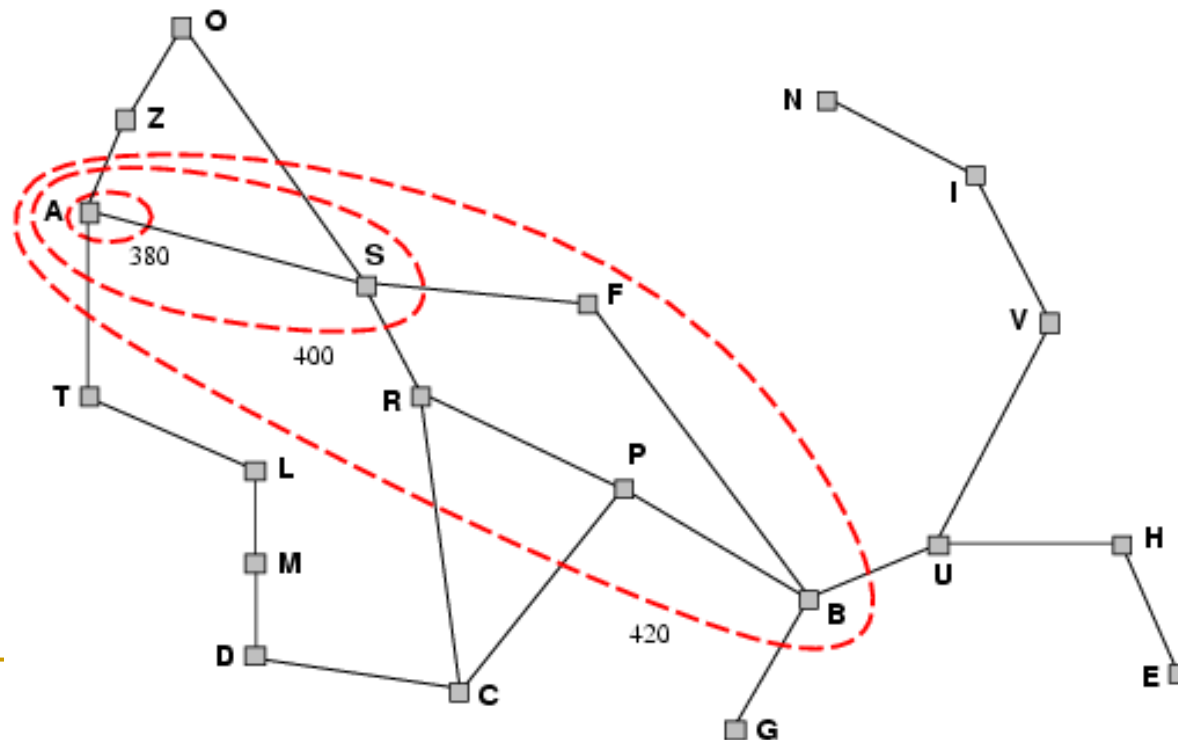


$$\begin{aligned} f(G_2) &= g(G_2) \text{ pues } h(G_2) = 0 \\ &> g(G_1) \text{ desde que } G_2 \text{ es suboptimo} \\ &\geq f(n) \text{ desde que } h \text{ es admisible} \end{aligned}$$

Como  $f(G_2) > f(n)$   $A^*$  nunca seleccionará  $G_2$  para expandir

# Optimalidad de $A^*$

- $A^*$  expande los nodos en orden creciente del valor de  $f$
- Gradualmente agrega " $f$ -contornos" de nodos
- Contorno  $i$  tiene todos los nodos con  $f=f_i$ , donde  $f_i < f_{i+1}$



---

# Propiedades de $A^*$

- Completo? Si (a menos que existan muchos nodos infinitos con  $f \leq f(G)$  )
  - Tiempo? Exponencial
  - Espacio? Guarda todos los nodos en memoria
  - Optimo? Si
-

# A\* prueba de consistencia

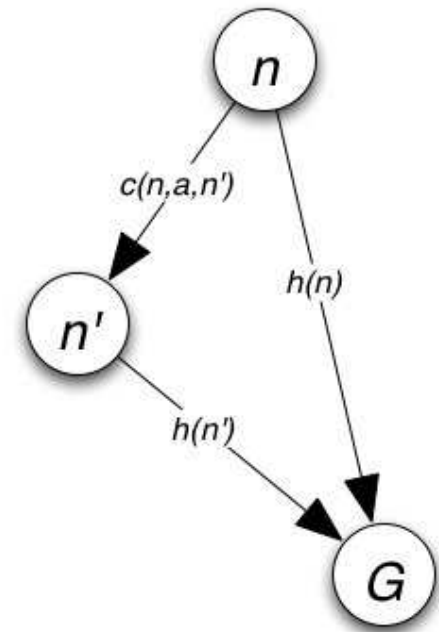
- Una heurística es consistente

$$h(n) \leq c(n, a, n') + h(n')$$

- Si  $h$  es consistente, entonces

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &\geq f(n) \end{aligned}$$

- $f(n)$  es no decreciente a través de cualquier camino



---

# Búsqueda heurística con memoria limitada

- Algunas soluciones al problema de espacio de  $A^*$  (matienen completitud y optimalidad)
    - Profundidad-iterativa  $A^*$  (IDA\*)
      - Información clave  $f=g+h$  en lugar de la profundidad
    - Búsqueda Recursiva Primero el Mejor (RBFS)
    - Memoria-limitada Simplificada  $A^*$  (SMA\*)
-

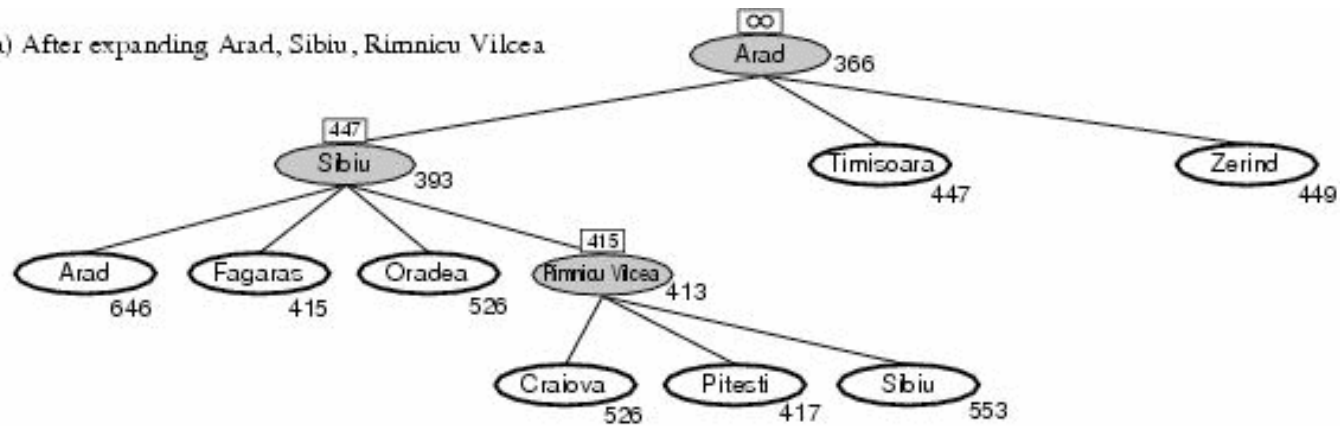
---

# Busqueda Recursiva Primero el Mejor RBFS

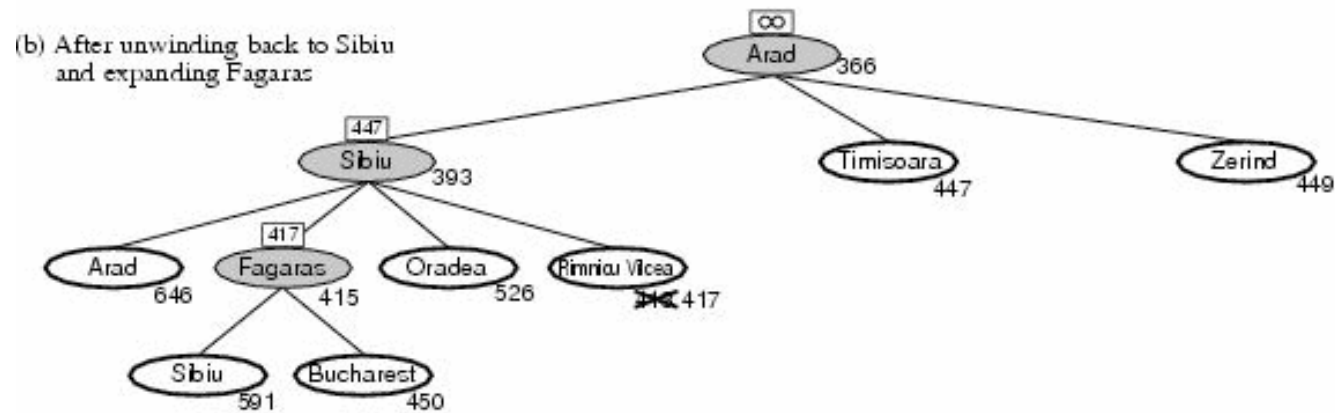
- Mantiene un registro del valor de  $f$  del mejor camino alternativo disponible
    - Si el nodo actual exede este límite entonces se hace vuelta atrás al valor  $f$  alternativo.
    - Después de la vuelta atrás el valor  $f$  del nodo es el mejor valor  $f$  de sus hijos.
    - Re-expansion de este resultado es todavia posible.
-

# Busqueda Recursiva Primero el Mejor RBFS ejemplo

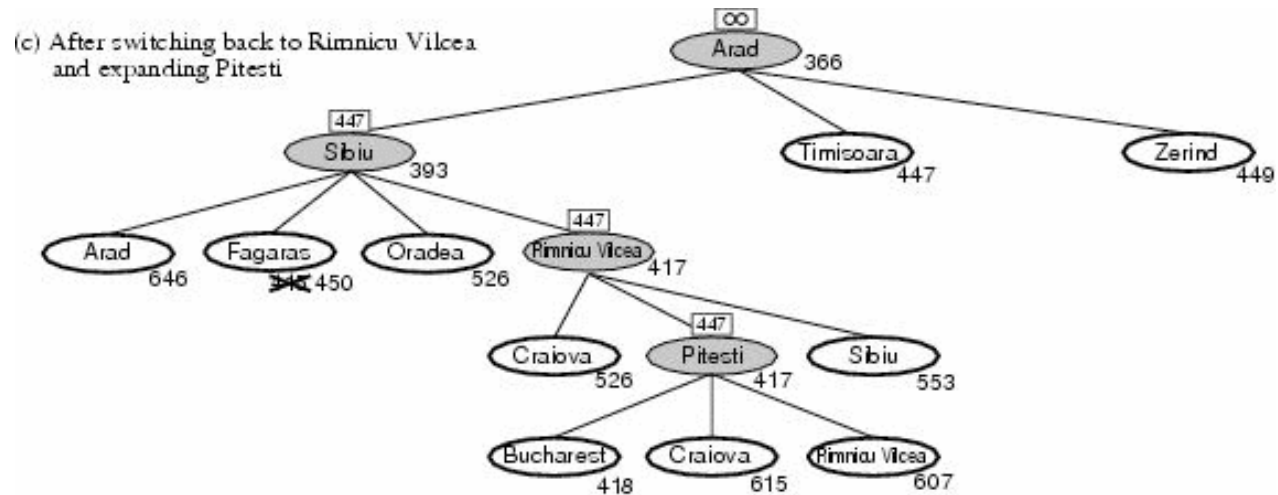
(a) After expanding Arad, Sibiu, Rimnicu Vilcea



# Busqueda Recursiva Primero el Mejor RBFS ejemplo



# Busqueda Recursiva Primero el Mejor RBFS ejemplo



---

# Busqueda Recursiva Primero el Mejor

## RBFS evaluación

RBFS es un poco mas eficiente que IDA\*

-Aunque realiza excesiva regeneracion de nodos

Igual que A\*, óptimo si  $h(n)$  es admisible

Complejidad de espacio es  $O(bd)$ .

-IDA\* retiene solamente un solo nodo en memoria (el valor actual del limite del f-costo)

Complejidad en el Tiempo difícil de caracterizar

-.

IDA\* y RBFS sufren de memoria ***muy pequeña.***

---

---

# Memoria-limitada Simplificada A\*

## (SMA\*)

- Usa toda la memoria disponible.
    - I.e. expande mejores hojas hasta que la memoria disponible este llena
    - Cuando está llena, SMA\* elimina el peor nodo hoja (mas alto  $f$ -valor)
    - Como RFBS hace vuelta atras hacia el nodo padre
  - Que pasa si todos los nodos hojas tienen el mismo  $f$ -valor?
    - El mismo nodo puede ser escogido tanto para expansión como borrado.
    - SMA\* soluciona esto expandiendo la mejor hoja *mas nueva* y borrando la peor hoja *mas antigua*.
  - SMA\* es completa si la solución es alcanzable, óptima si solución óptima es alcanzable
-

# Funciones Heurísticas

Ejemplo., para el 8-puzzle: □

- $h_1(n)$  = numero de piezas fuera de lugar
- $h_2(n)$  = distancia total de Manhattan

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1(S) = ?$
- $h_2(S) = ?$  □

# Funciones Heurísticas : Admisibilidad

Ejemplo., para el 8-puzzle: □

- $h_1(n)$  = numero de piezas fuera de lugar
- $h_2(n)$  = distancia total de Manhattan

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1(S) = ?$  8
- $h_2(S) = ?$   $3+1+2+2+2+3+3+2 = 18$
- Son admisibles ambas pues son menores que el costo real

---

# Funciones Heurísticas : Dominación

- $b^*$  = factor efectivo de ramificación
- $N$  = Número total de nodos generados por  $A^*$
- $d$  = profundidad de la solución

$$N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$$

Ejemplo:

si  $A^*$  encuentra una solución con  $d=5$  usando  $N = 52$ , entonces  $b^* = 1.92$

Conclusión:

Una heurística bien diseñada tiene valores de  $b^*$  que tienen valores cercanos a 1

---

# Funciones Heurísticas : Dominación

d	Costo de búsqueda			$b^*$		
	BPI	$A^*(h_1)$	$A^*(h_2)$	BPI	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	-----	539	113	.....	1.44	1.23
16	-----	18094	1219	.....	1.48	1.28
18	-----	39135	1641	.....	1.48	1.26

# Funciones Heurísticas : Dominación

Si  $h_2(n) \geq h_1(n)$  para todos los  $n$  (ambos admisibles)  
entonces  $h_2$  **domina**  $h_1$   $h_2$  es mejor para buscar

Costo típico de búsqueda (numero promedio de nodos expandidos):

- $d=12$

BPI = 3,644,035 nodos

$A^*(h_1) = 227$  nodos

$A^*(h_2) = 73$  nodos

- $d=24$

BPI = muchos nodos

$A^*(h_1) = 39,135$  nodos

$A^*(h_2) = 1,641$  nodos

Dadas cualquier heurística admisible  $h_a, h_b,$

---

$h(n) = \max(h_a(n), h_b(n))$  Es también admisible y domina a ambas

---

# Funciones Heurísticas : Relajando Problemas

- Un problema con pocas restricciones en las acciones es llamado un **problema relajado**
  - **Ejemplo:**
    - Si las reglas del 8-puzzle son relajadas de tal manera que una pieza pueda moverse **a cualquier lugar**, en lugar de cualquier lugar adyacente entonces  $h_1(n)$  nos da la solución mas corta
    - Si las reglas son relajadas de tal manera que la pieza pueda moverse **un cuadrado adyacente en cualquier direccion**, incluso havia un cuadrado ocupado, entonces  $h_2(n)$  da la solución mas corta
  - **Idea I:** El costo de la solución óptima de un problema relajado, no es más grande que el costo de la solución óptima del problema real
    - **Costo problema relajado < costo problema real**
  - **Idea II:** El costo de una solución óptima para un problema relajado es una **heurística admisible** para el problema original
-

---

# Funciones Heurísticas : Inventando Heurísticas

- ¿Será posible para una máquina inventar automáticamente heurísticas?

- La respuesta es sí, Ejemplo:

Una pieza puede moverse del cuadrado A hacia B si A esta horizontalmente o verticalmente adyacente a B y B está desocupado

- Removiendo condiciones se tienen los siguientes problemas relajados
    - Una pieza puede moverse de A hacia B si A es adyacente a B
    - Una pieza puede moverse de A hacia B si B esta desocupado
    - Una pieza puede moverse de A hacia B
-

---

# Referencias Bibliográficas

- Capitulo 4 secc. 1 y 2 Artificial Intelligence: A Modern Approach, Russell and Norvig
- <http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/Search>

