Algoritmi di ricerca informati

Capitolo 4, sezioni 1–2, 4

Sommario

- ♢ Ricerca best-first
- ♢ Ricerca A*
- ♢ Euristiche
- ♢ Hill-climbing
- ♢ Simulated annealing

Ripasso: ricerca generale

function GENERAL-SEARCH(*problem,* QUEUING-FN) **returns** a solution, or failure

nodes ← MAKE-QUEUE(MAKE-NODE(INITIAL-STATE[*problem*]))
loop do
  if *nodes* is empty **then return** failure
  node ← REMOVE-FRONT(*nodes*)
  if GOAL-TEST[*problem*] applied to STATE(*node*) succeeds **then return** *node*
  nodes ← QUEUING-FN(*nodes*, EXPAND(*node*, OPERATORS[*problem*]))
end

Una strategia è definita scegliendo *l'ordine di espansione dei nodi*

Ricerca Best-first

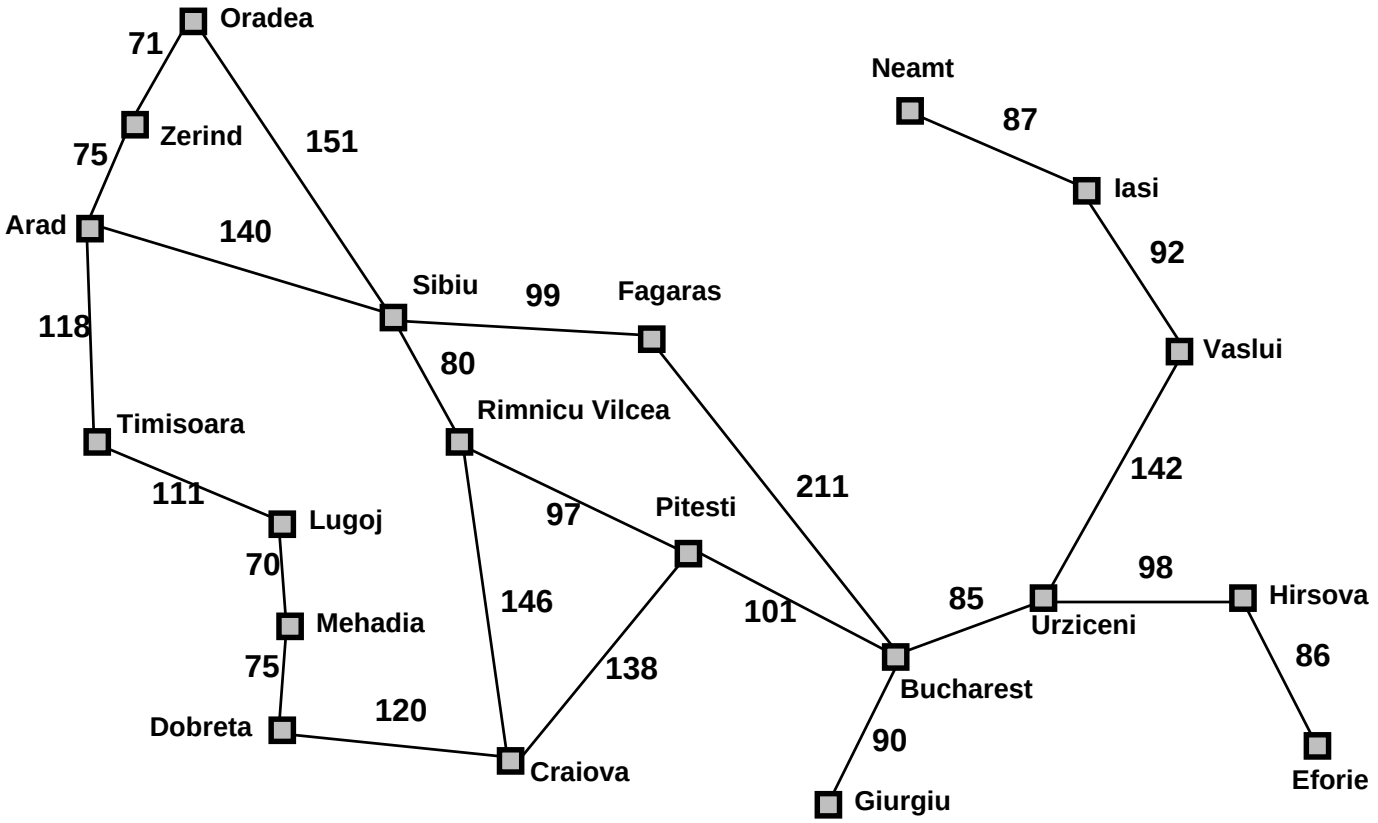
Idea: usare una *funzione di valutazione* per ogni nodo
– stima di "desiderabilità"

$\Rightarrow$ Espandere il nodo non espanso più desiderabile

Implementazione:
$\textsc{QueueingFn}$ = inserire i successori in ordine decrescente di desiderabilità

Casi speciali:
ricerca greedy
ricerca A*

Romania con costo dei passi in km



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

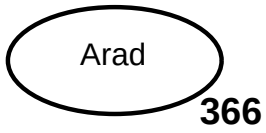
Ricerca greedy

Funzione di valutazione $h(n)$ (euristica)
$=$ stima del costo dal nodo n al $goal$

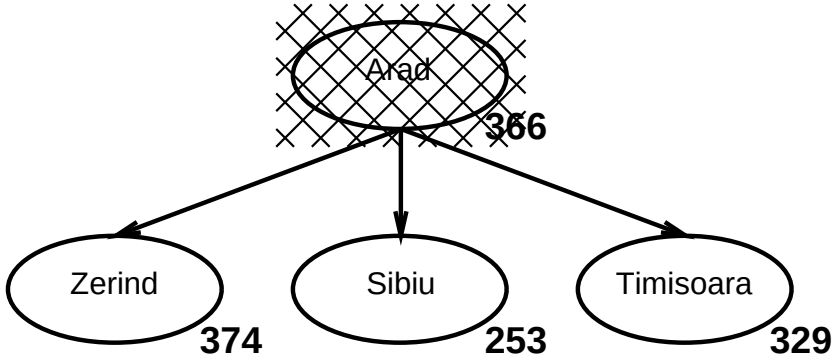
E.g., $h_{\mathrm{SLD}}(n) =$ distanza in linea d'aria da n a Bucharest

La ricerca greedy espande il nodo che $appare$ essere il più vicino al goal

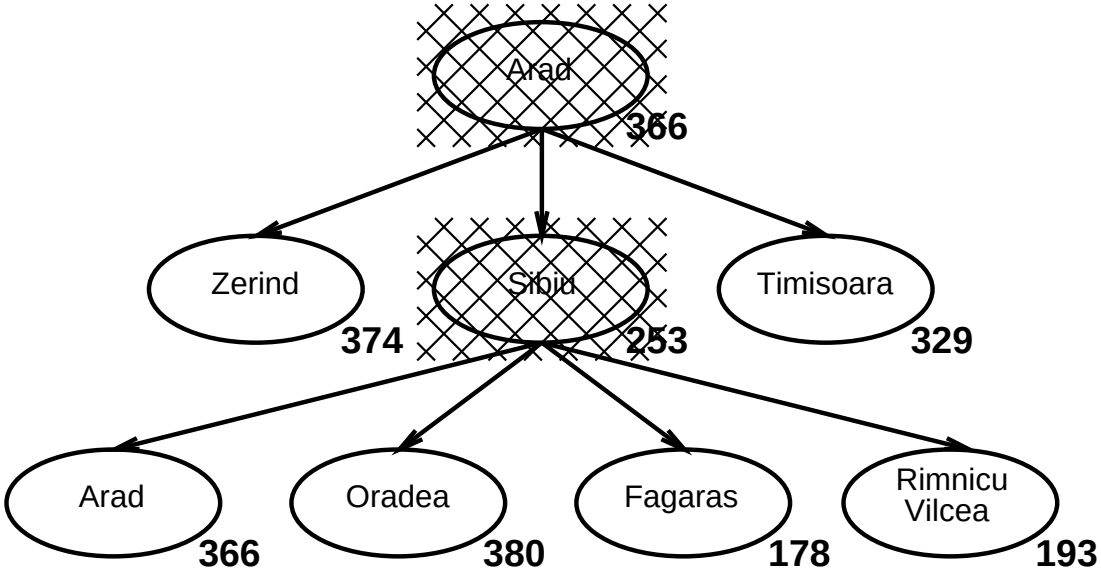
Esempio di ricerca greedy



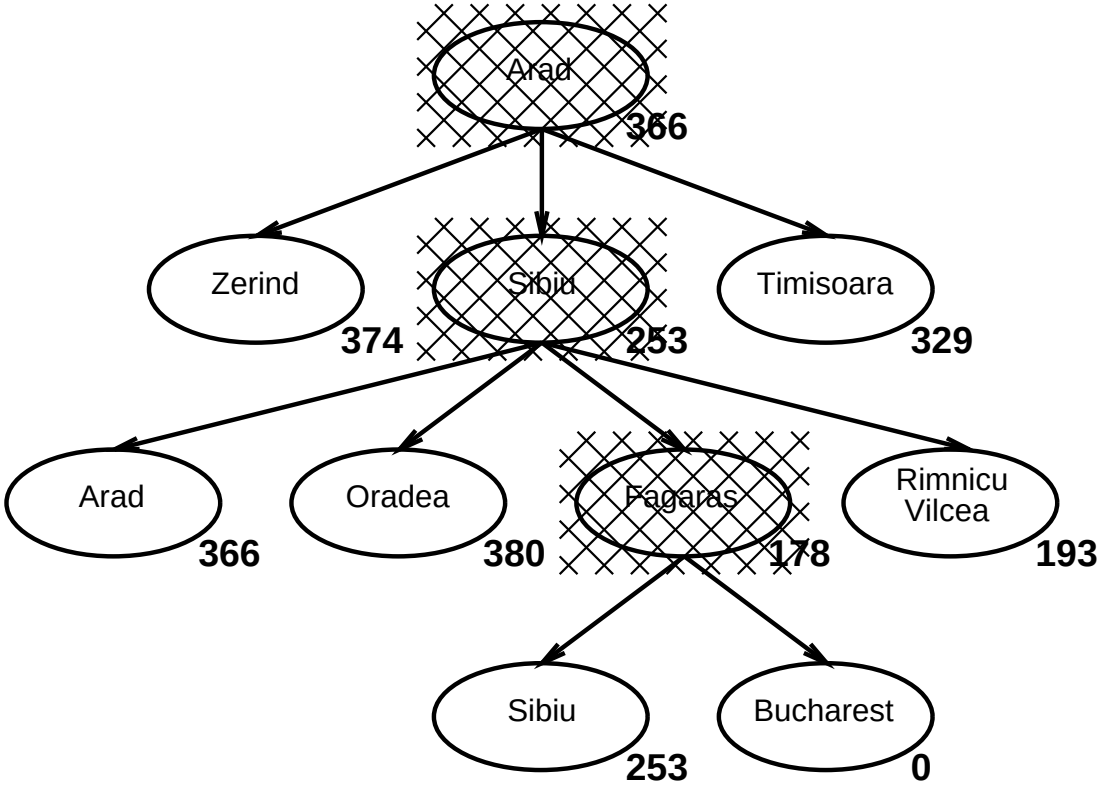
Esempio di ricerca greedy



Esempio di ricerca greedy



Esempio di ricerca greedy



Proprietà della ricerca greedy

Completa?? No – può restare intrappolata in cicli, per es.
Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$
Completa in spazi finiti con controllo di ripetizione di stati

Tempo?? $O(b^m)$, ma l'uso di una buona euristica può dare miglioramenti enormi

Spazio?? $O(b^m)$—mantiene tutti i nodi in memoria

Ottima?? No

Ricerca A*

Idea: evitare di espandere cammini che sono gia' costosi

Funzione di valutazione $f(n) = g(n) + h(n)$

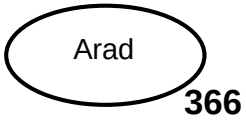
$g(n) =$ costo gia' avuto per raggiungere n
$h(n) =$ costo stimato da n al goal
$f(n) =$ costo totale stimato del cammino attraverso n al goal

La ricerca A* usa un'euristica $ammissibile$
cioe': $h(n) \leq h^*(n)$ dove $h^*(n)$ e' il costo $vero$ da n.

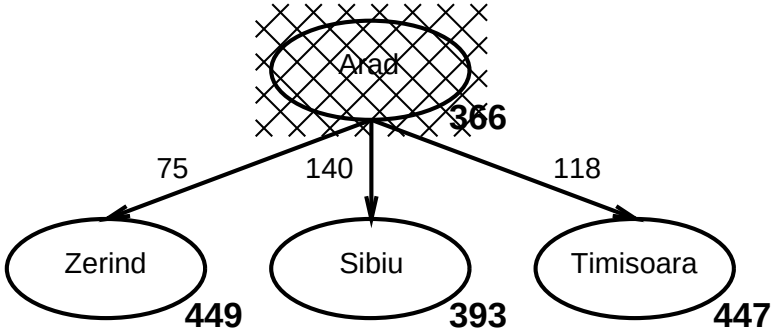
Es.: $h_{\mathrm{SLD}}(n)$ non sovrastima mai la reale distanza

Teorema: la ricerca A* e' ottima

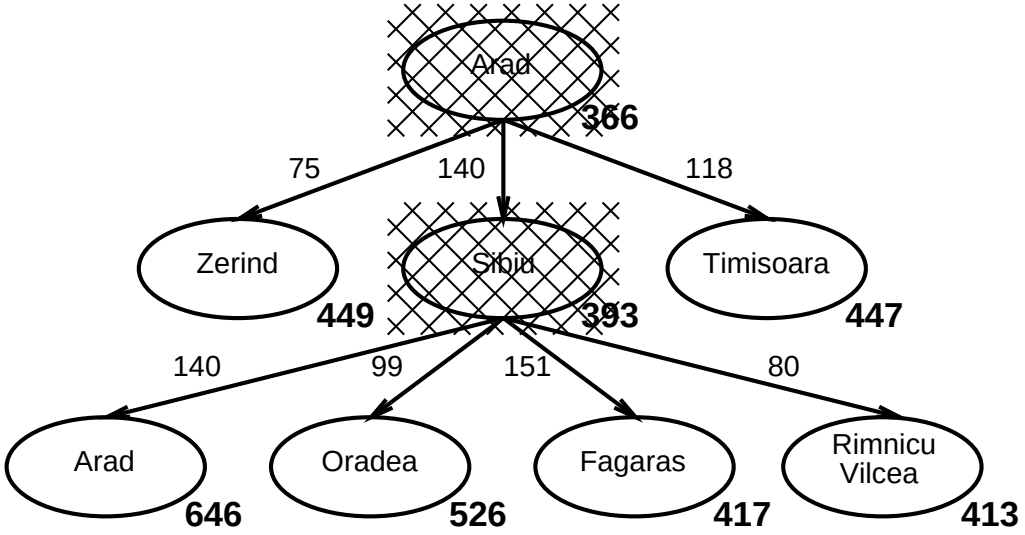
Esempio di ricerca A*



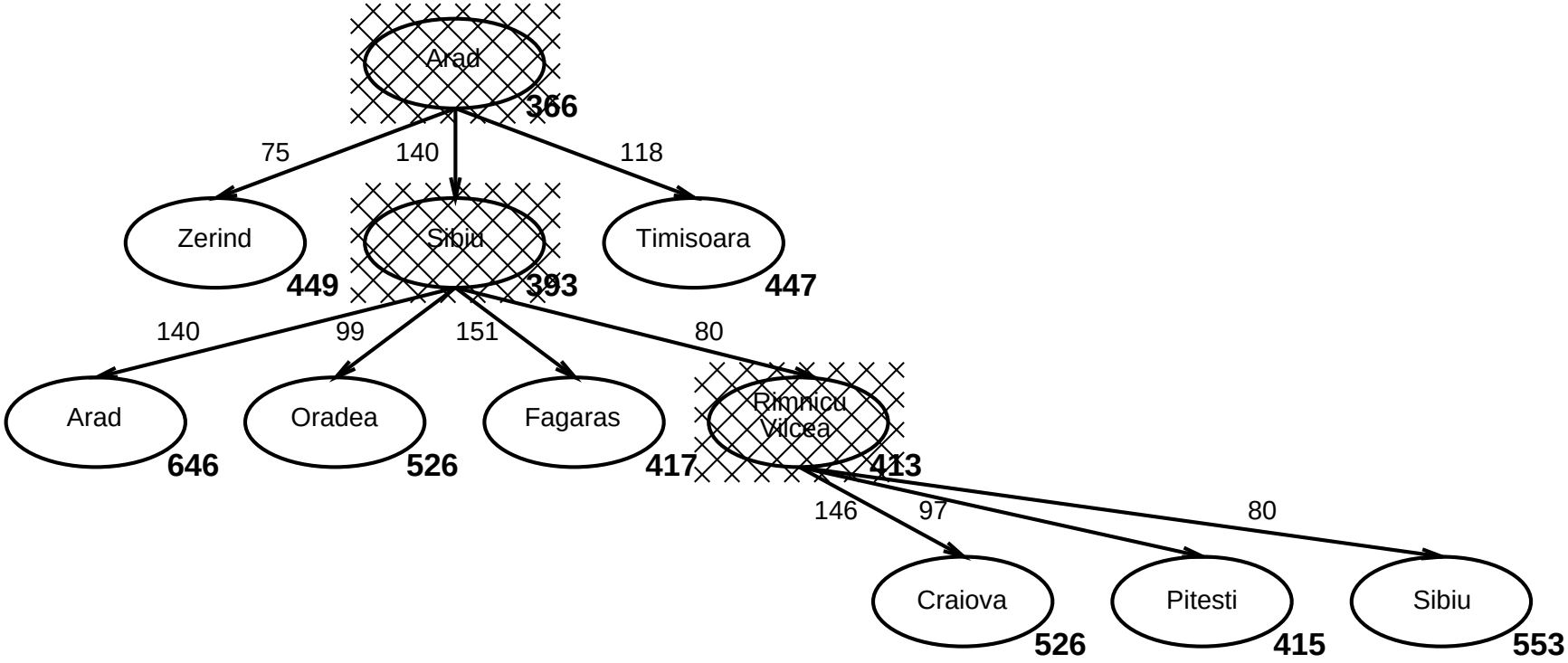
Esempio di ricerca A*



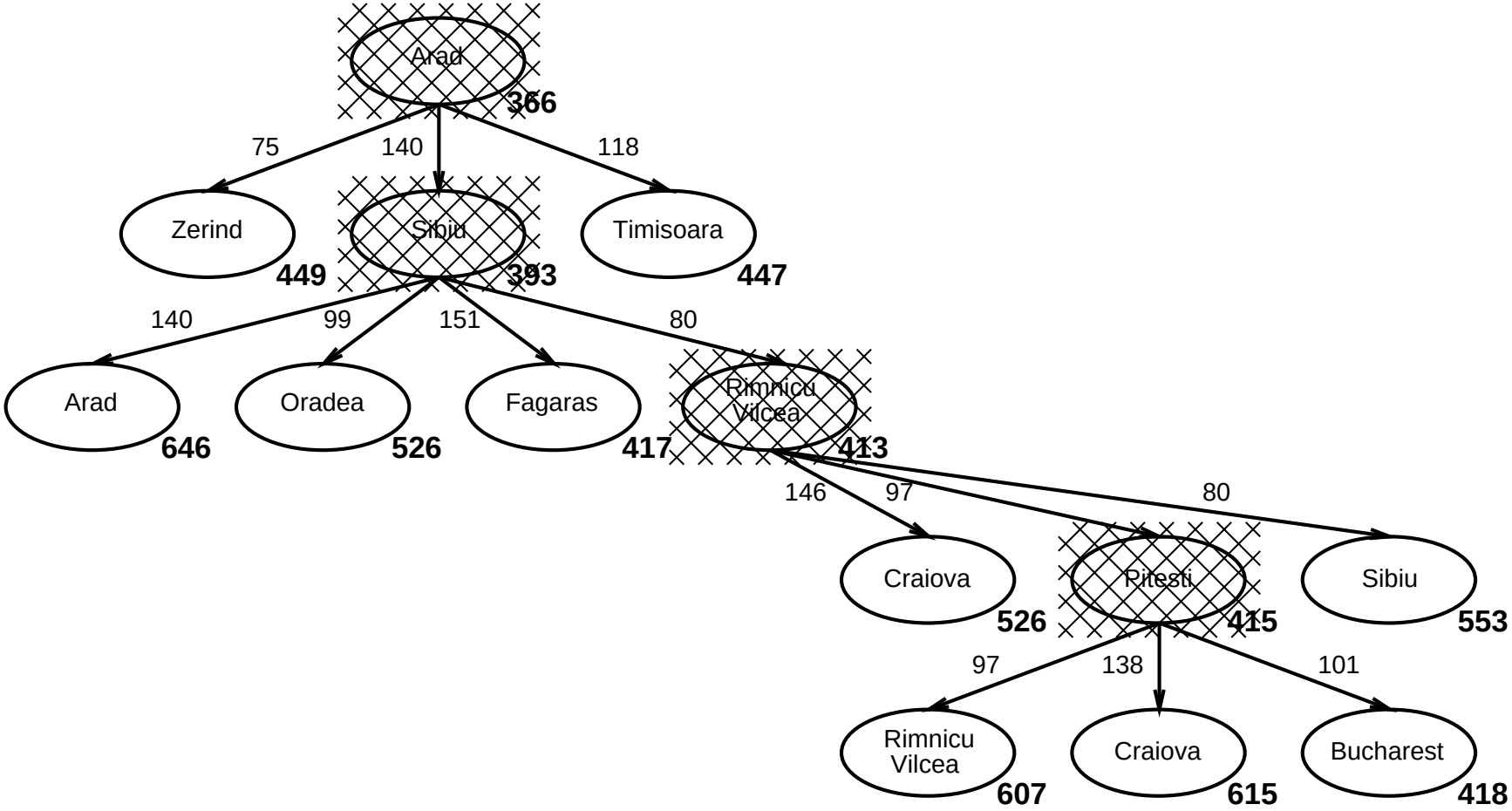
Esempio di ricerca A*



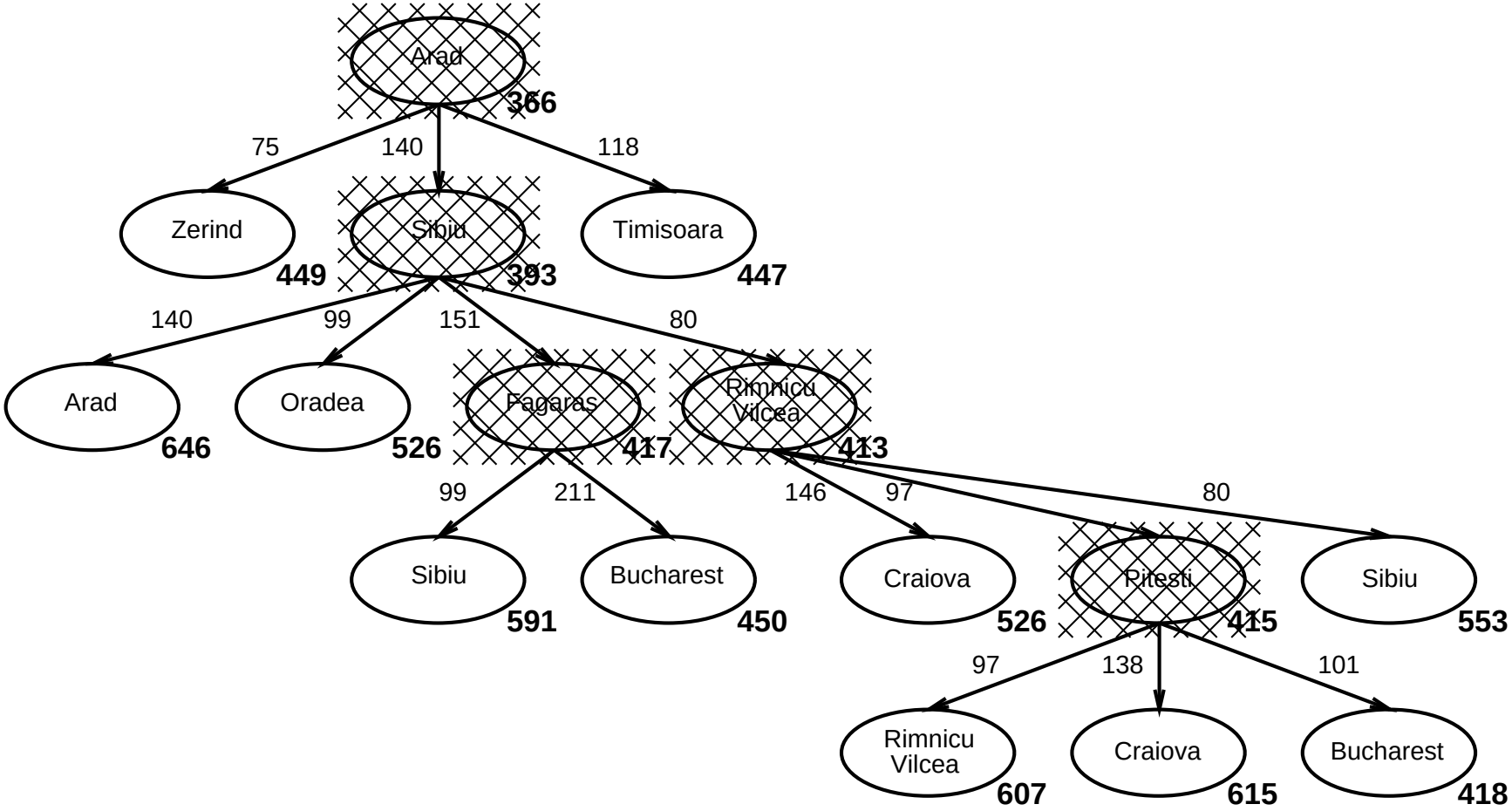
Esempio di ricerca A^*



Esempio di ricerca A^*

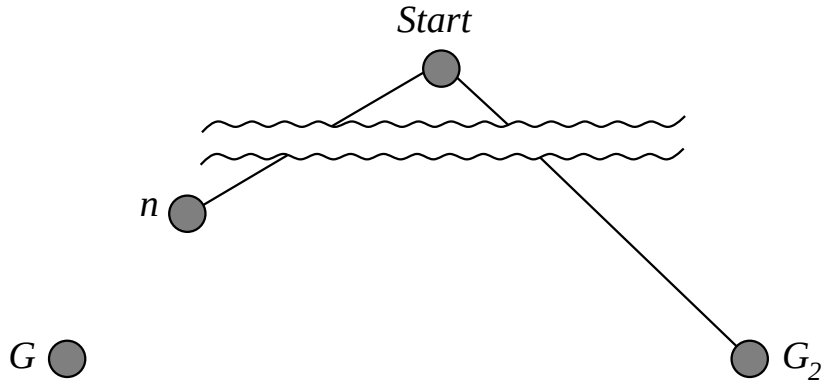


Esempio di ricerca A*



Ottimalita' di A*

Supponiamo un goal sub-ottimo G_2 e' stato generato ed e' nella coda. Sia n un nodo non ancora espanso su un cammino minimo verso un goal ottimo G.



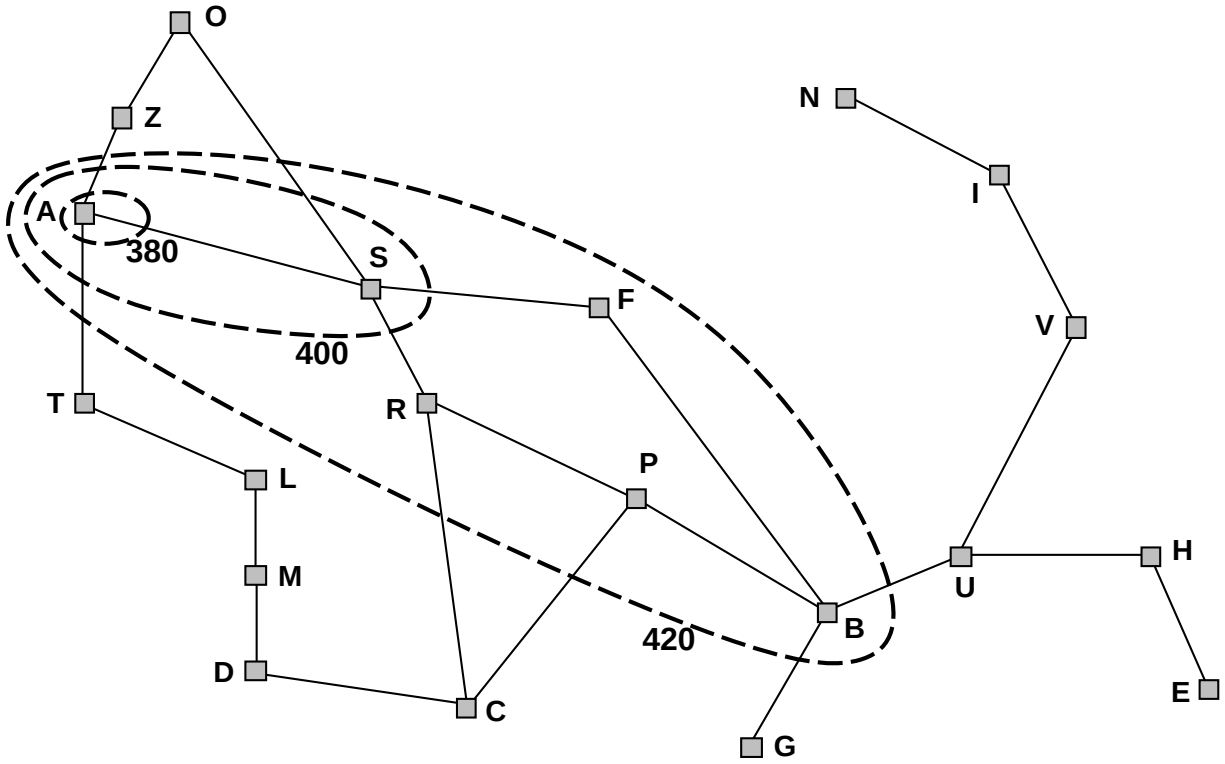
$$\begin{aligned} f(G_2) &= g(G_2) && \text{dato che } h(G_2) = 0 \\ &> g(G) && \text{dato che } G_2 \text{ e}' \text{ sub} - \text{ottimo} \\ &\geq f(n) && \text{dato che } h \text{ e}' \text{ ammissibile} \end{aligned}$$

Dato che $f(G_2) > f(n)$, A* non selezionera' mai G_2 per espanderlo

Ottimalita' di A* (piu' utile)

Lemma: A* espande i nodi in ordine di valore di f

Gradualmente, aggiunge dei "contorni di f" dei nodi
Il contorno i ha tutti i nodi con $f = f_i$, dove $f_i < f_{i+1}$



Proprieta' di A*

Completa?? Si, a meno che non ci sia un numero infinito di nodi con $f \leq f(G)$

Tempo?? Esponenziale in [errore relativo in h $\times$ lunghezza di sol.]

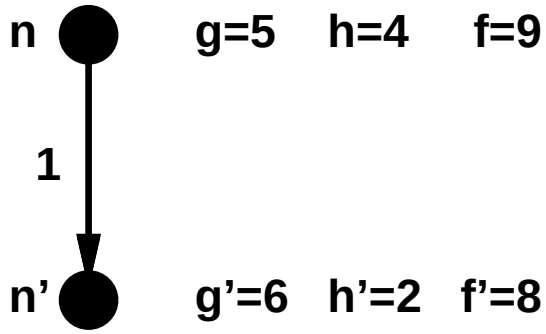
Spazio?? Mantiene tutti i nodi in memoria

Ottima?? Si—non puo' espandere f_{i+1} finche' f_i non e' finita

Prova del lemma: Pathmax

Per alcune euristiche ammissibili, f puo' $decrementare$ lungo un cammino

Es.: supponiamo che n' sia un successore di n



Ma questo getta via dell'informazione!
$f(n) = 9 \Rightarrow$ il vero costo di un cammino che passa da n e' ≥ 9
Quindi anche il vero costo di un cammino che passa da n' e' ≥ 9

Modifica pathmax di A*:
Invece di $f(n') = g(n') + h(n')$, usare $f(n') = max(g(n') + h(n'), f(n))$

Con pathmax, f e' sempre non-decrescente lungo un qualsiasi cammino