CS 480

Introduction to Artificial Intelligence

February 1, 2022

Announcements / Reminders

- **Please follow the Week 03 TO DO List**
- **Quiz #01: due on Sunday (02/06) at 11:00 PM CST**
- **Written Assignment #1 will be posted within a week**
- **Programming Assignment #1 will be posted within 1.5 - 2 weeks**
- **Exam dates (consider fixed):**
  - **Midterm: February 24, 2022 during lecture time**
  - **Final: April 28, 2022 during lecture time**

Plan for Today

- **Problem Solving: Searching**
  - **Uninformed search: continued**
  - **Informed search**
    - **Hill Climbing algorithm**
    - **Greedy Best First algorithm**
    - **A* algorithm**

Designing the Searching Problem

Analyze and define the Problem / Task

Model and buid the State Space

Select searching algorithm

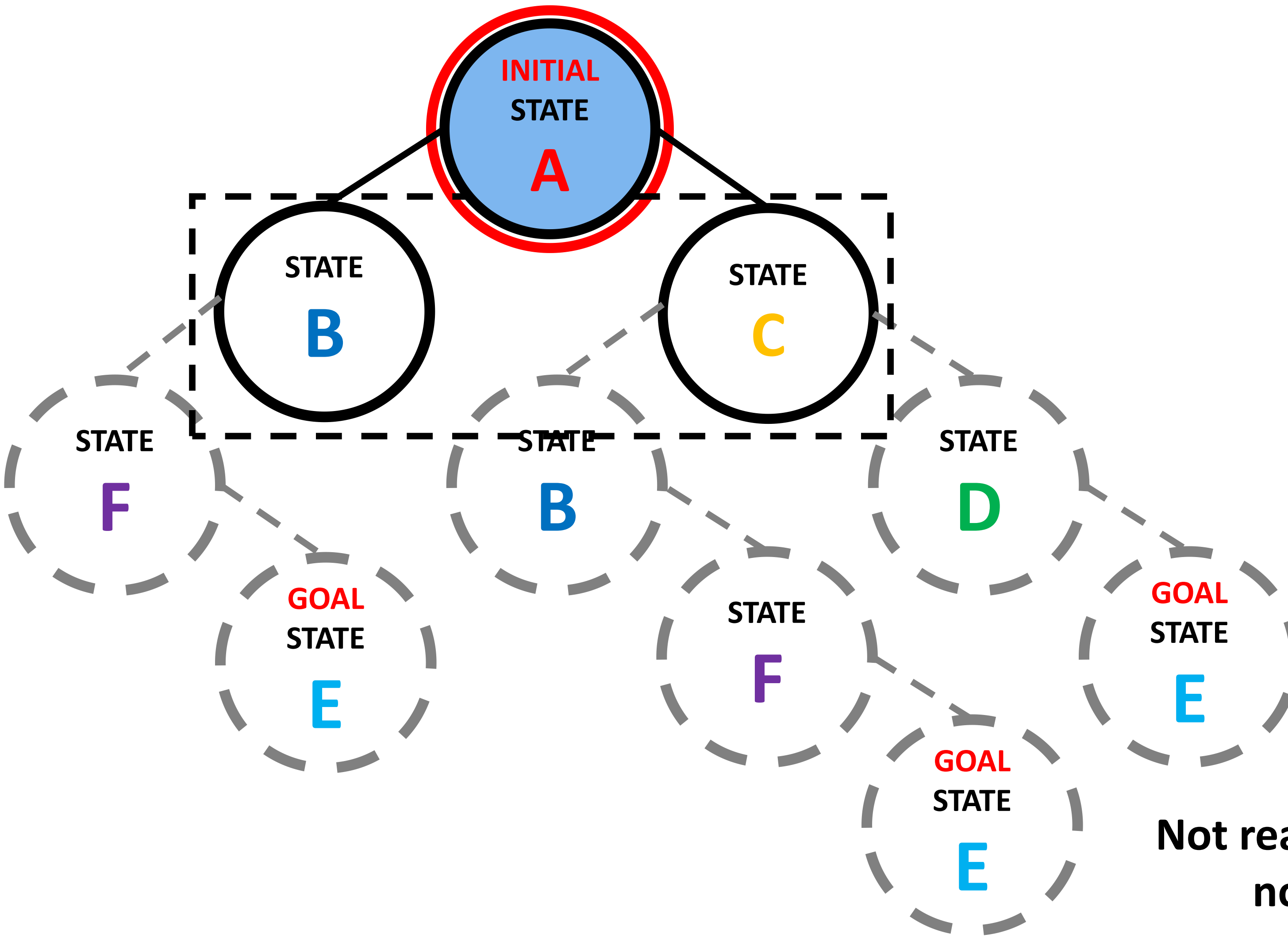
Search

Selected Searching Algorithms

- Searching Algorithms
  - **Uninformed** Search
    - Breadth First Search
    - Depth First Search
    - Brute Force Search
    - Uniform Cost Search
  - **Informed** Search
    - Best First Search
    - Branch and Bound
    - A* Algorithm
    - Hill Climbing Algorithm
    - Constraint Satisfaction Search
    - α - β Pruning
    - Min-Max Search

Uninformed search: an agent <u>has no estimate</u> how far it is from the goal
Informed search: an agent <u>can estimate</u> how far it is from the goal

Expansion: Which Node to Expand?



Not reached yet nodes

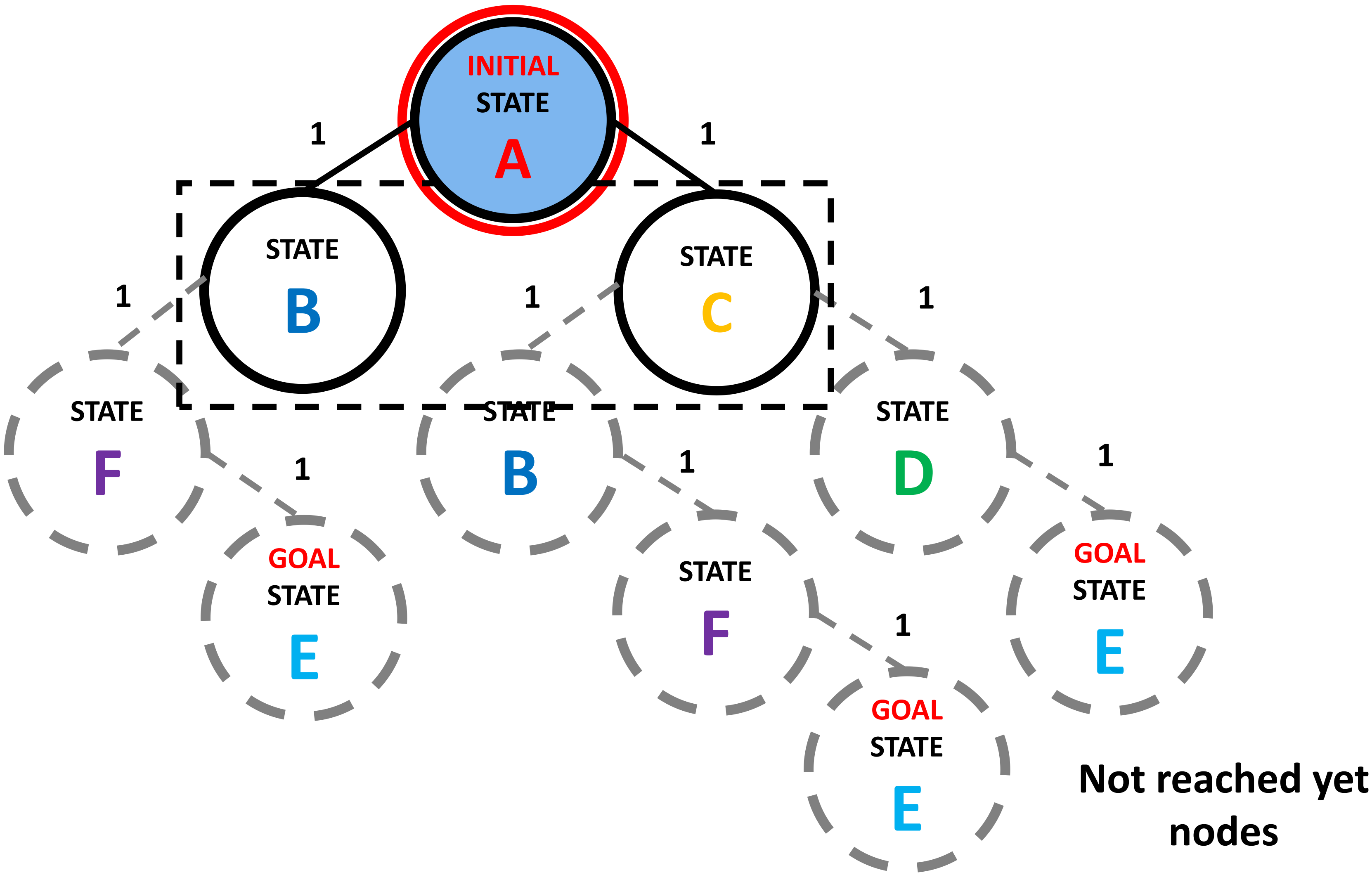
Evaluation function

Calculate / obtain:

f(n) = f(State n)
f(n) = f(relevant information about State n)

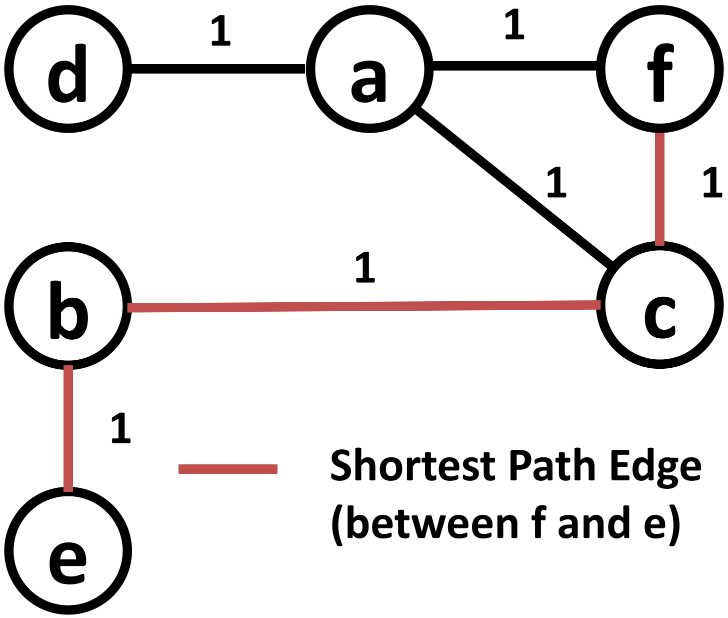
A state n with minimum f(n) should be chosen for expansion
What about ties?

Search Tree: Uniform Action Cost



Uniform Cost Search | Dijkstra's Algo

Weighted Graph G



Popular algorithms:

- **Dijkstra's algorithm**

Shortest Path Problem

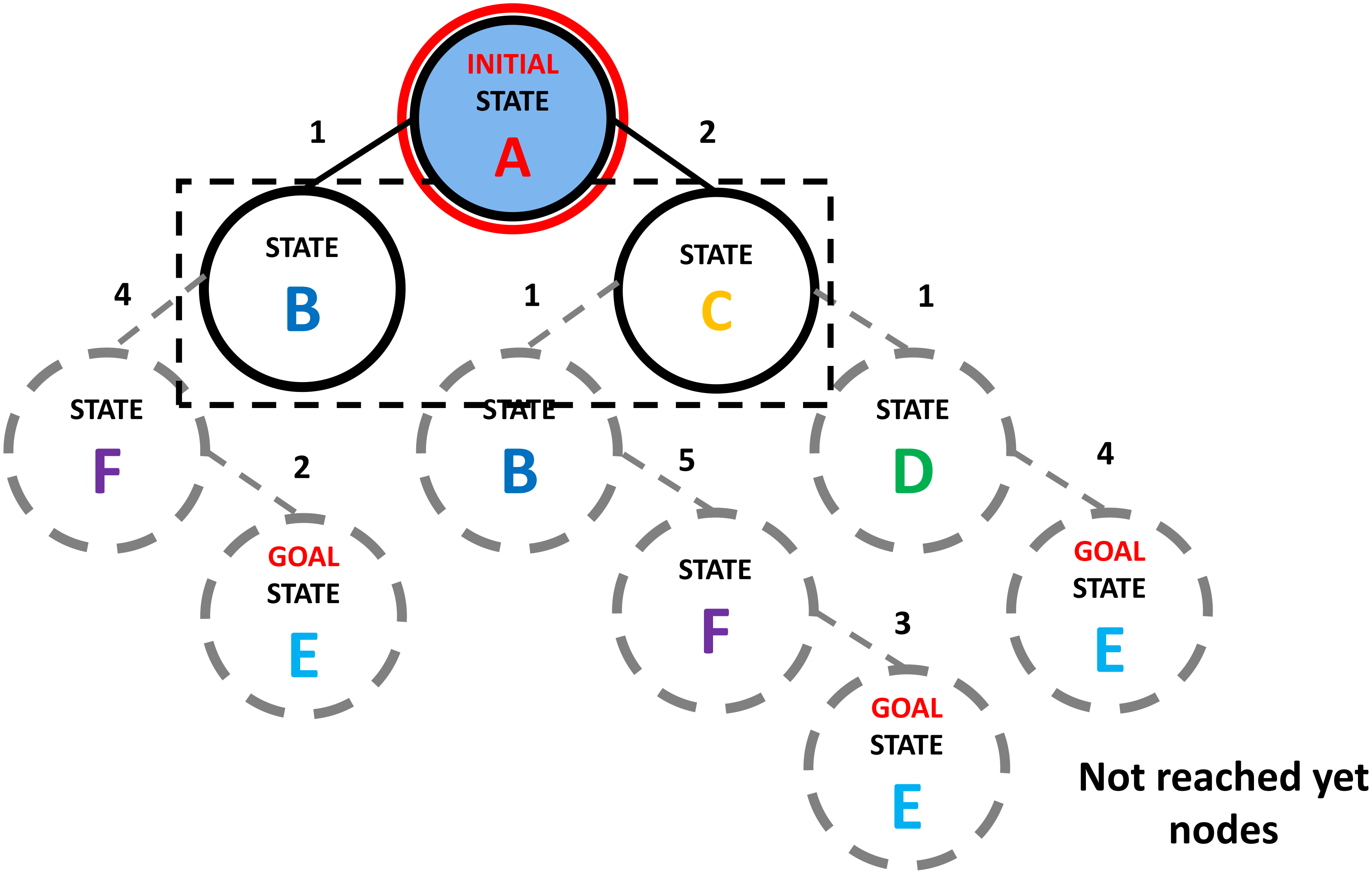
Shortest path problem:
Given a weighted graph $G(V, E, w)$ and two vertices a, b in V, find the shortest path between vertices a and b (**all edge weights are equal**).

BFS and UCS: Pseudocode

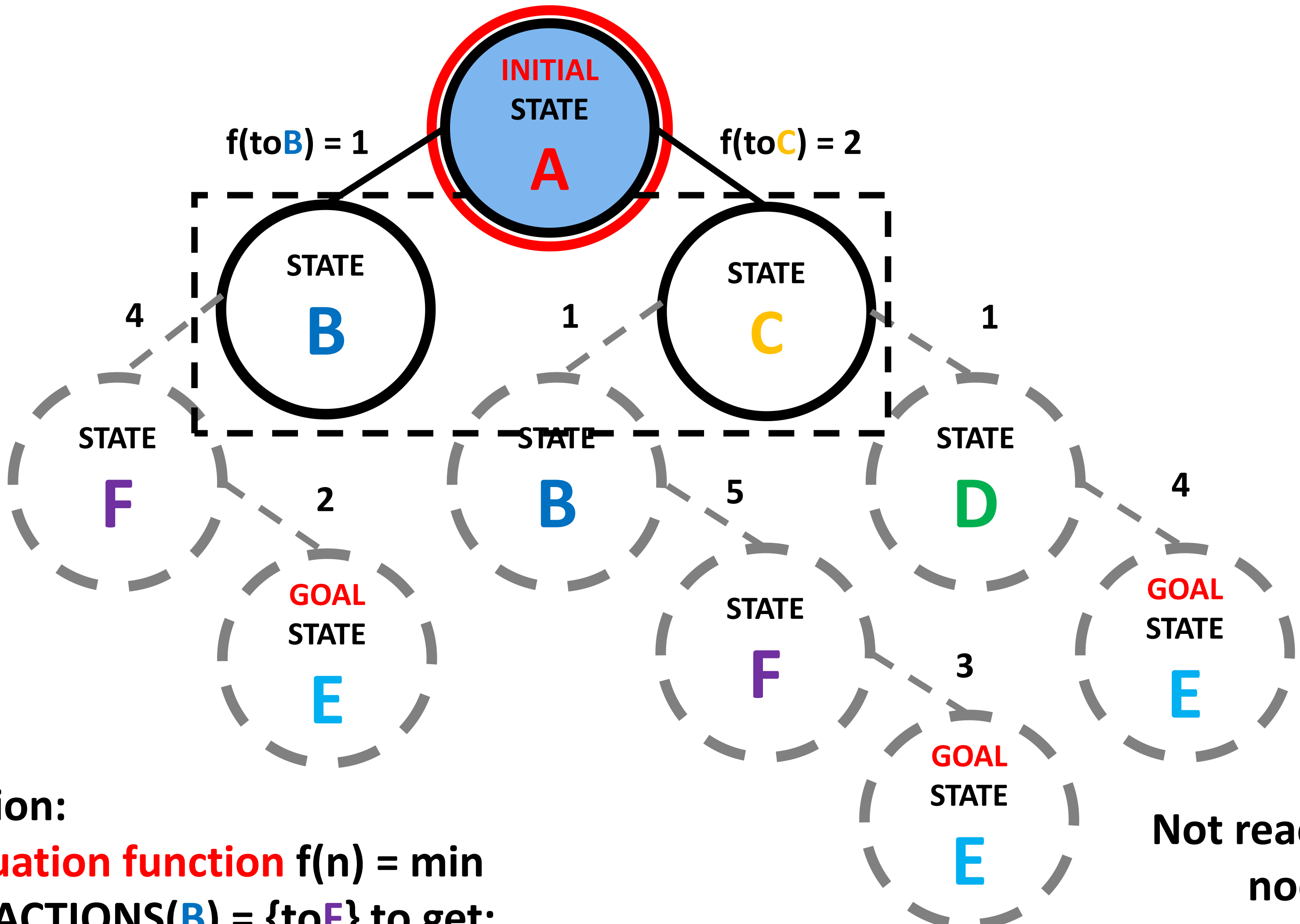
```
function BREADTH-FIRST-SEARCH(problem) returns a solution node or failure
  node ← NODE(problem.INITIAL)
  if problem.IS-GOAL(node.STATE) then return node
  frontier ← a FIFO queue, with node as an element
  reached ← {problem.INITIAL}
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if problem.IS-GOAL(s) then return child
      if s is not in reached then
        add s to reached
        add child to frontier
  return failure

function UNIFORM-COST-SEARCH(problem) returns a solution node, or failure
  return BEST-FIRST-SEARCH(problem, PATH-COST)
```

Search Tree: Variable Action Cost



Search Tree: Variable Action Cost

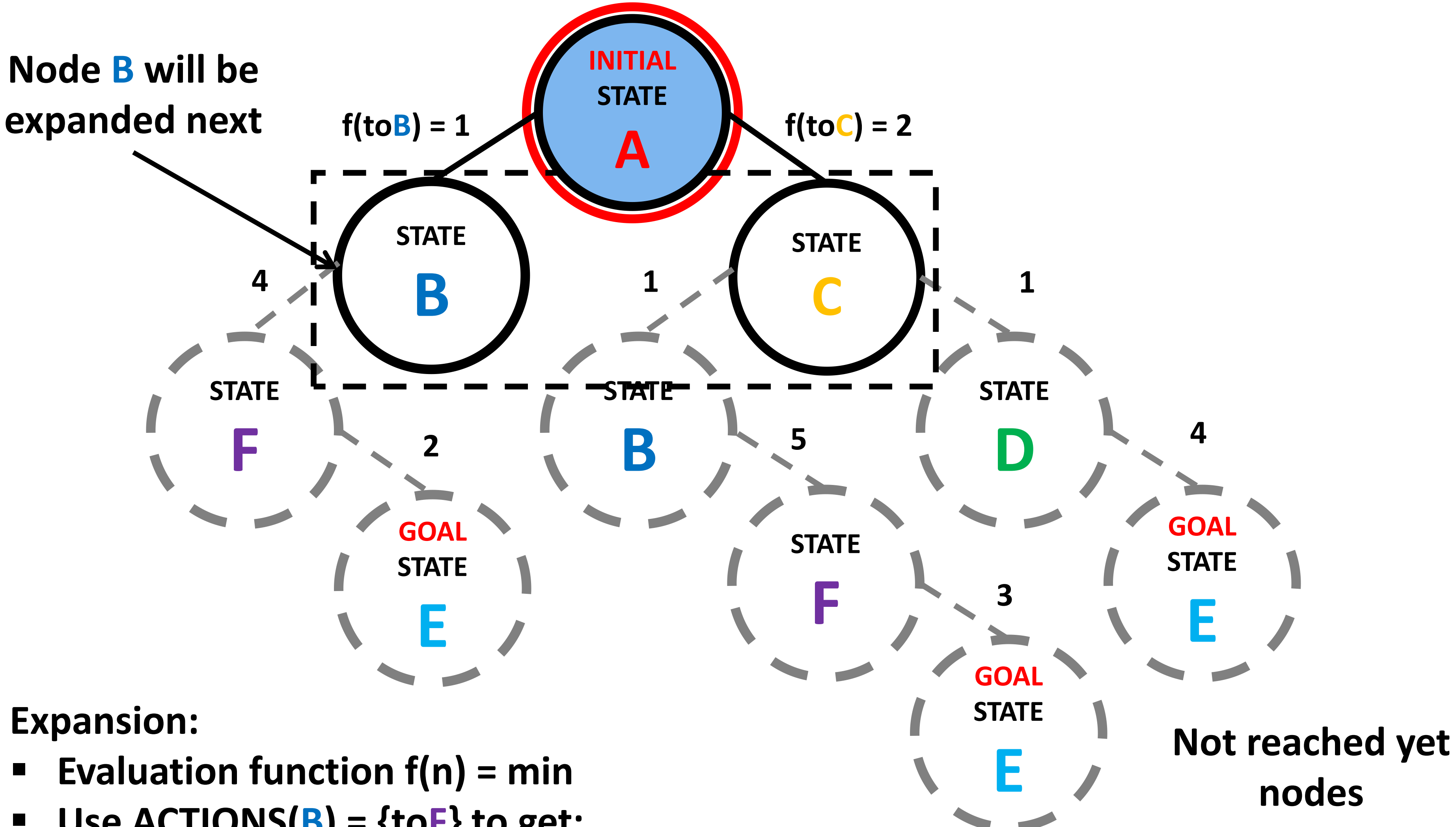


Expansion:

- Evaluation function f(n) = min
- Use ACTIONS(B) = {toF} to get:
- RESULT(B, toF) = F

Not reached yet nodes

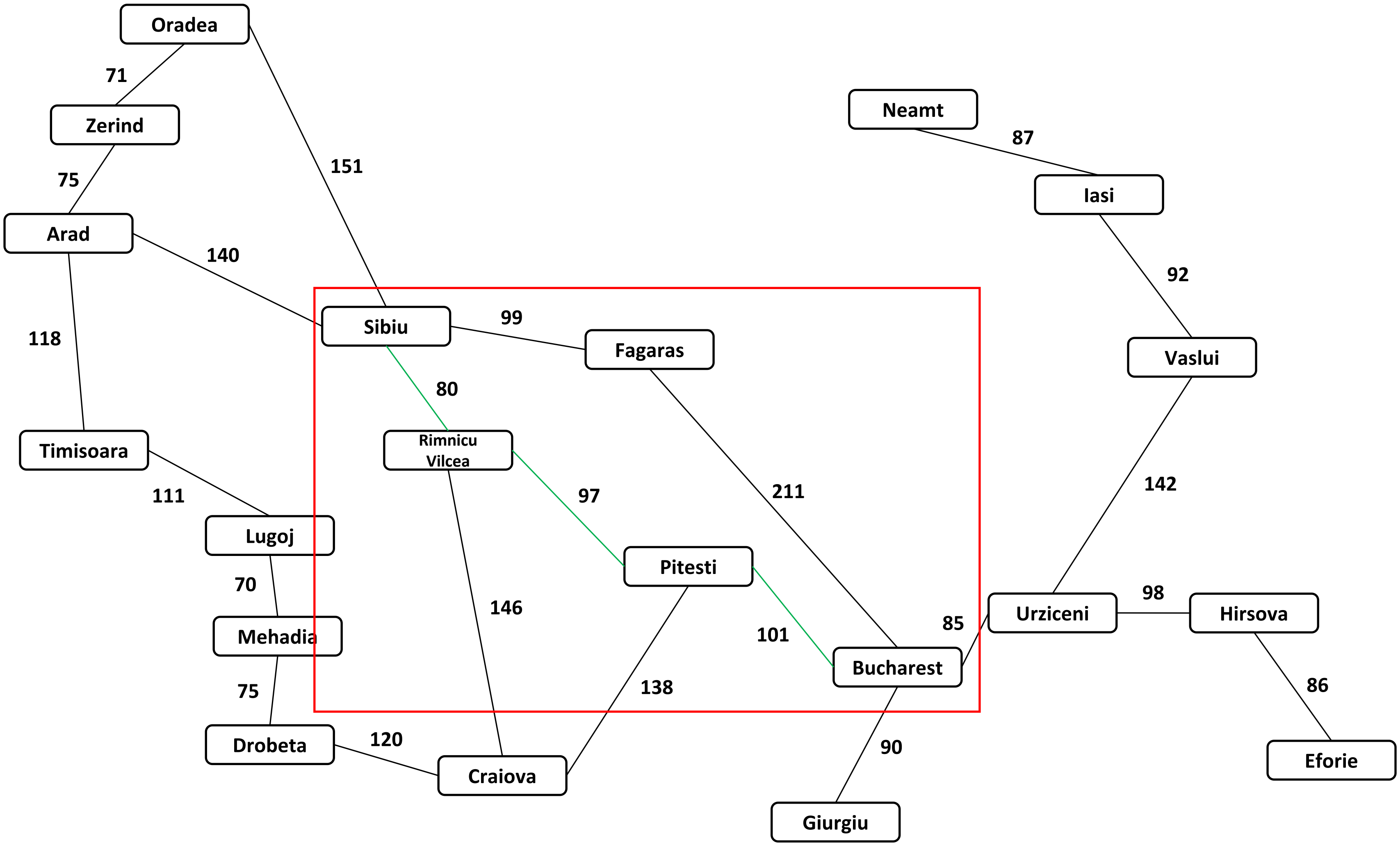
Search Tree: “Take Best First”



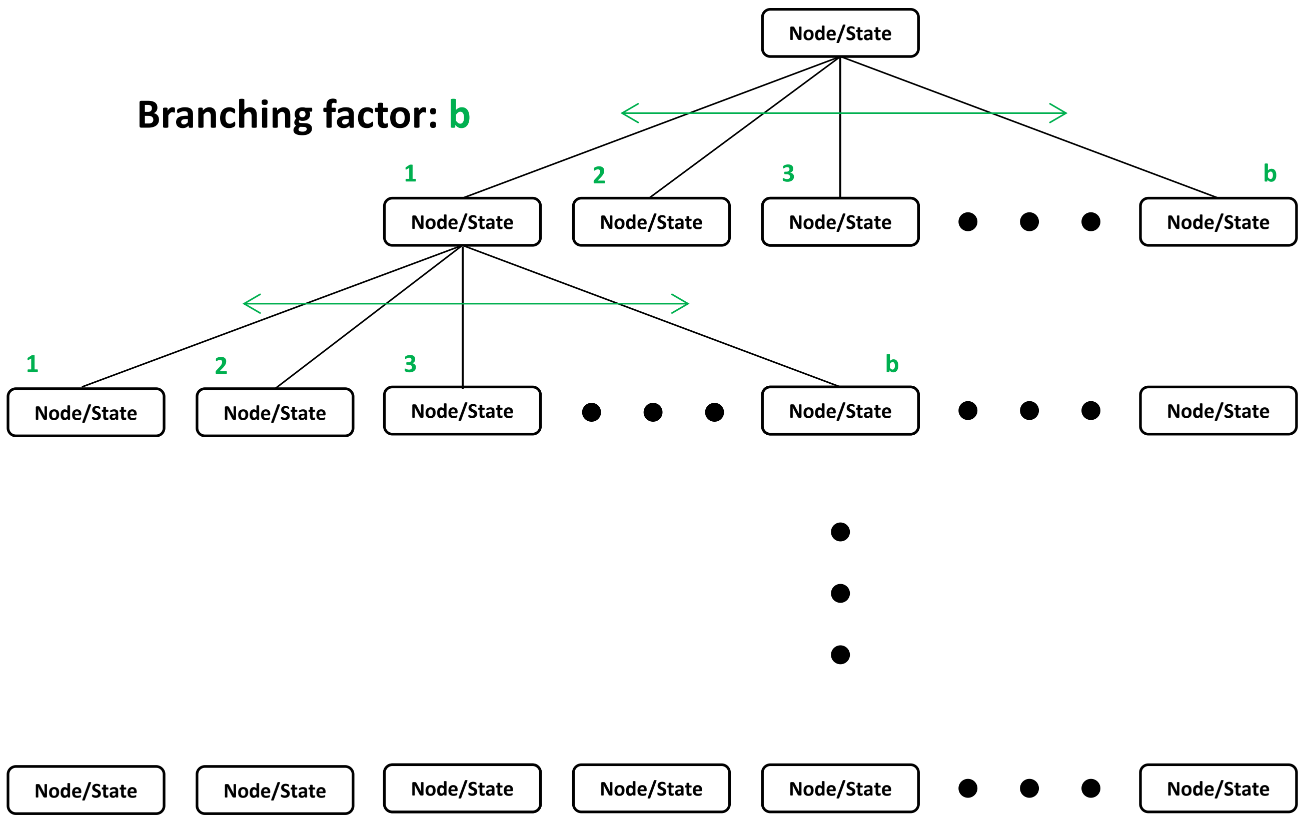
Expansion:

- Evaluation function f(n) = min
- Use ACTIONS(B) = {toF} to get:
- RESULT(B, toF) = F

“Take Best First” Search: Issue



Let’s Go Back to Depth First Search



Depth: 0 | $N_0 = 1$

Depth: 1 | $N_1 = b$

Depth: 2 | $N_2 = b^2$

•
•
•

Depth: d | $N_d = b^d$

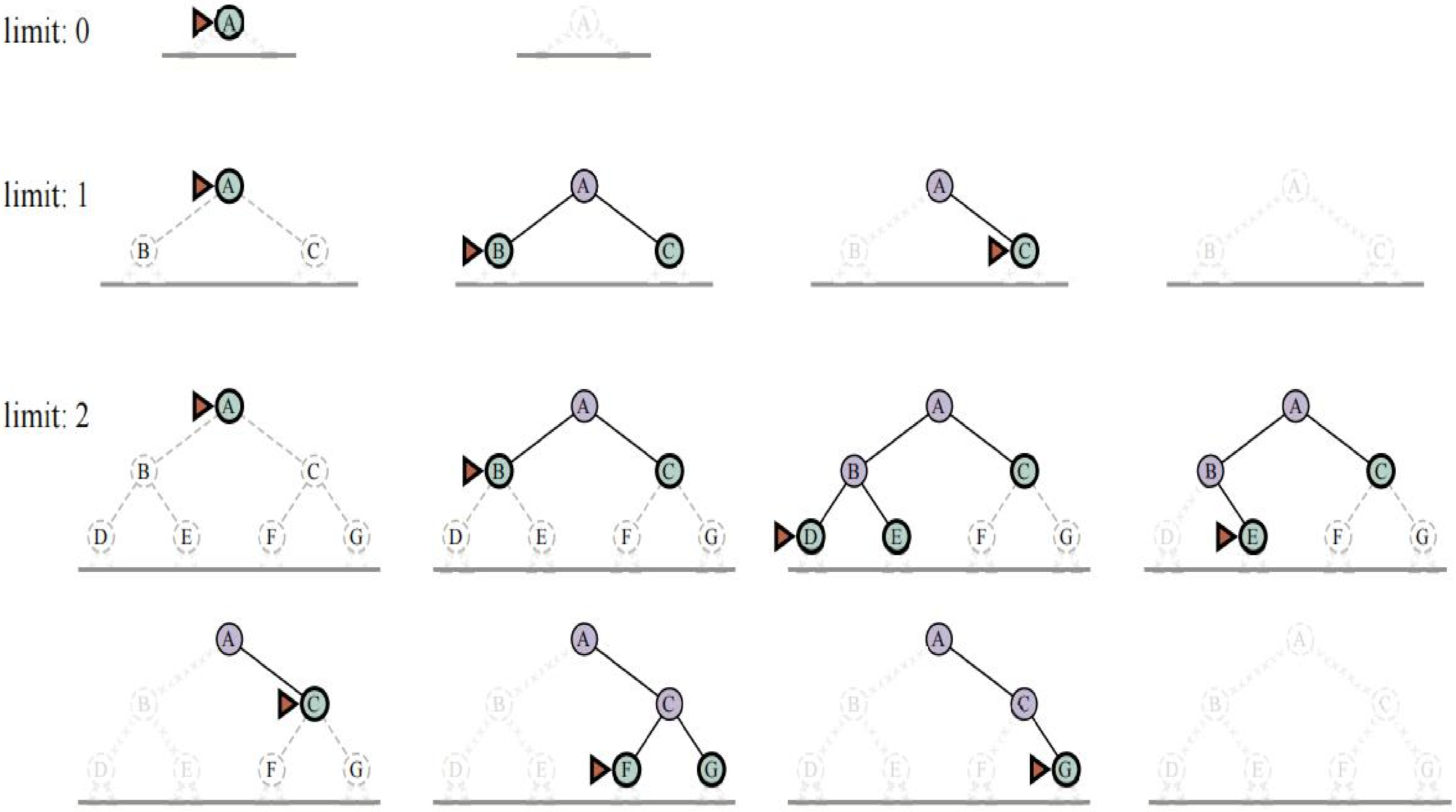
Tree depth is an issue!

“Controlled” DFS: Pseudocode

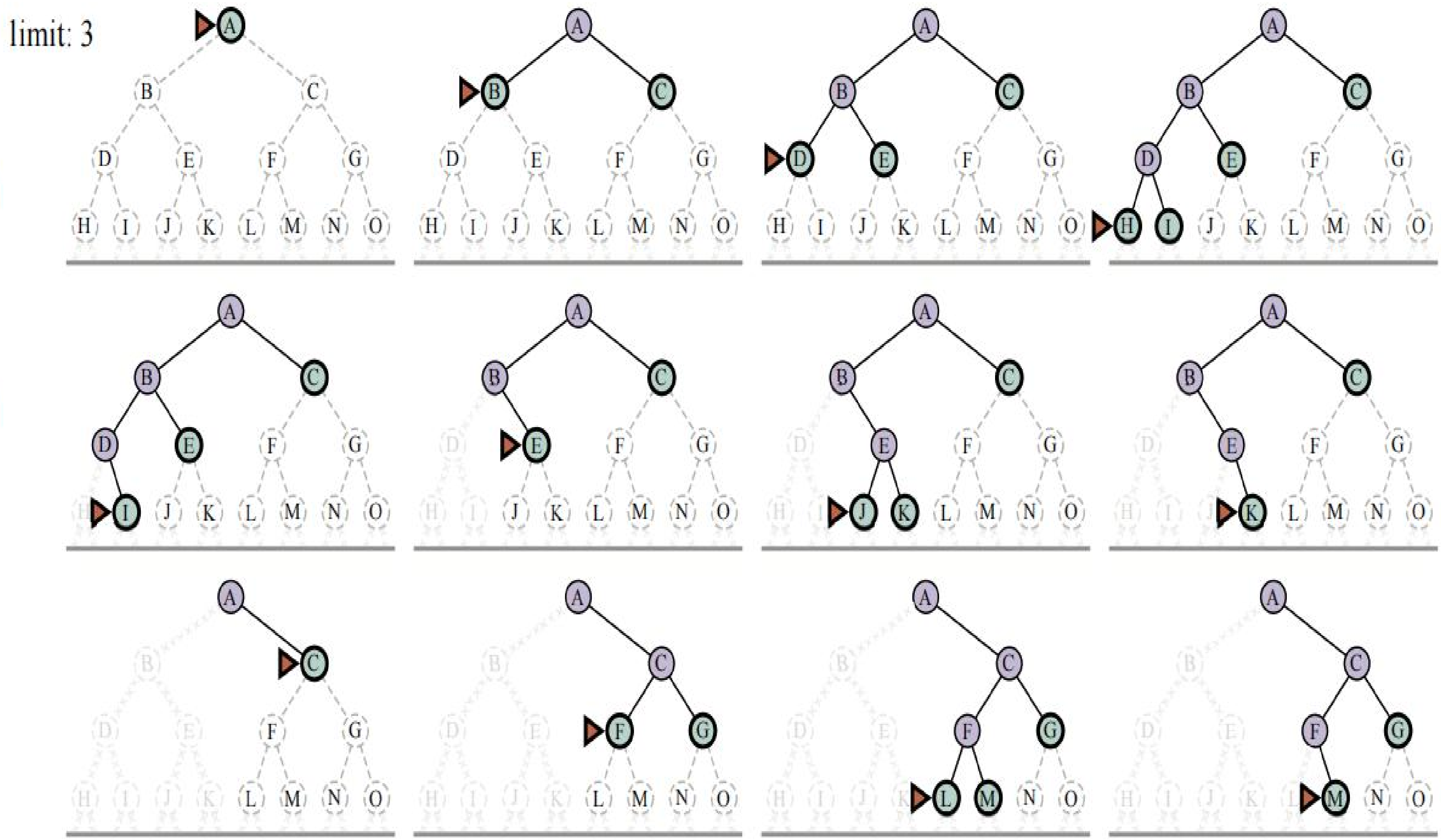
```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution node or failure
  for depth = 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
    if result ≠ cutoff then return result

function DEPTH-LIMITED-SEARCH(problem, ℓ) returns a node or failure or cutoff
  frontier ← a LIFO queue (stack) with NODE(problem.INITIAL) as an element
  result ← failure
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    if DEPTH(node) > ℓ then
      result ← cutoff
    else if not IS-CYCLE(node) do
      for each child in EXPAND(problem, node) do
        add child to frontier
  return result
```

Iterative Deepening DFS: Illustration



Iterative Deepening DFS: Illustration



Uninformed Search

- **Traverse the search tree, possibly through all legal paths, to find a solution / reach goal state**
- **Search tree size can be large (or infinite)**
  - **Use node expansion / generation as you traverse**
- **Avoid repeated states (those cause "loops")**
  - **Keep track of already visited states**
- **Search tree depth can be a challenge**
  - **Use Iterative Deepening or Depth Limits**
- **Extra problem information? → informed search**

Uninformed Search Algorithms

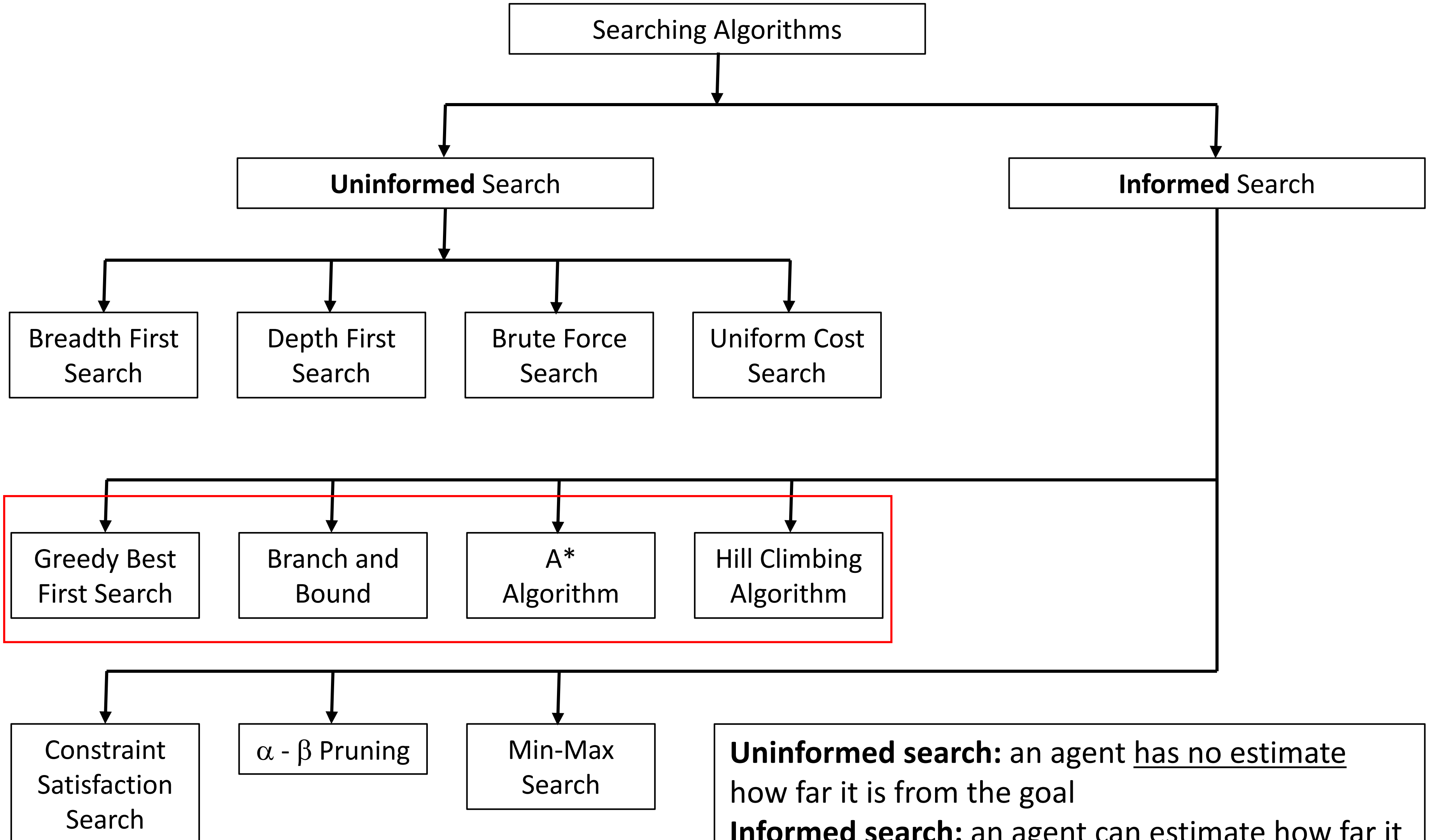
Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[1]	Yes[1,2]	No	No	Yes[1]	Yes[1,4]
Optimal cost?	Yes[3]	Yes	No	No	Yes[3]	Yes[3,4]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

Figure 3.15 Evaluation of search algorithms. b is the branching factor; m is the maximum depth of the search tree; d is the depth of the shallowest solution, or is m when there is no solution; ℓ is the depth limit. Superscript caveats are as follows: [1] complete if b is finite, and the state space either has a solution or is finite. [2] complete if all action costs are $\geq \epsilon > 0$; [3] cost-optimal if action costs are all identical; [4] if both directions are breadth-first or uniform-cost.

DFS vs. BFS: When to Use? Tips

- **Breadth First Search**
  - **branching factor b is not excessive**
  - **solution is expected to exist at a reasonable level (depth d is reasonable)**
  - **some search paths can be very deep**
- **Depth First Search**
  - **branching factor b is relatively large**
  - **solution is expected to exist at a relatively shallow level (depth d is low)**
  - **search paths are not excessively deep**

Selected Searching Algorithms



Uninformed search: an agent has no estimate how far it is from the goal
Informed search: an agent can estimate how far it is from the goal

Informed Search: the Idea

When traversing the search tree use domain knowledge / heuristics to avoid search paths that are likely to be fruitless

Informed Search and Heuristics

Informed search relies on domain-specific knowledge / hints that help locate the goal state.

$$h(n) = h(\text{State } n)$$

$$h(n) = n(\text{relevant information about State } n)$$

h(n) : heuristic function - estimated cost of the cheapest path from State n to the goal state

Evaluation function

Calculate / obtain:

f(n) = f(State n)
f(n) = f(relevant information about State n)

A state n with minimum (or maximum) f(n) should be chosen for expansion
What about ties?

Best-First Search

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure
```

Best-First Search is really a class of search algorithms that:

- **Use the evaluation function f(n) to pick next action**
- **Keep track of visited states**
- **Keep track of frontier states**
- **Evaluation function f(n) choice controls their behavior**

Informed Search and Heuristics

Informed search relies on domain-specific knowledge / hints that help locate the goal state.

$$h(n) = h(\text{State } n)$$

$$h(n) = n(\text{relevant information about State } n)$$

h(n) : heuristic function - estimated cost of the cheapest path from State n to the goal state

Hill Climbing Search

- **The most primitive informed search approach**
  - **a naive greedy algorithm**
  - **evaluation function: the cost of next move**
  - **does not care about the "bigger picture" (for example: total search path cost)**
- **Practicalities:**
  - **usually does not keep track of search history:**
    - **not tracking visited nodes → loops!**
    - **not tracking frontier nodes → does not look at alternatives**

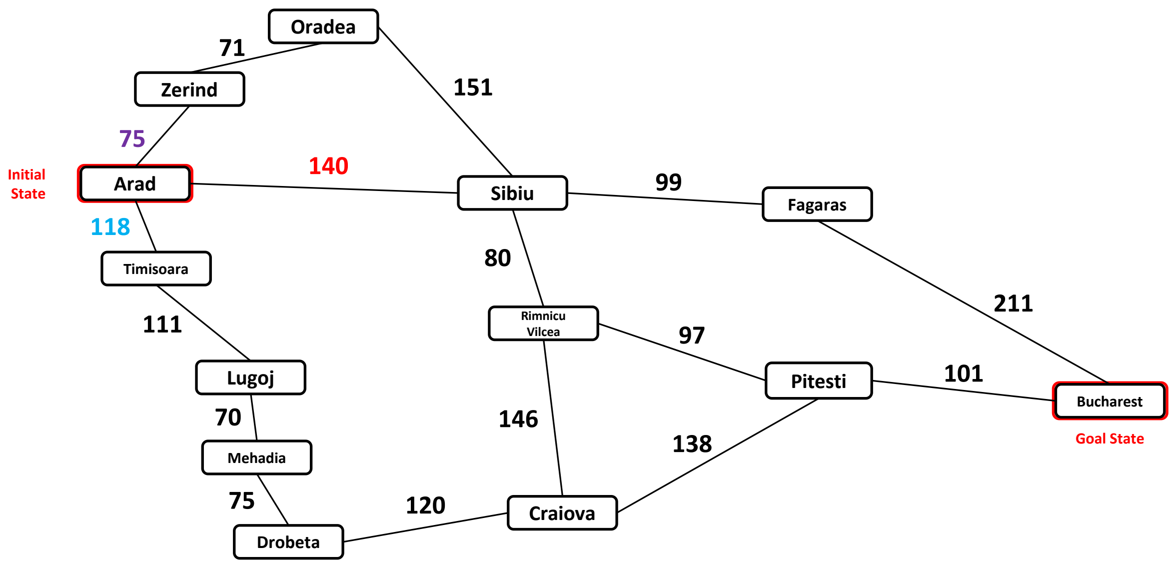
Hill Climbing: Evaluation function

Calculate / obtain:

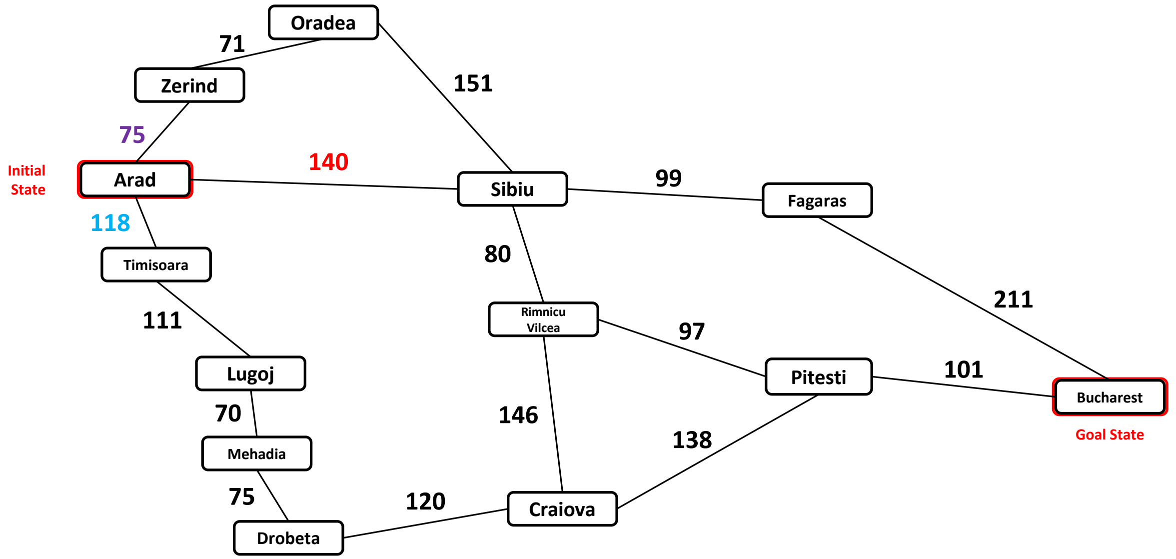
f(n) = **ACTION-COST(State$_a$, toState$_n$, State$_n$)**

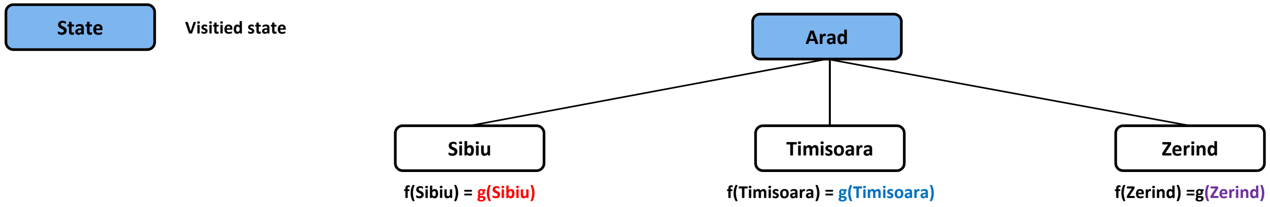
A state n with minimum (or maximum) f(n) should be chosen for expansion

Dracula's Roadtrip: Hill Climbing

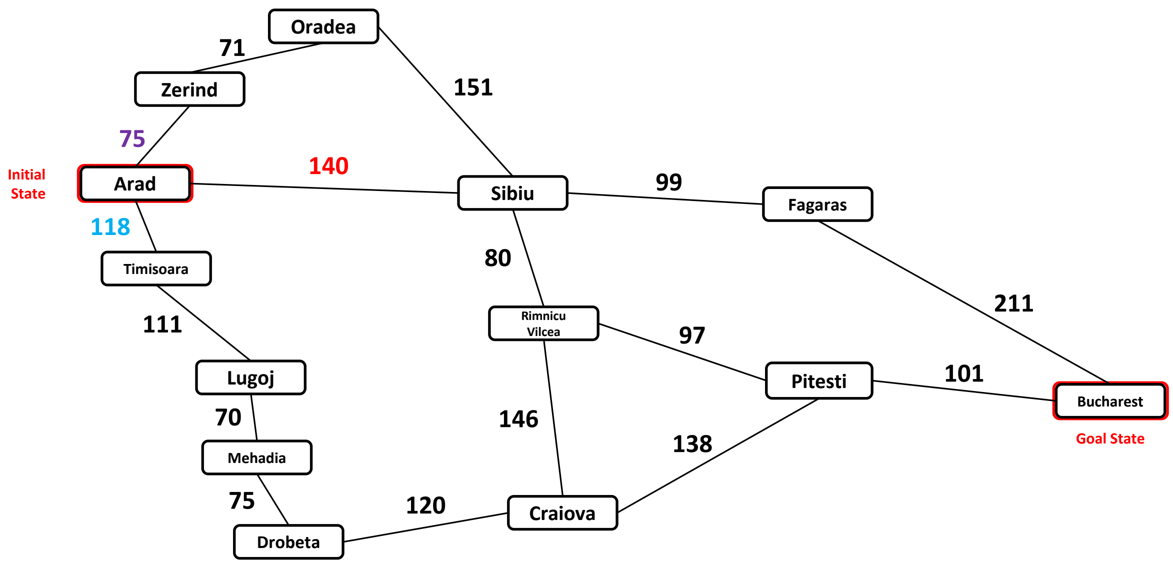


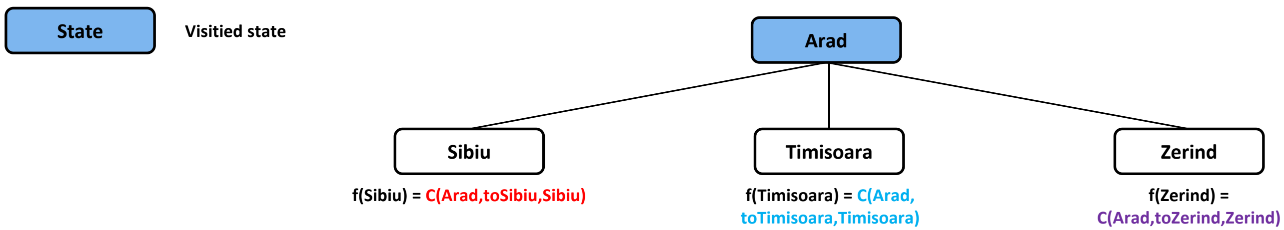
Dracula’s Roadtrip: Hill Climbing



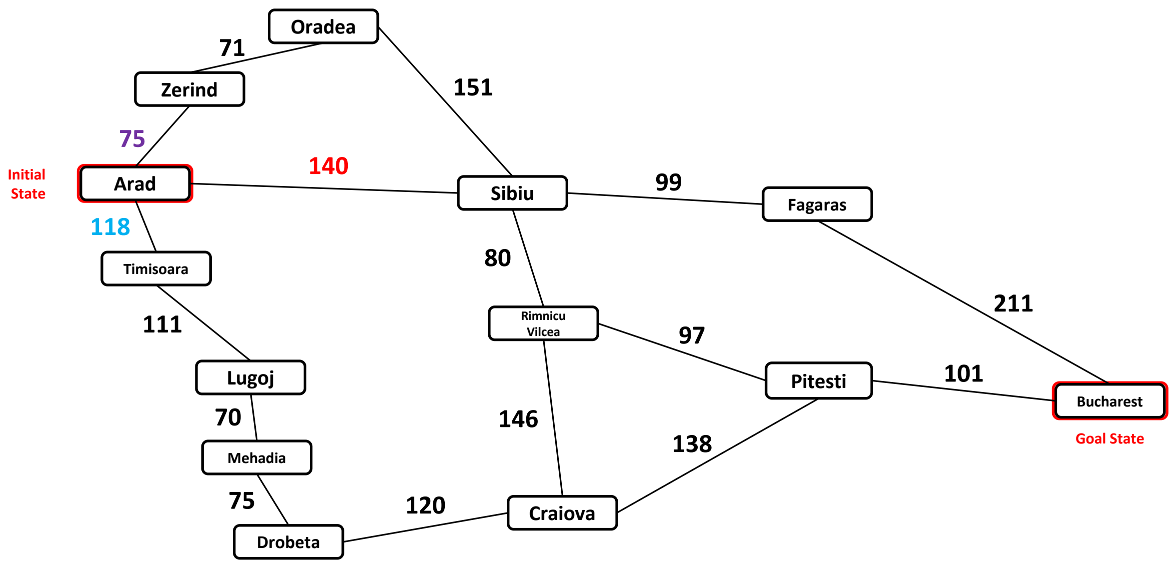


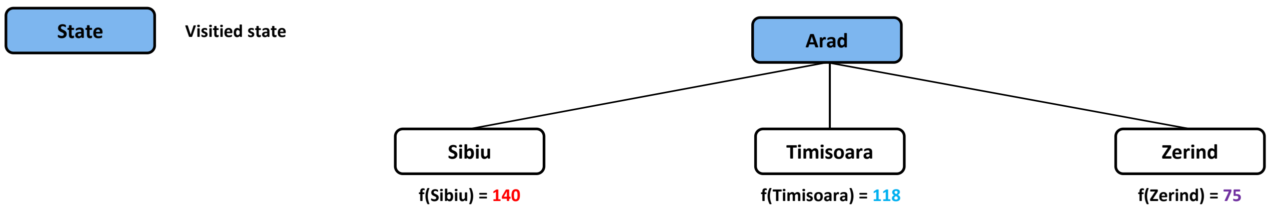
Dracula’s Roadtrip: Hill Climbing



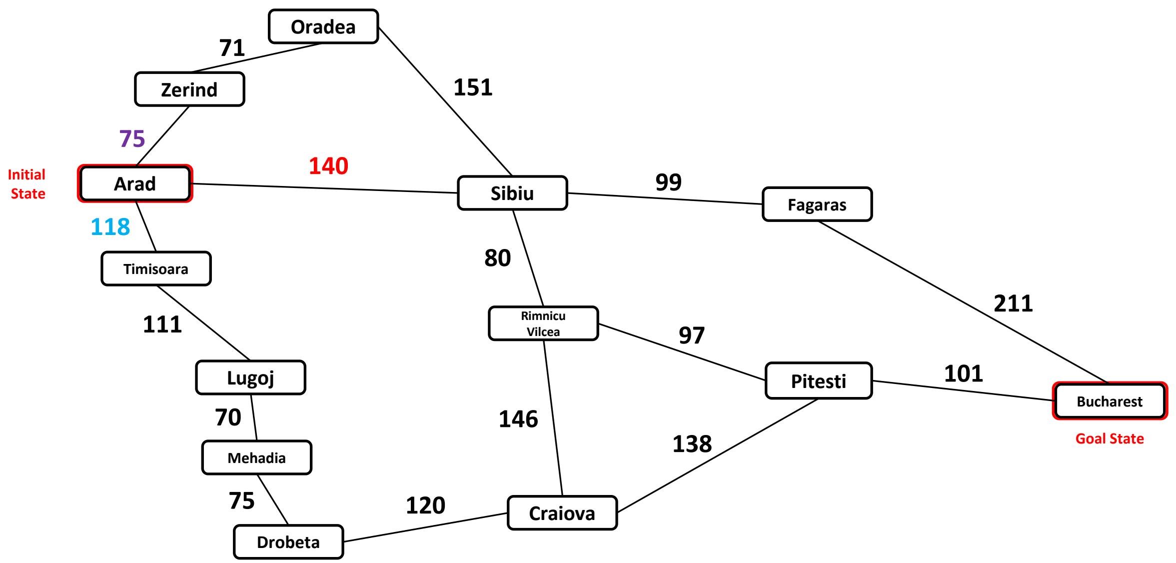


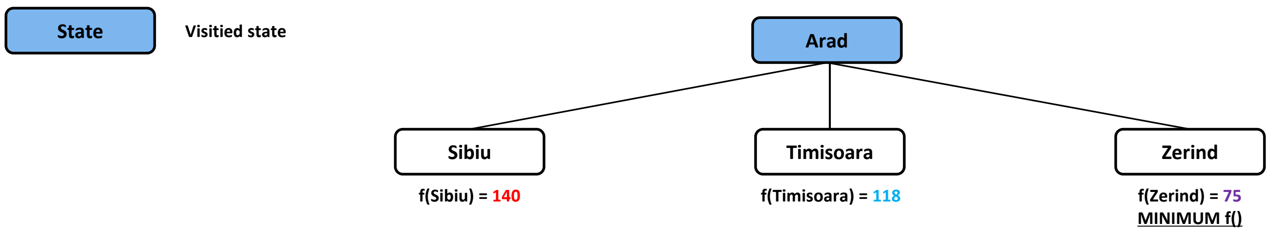
Dracula's Roadtrip: Hill Climbing



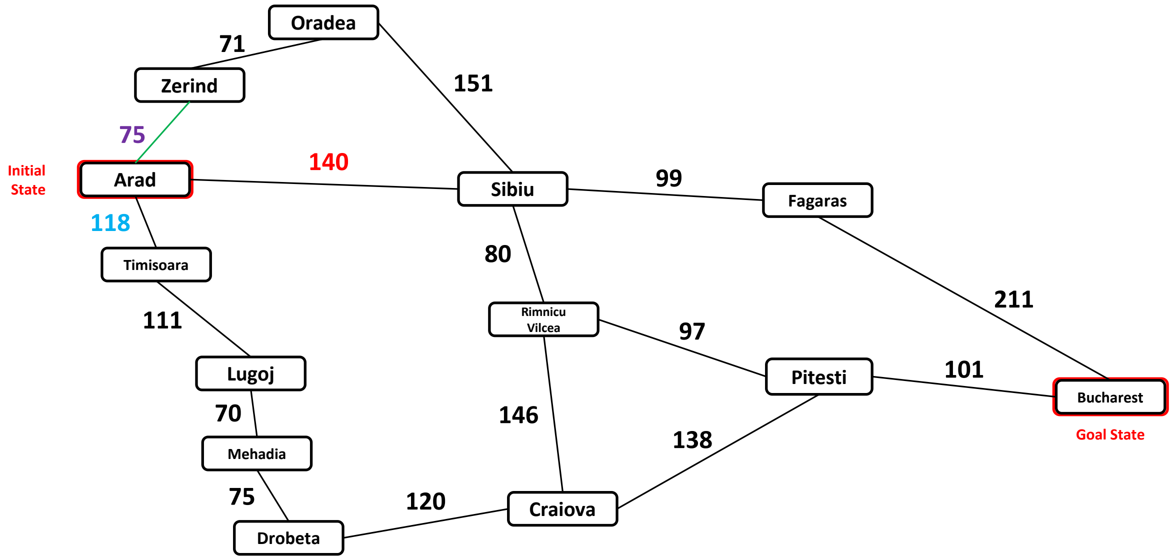


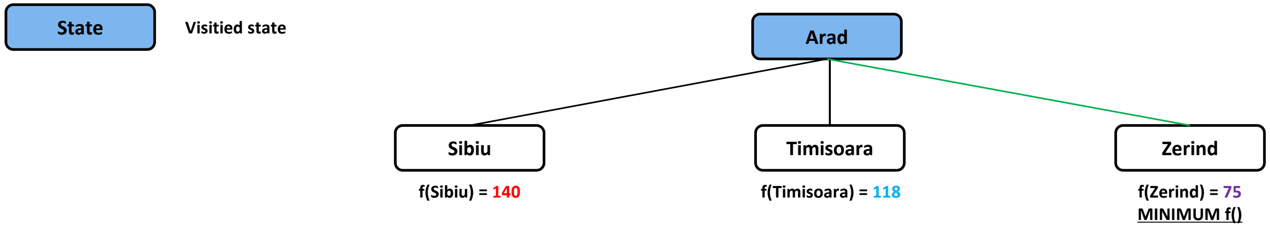
Dracula's Roadtrip: Hill Climbing



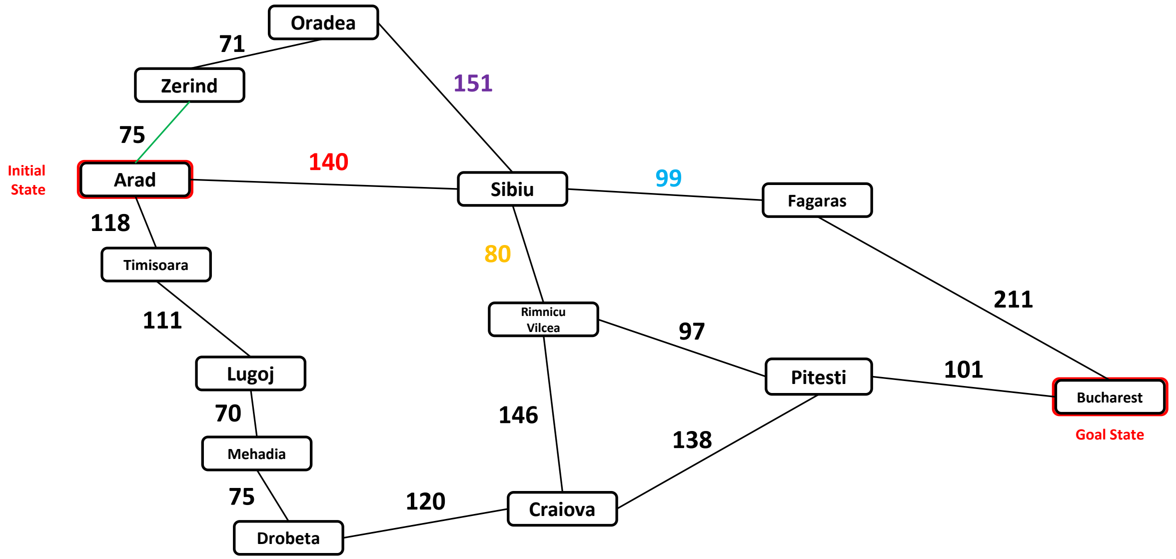


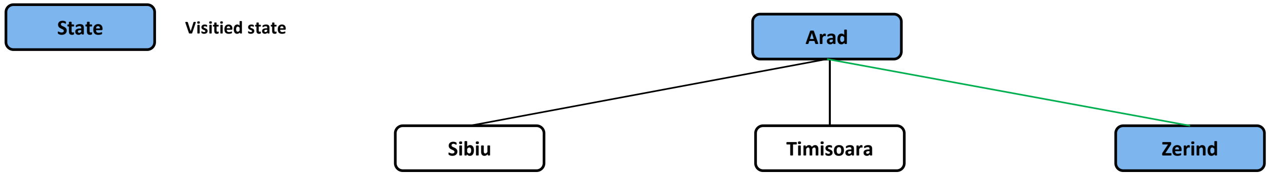
Dracula’s Roadtrip: Hill Climbing



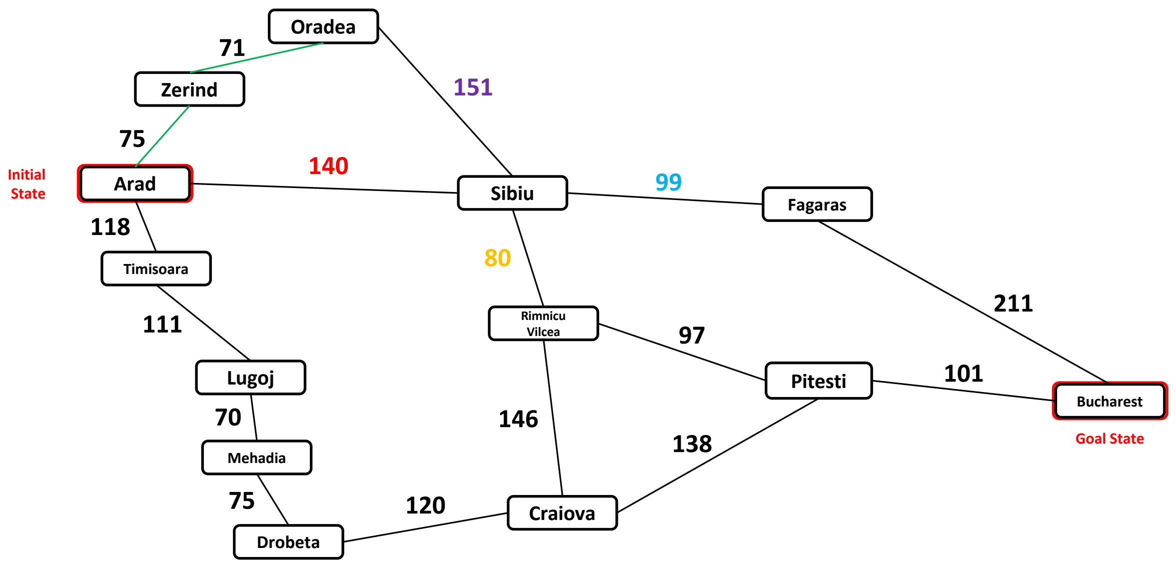


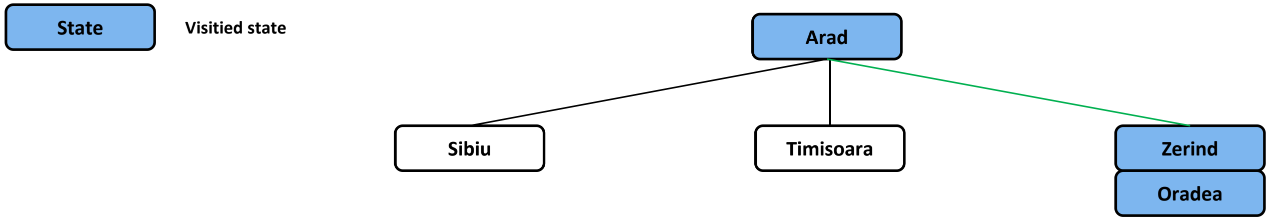
Dracula's Roadtrip: Hill Climbing



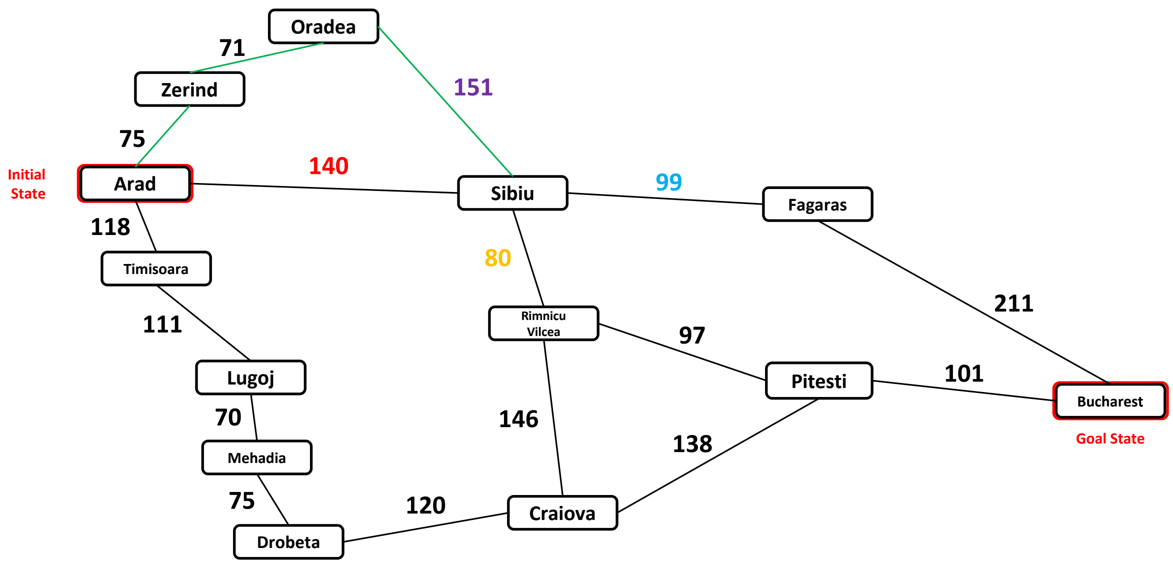


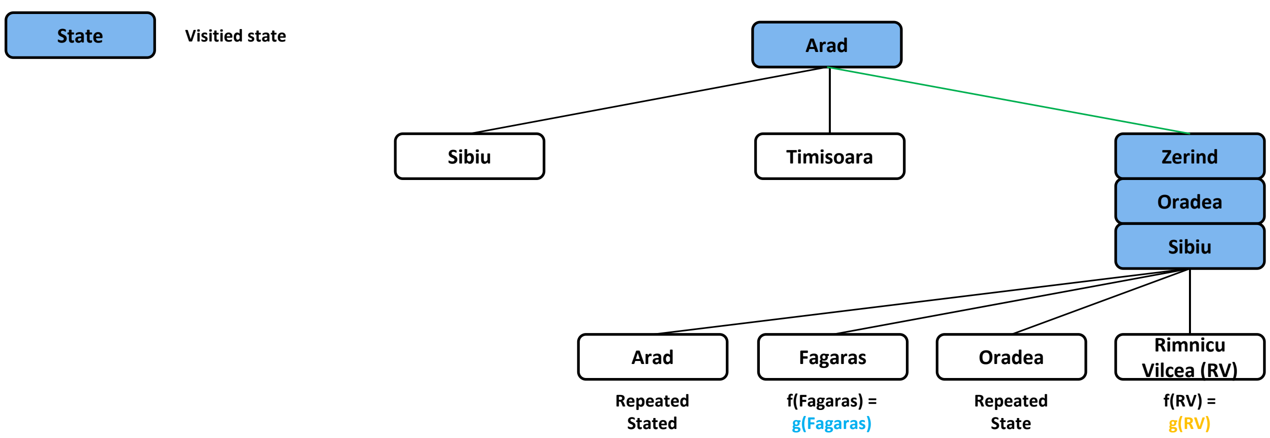
Dracula’s Roadtrip: Hill Climbing



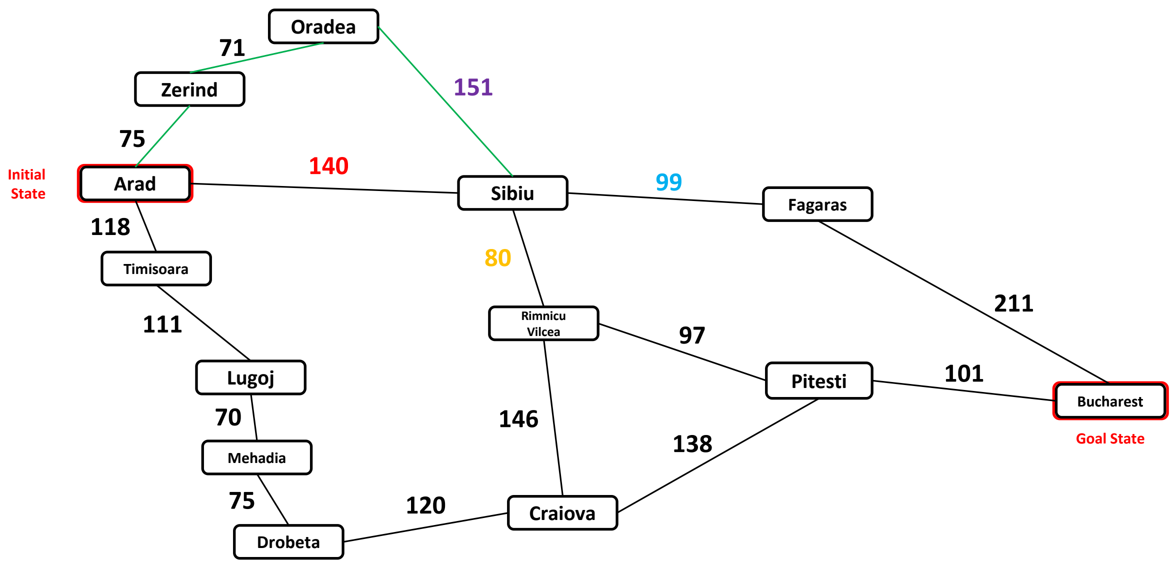


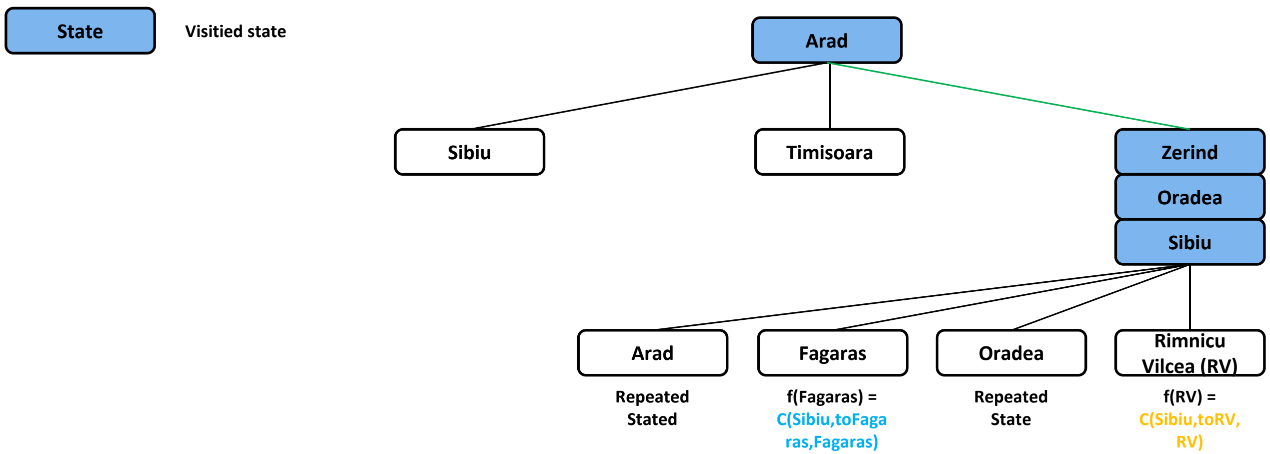
Dracula's Roadtrip: Hill Climbing



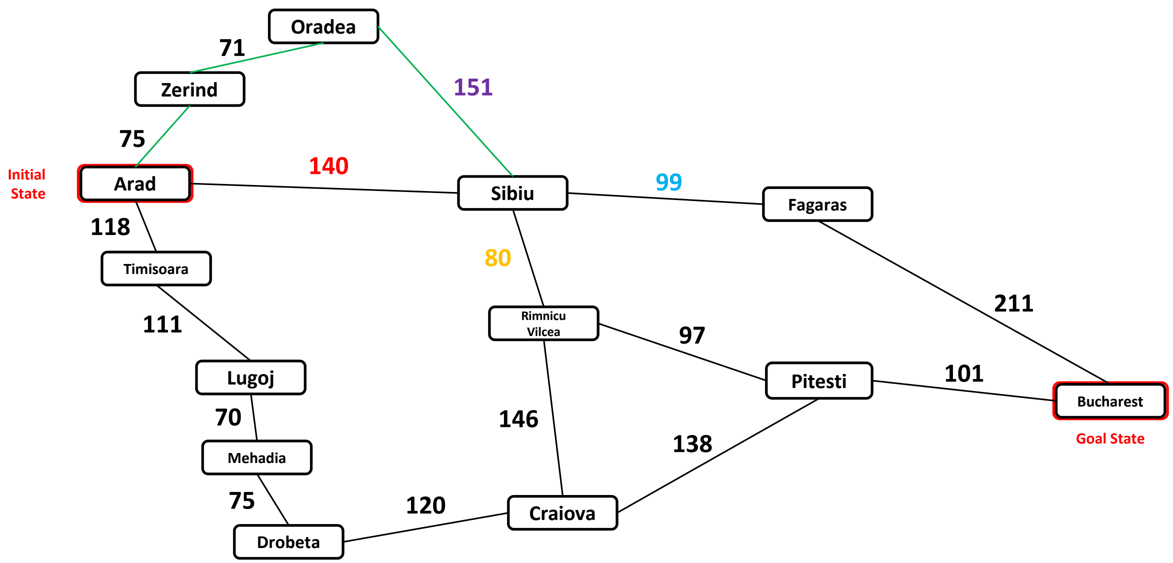


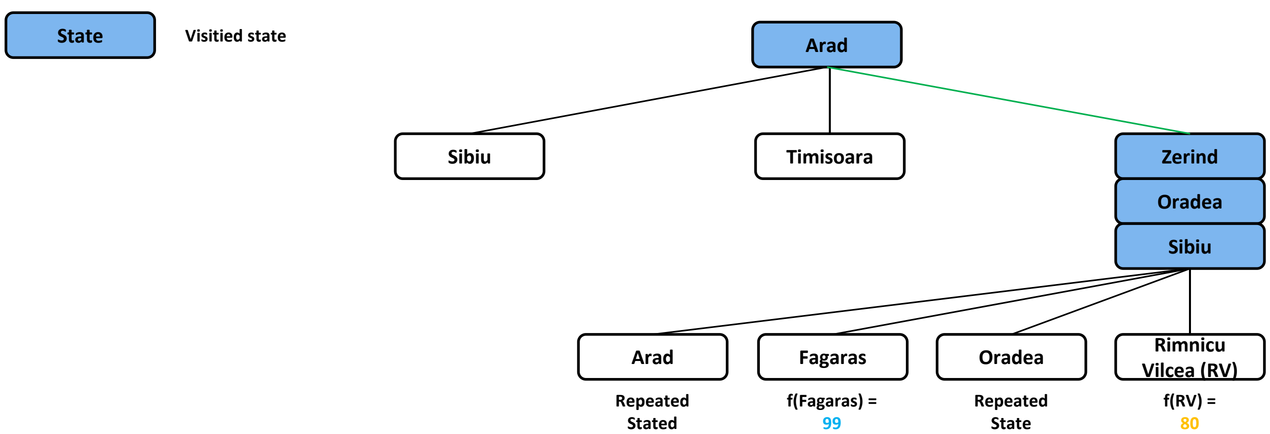
Dracula's Roadtrip: Hill Climbing



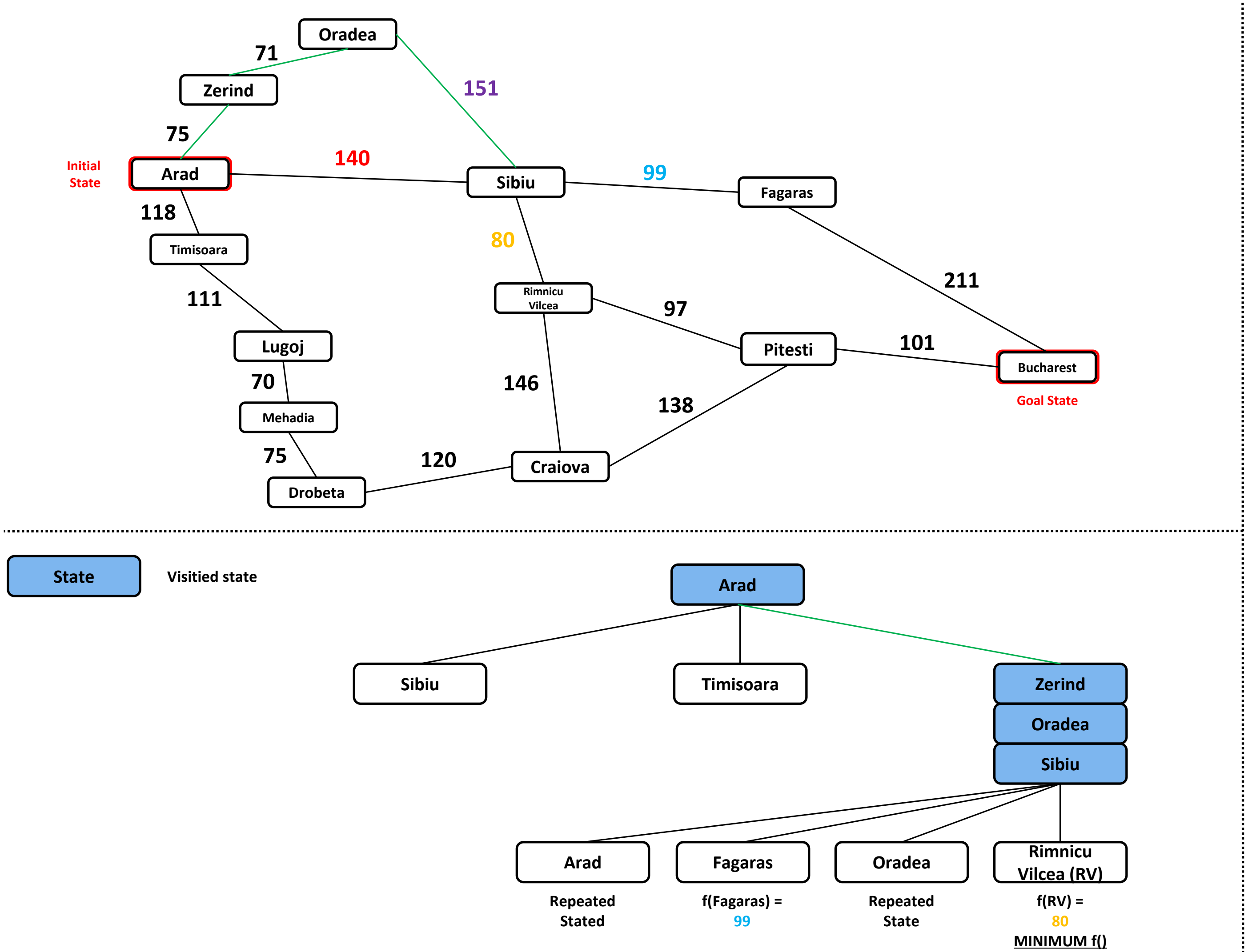


Dracula’s Roadtrip: Hill Climbing

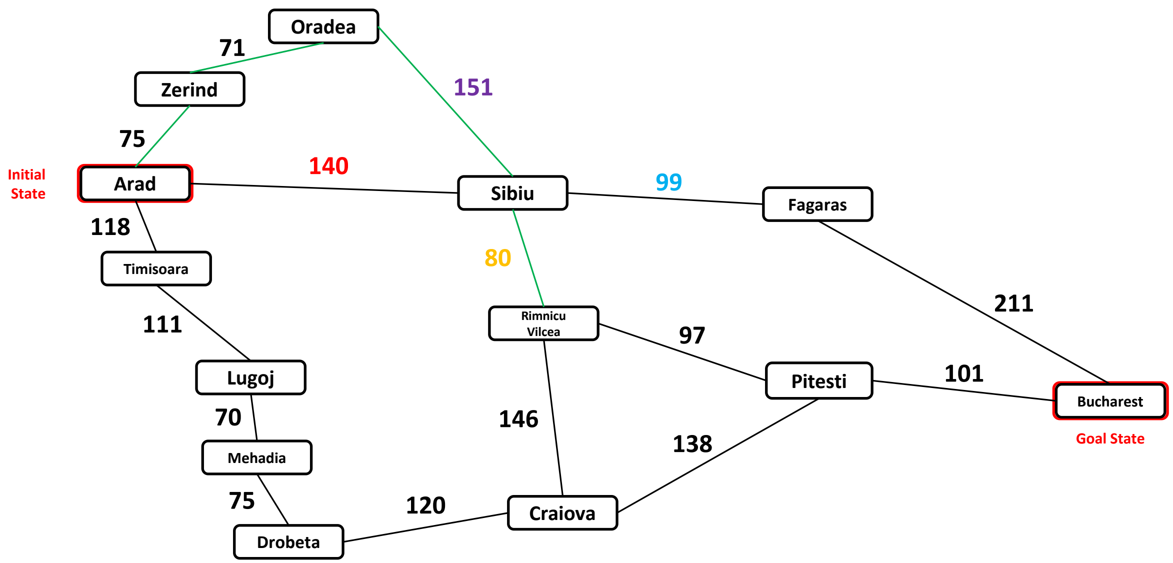


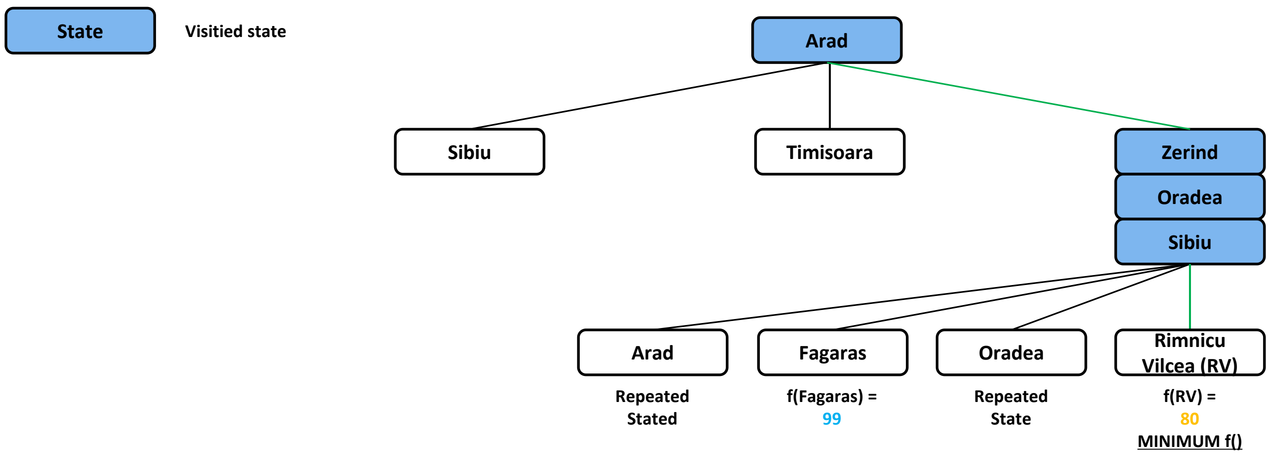


Dracula's Roadtrip: Hill Climbing

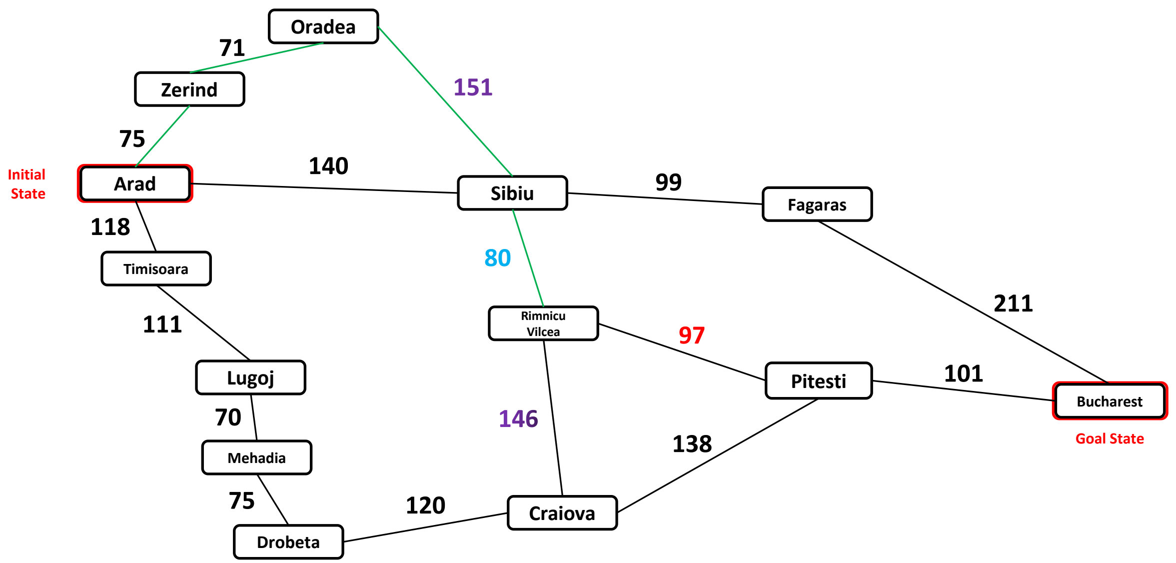


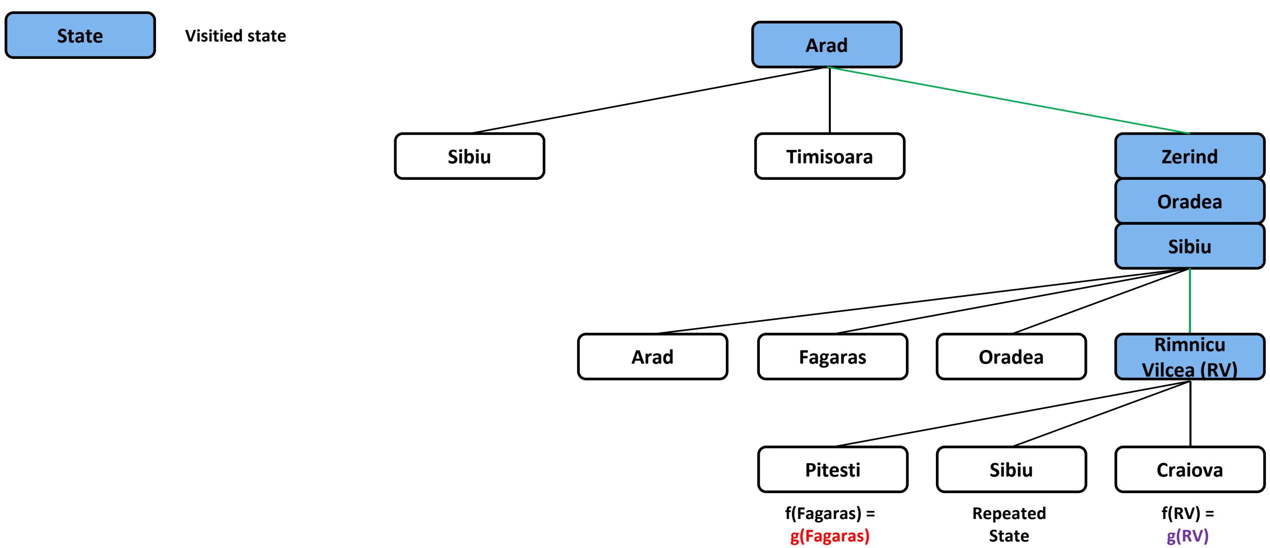
Dracula's Roadtrip: Hill Climbing



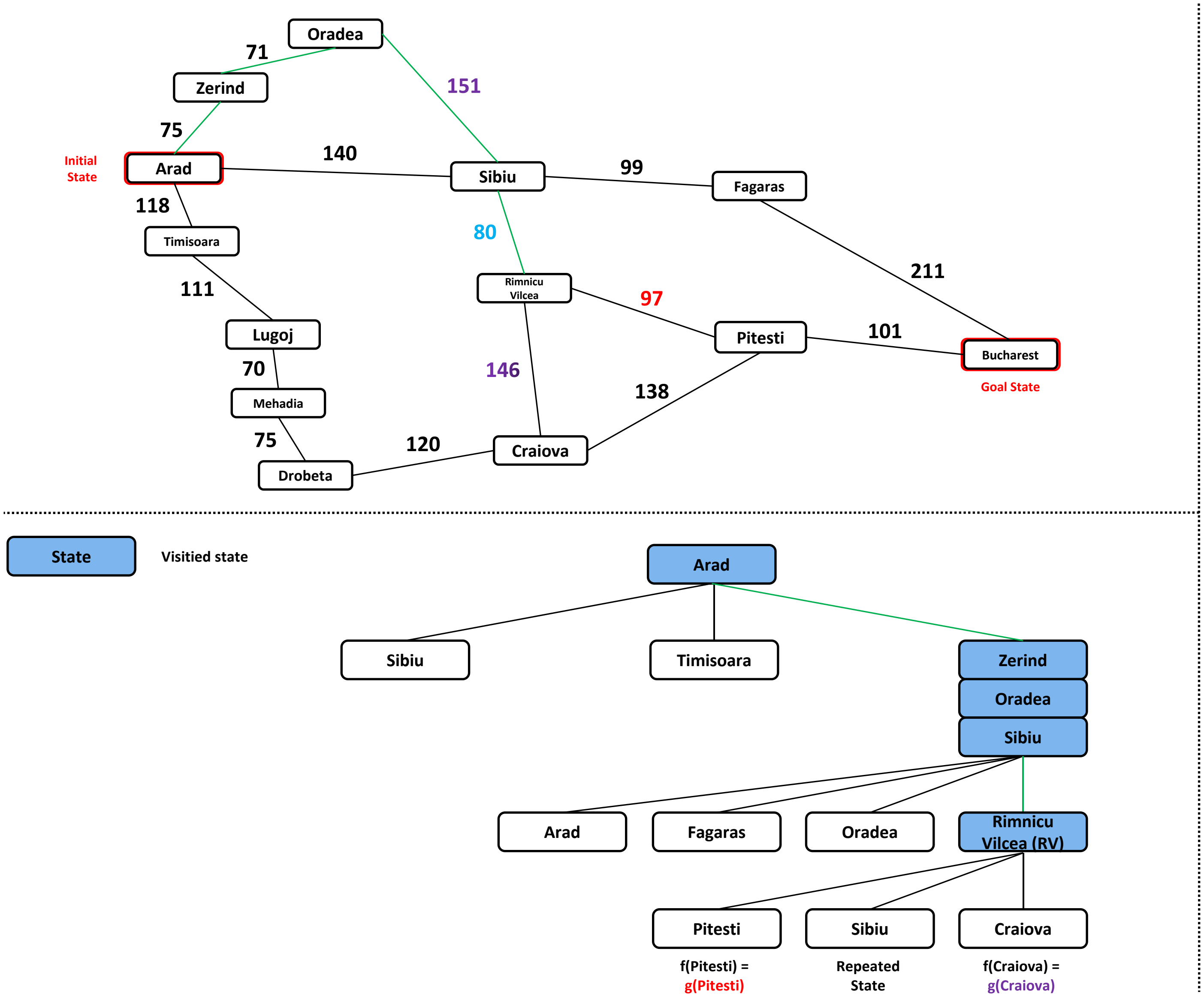


Dracula's Roadtrip: Hill Climbing

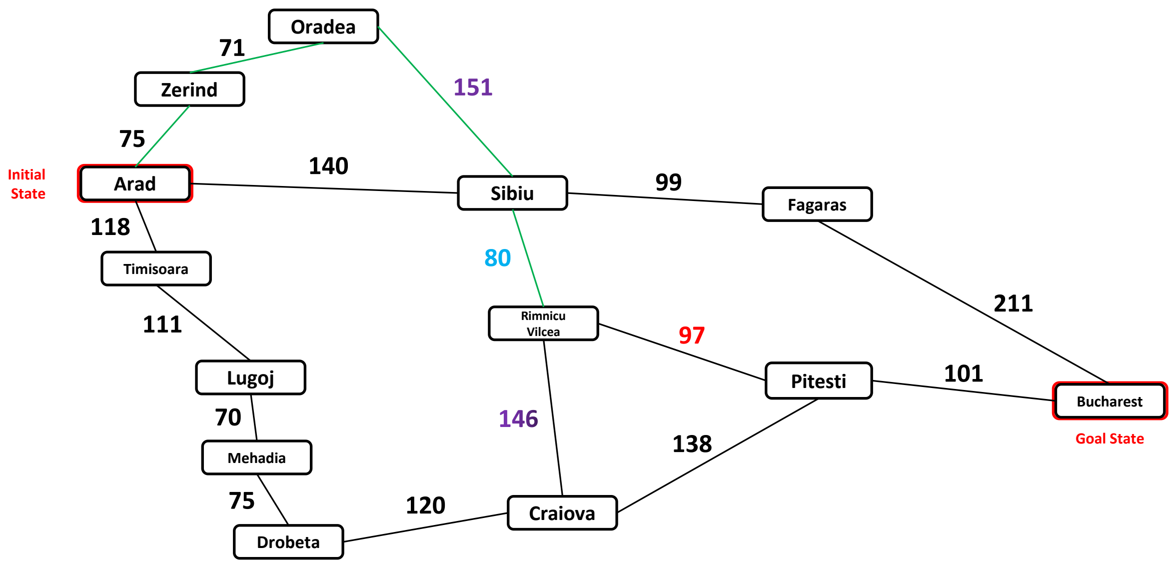


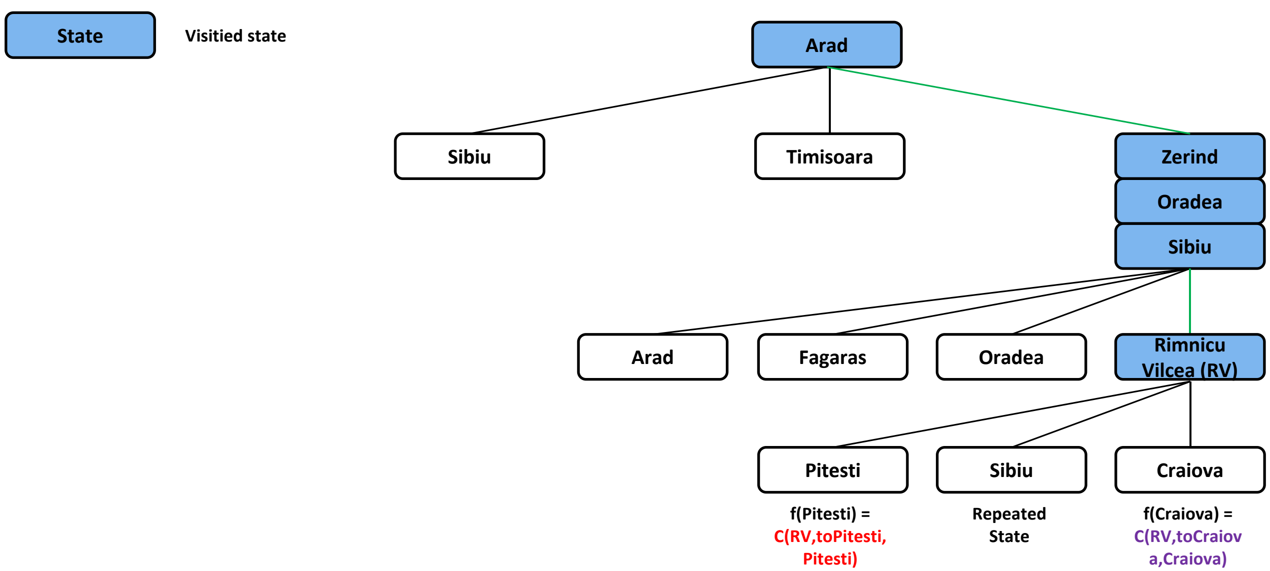


Dracula's Roadtrip: Hill Climbing

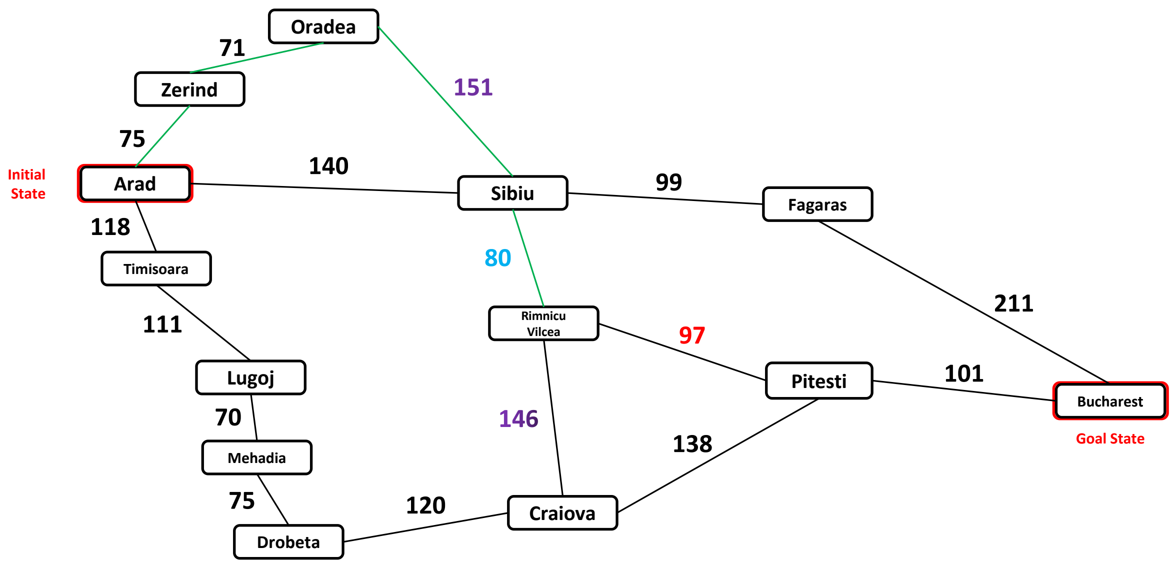


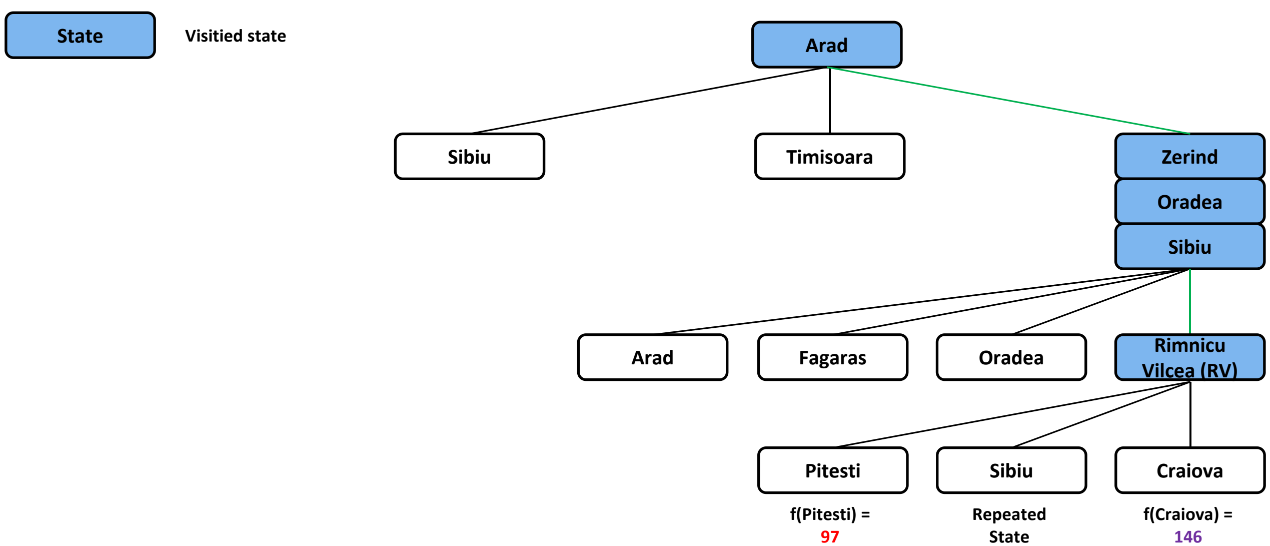
Dracula's Roadtrip: Hill Climbing



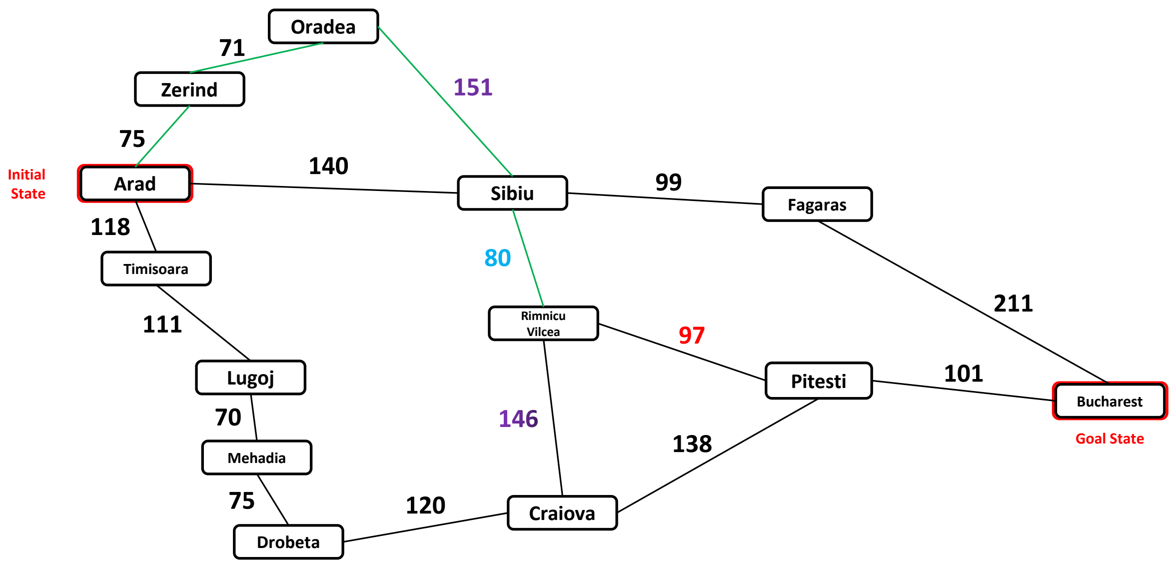


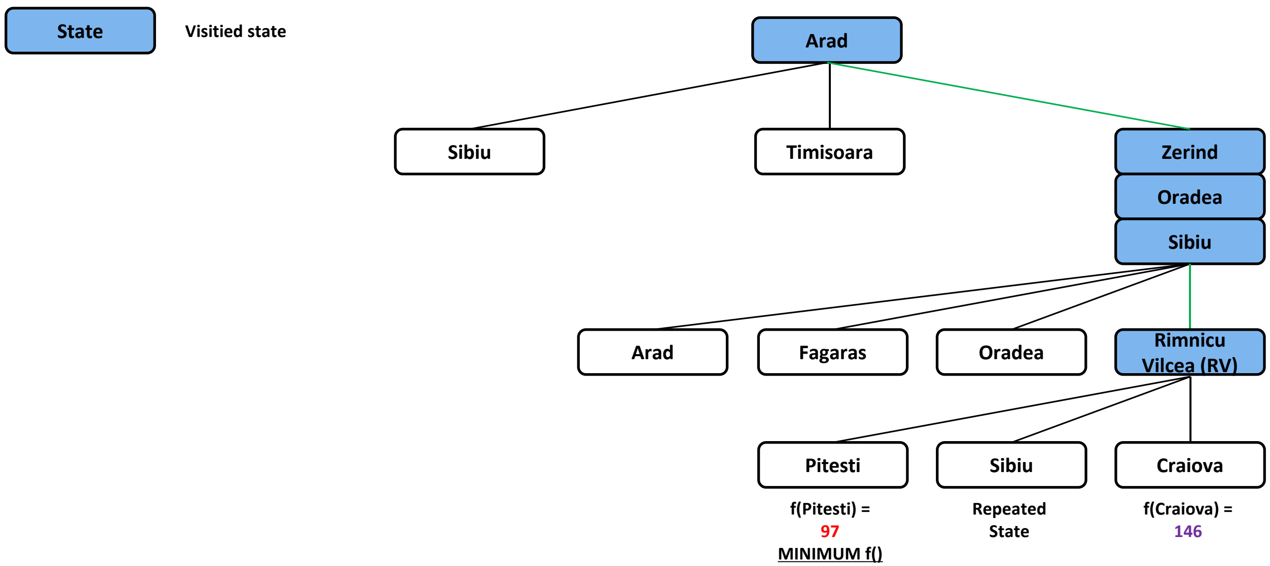
Dracula’s Roadtrip: Hill Climbing



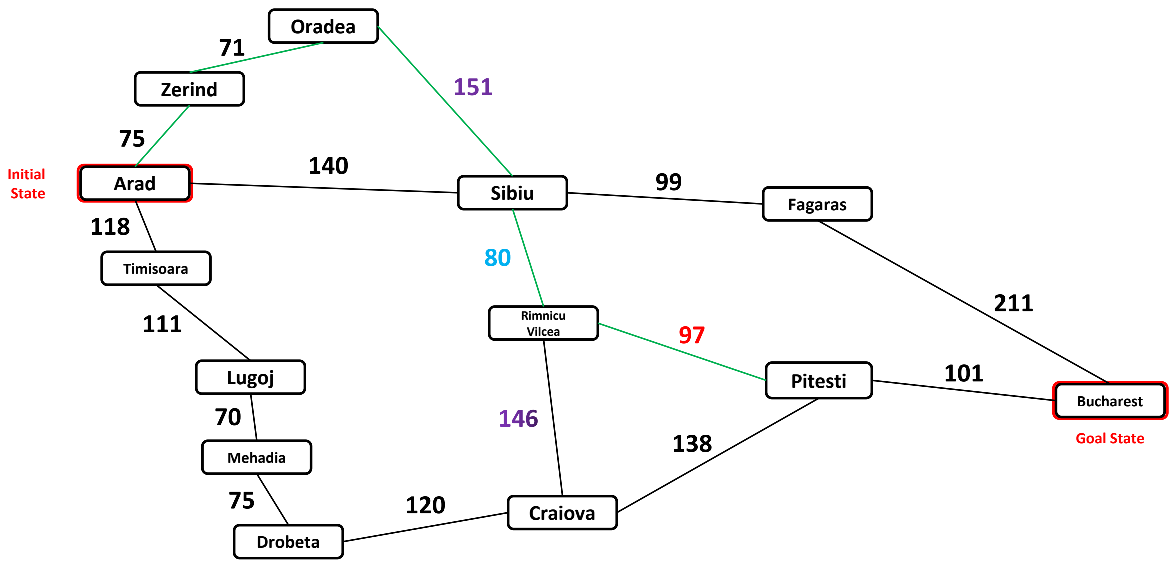


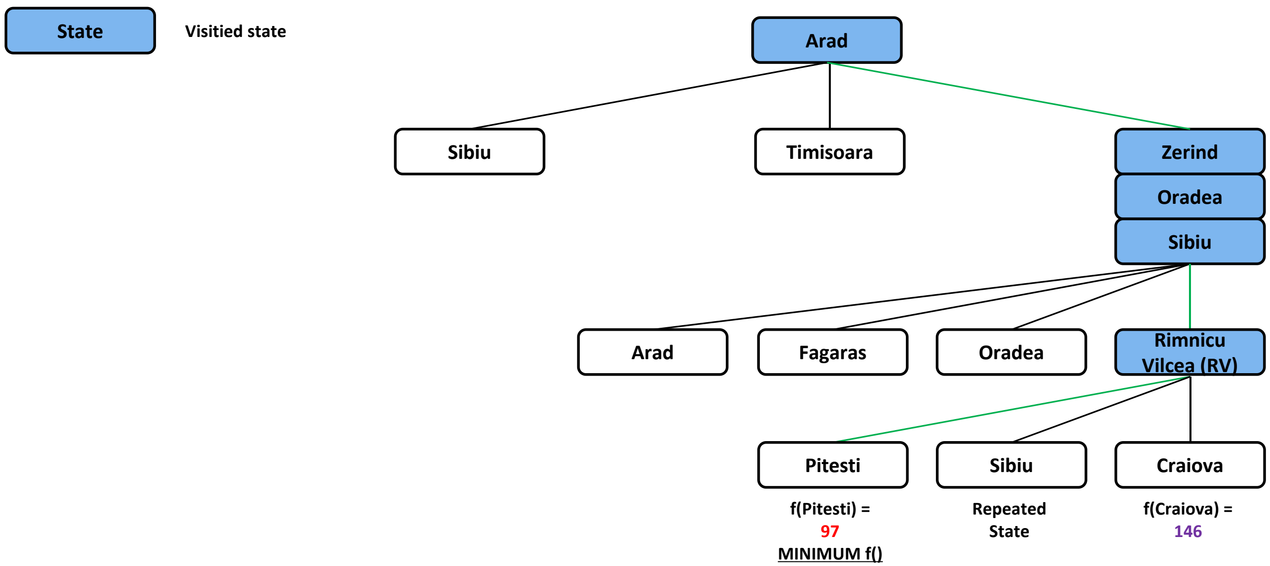
Dracula's Roadtrip: Hill Climbing



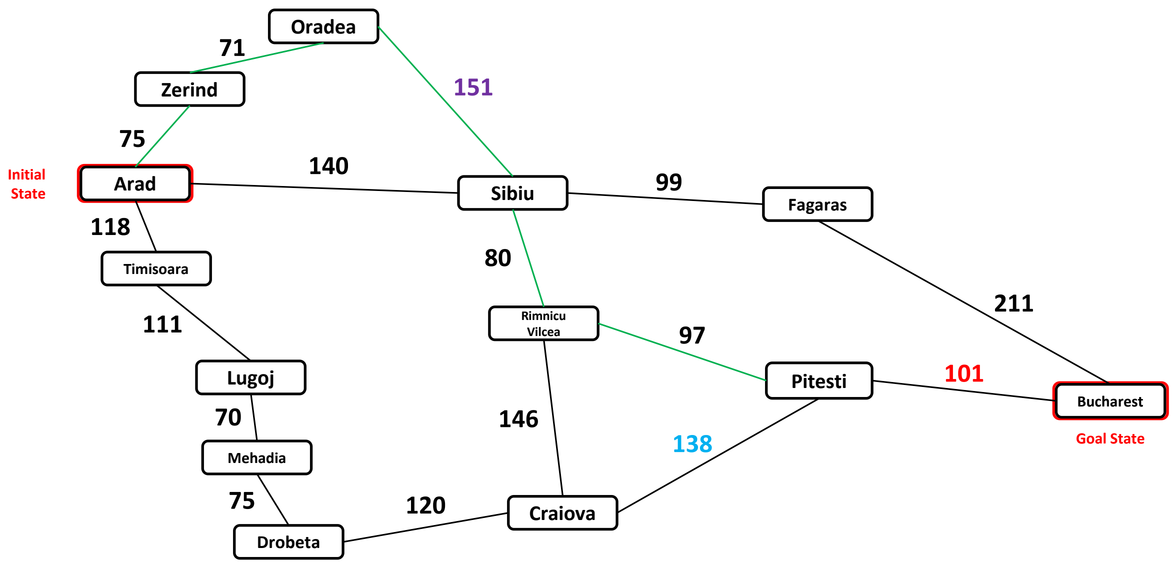


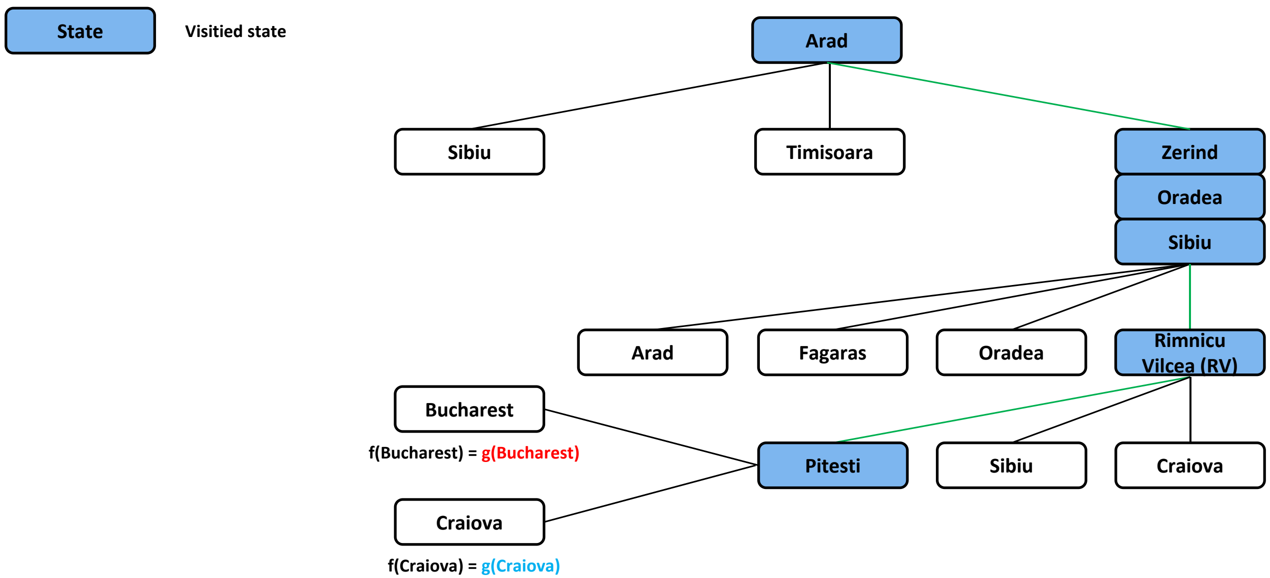
Dracula’s Roadtrip: Hill Climbing



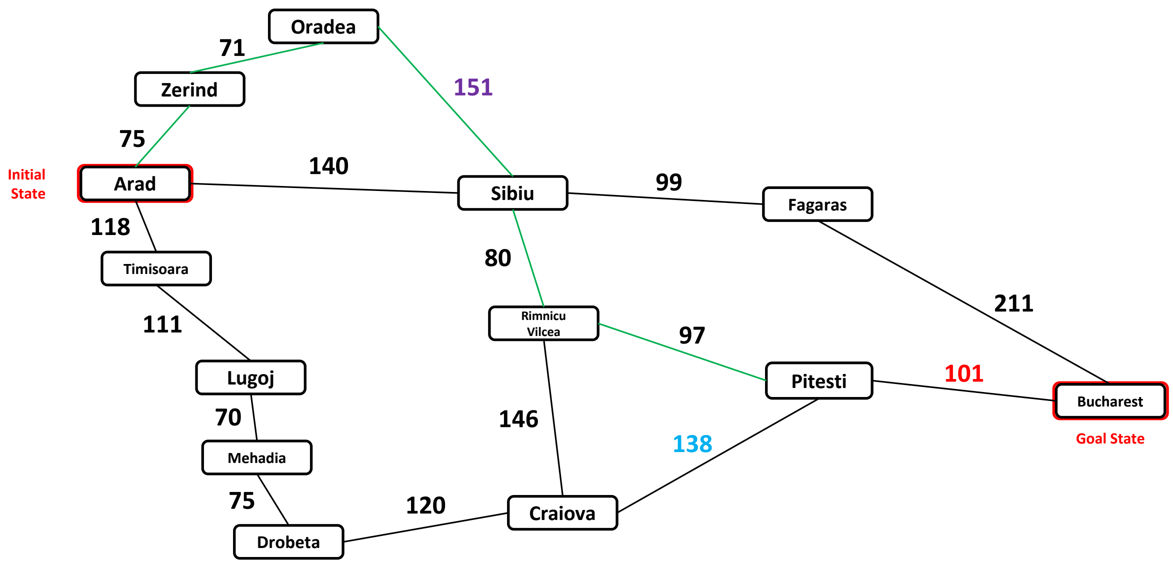


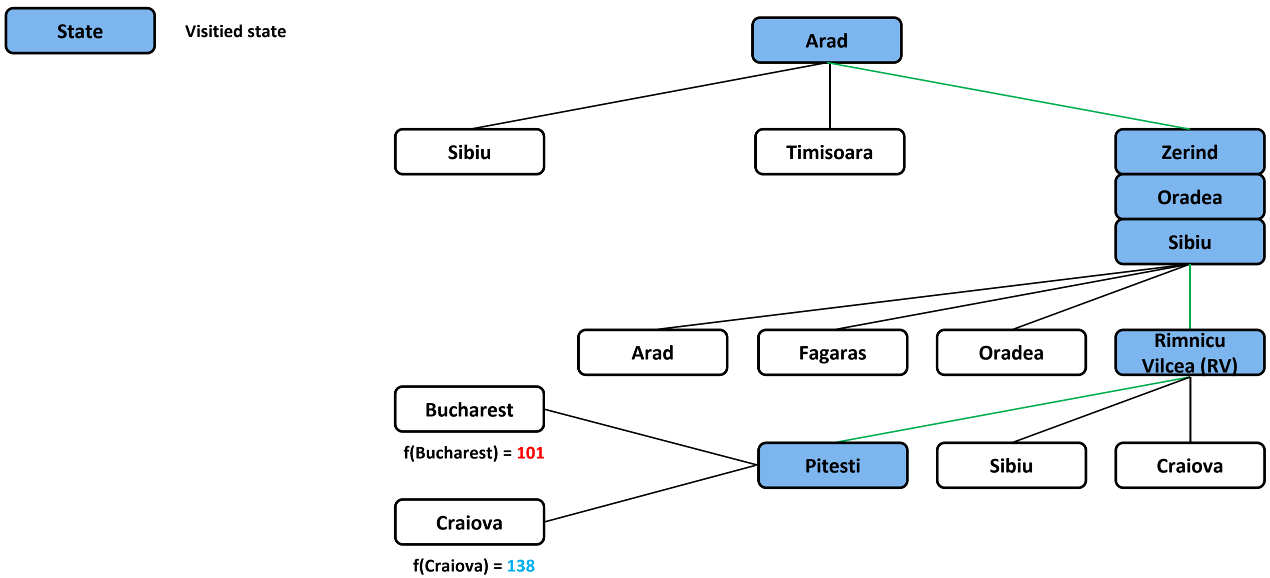
Dracula’s Roadtrip: Hill Climbing



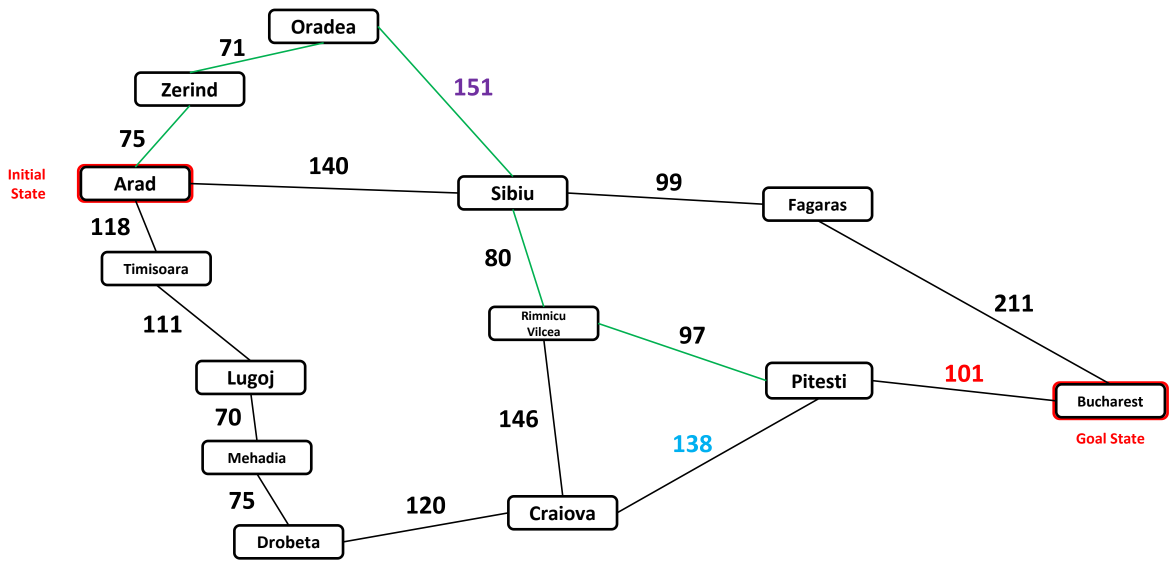


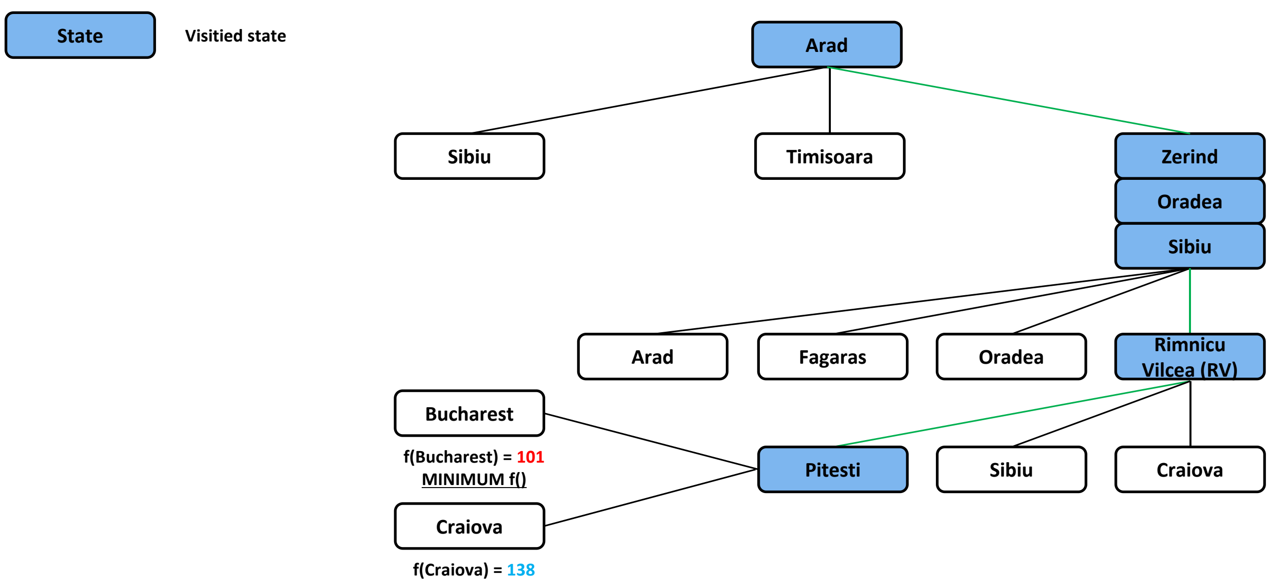
Dracula’s Roadtrip: Hill Climbing



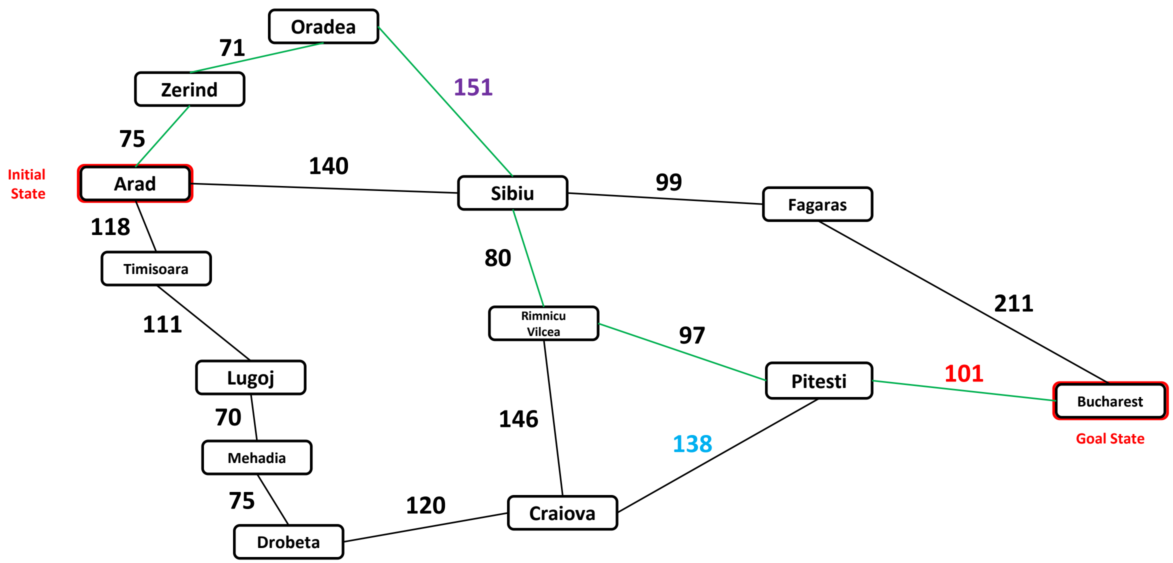


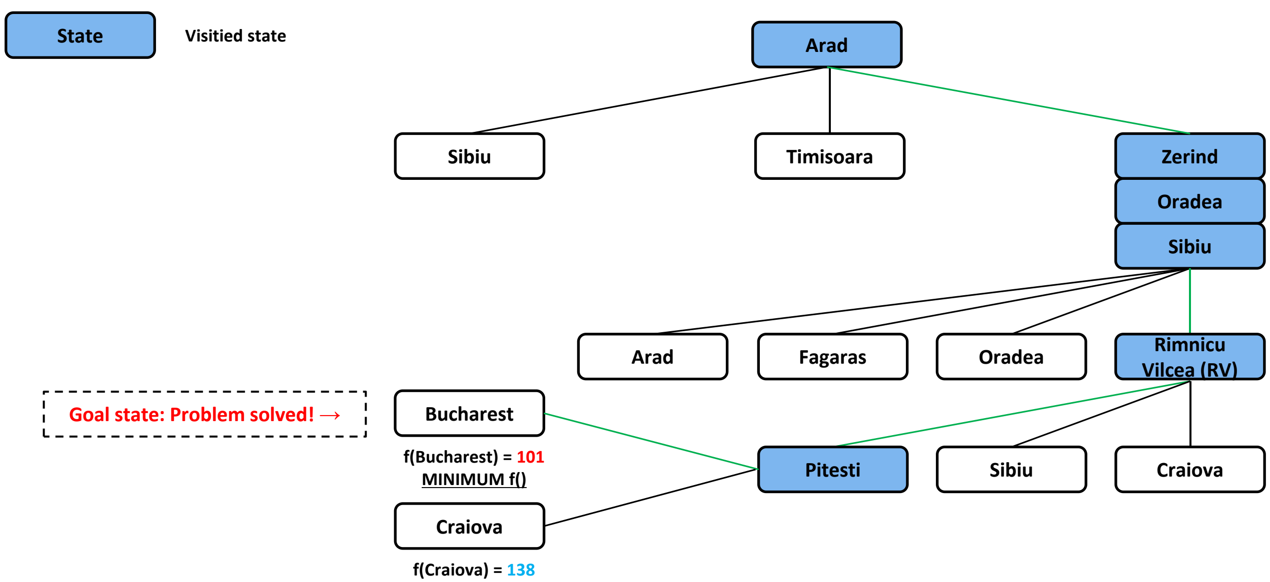
Dracula's Roadtrip: Hill Climbing



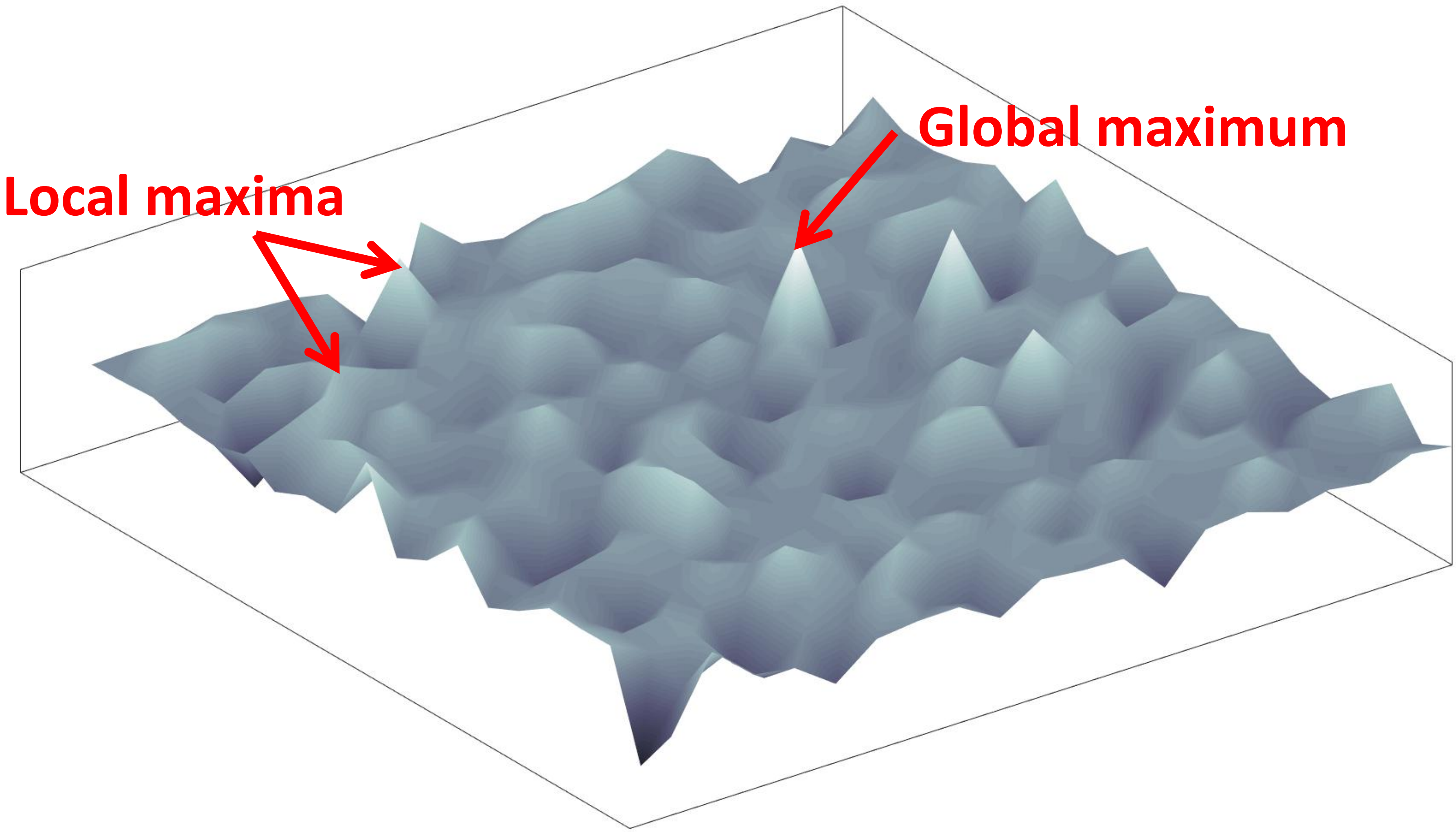


Dracula's Roadtrip: Hill Climbing

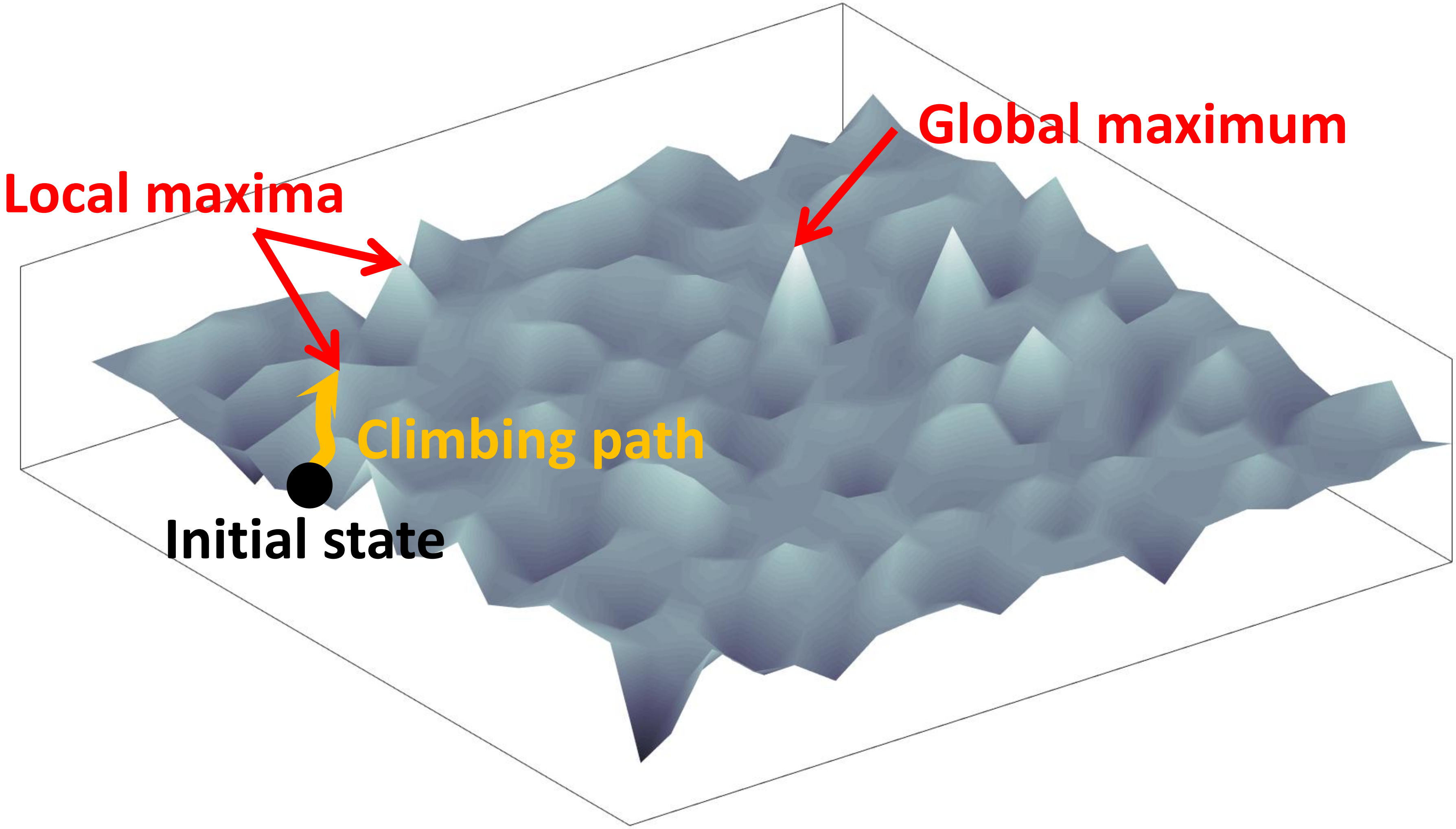




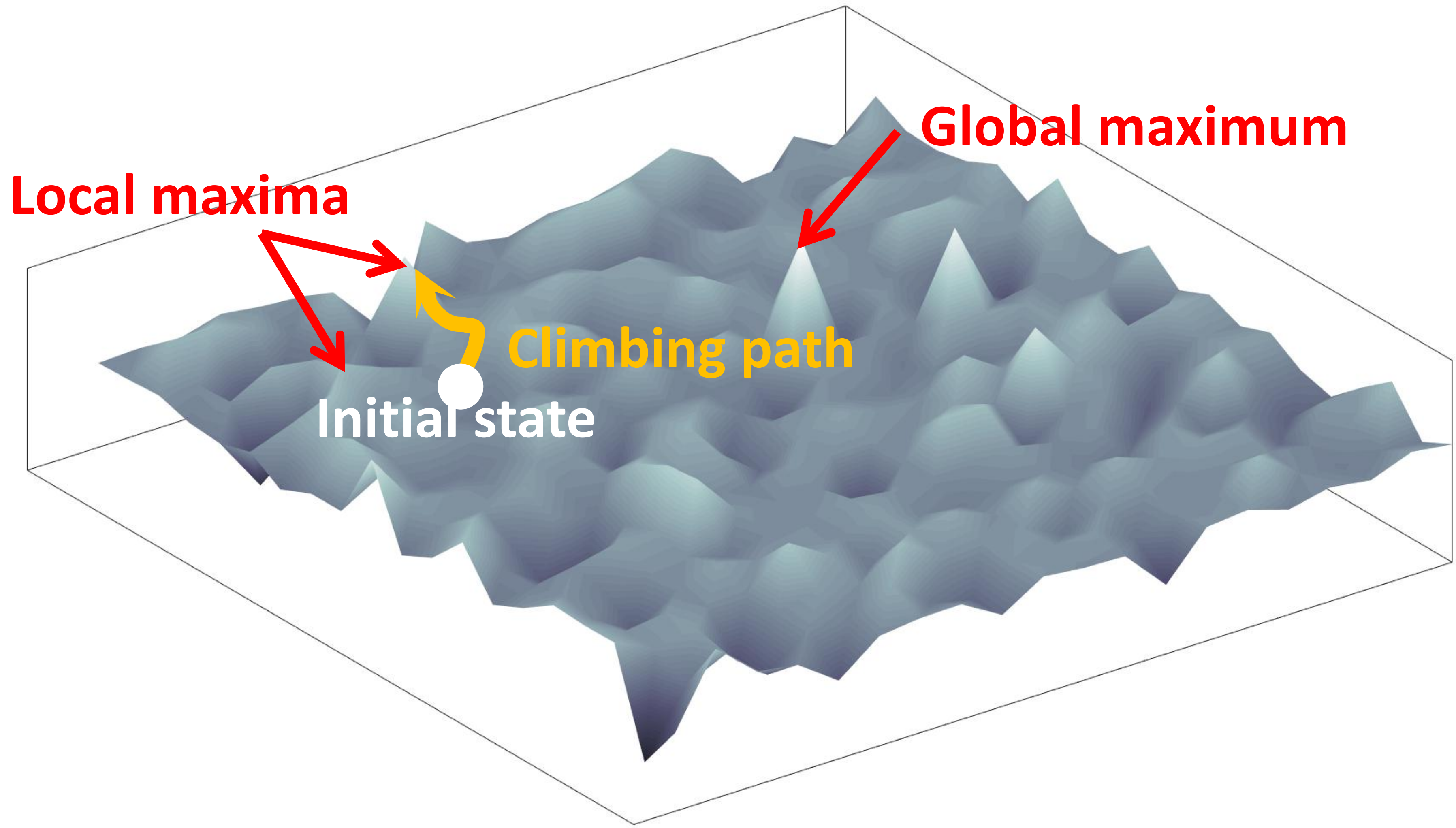
Hill Climbing Problems: Local Maxima



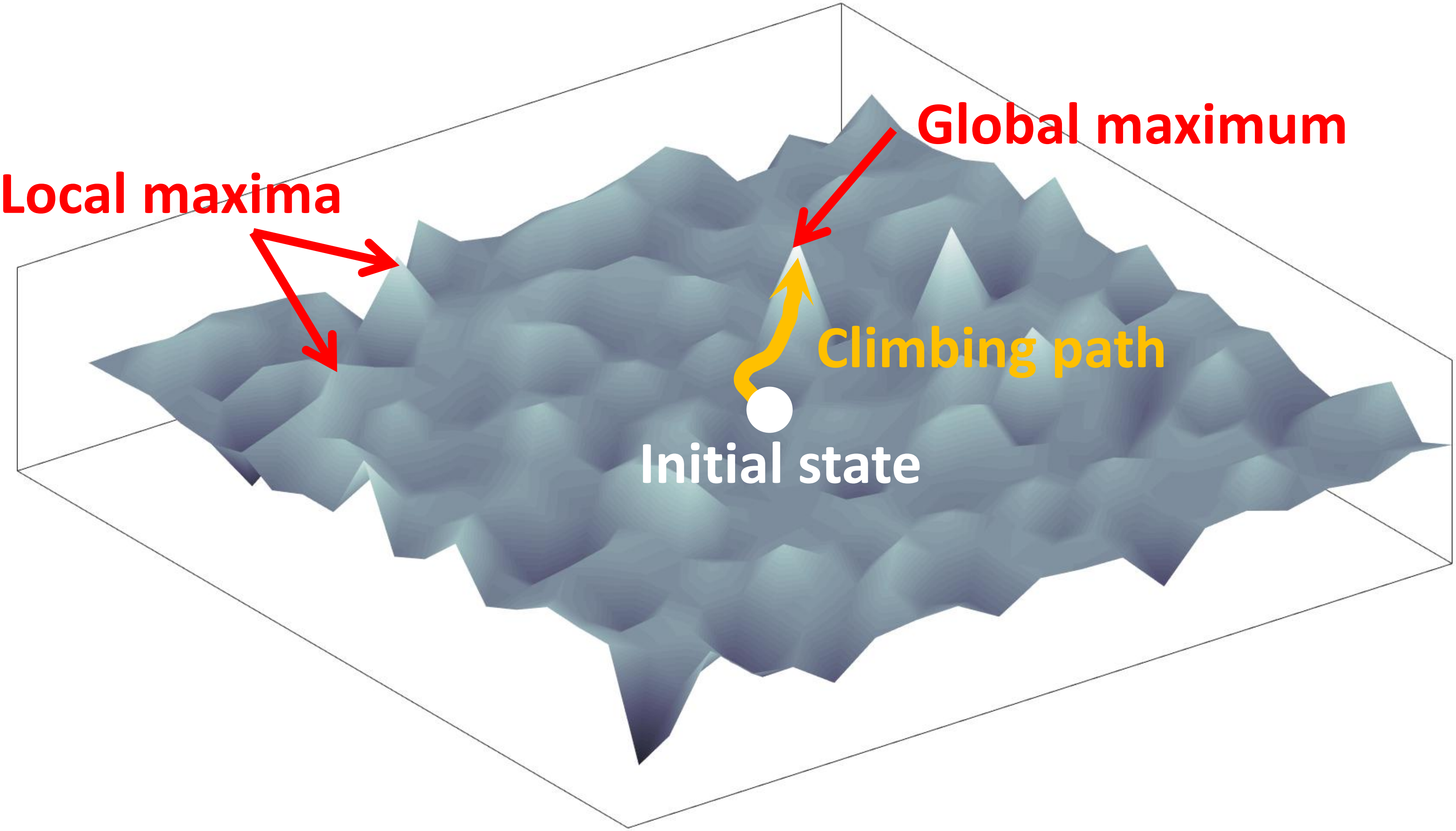
Hill Climbing Problems: Local Maxima



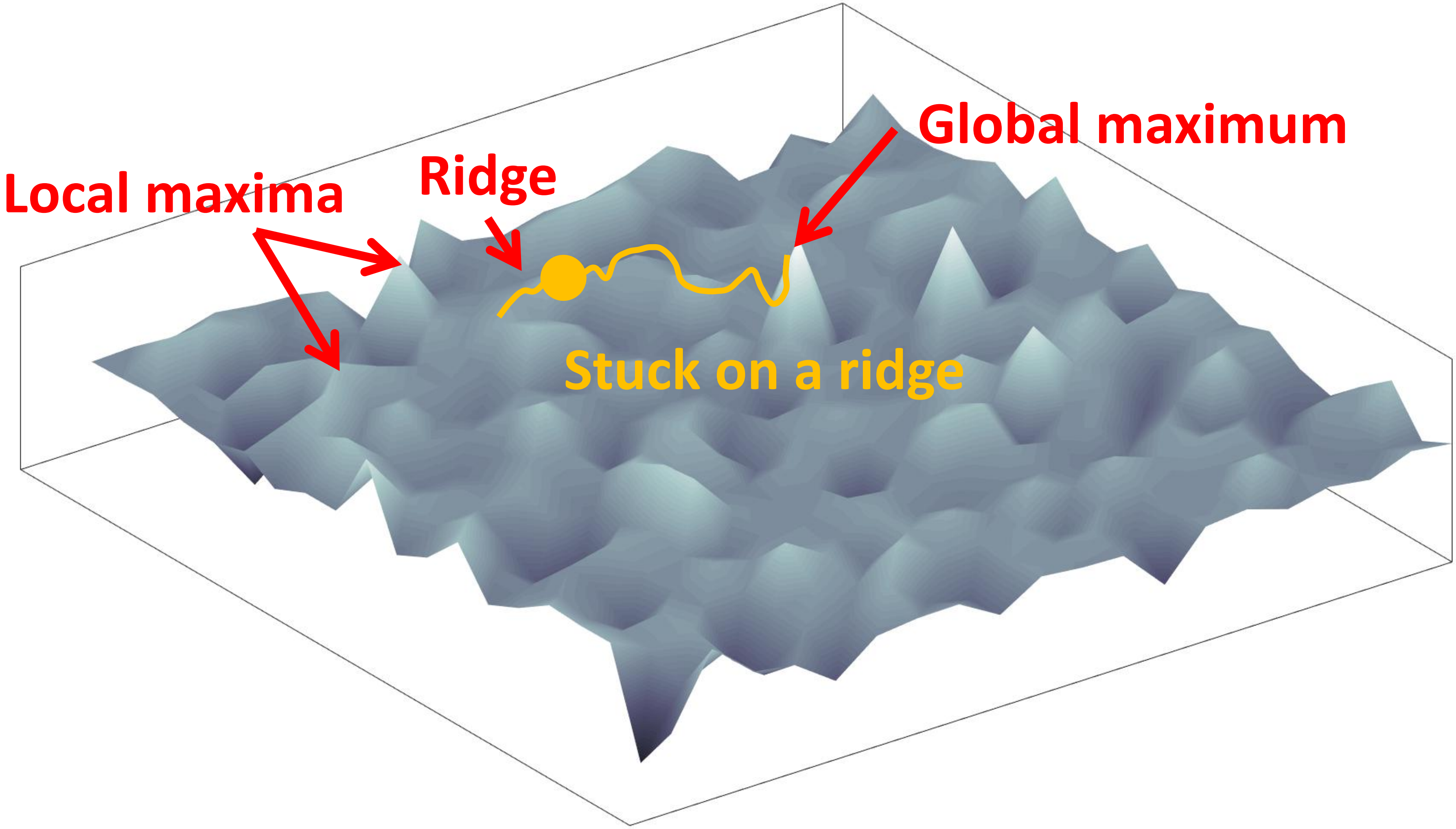
Hill Climbing Problems: Local Maxima



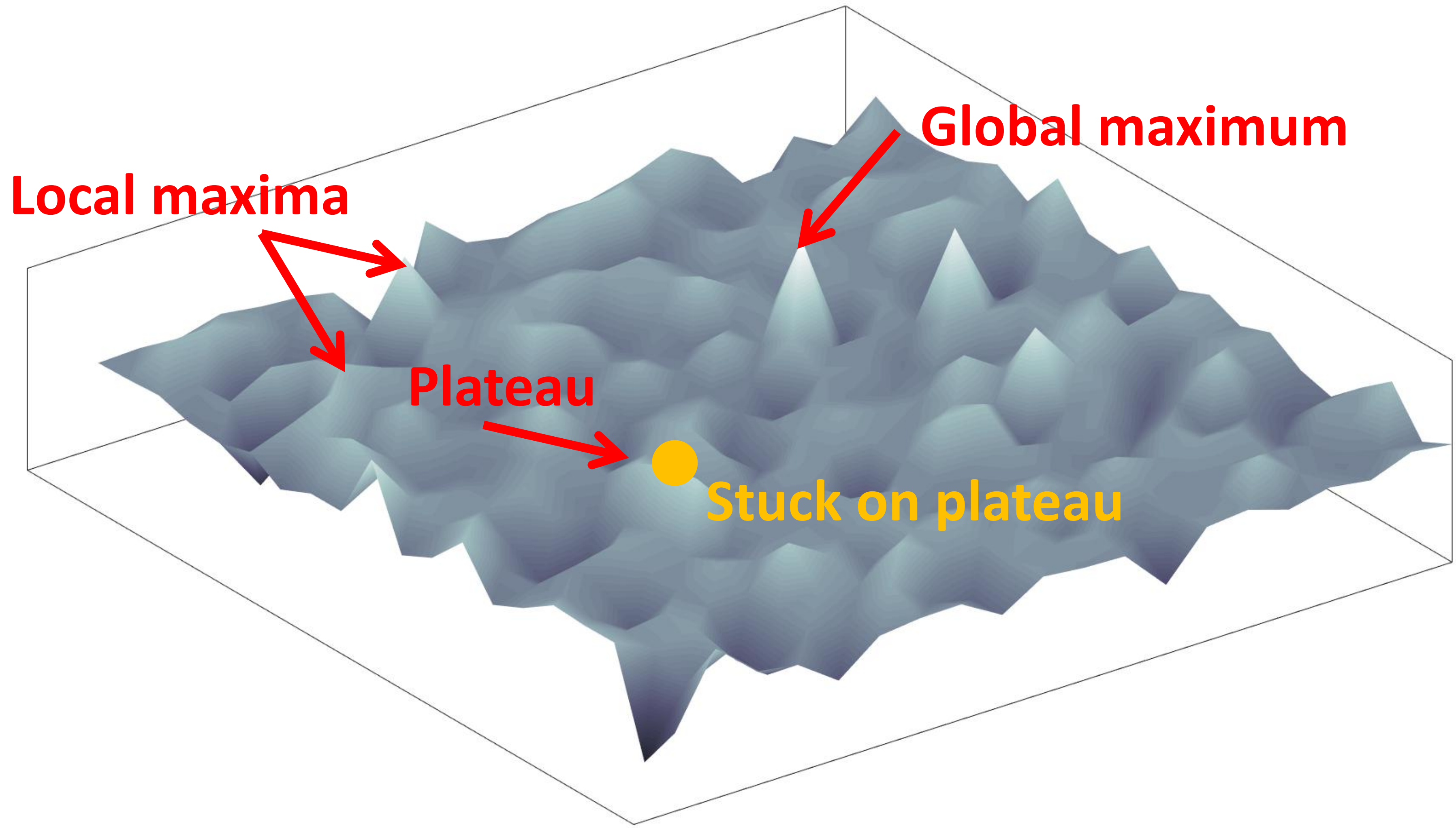
Hill Climbing Problems: Local Maxima



Hill Climbing Problems: Ridges



Hill Climbing Problems: Plateaus



Greedy Best First Search

- **Also a rather primitive informed search approach**
  - **a naive greedy algorithm**
  - **evaluation function: heuristics h(n)**
  - **tries to not "move farther away" from the goal**
  - **does not care about the total path cost**
- **Practicalities:**
  - **it keep track of search history:**
    - **tracks visited states / nodes**
    - **tracks frontier states / nodes**

Greedy Best First: Evaluation Function

Calculate / obtain:

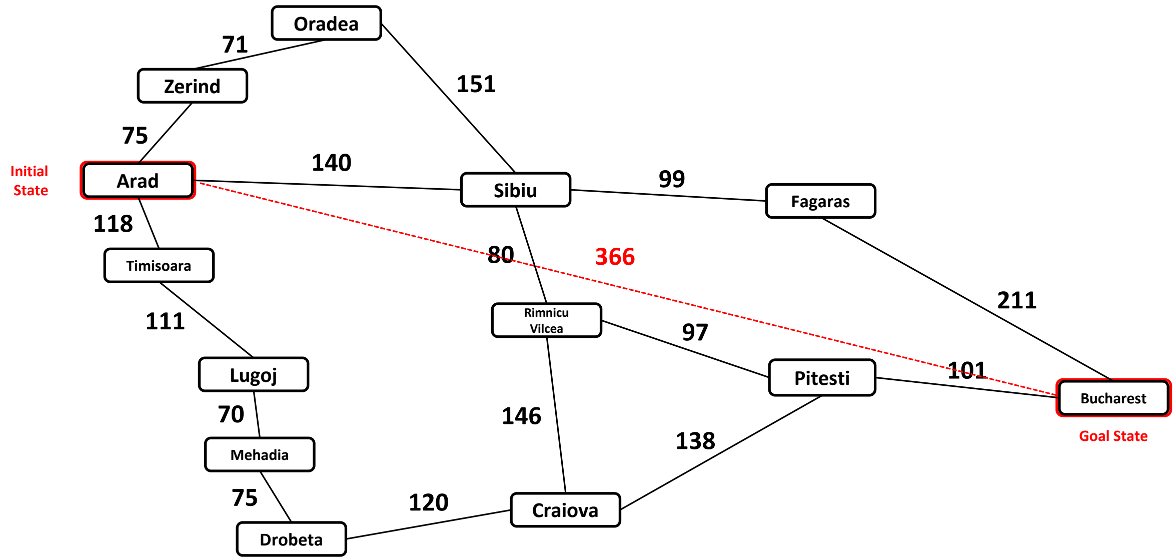
$$f(n) = \mathbf{h(State_n)}$$

A state n with minimum (or maximum) f(n) should be chosen for expansion

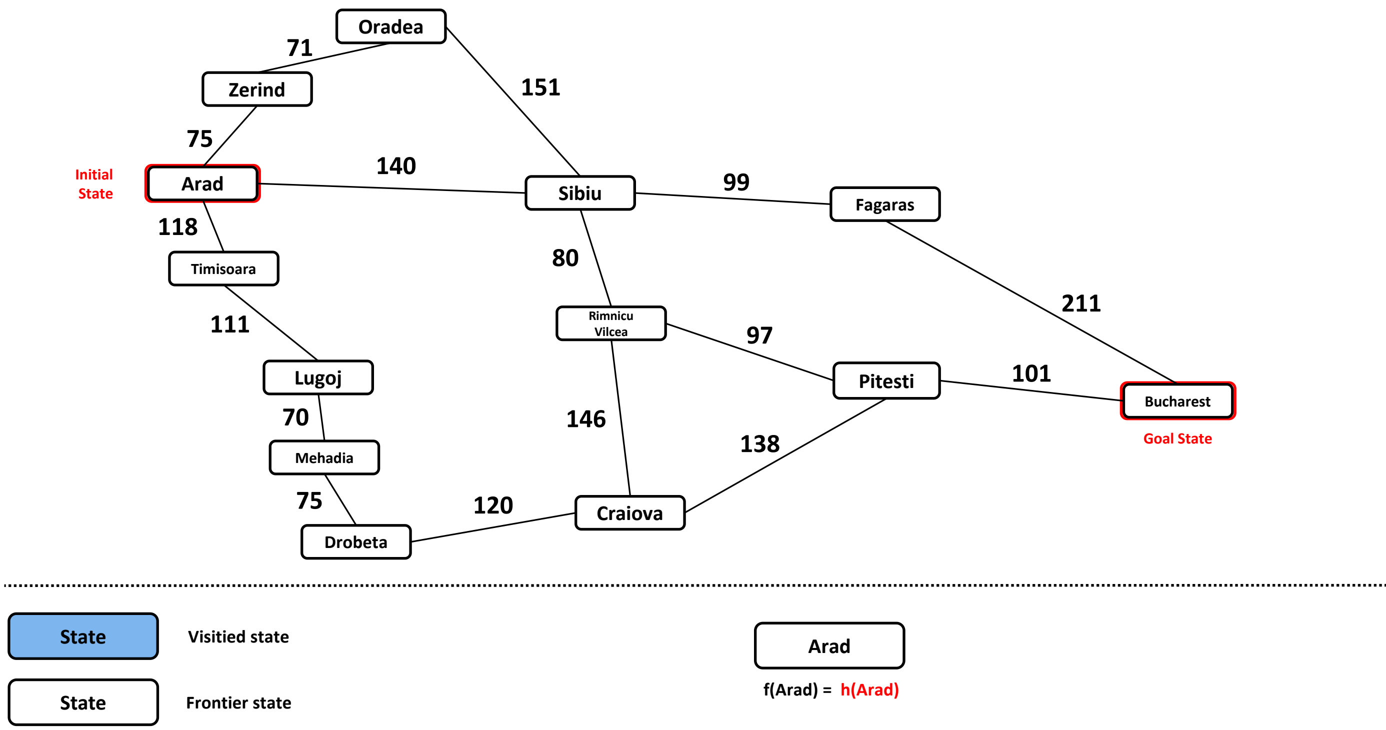
Dracula's Roadtrip: Heuristics h(n)

For this particular problem the heuristic function h(n) is defined by a straight-line (Euclidean) distance between two states (cities).

"As the crows flies" in other words.



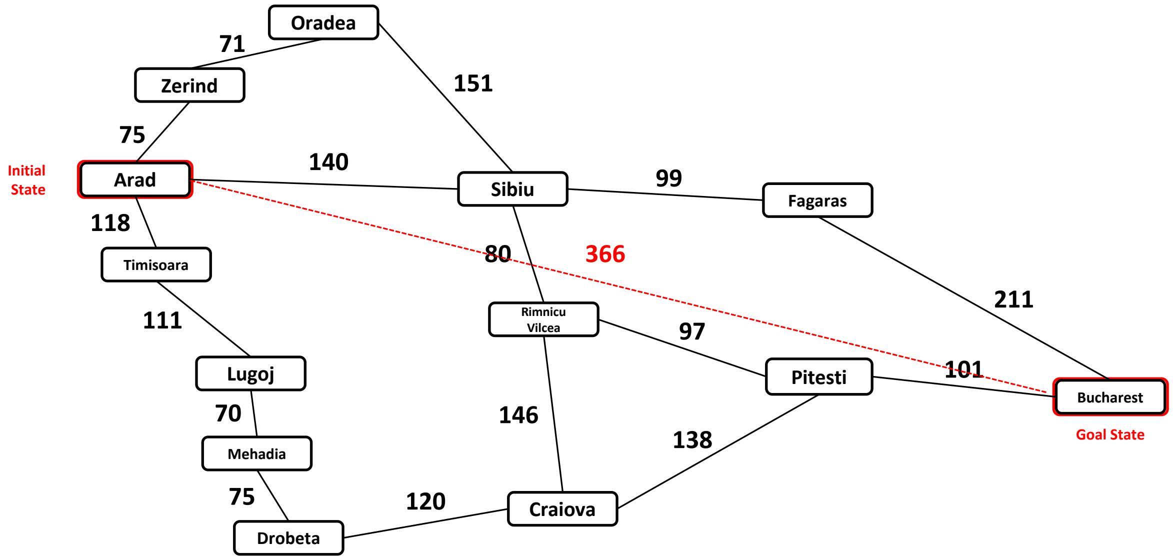
Dracula's Roadtrip: Greedy Best First



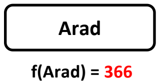
Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First



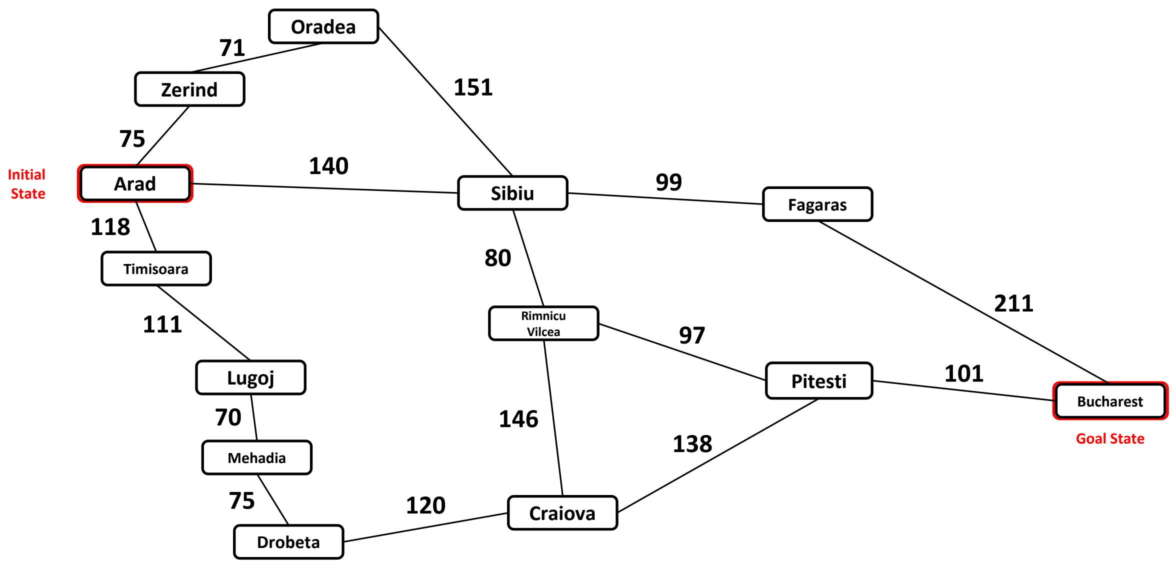
State	Visitied state
State	Frontier state

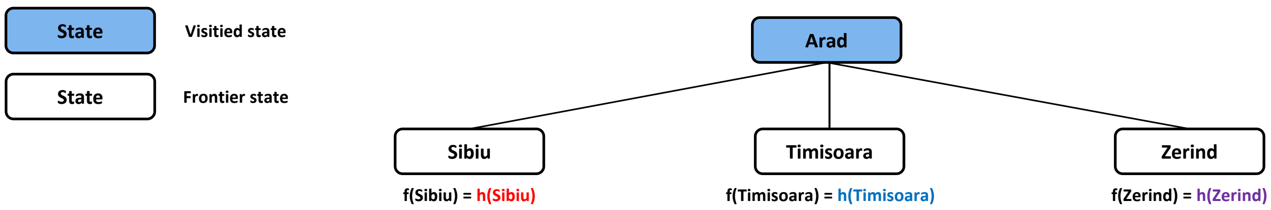


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First

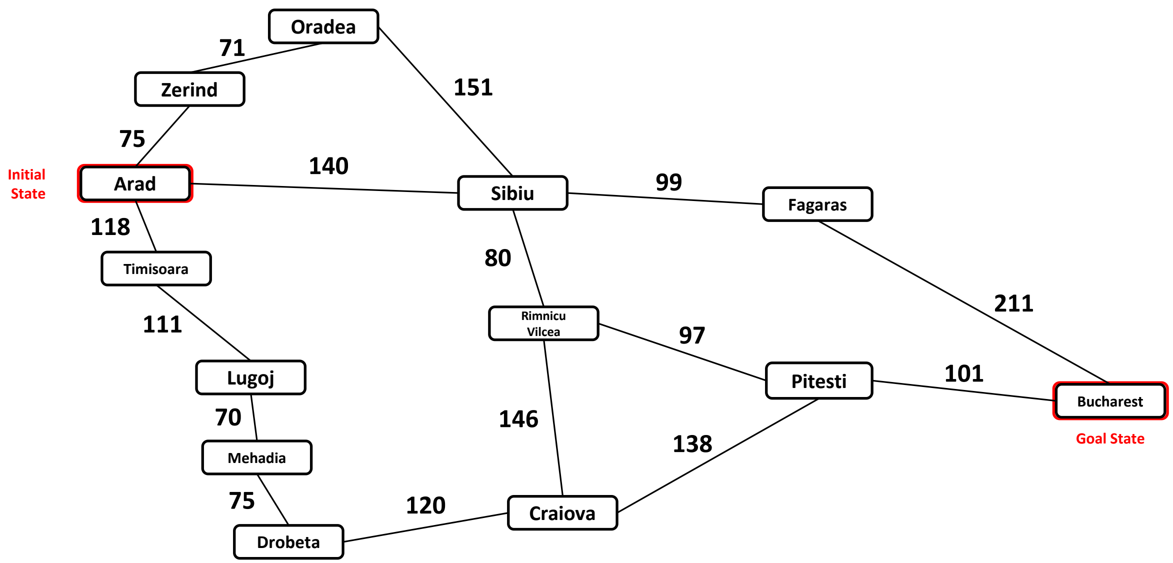


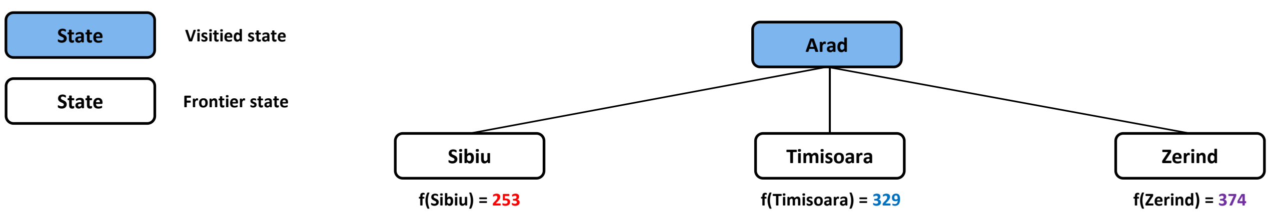


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: Greedy Best First

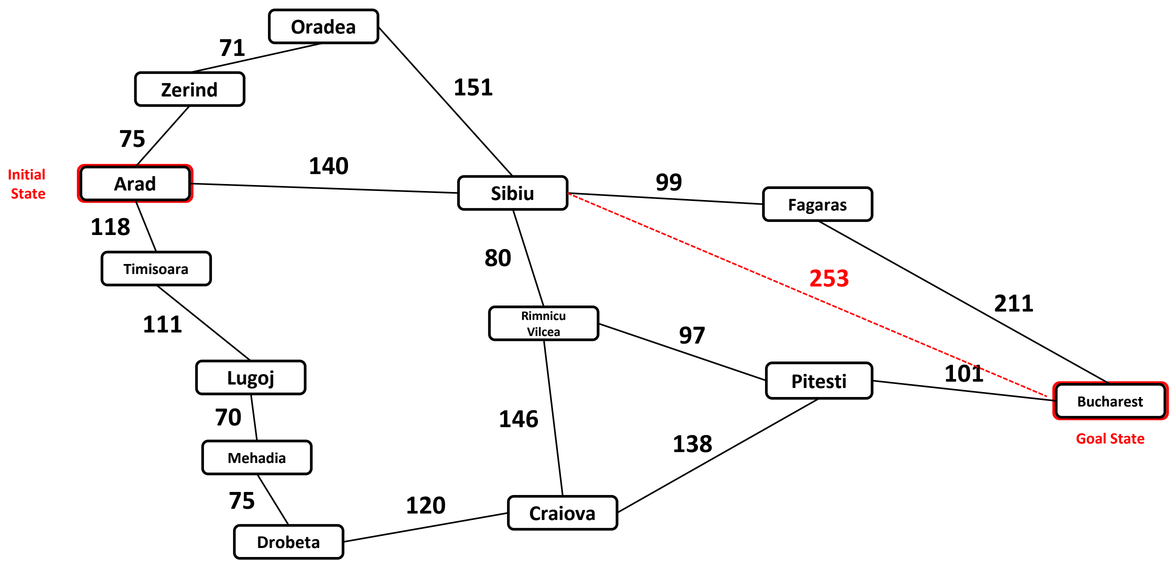


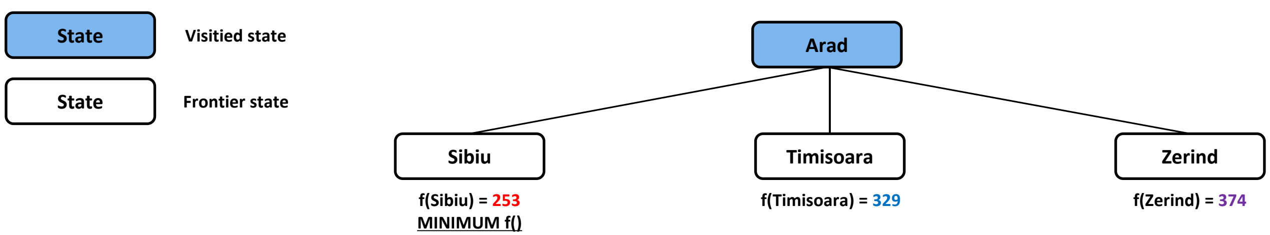


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First

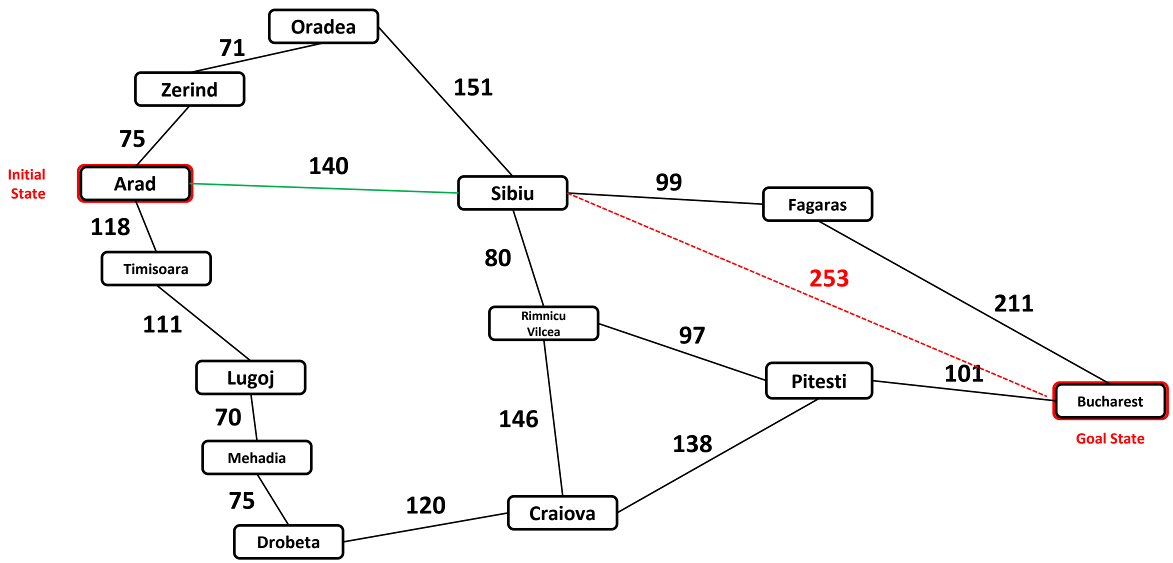


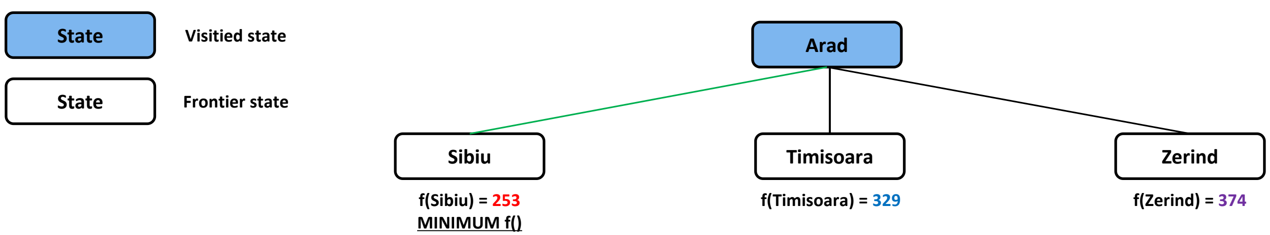


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First

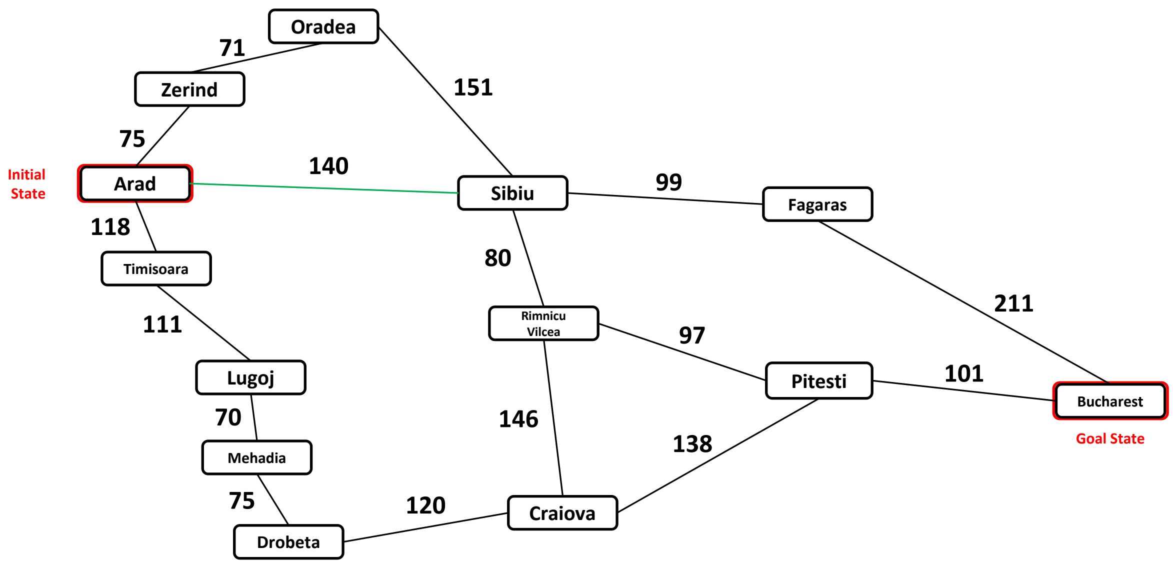


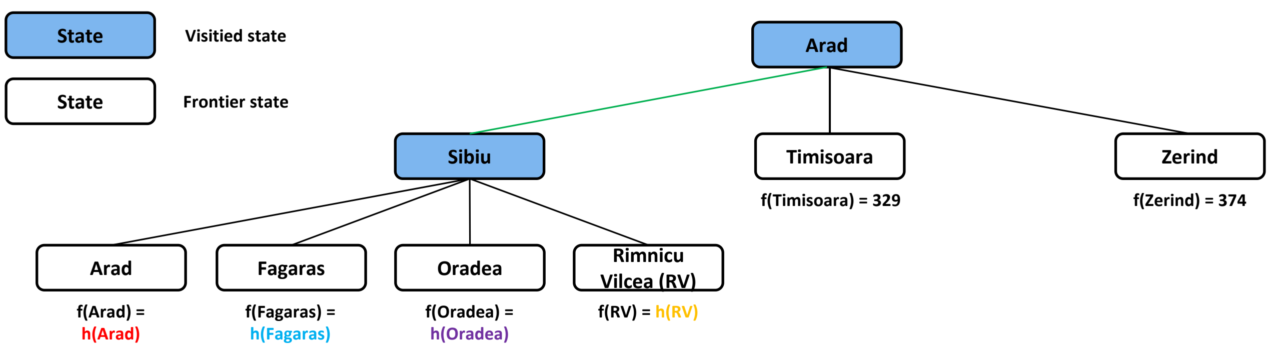


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First

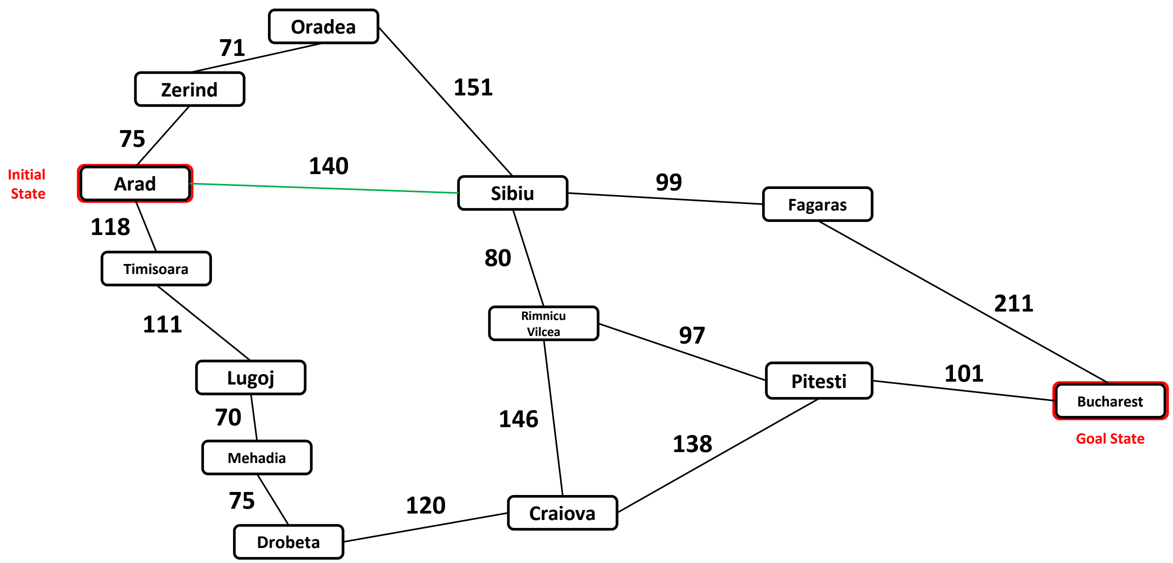


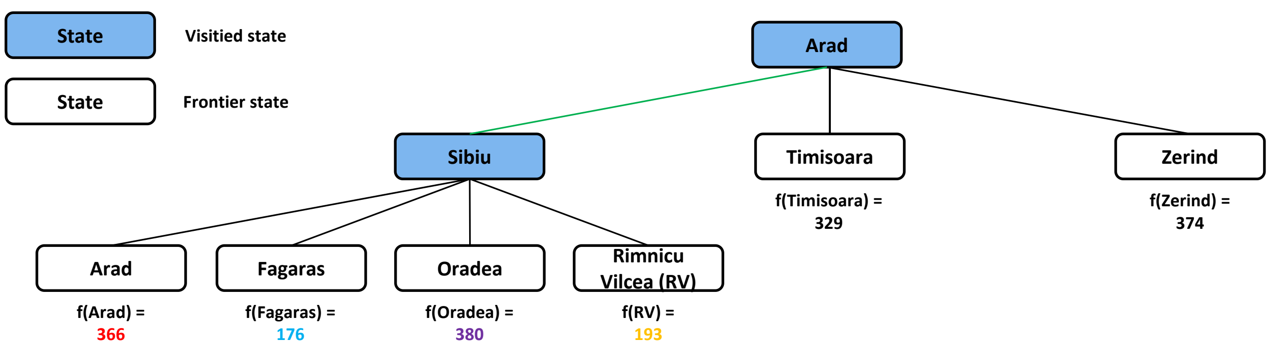


<u>Straight-line</u> distance to Bucharest (h(State)):

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

Dracula's Roadtrip: Greedy Best First

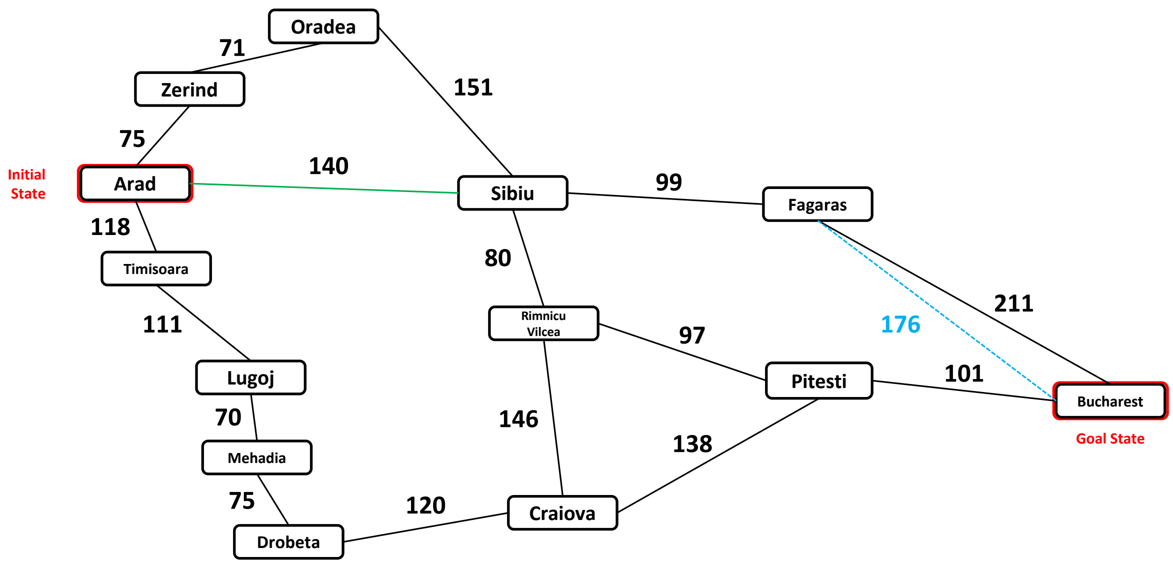


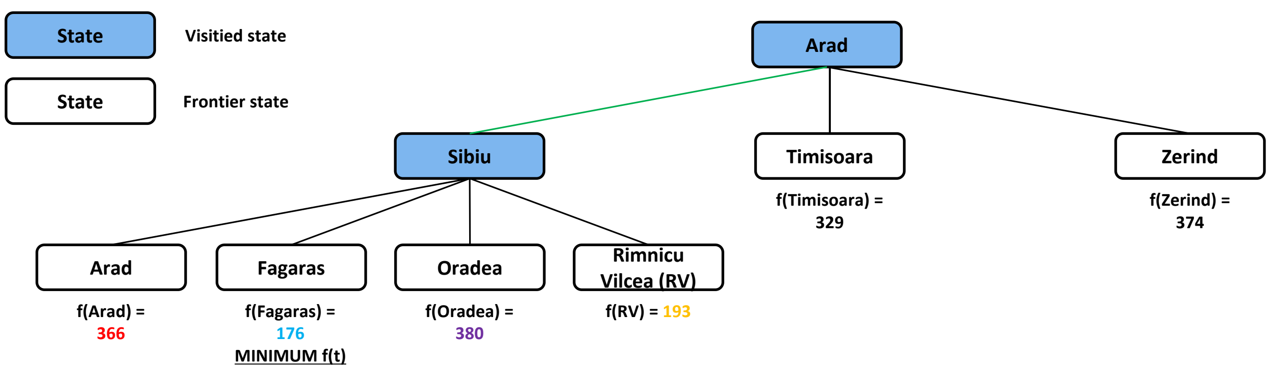


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: Greedy Best First

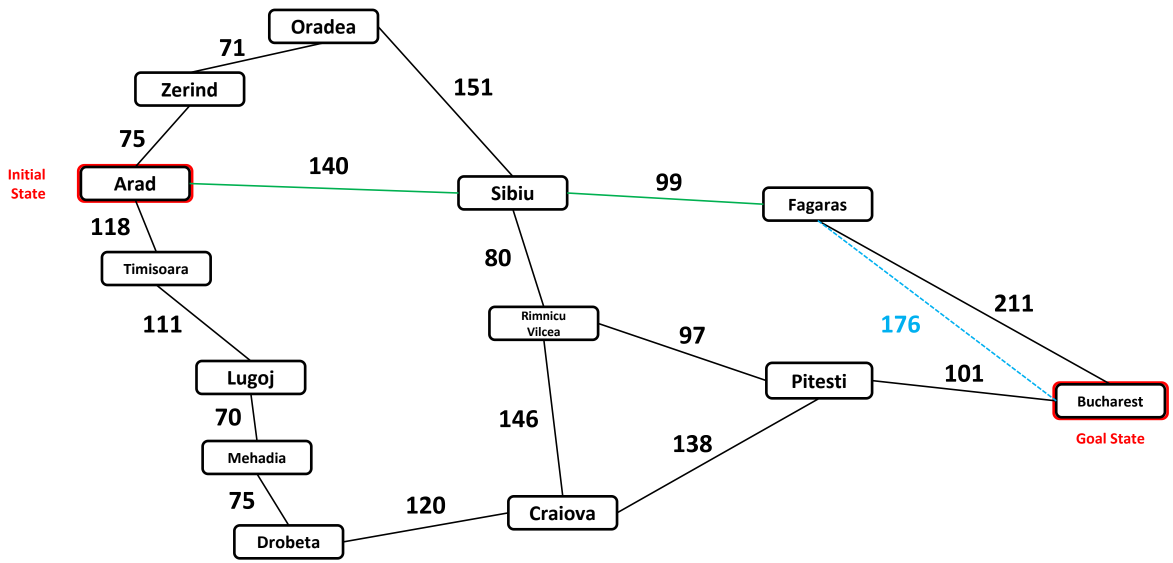


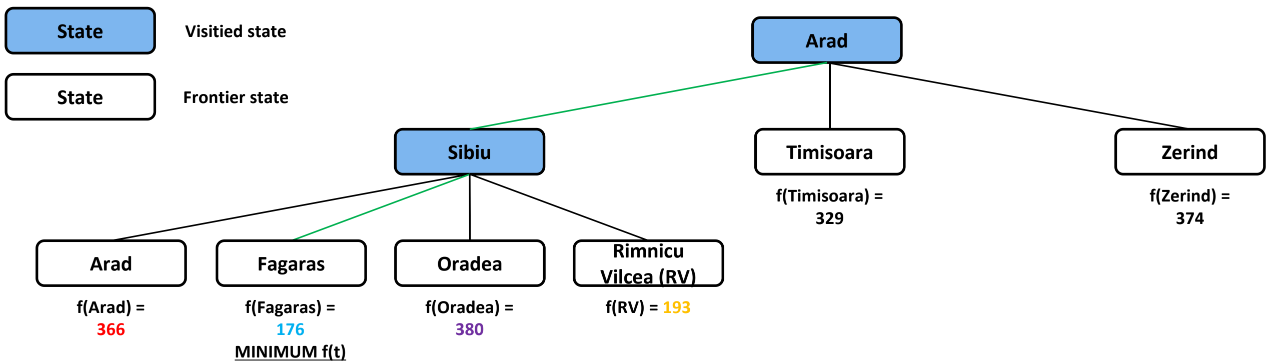


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First

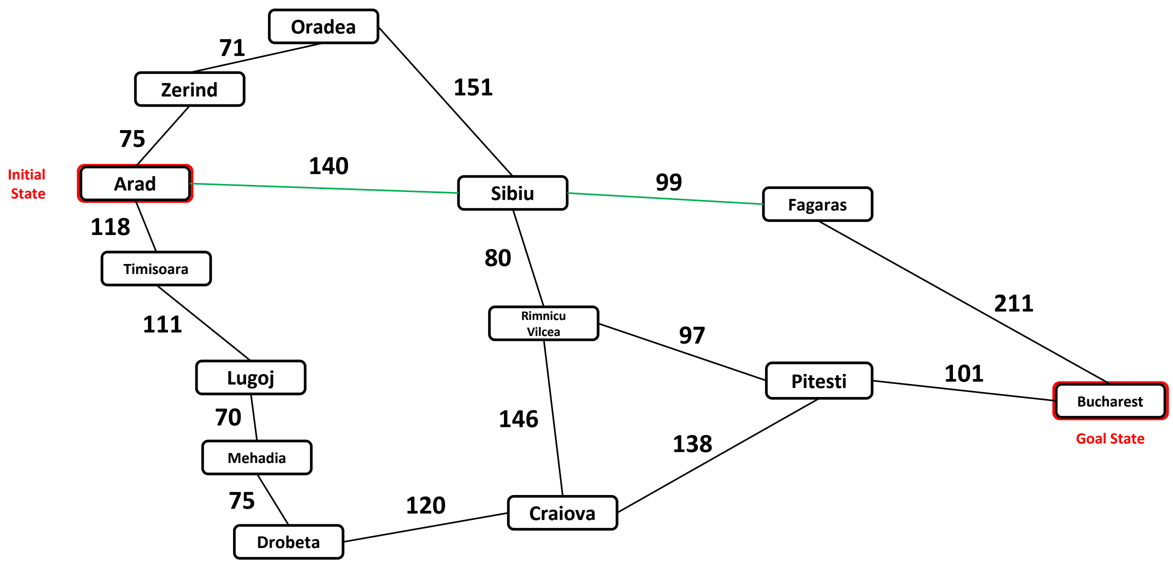


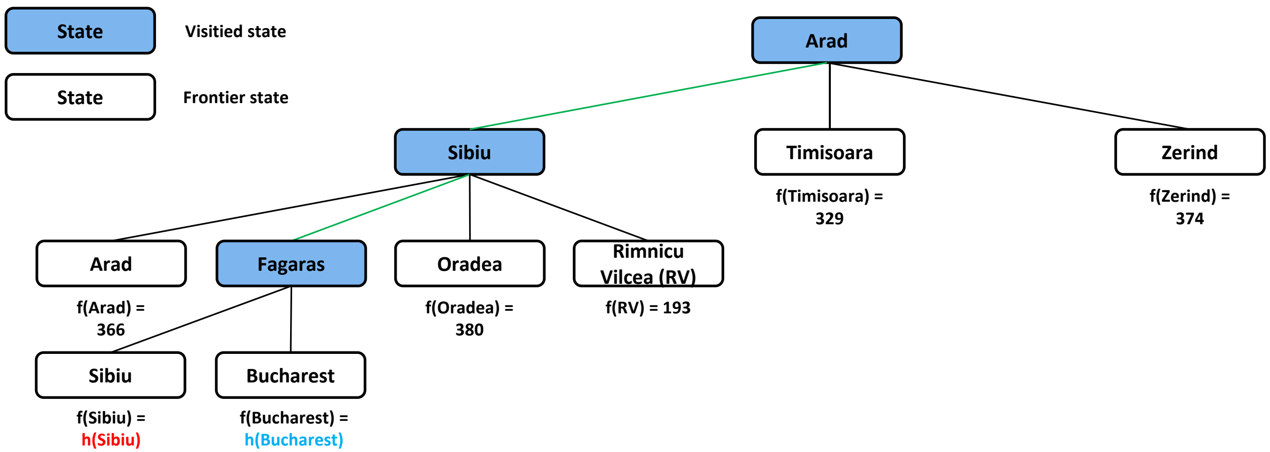


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First

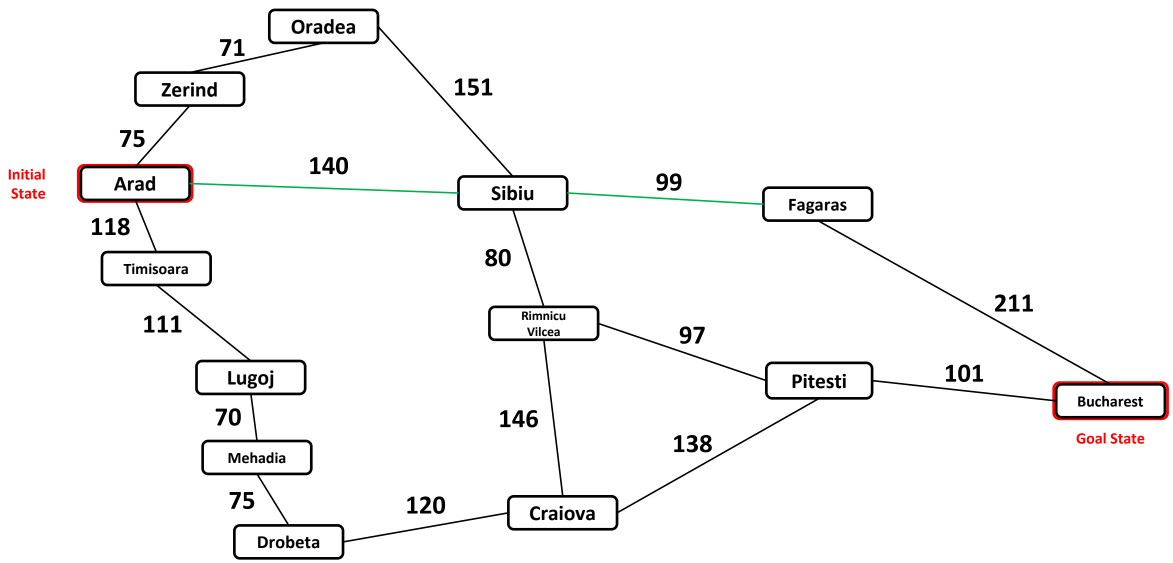


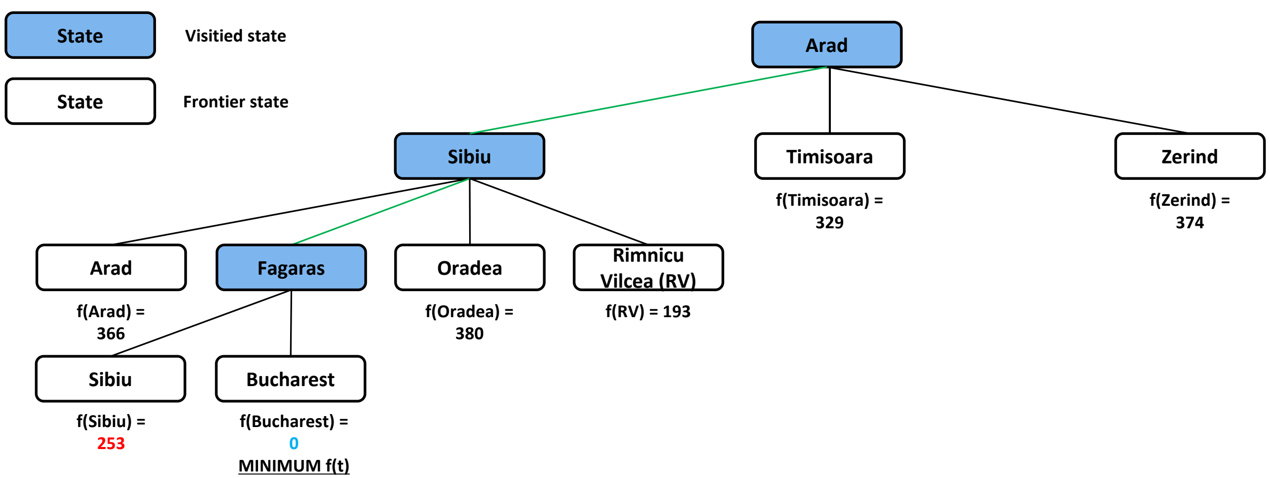


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: Greedy Best First

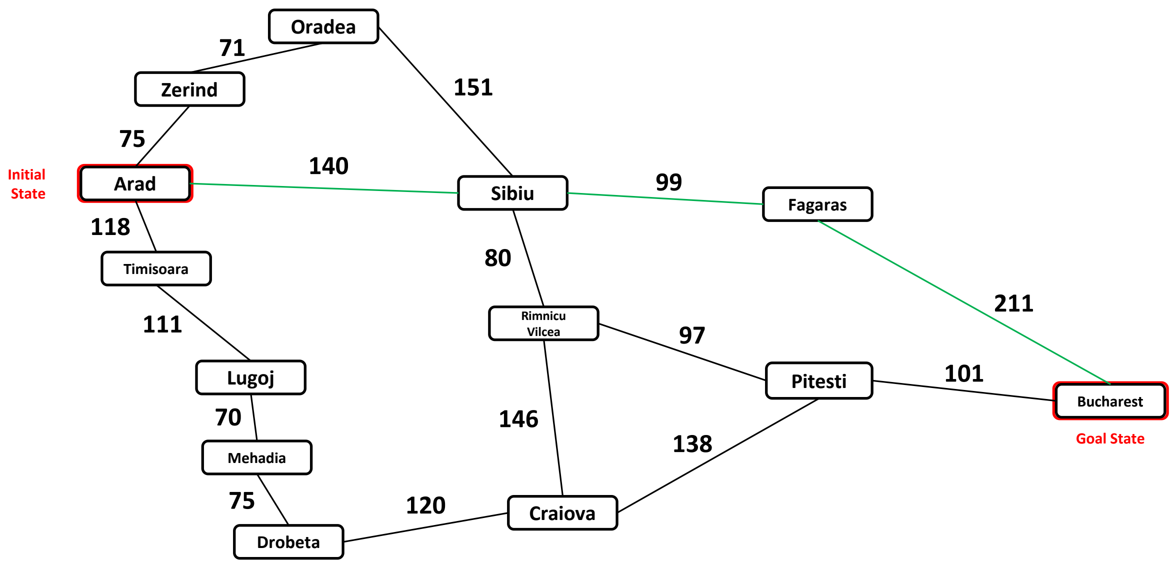


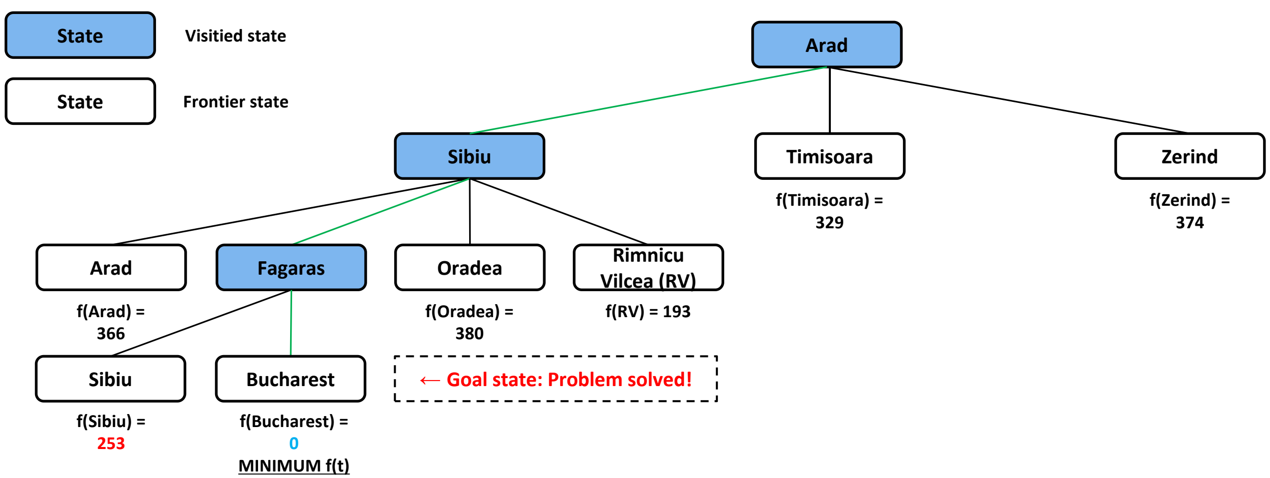


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: Greedy Best First





Straight-line distance to Bucharest (h(State)):

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

A* Algorithm: Evaluation Function

Calculate / obtain:

$f(n) = \mathbf{g(State_n)} + \mathbf{h(State_n)}$

where:

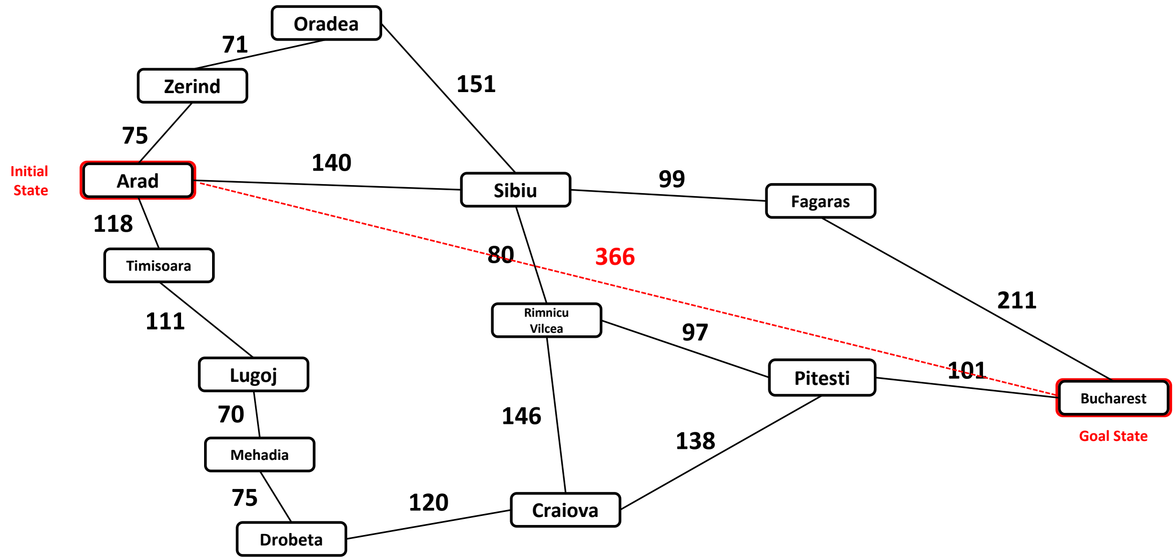
- **g(n) - initial node to node n path cost**
- **h(n) - estimated cost of the best path that continues from node n to a goal node**

A state n with minimum (maximum) f(n) should be chosen for expansion

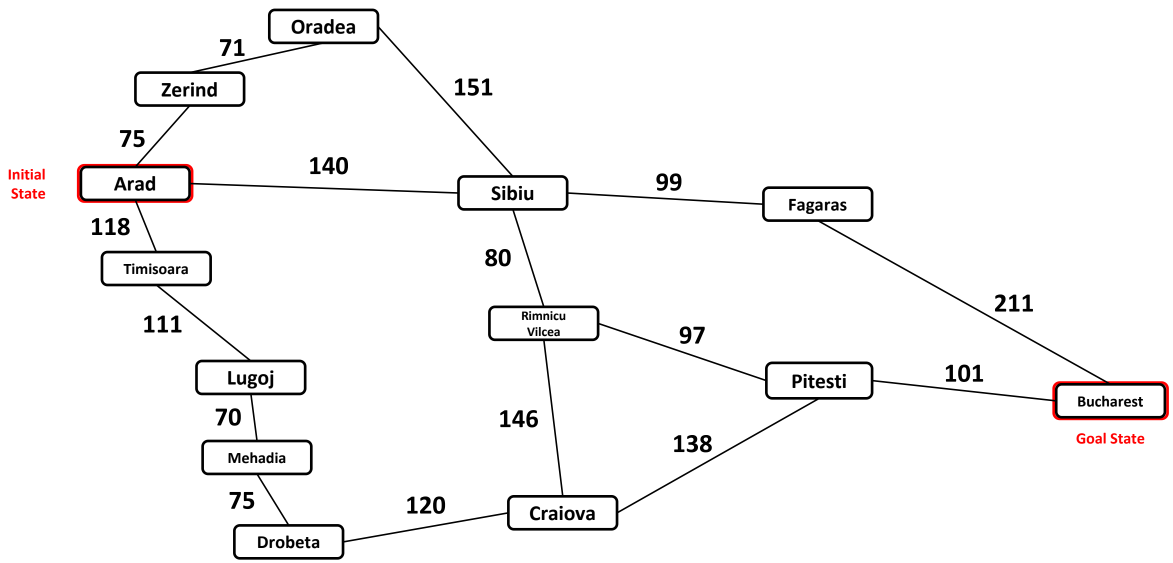
Dracula’s Roadtrip: Heuristics h(n)

For this particular problem the heuristic function h(n) is defined by a **straight-line (Euclidean) distance** between two states (cities).

“As the crows flies” in other words.



Dracula's Roadtrip: A* Search



State	Visitied state
State	Frontier state

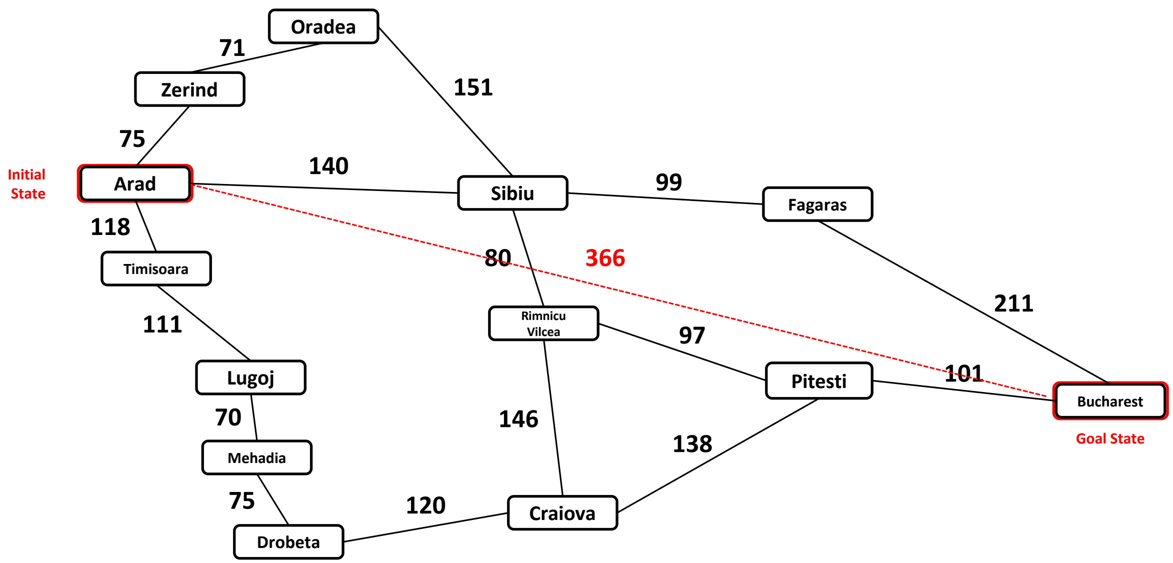
Arad

f(Arad) = g(Arad) + h(Arad)

Straight-line distance to Bucharest (h(State)):

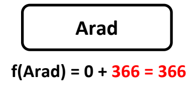
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadi	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Dracula's Roadtrip: A* Search



State — Visitied state

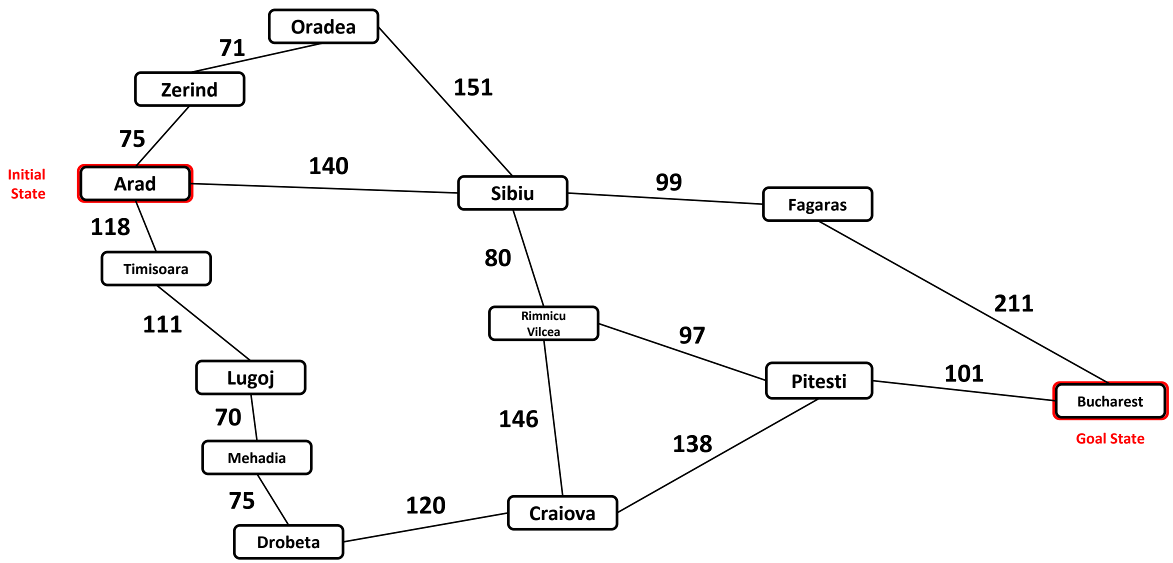
State — Frontier state

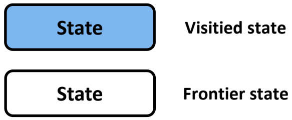


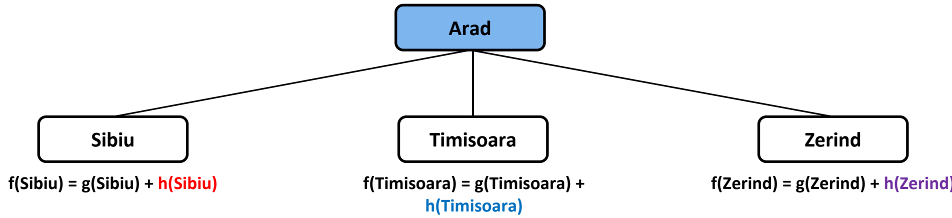
Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: A* Search



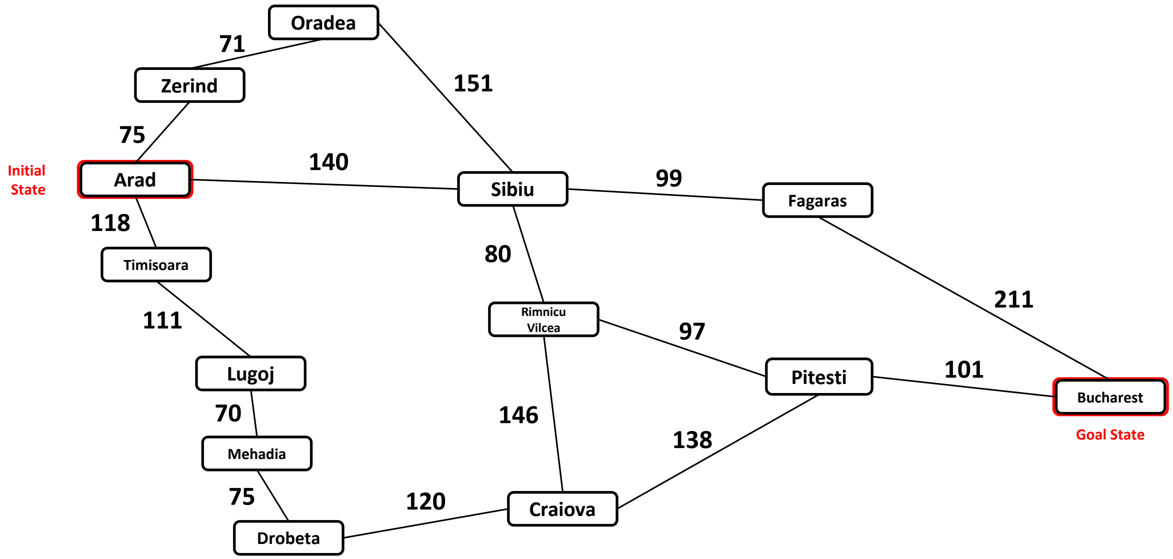


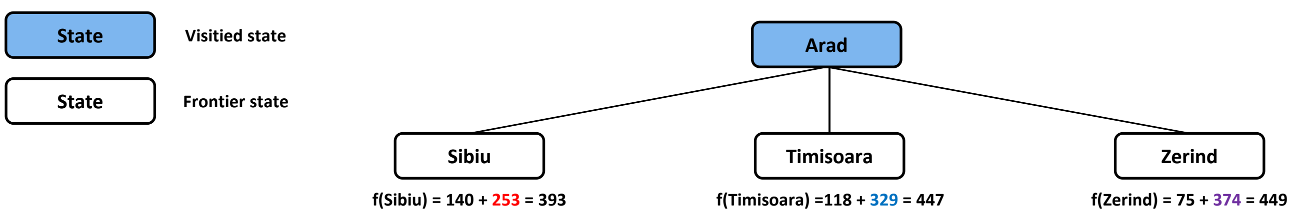


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

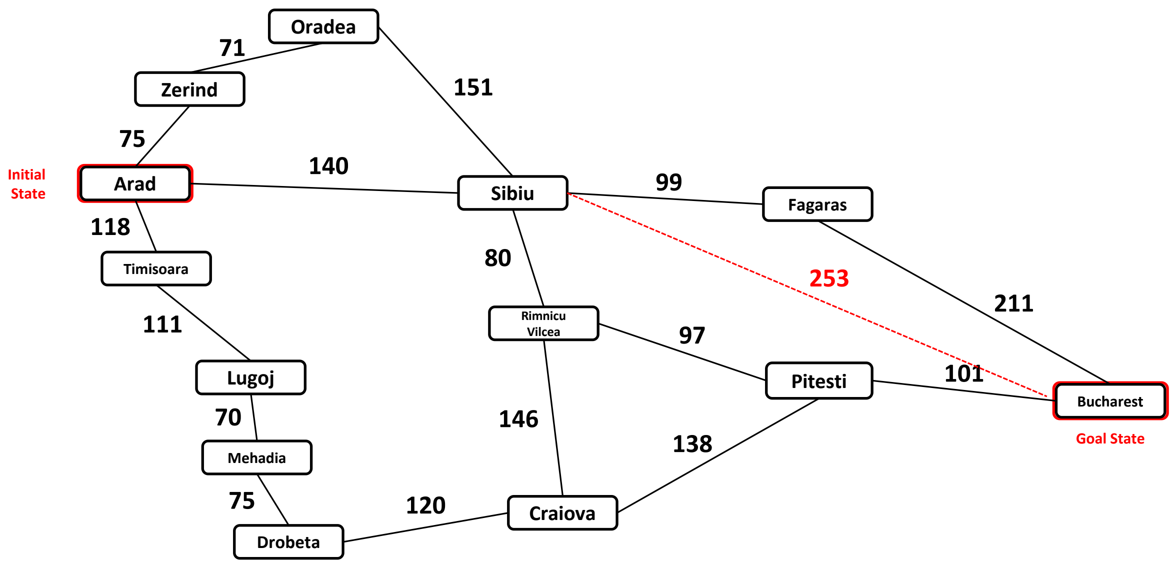


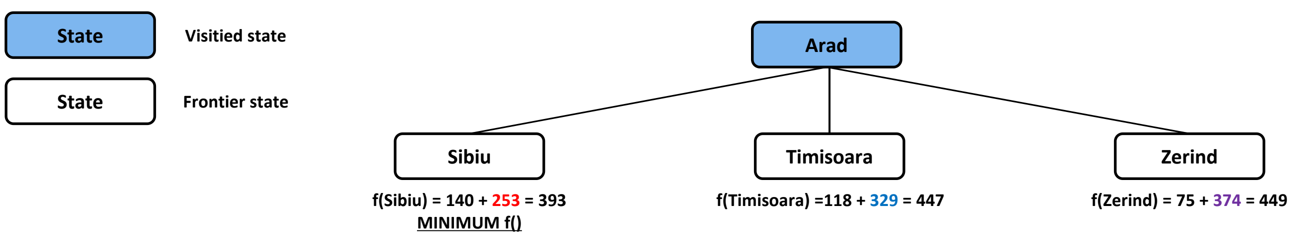


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

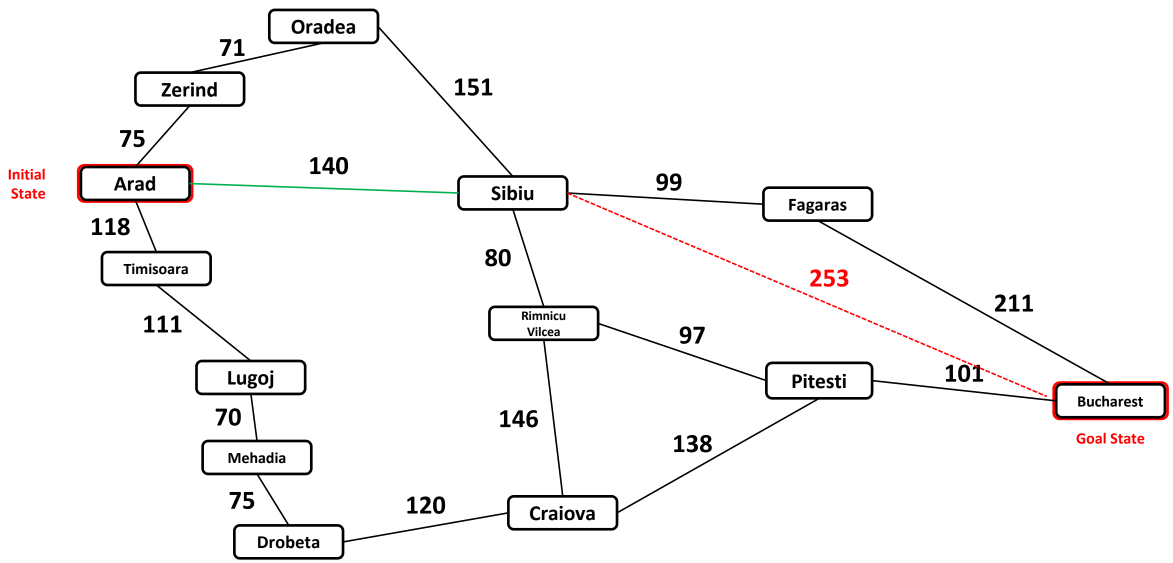


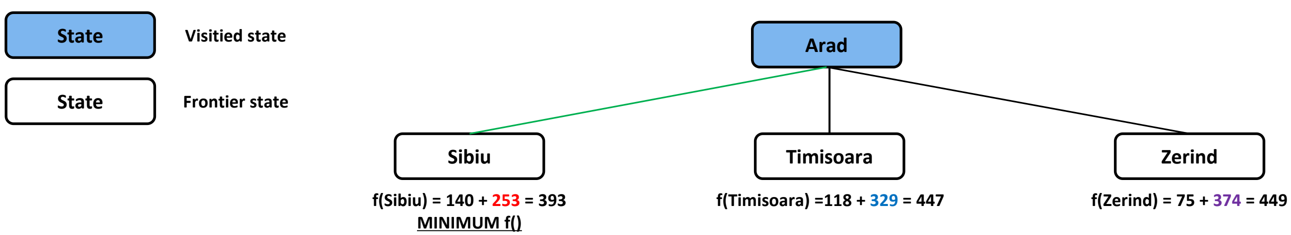


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

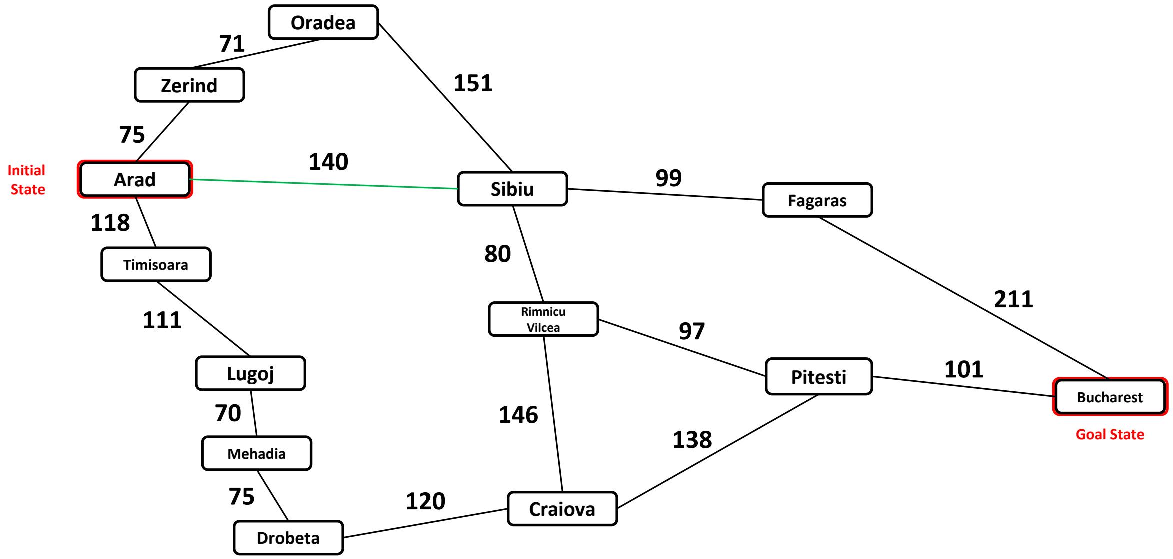


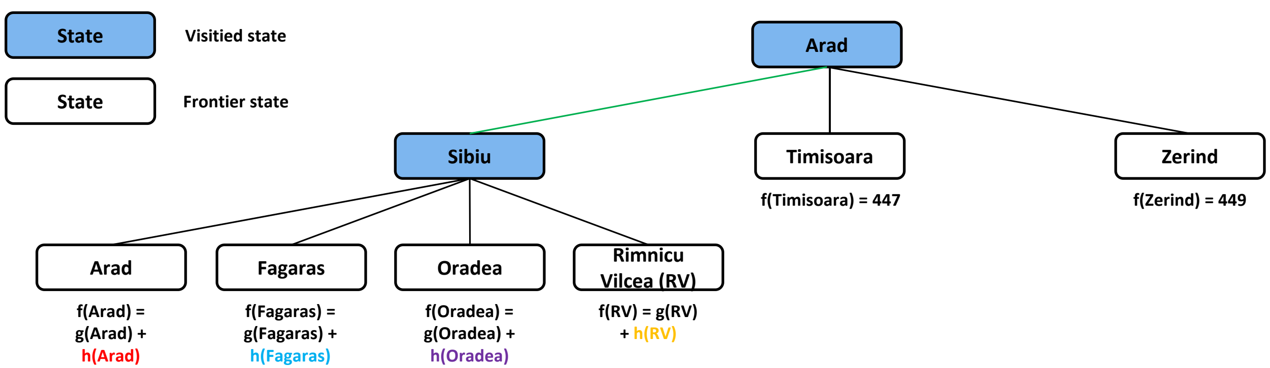


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

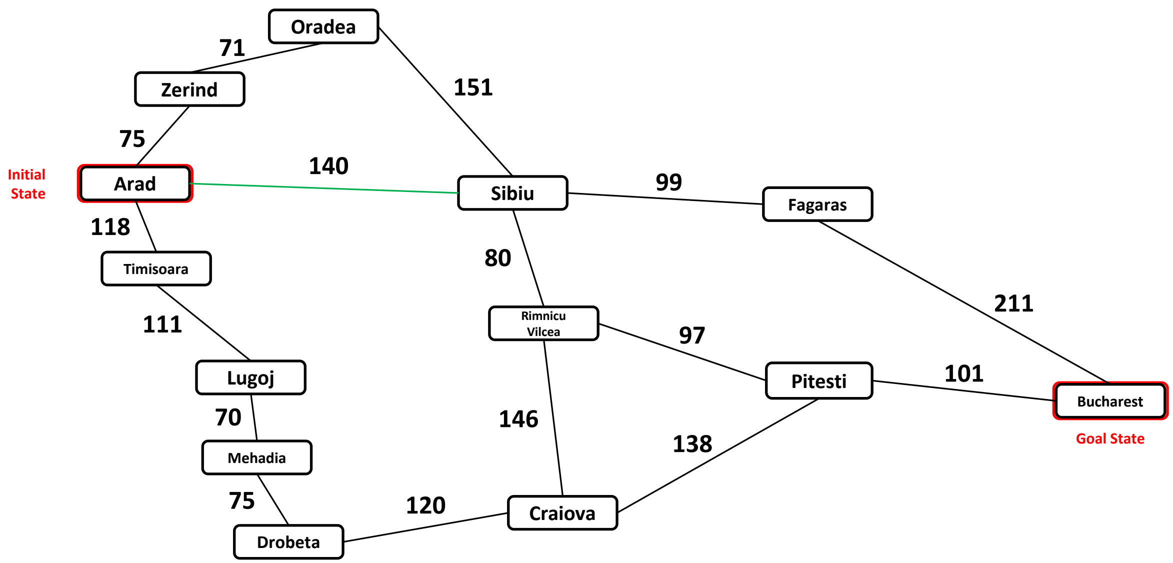


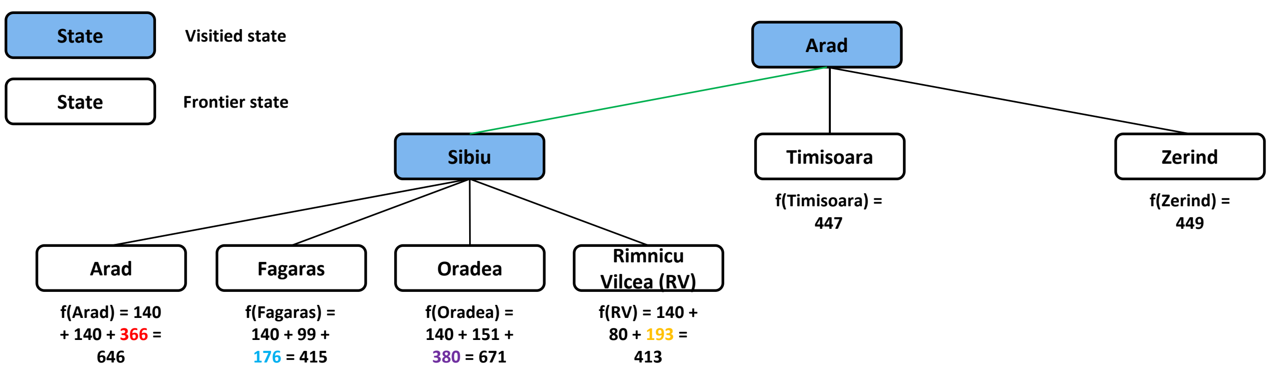


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: A* Search

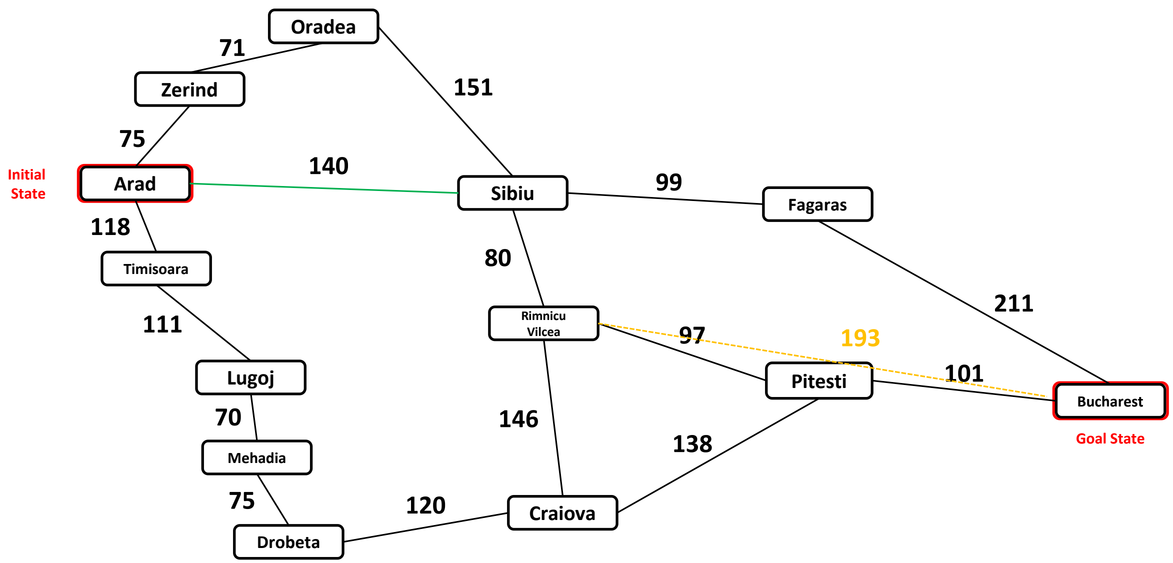


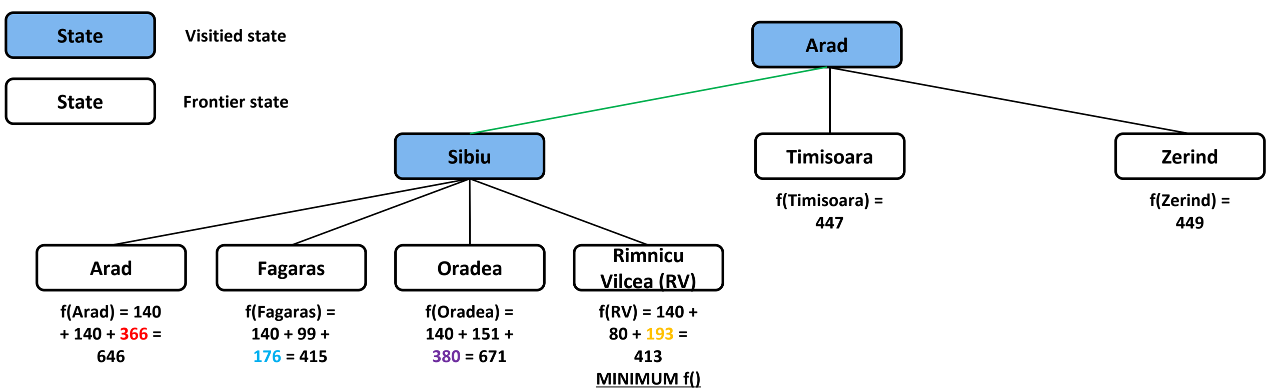


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: A* Search

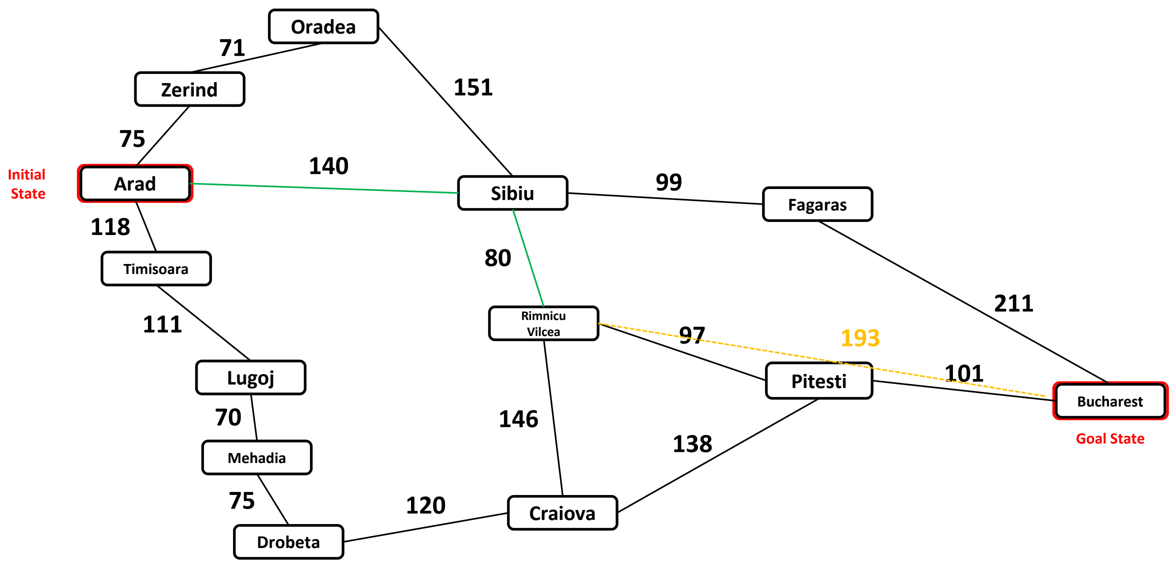


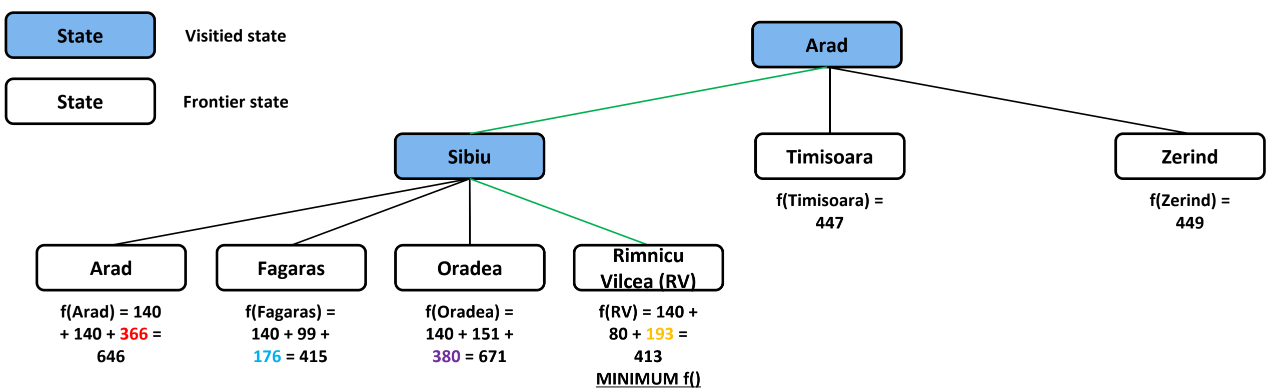


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

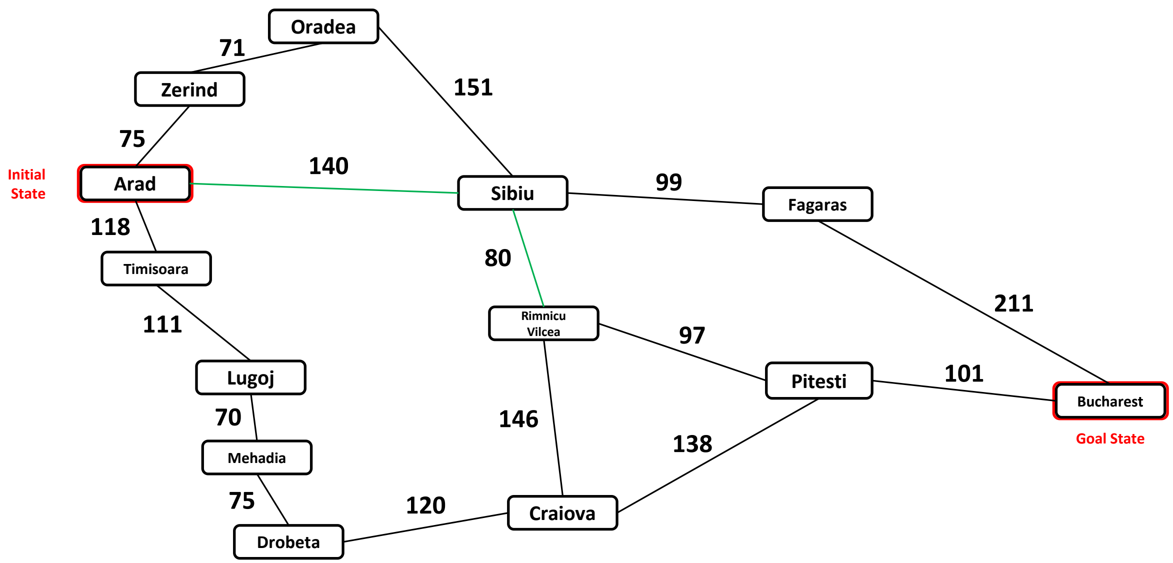


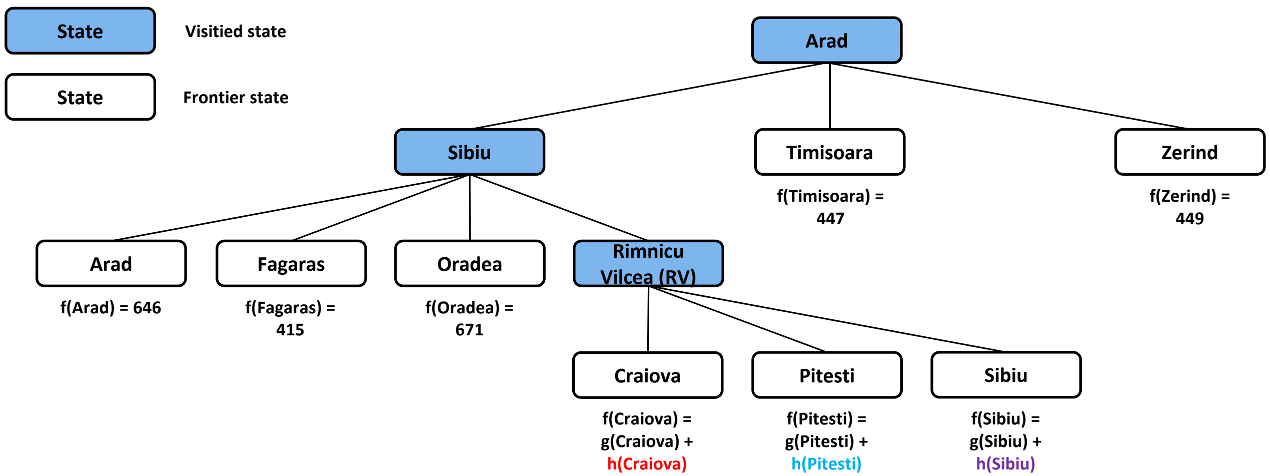


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: A* Search

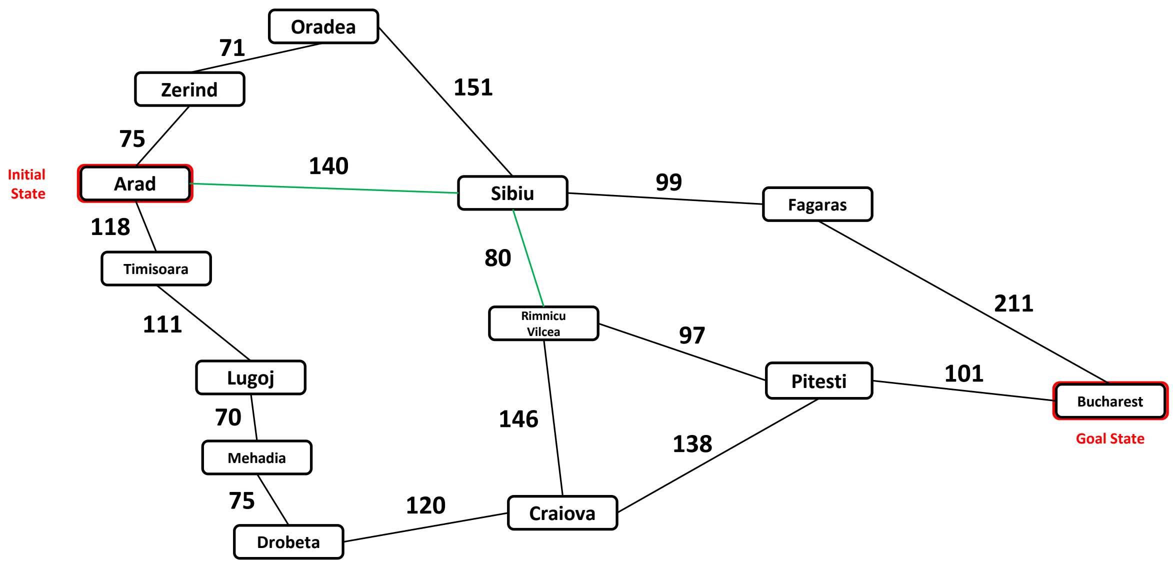


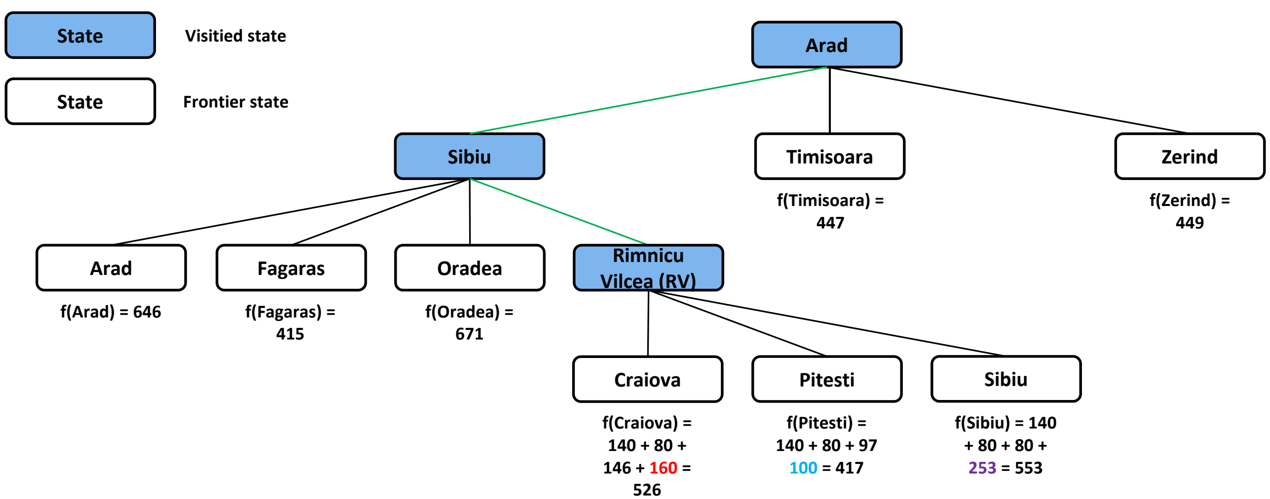


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

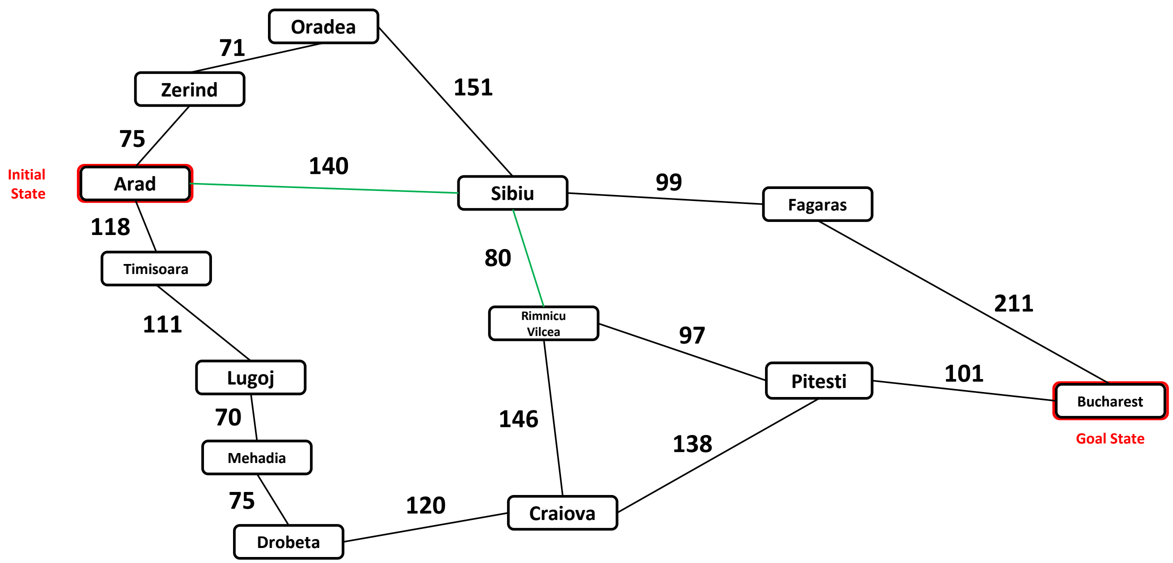


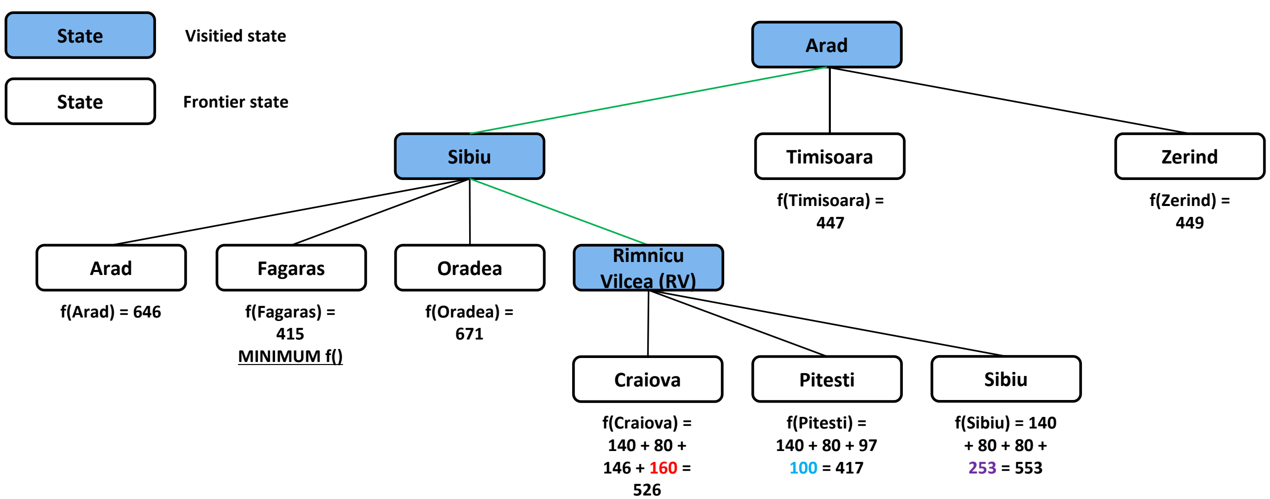


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

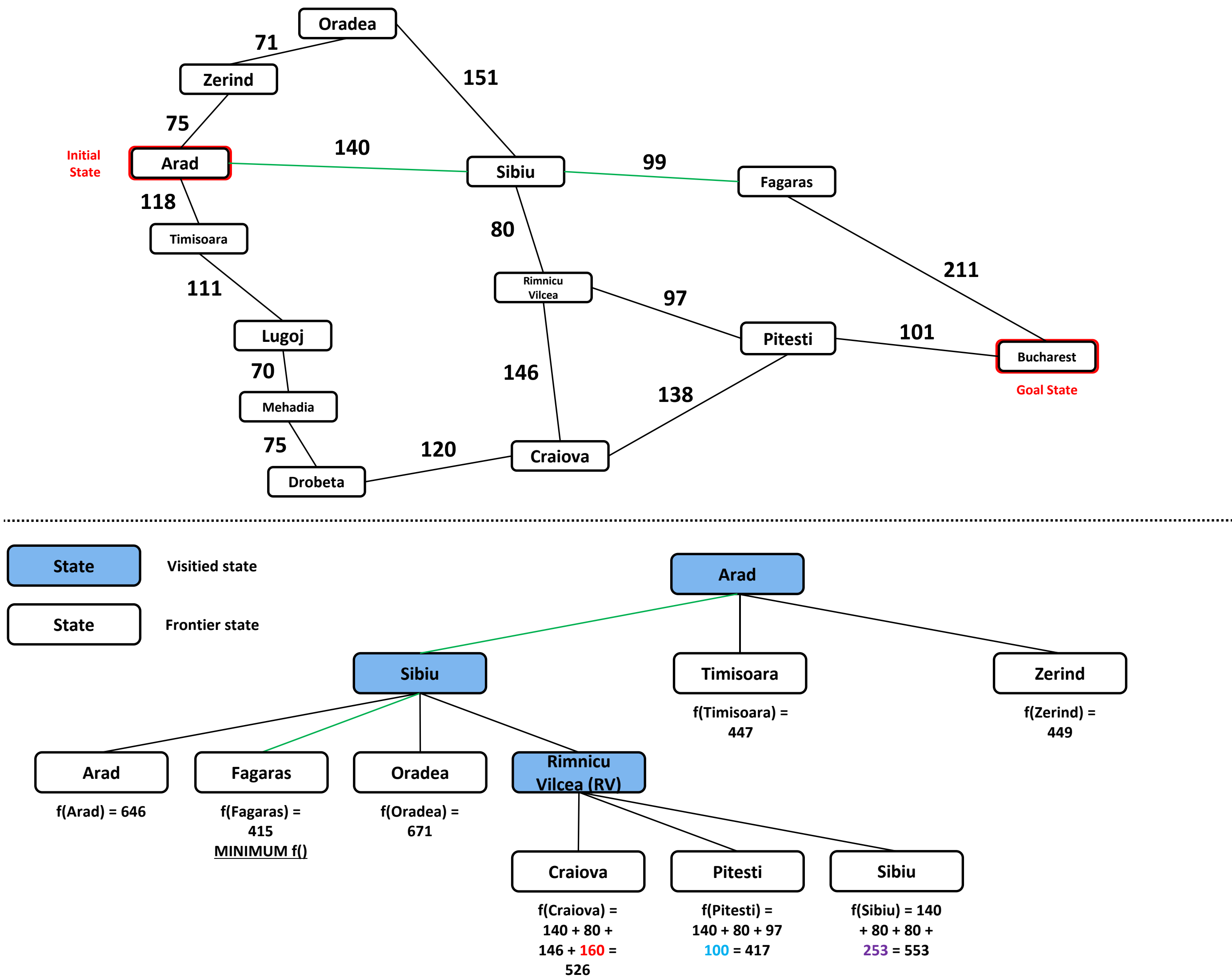




Straight-line distance to Bucharest (h(State)):

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

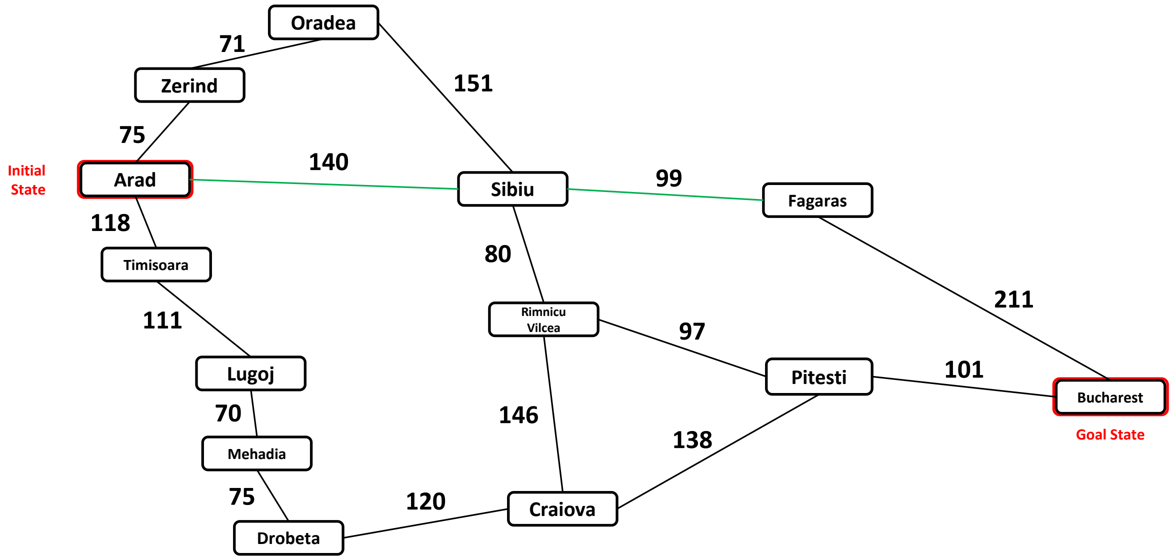
Dracula's Roadtrip: A* Search

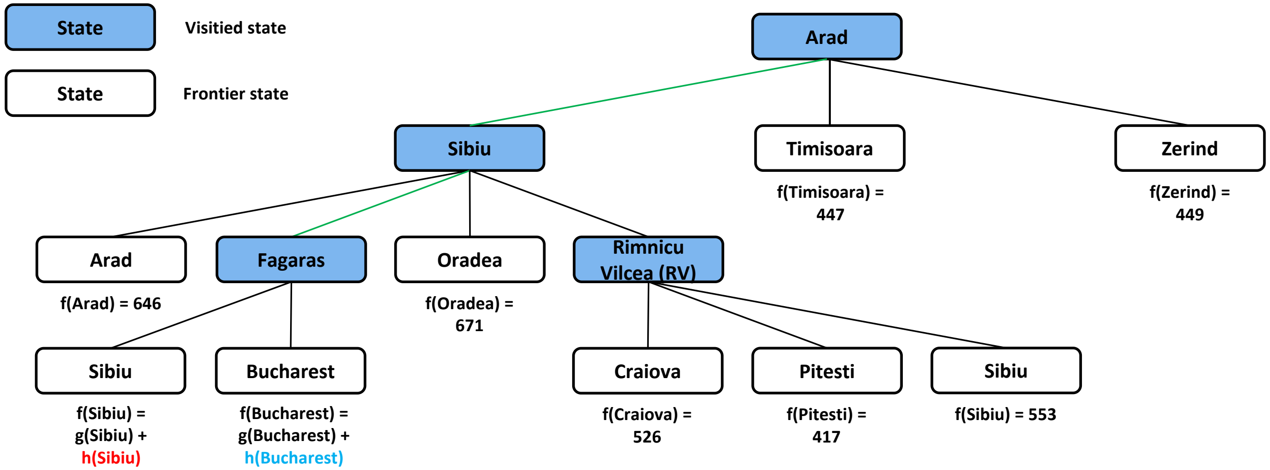


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

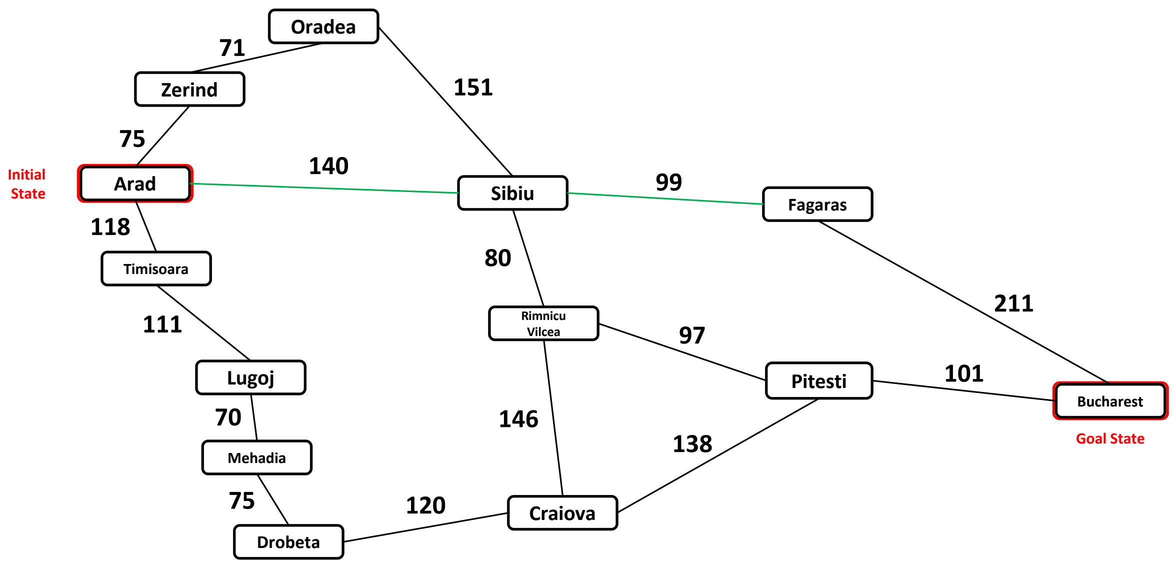


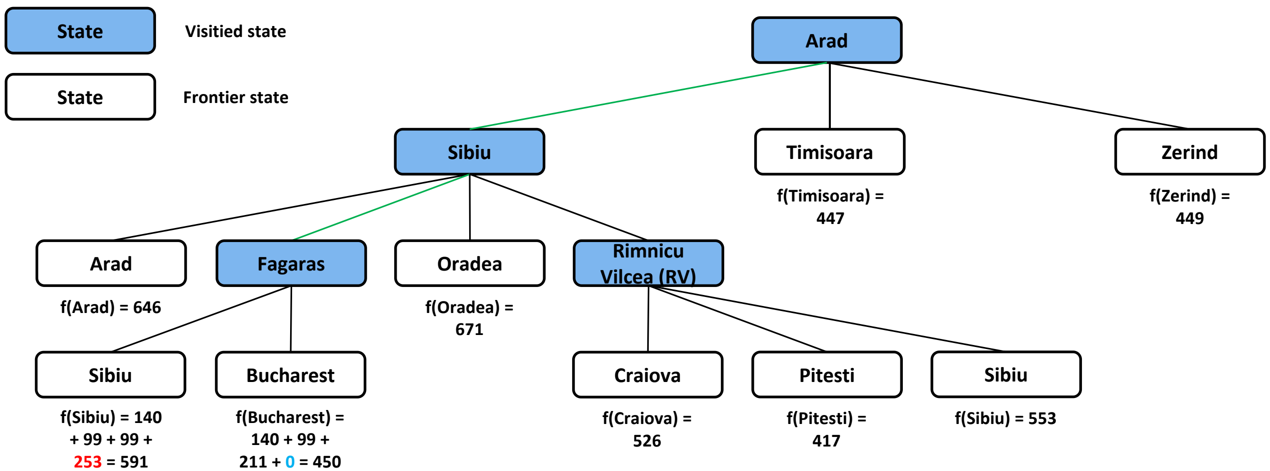


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

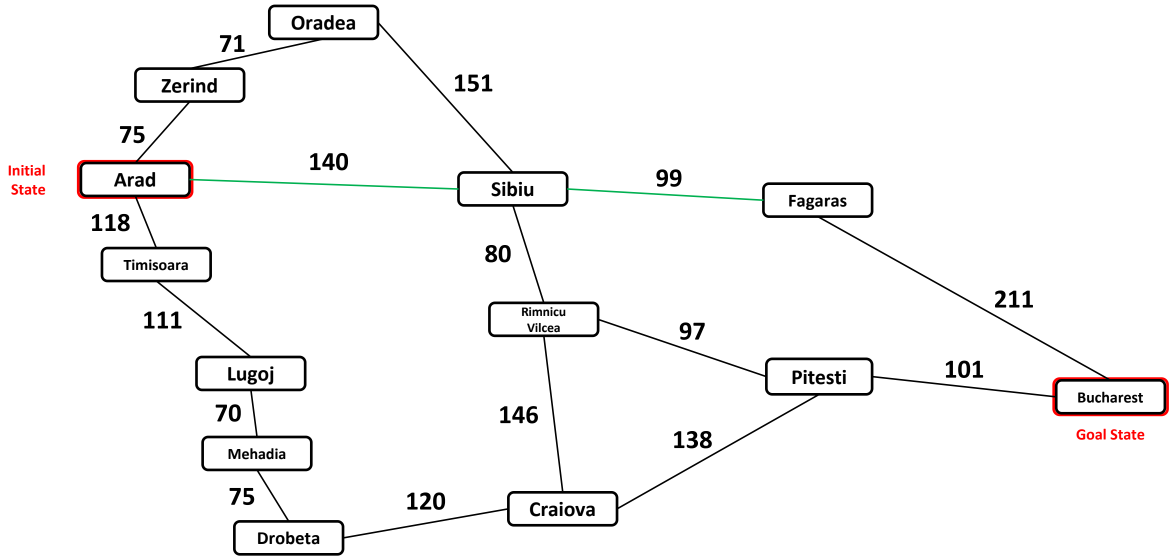


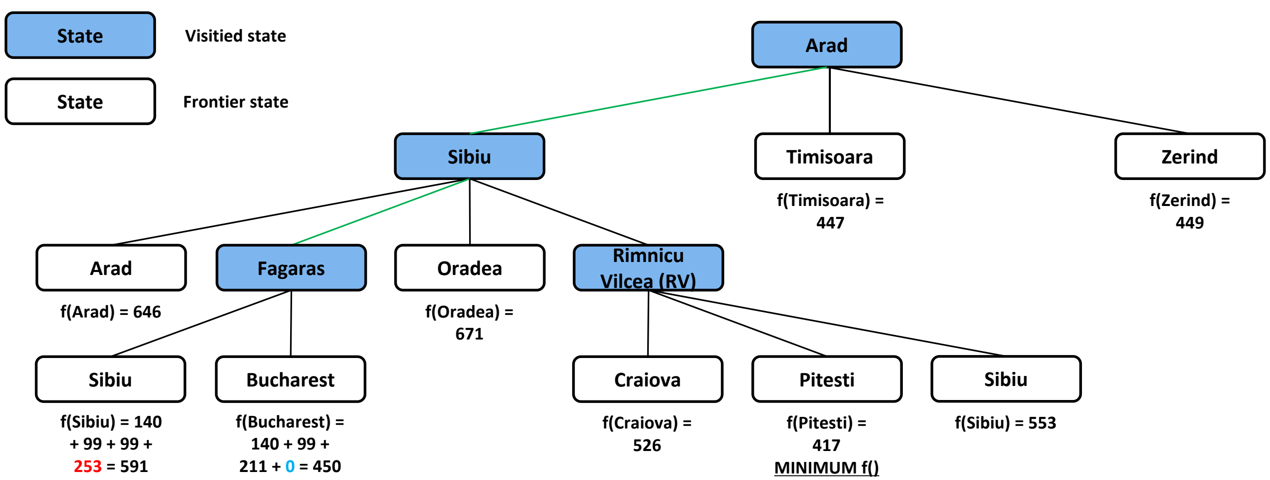


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

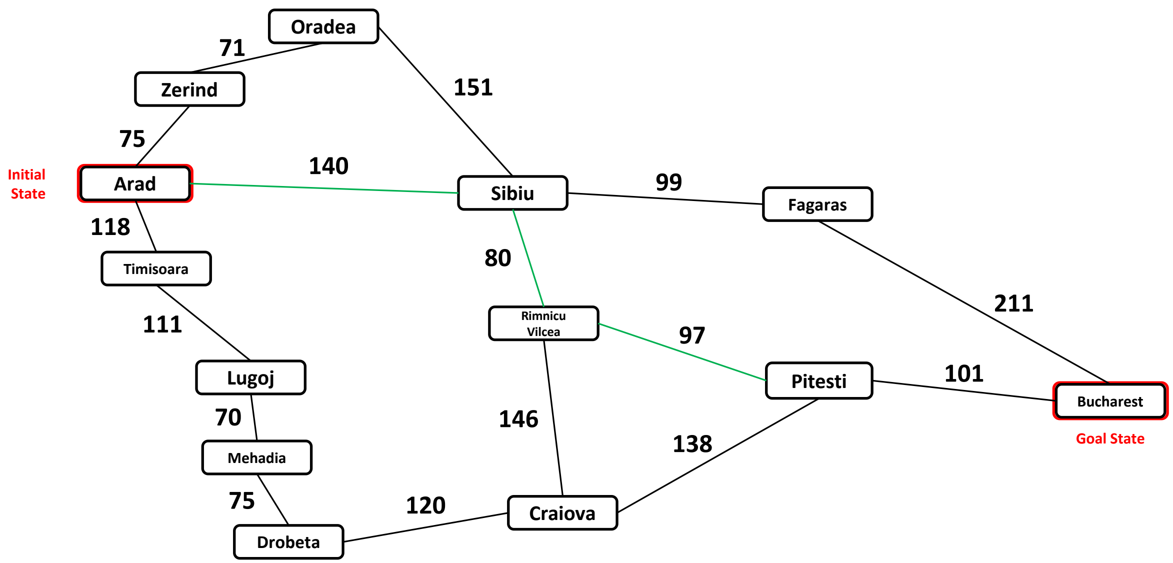


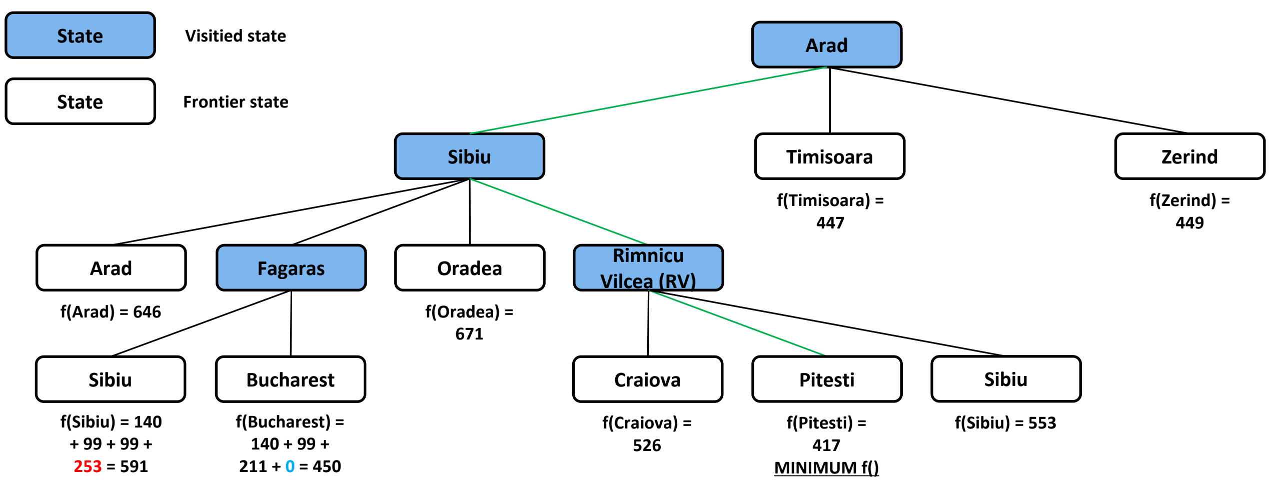


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

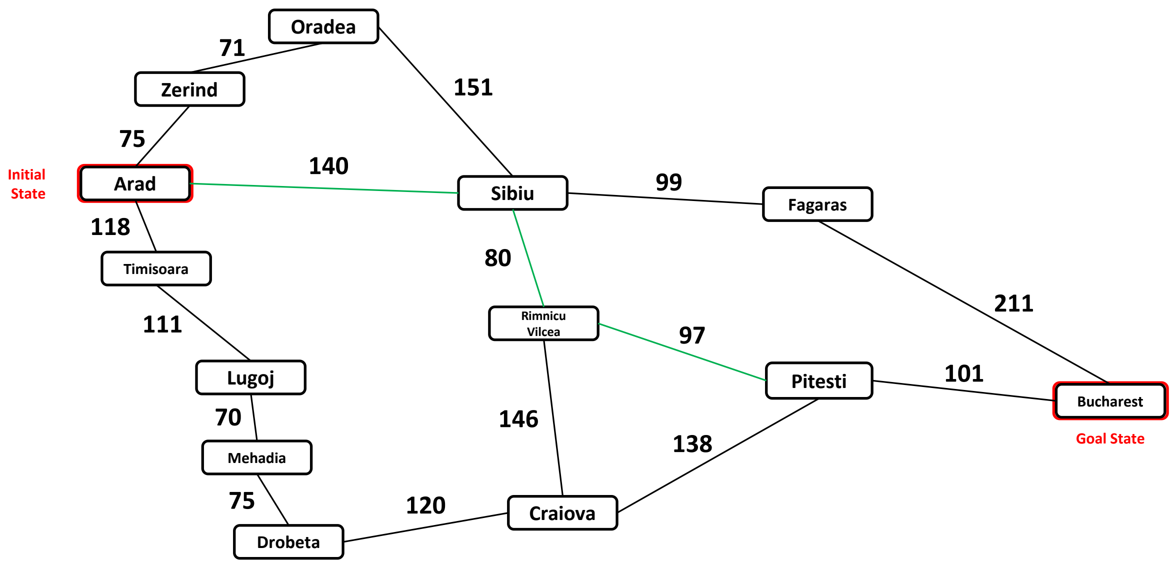


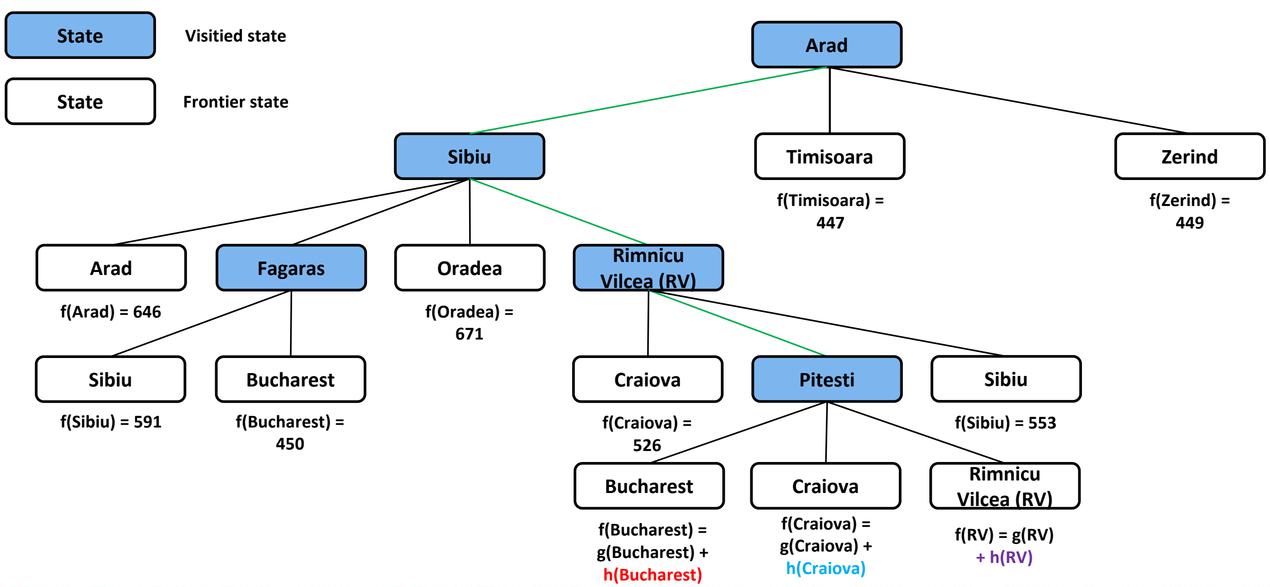


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

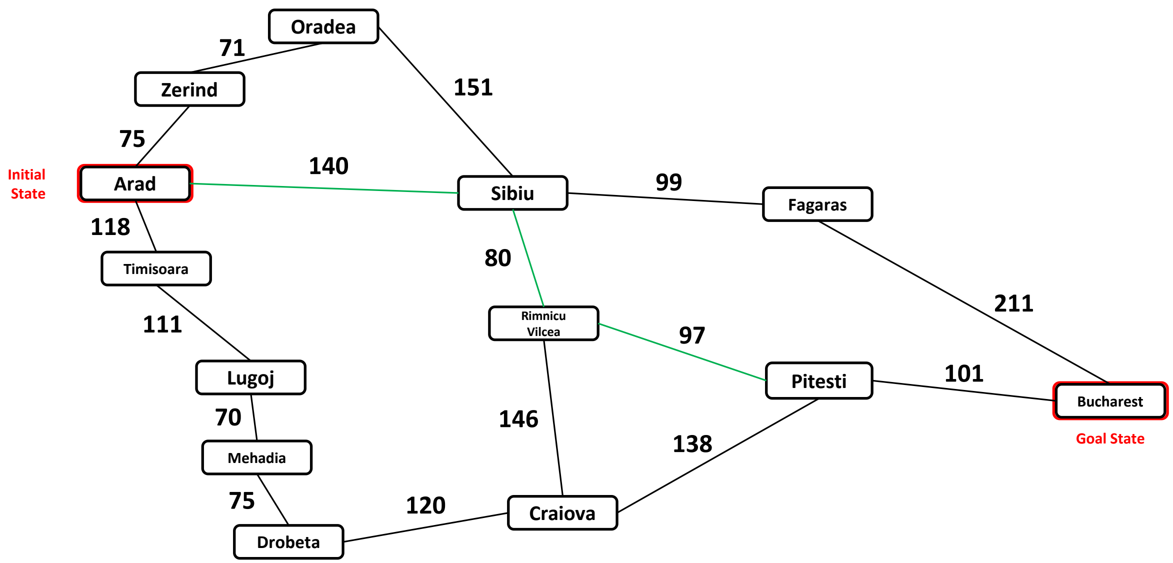


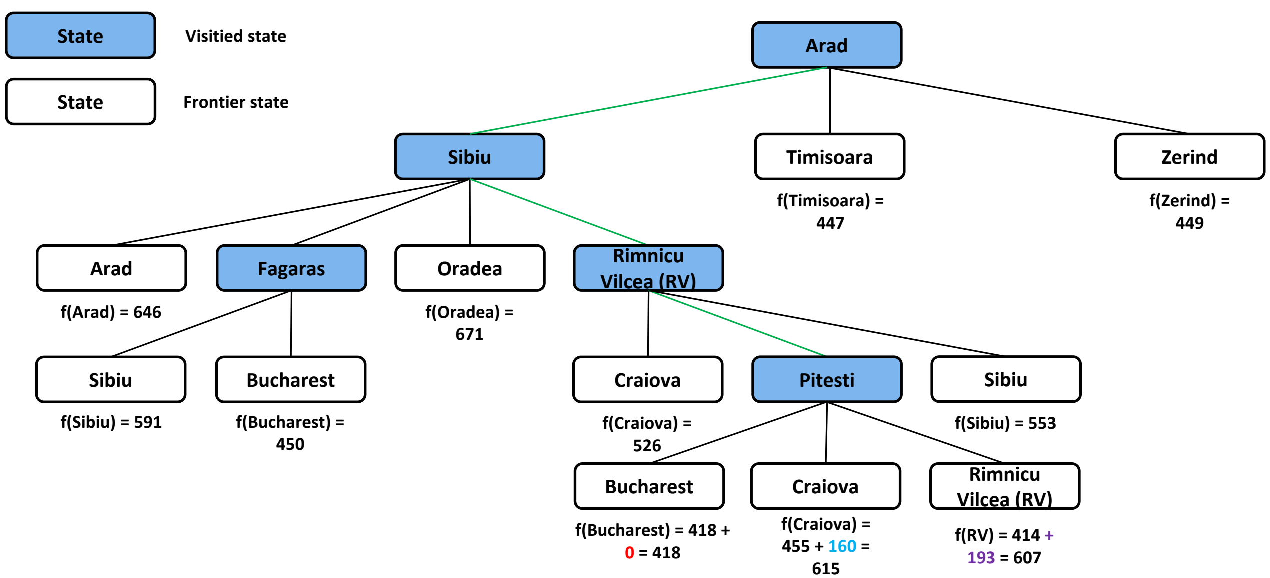


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

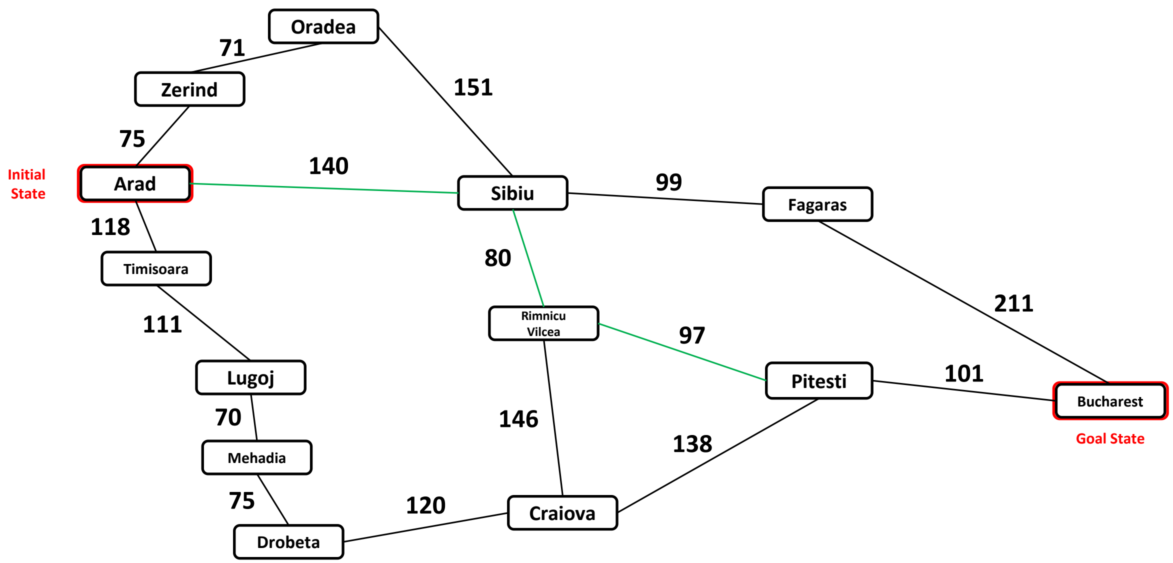


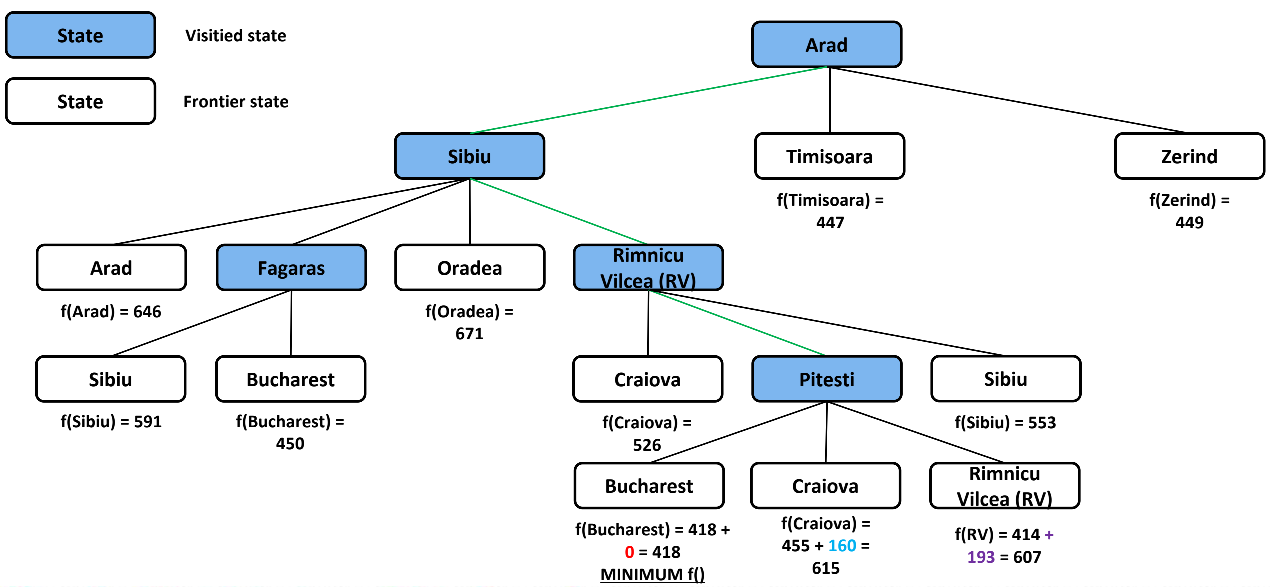


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: A* Search

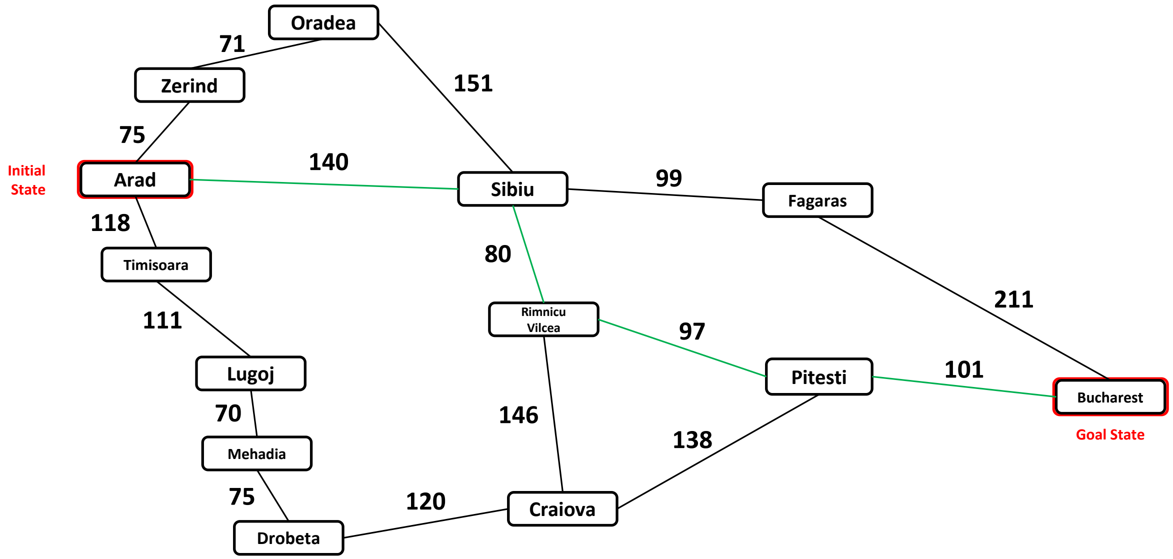


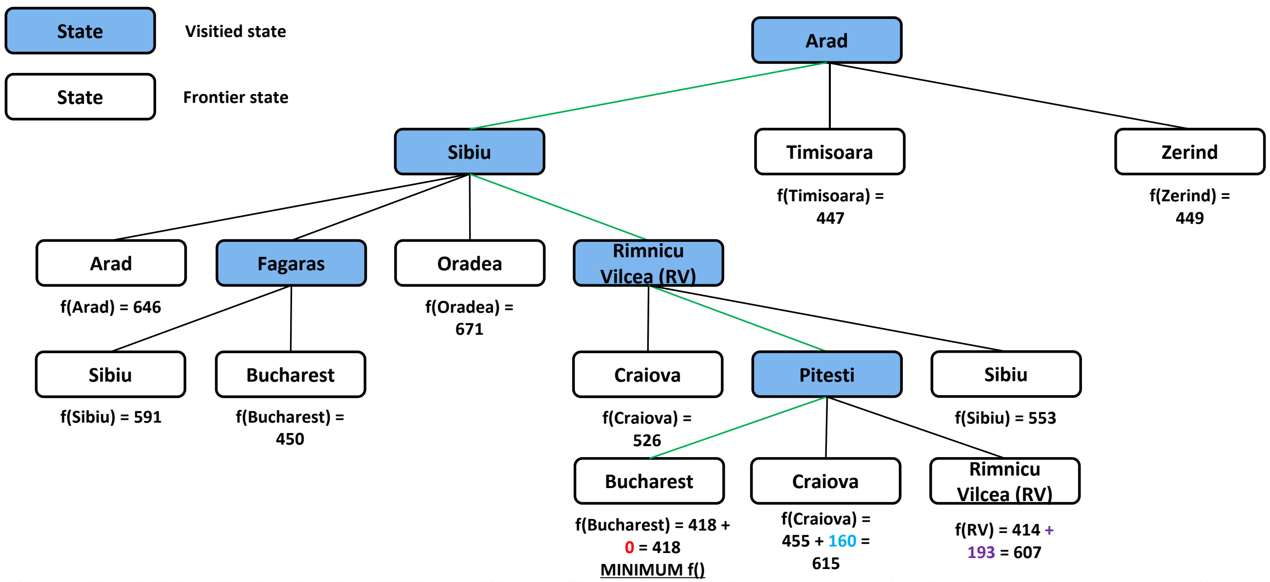


Straight-line distance to Bucharest (h(State)):

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

Dracula’s Roadtrip: A* Search

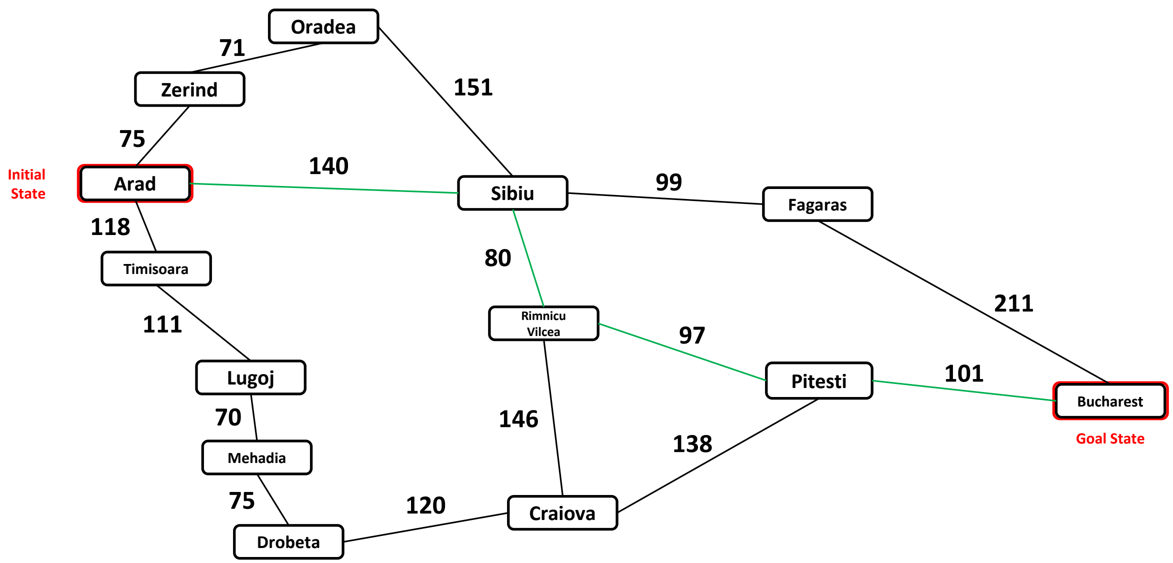


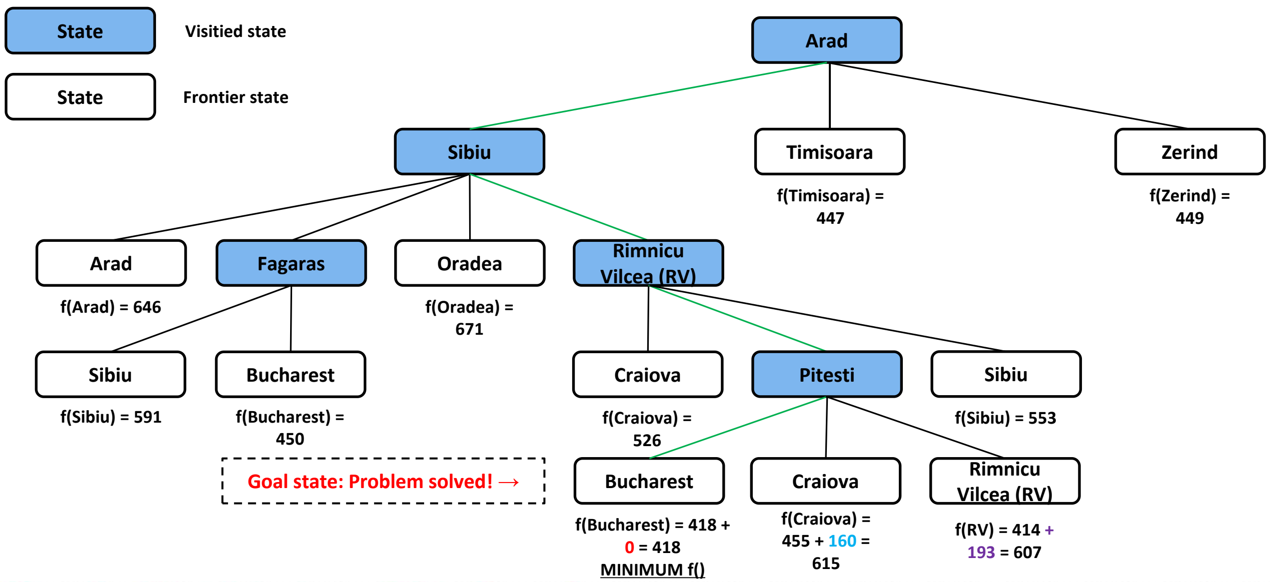


Straight-line distance to Bucharest (h(State)):

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

Dracula's Roadtrip: A* Search

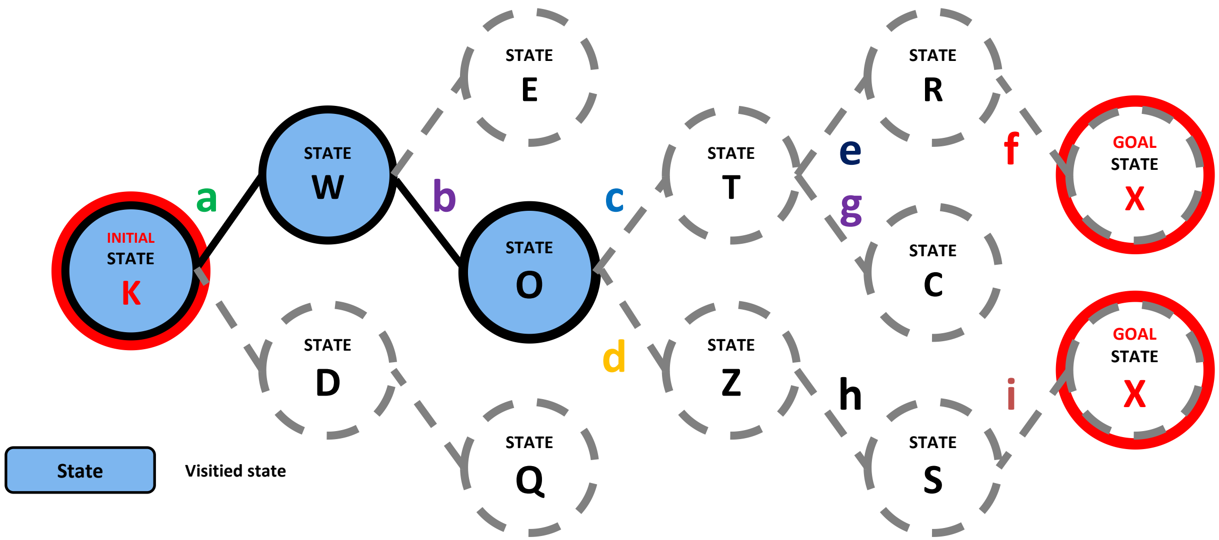




Straight-line distance to Bucharest (h(State)):

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

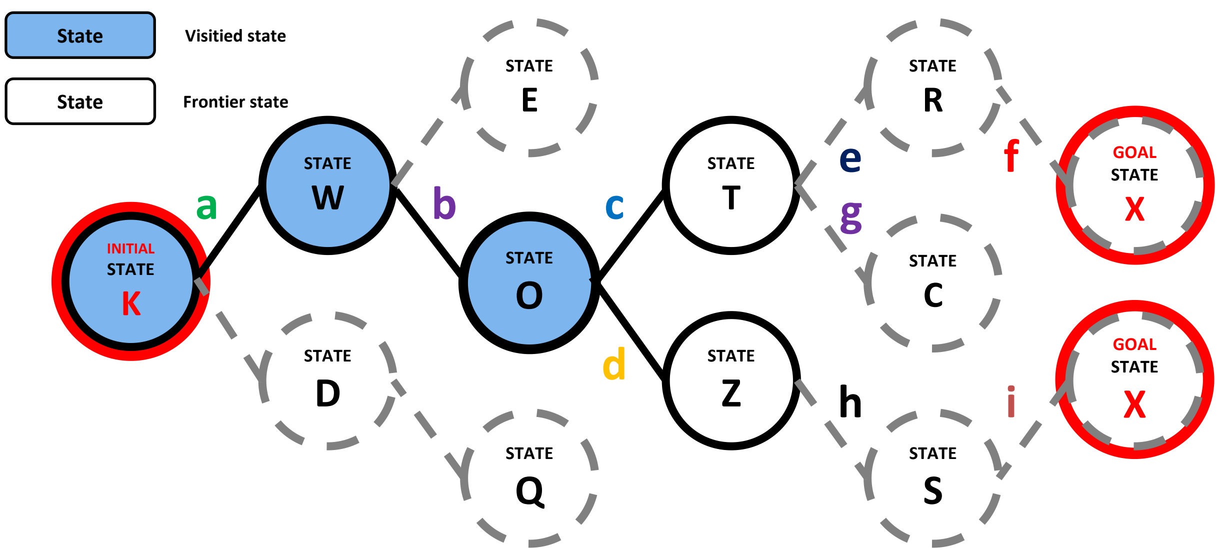
Hill Climbing Search vs. A* Search



Best First Search:
Go to T or Z?
f(T) = c

f(Z) = d

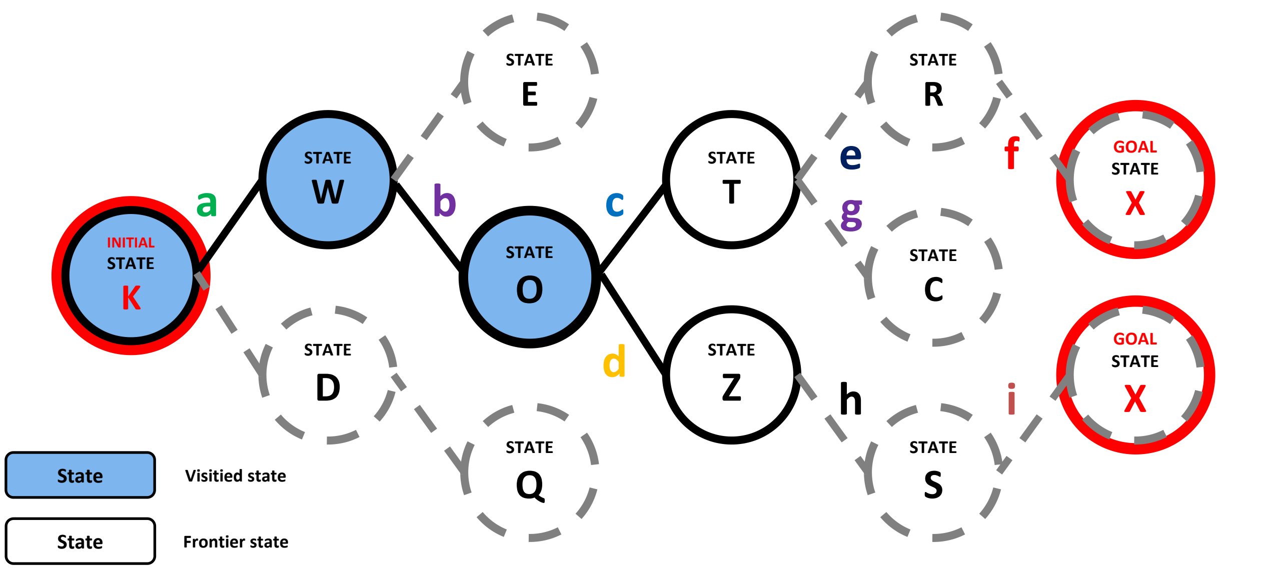
Pick state with min f()



A* Search:
Expand T or Z?
f(T) = g(T) + h(T)
f(T) = a + b + c + h(T)

f(Z) = g(Z) + h(Z)
f(T) = a + b + d + h(Z)
Pick state with min f()

Greedy Best First Search vs. A* Search



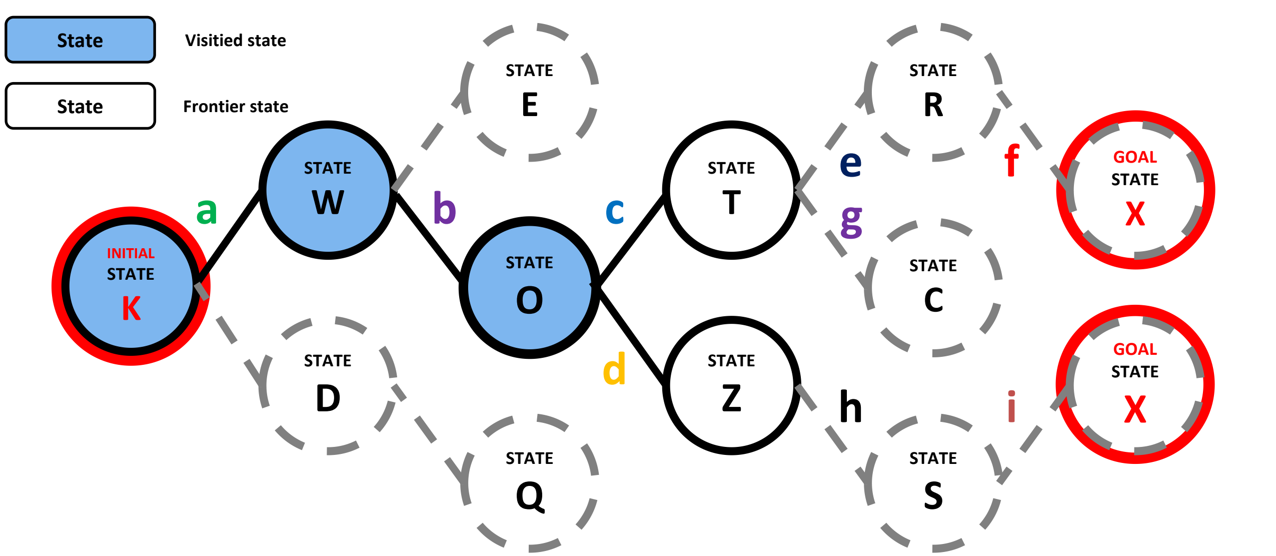
Greedy Best First:

Expand T or Z?

f(T) = h(T)

f(Z) = h(Z)

Pick state with min f()



A* Search:

Expand T or Z?

f(T) = g(T) + h(T)

f(T) = a + b + c + h(T)

f(Z) = g(Z) + h(Z)

f(T) = a + b + d + h(Z)

Pick state with min f()