Artificial Intelligence- Informed Search and Exploration

Amir Aavani
Amir@Aavani.net
http://Amir.Aavani.net/CS/AI-86

22. Oktober 2007

Outline

- Informed Search Strategies
- Best-First Search
- A^*

Informed Search Strategies

- Usually, blind search strategies are inefficient due to large
- Uses problem-specific knowledge beyond the problem definition.
- They need to heuristic function(s).
- The general approach is called "best-first search"

Best-First Search

- Expands the best node first.
- Which node is the best node?
- Accurately determining best node $\Rightarrow$ No need to search.
- How to find which node is the best?

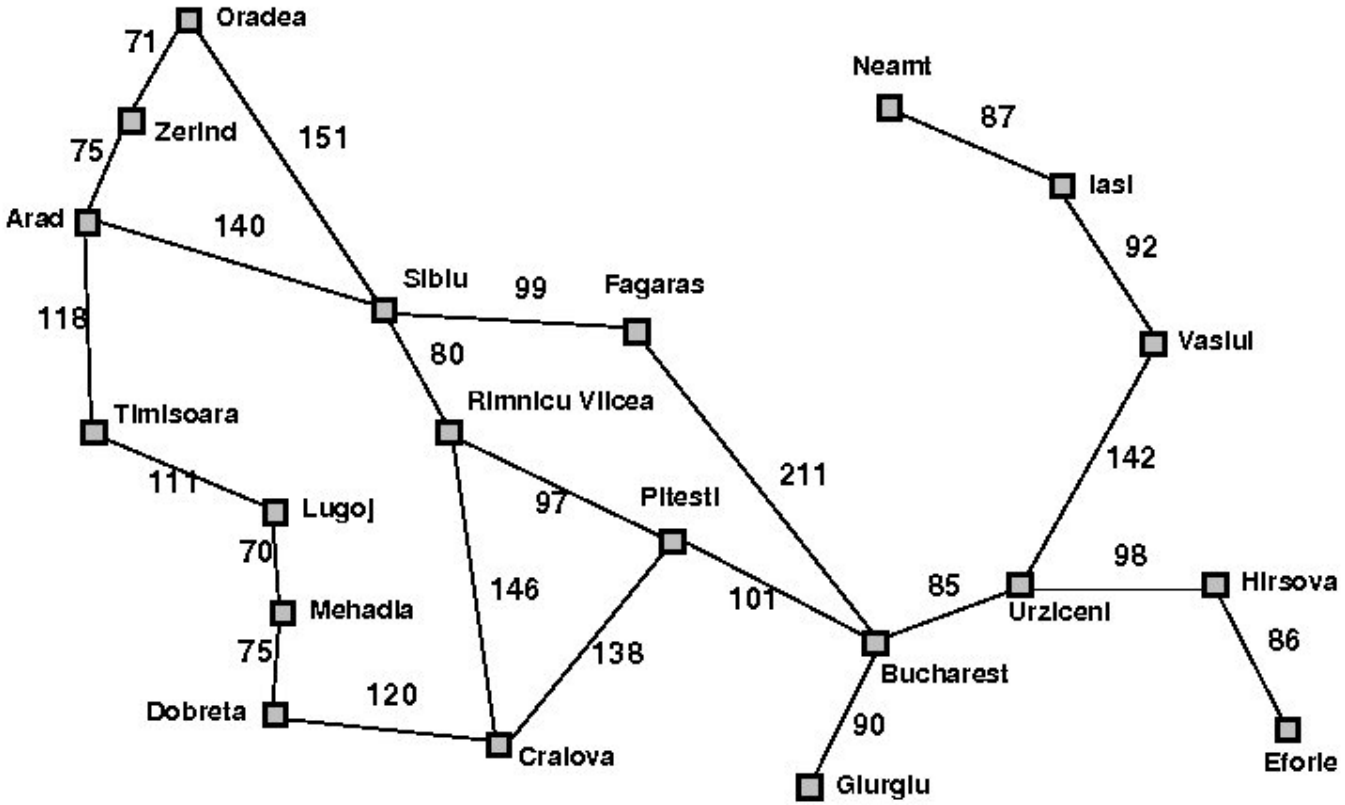
Best-First Search, Cont

- How to find which node is the best?
- Evaluation Function ($f(n)$): desirability of node n.
- Heuristic function ($h(n)$): estimates cheapest path cost to goal.
- h provides additional information about the problem.
- Vacuum cleaner: number of dirty cells.

Greedy Best-First Search

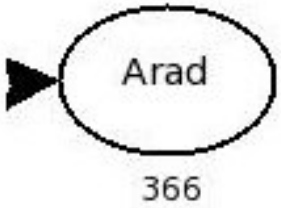
- Expands the node that appears to be closest to goal.
- i.e., $f(n) = h(n)$
- Romania problem: $h_{SLD}(n)$= straight-line-distance from n to Bucharest.
- $h_{SLD}(Bucharest) = 0$

Greedy Best-First Search, Cont

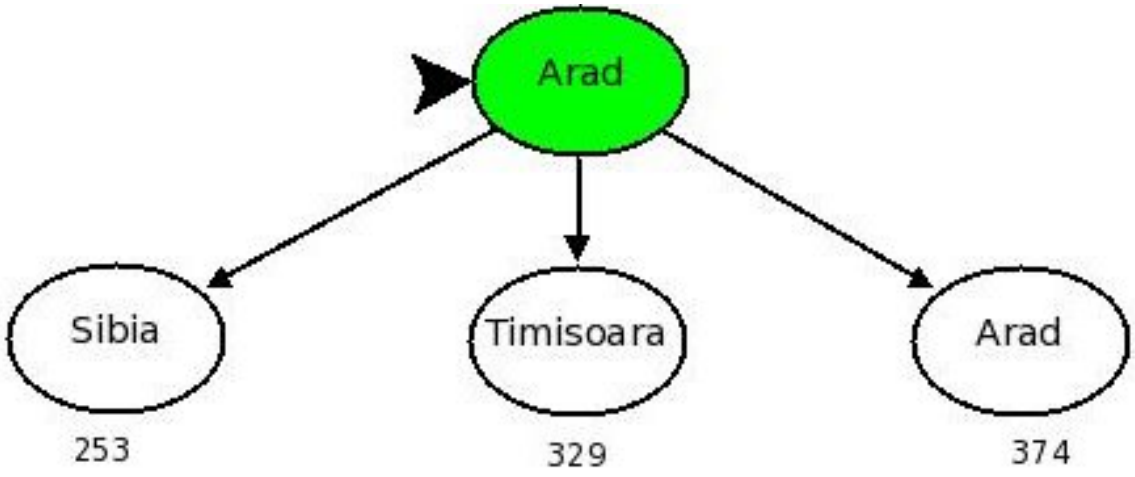


Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329

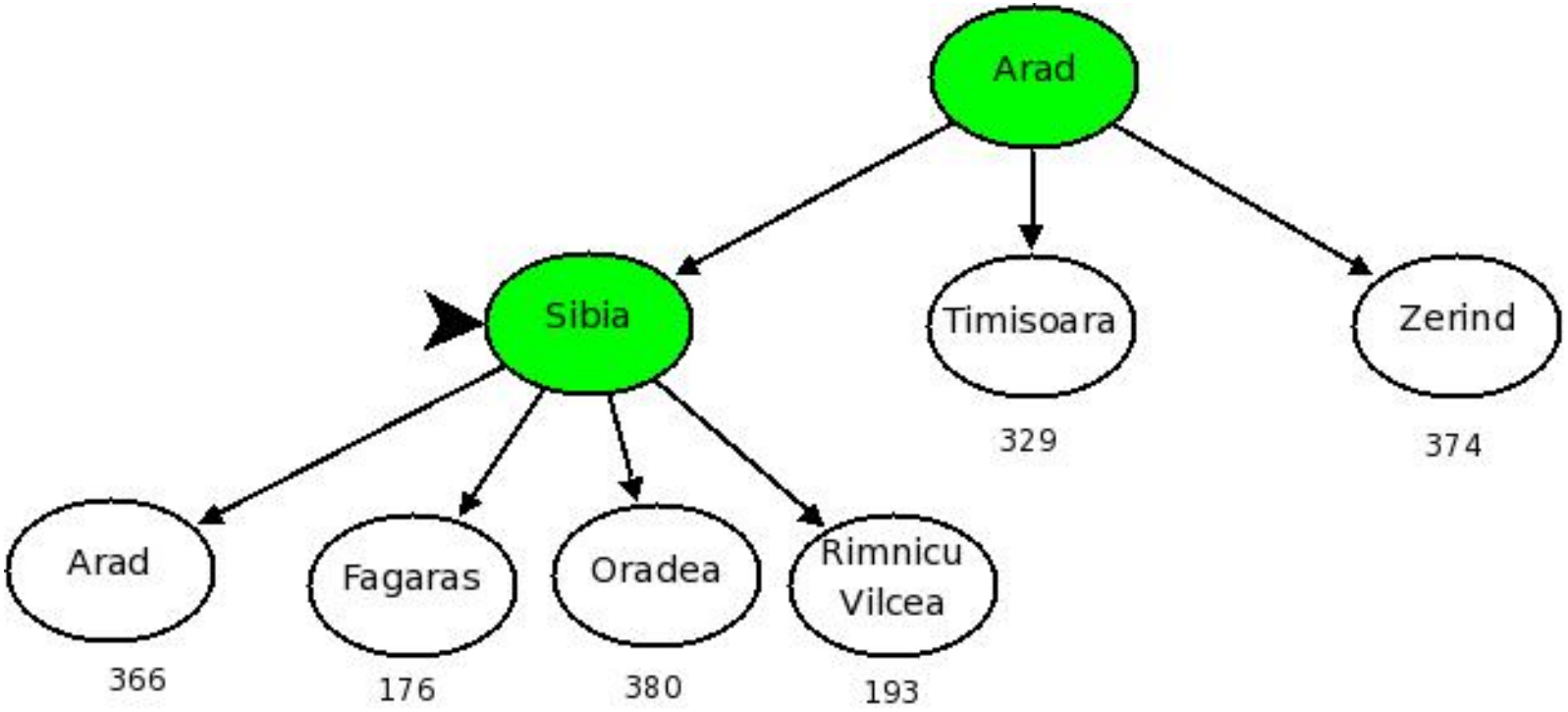
Greedy Best-First Search, Cont



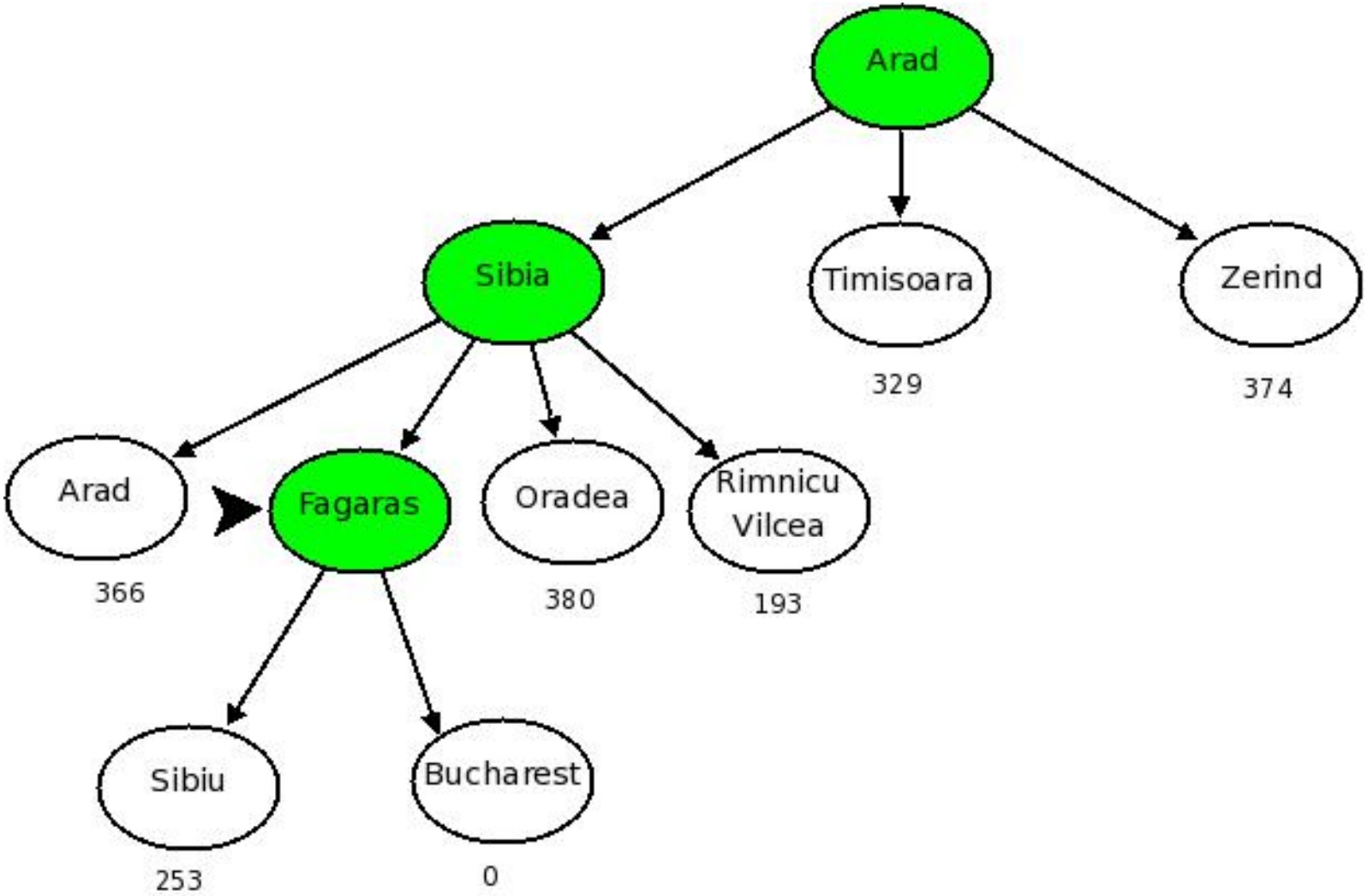
Greedy Best-First Search, Cont



Greedy Best-First Search, Cont



Greedy Best-First Search, Cont



Greedy Best-First Search, Cont

- Complete:
- Time Complexity:
- Space Complexity:
- Optimality:

Greedy Best-First Search, Cont

- Complete: No, may get stuck in loops, e.g., Iasi $\Rightarrow$ Neamt $\Rightarrow$ Iasi $\Rightarrow$ Neamt
- Time Complexity:
- Space Complexity:
- Optimality:

Greedy Best-First Search, Cont

- Complete: No, may get stuck in loops, e.g.,
  Iasi $\Rightarrow$ Neamt $\Rightarrow$ Iasi $\Rightarrow$ Neamt
- Time Complexity: $O(b^m)$, it is very dependant to h function.
- Space Complexity:
- Optimality:

Greedy Best-First Search, Cont

- Complete: No, may get stuck in loops, e.g.,
  Iasi $\Rightarrow$ Neamt $\Rightarrow$ Iasi $\Rightarrow$ Neamt
- Time Complexity: $O(b^m)$, it is very dependant to h function.
- Space Complexity: $O(b^m)$
- Optimality:

Greedy Best-First Search, Cont

- Complete: No, may get stuck in loops, e.g.,
  Iasi $\Rightarrow$ Neamt $\Rightarrow$ Iasi $\Rightarrow$ Neamt
- Time Complexity: $O(b^m)$, it is very dependant to h function.
- Space Complexity: $O(b^m)$
- Optimality: No.

Greedy Best-First Search, Cont

- Follows single path to goal, like DFS.
- It is complete in finite space with repeated-state checking.
- May expands an expensive path.

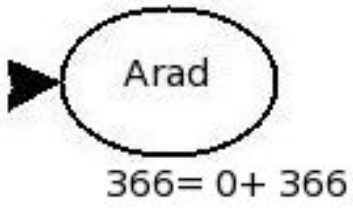
A^* Search

- avoids expanding paths that are already expensive.
- $f(n) = g(n) + h(n)$
- $g(n) =$ cost to reach to n.
- $h(n) =$ estimated cost to goal to n.
- $f(n) =$ estimated cost of path passing through n.

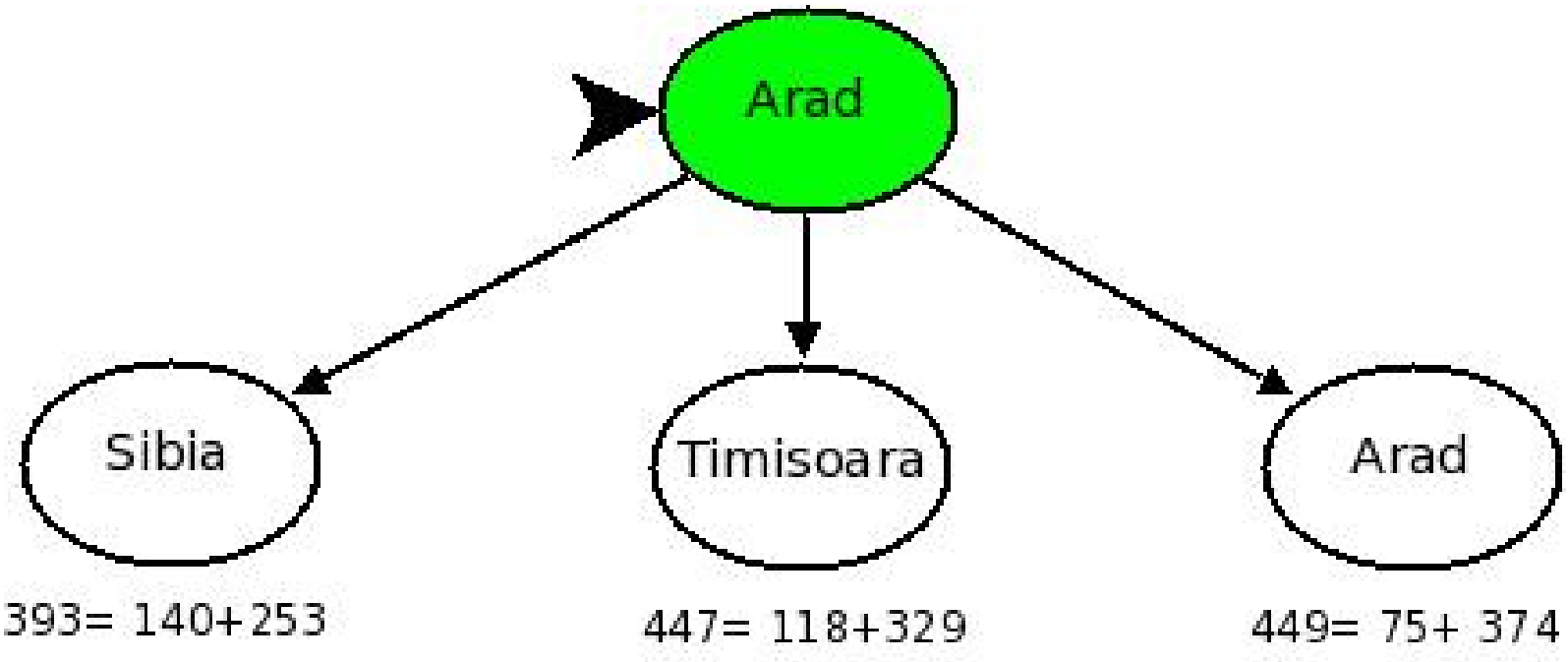
A^* Search, Cont

- Admissible heuristic function is a function that never over-estimates.
- $g(n)$ is the exact cost of path to n.
- So, $f(n)$ is always under-estimate.
- h_{SLD} is an admissible heuristic function.

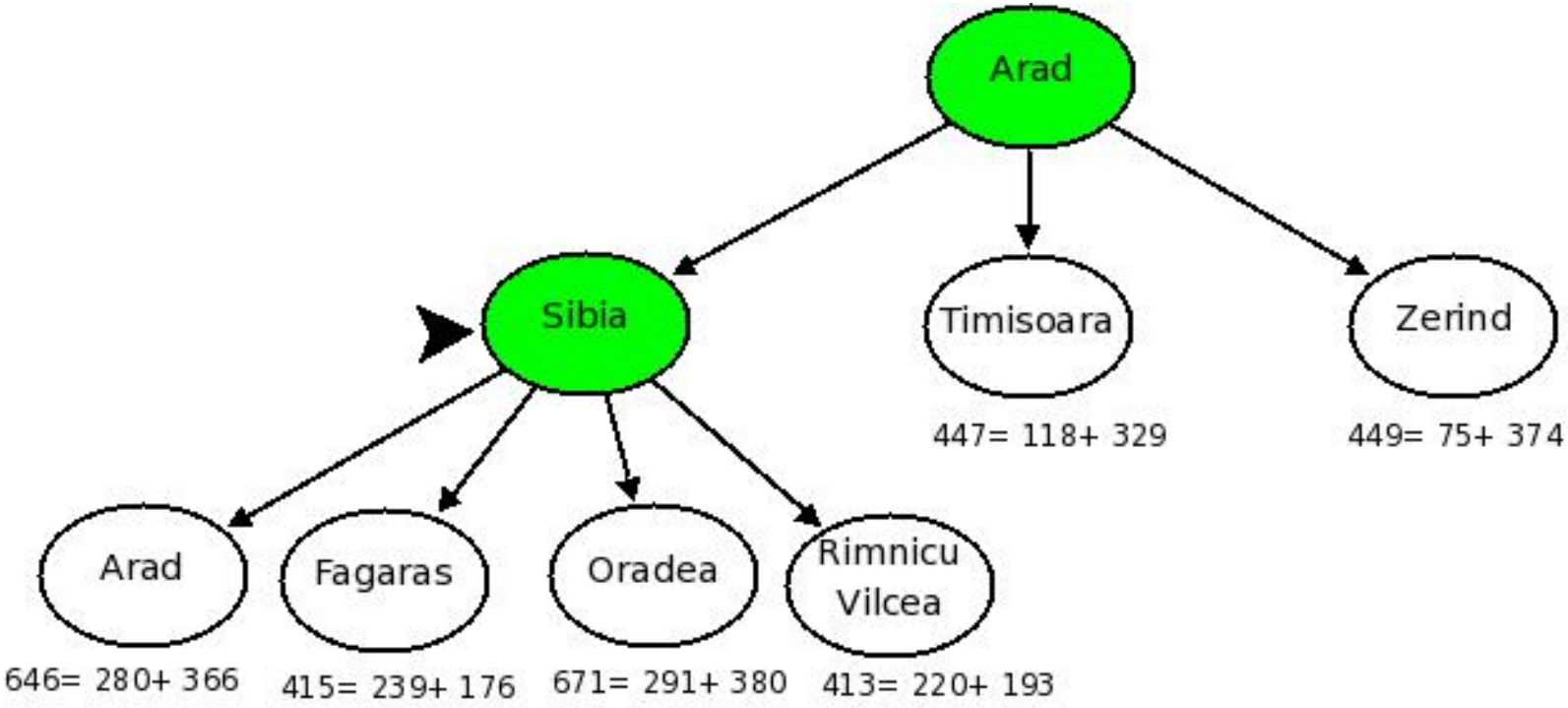
A^* Search, Cont



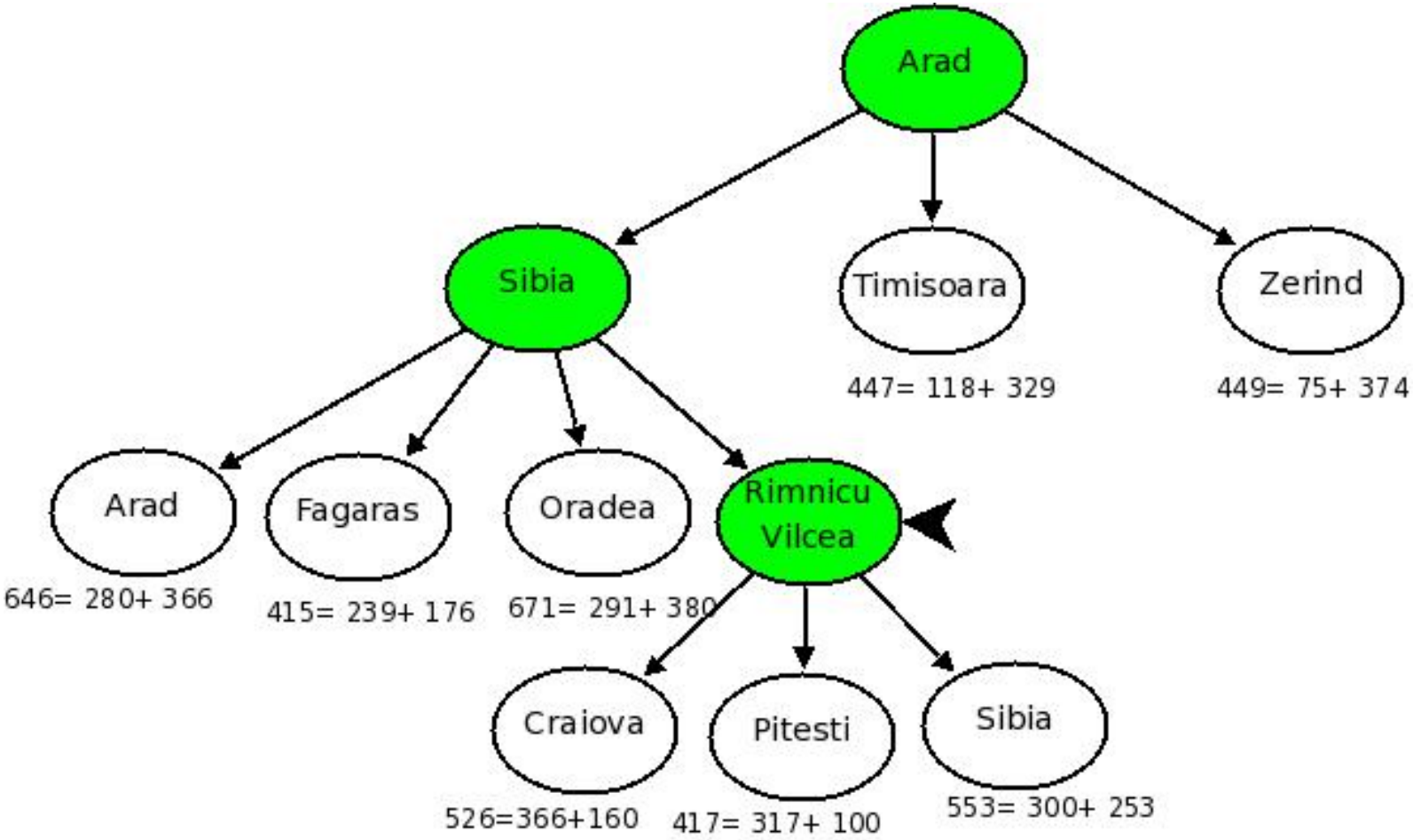
A^* Search, Cont



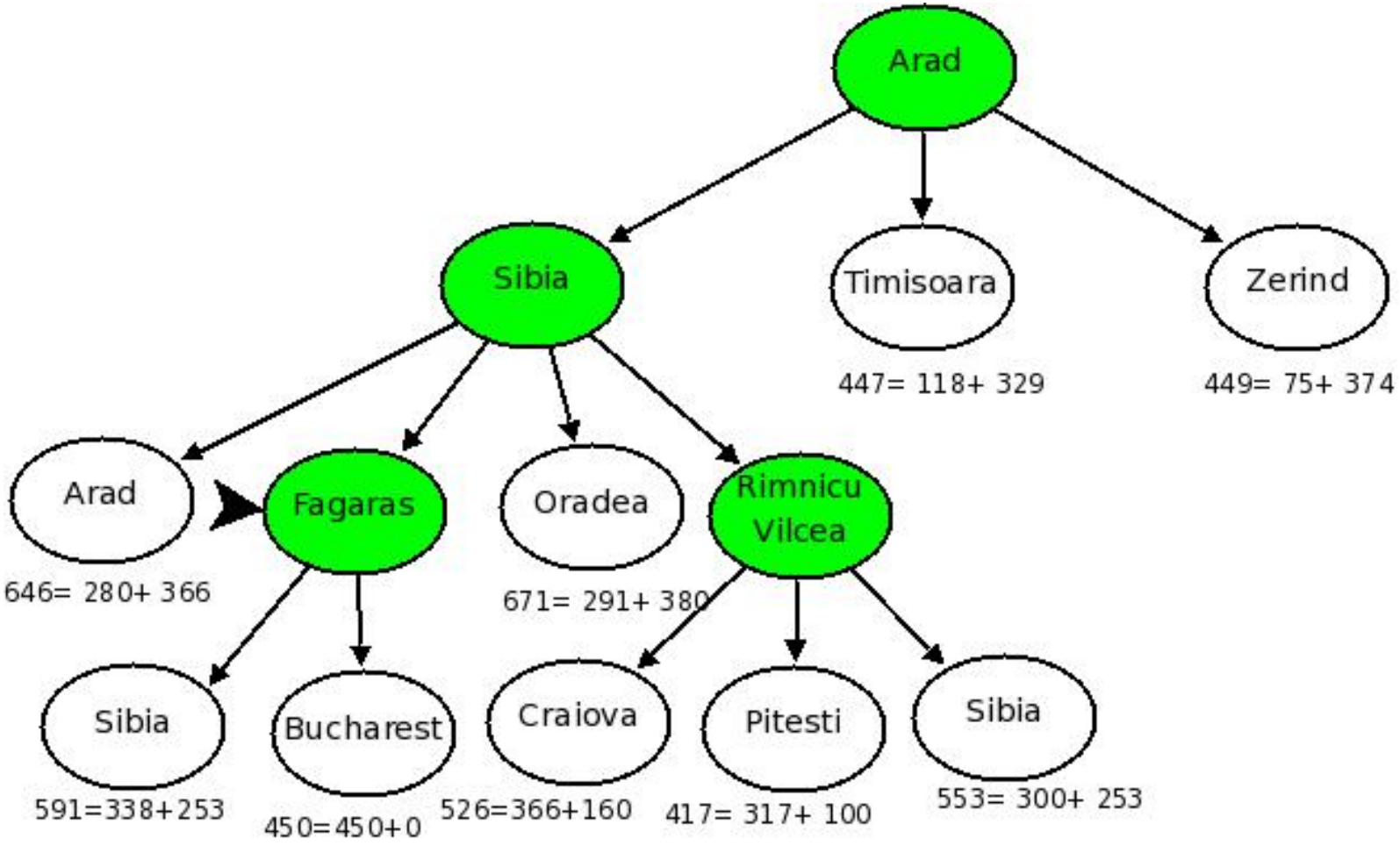
A^* Search, Cont



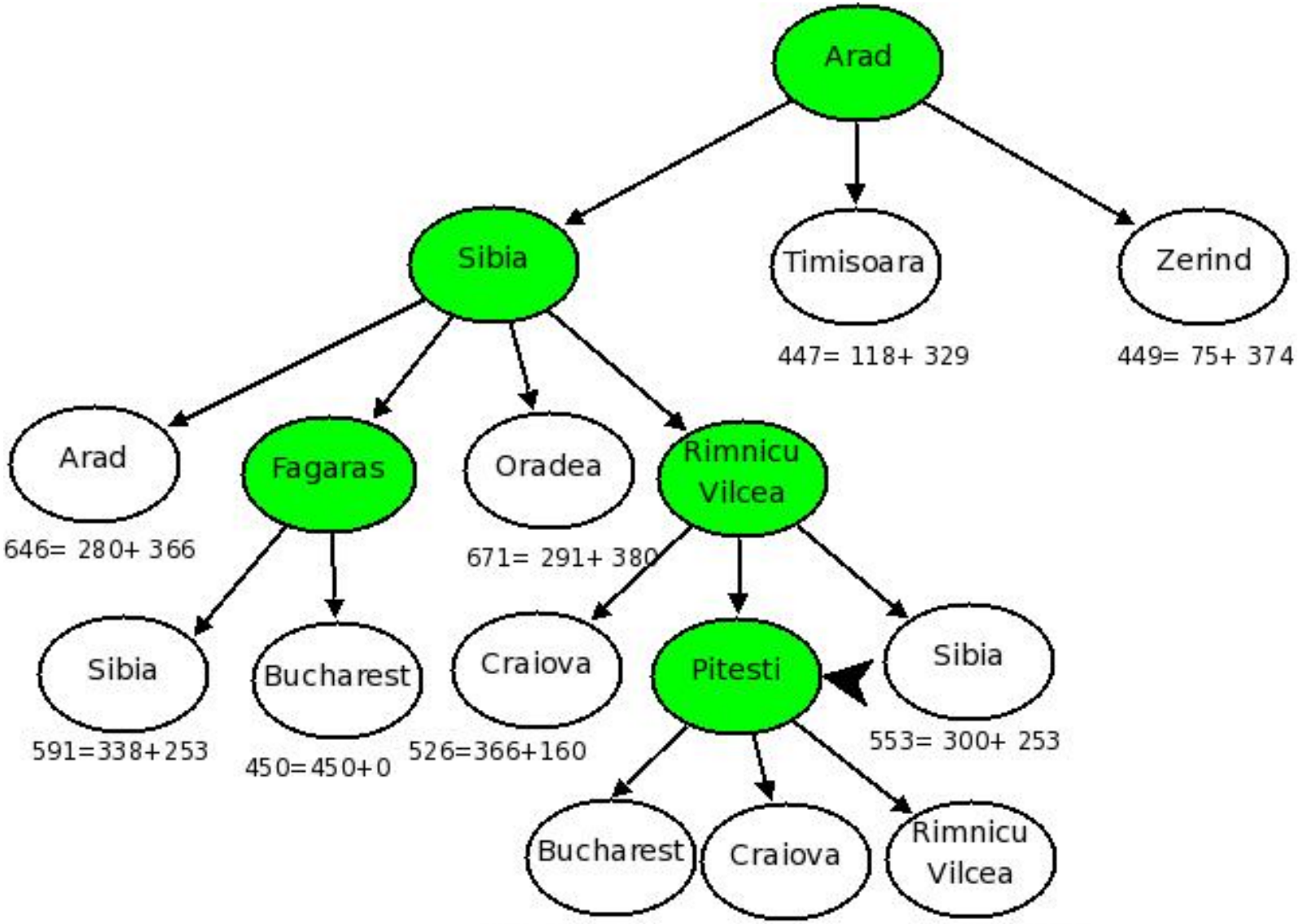
A^* Search, Cont



A^* Search, Cont



A^* Search, Cont



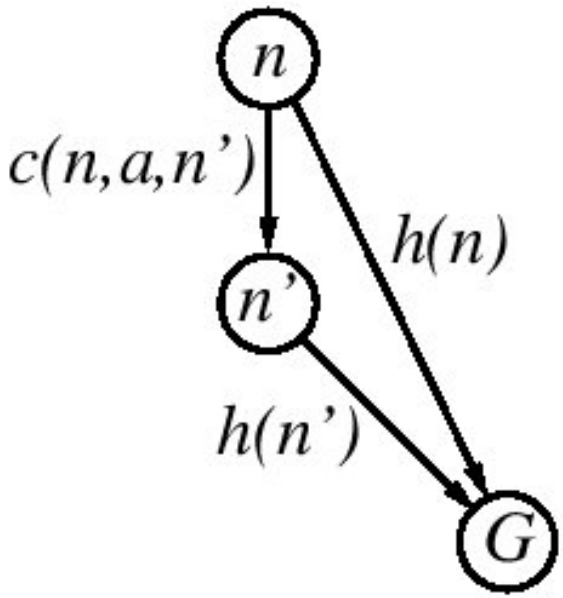
Optimality of A^* Search

Optimality of A^* Search

- Let G_2 be a suboptimal goal.
- Let n be an unexpanded on shortest path to optimal goal (G_1).
- $f(G_2) = g(G_2)$ $\quad$ $h(G_2) = 0$
- $> g(G_1)$ $\quad$ G_2 is suboptimal goal.
- $\geq f(n)$
- Since $f(G_2) > f(n)$, A^* will never select G_2 for expansion.

Consistent Heuristic function

- A heuristic function is consistent if $h(n) \leq c(n, a, n') + h(n')$
- if h is consistent, we have
  $f(n') = g(n') + h(n')$
  $= g(n) + c(n, a, n') + h(n')$
  $\geq g(n) + h(n)$
  $= f(n)$



Consistent Heuristic function, Cont

- $f(n)$ is non-decreasing along any path.
- Every consistent function is also an admissible one.
- It is very hard to find a heuristic function which is admissible but not consistent.

Properties of A^*

- Complete:
- Time complexity:
- Space complexity:
- Optimality:

Properties of A^*, Cont

- Complete: Yes
- Time complexity:
- Space complexity:
- Optimality:

Properties of A^*, Cont

- Complete: Yes.
- Time complexity: Exponential in length of solution.
- Space complexity:
- Optimality:

Properties of A^*, Cont

- Complete: Yes.
- Time complexity: Exponential in length of solution.
- Space complexity: Keeps all nodes in memory.
- Optimality:

Properties of A^*, Cont

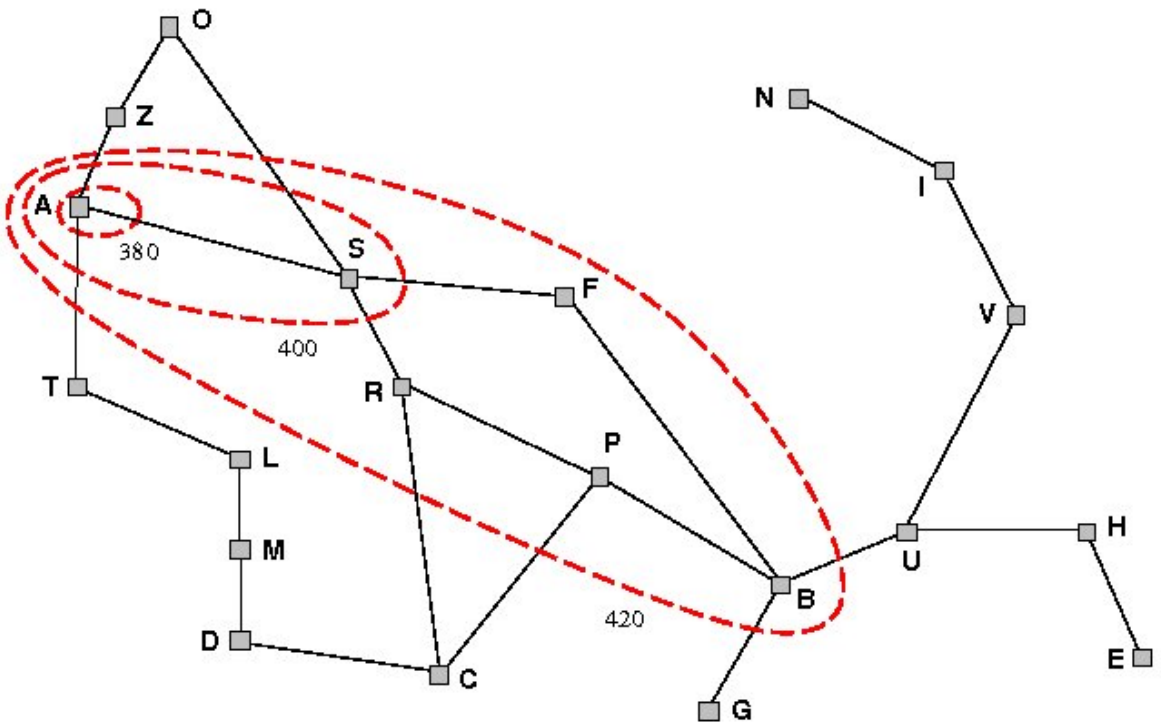
- Complete: Yes.
- Time complexity: Exponential in length of solution.
- Space complexity: Keeps all nodes in memory.
- Optimality: Yes.

Memory-bounded A^* Search

- The main draw-back of A^* is space.
- To bound the number of expanded nodes, some strategies can be used:
  - *RBFS*
  - *SMA**

IDA^* Search

- A^* expands nodes in order of non-decreasing f value.



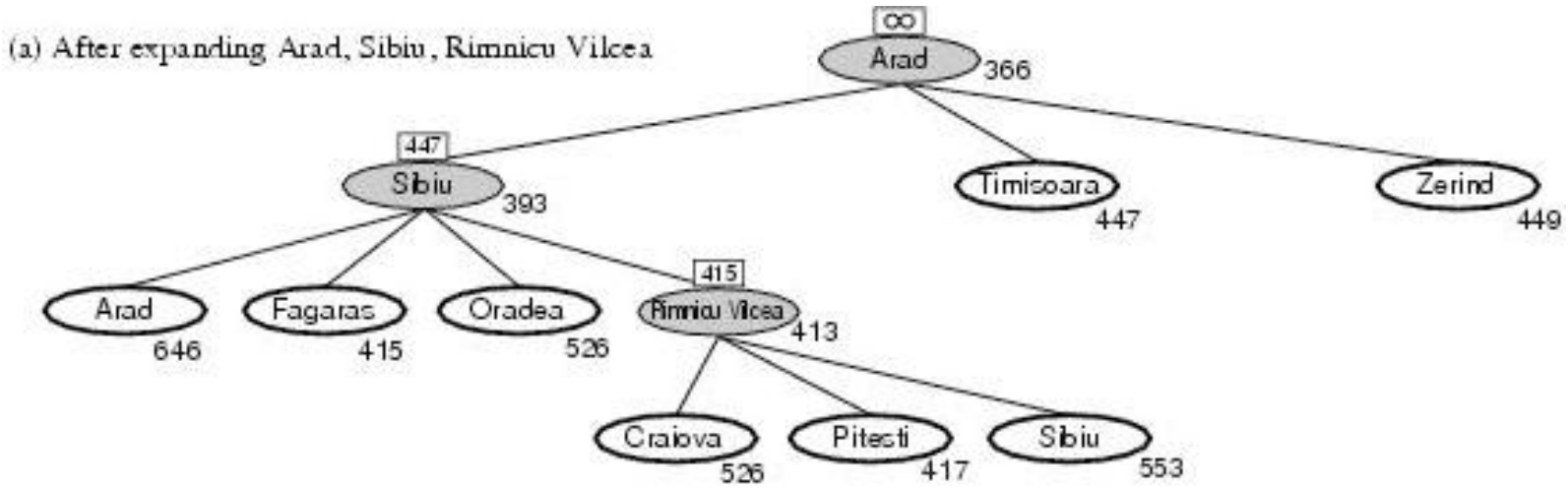
Recursive Best-First Search

- It is recursive version of best-first search method.
- RBFS uses linear memory.
- If current f-values exceeds this alternative f-value than backtrack to alternative path.
- Upon backtracking change f-value to best f-value of its children.
- Re-expansion of this result is thus still possible.

Recursive Best-First Search, Cont

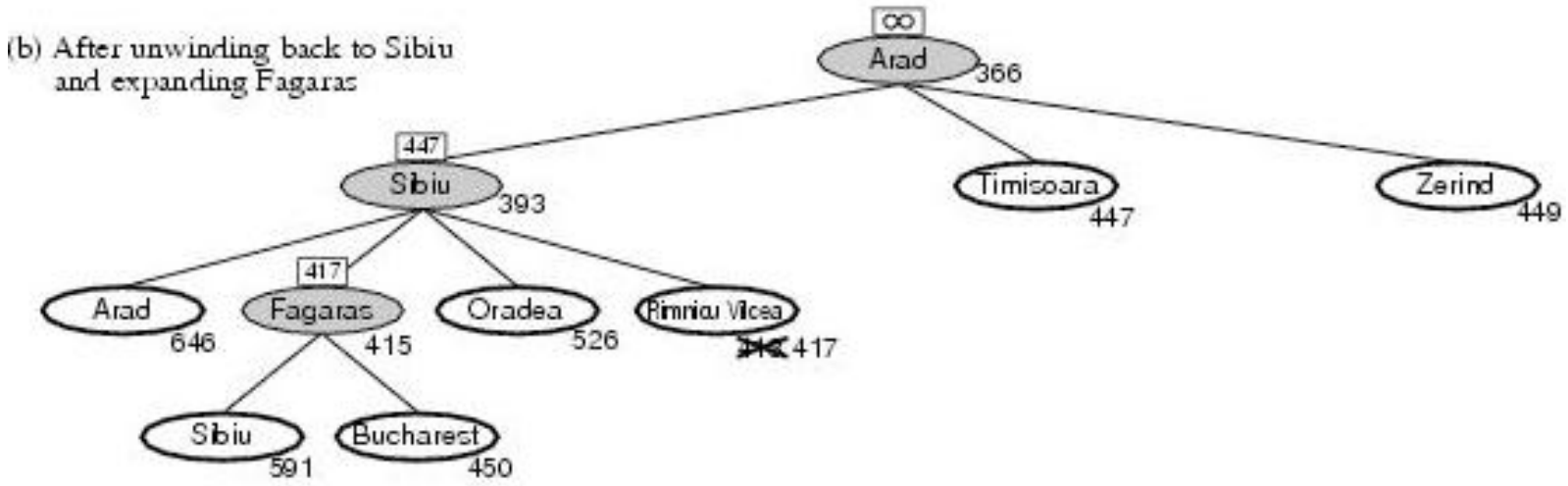
```
function RBFS (problem,node,f-limit) returns sol & f-cost
  if GOAL-TEST (problem, STATE (node)) then return node
  succ= EXPAND (node, problem)
  if succ is empty then return failure
  for-each s in succ do
    f [s]= max (g(s)+ h(s), f[node])
  repeat
    best= the lowest f-value node in succ
    if f[best]> f-limit then return failure, f[best]
    alternative= the second-lowest f-value among succ
    result,f[best]= RBFS (problem, best, min(f-limit,altern
    if result<> failure then return result
```

Recursive Best-First Search, Cont



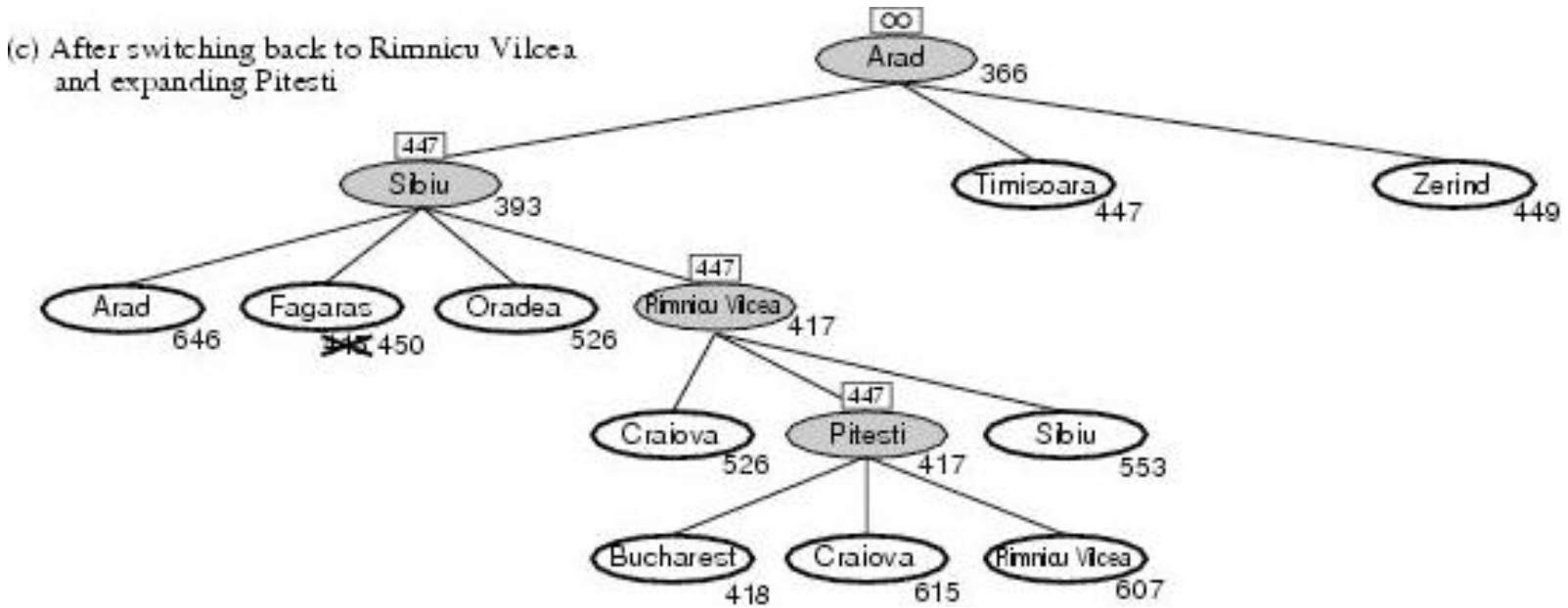
- The path is followed until Pitesti which has a f-value worse than the f-limit.

Recursive Best-First Search, Cont



- Unwind recursion and store best f-value for current best leaf Pitesti
  $result, f[best] = RBFS(problem, best, min(f_limit, alternative))$
- best is now Fagaras. Call RBFS for new best
  best value is now 450

Recursive Best-First Search, Cont



- best is now Rimnicu Viclea (again). Call RBFS for new best Subtree is again expanded.
  Best alternative subtree is now through Timisoara.
- Solution is found since because $447 > 417$.

Recursive Best-First Search, Properties

- RBFS is a bit more efficient than A^* but still excessive node generation (mind changes).
- Time complexity difficult to characterize.
- Space complexity is O(bd).

(Simplified) memory-bounded A^*

- Expand best leaf until available memory is full.
- When full, SMA^* drops worst leaf node (highest f-value).
- Like RFBS backup forgotten node to its parent.
- SMA^* is complete if solution is reachable, optimal if optimal solution is reachable.