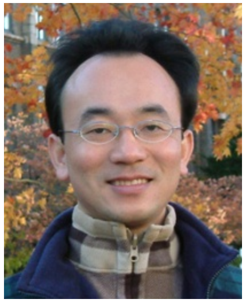
人工智慧
(Artificial Intelligence)
問題解決
(Problem Solving)

1092AI03
MBA, IM, NTPU (M5010) (Spring 2021)
Wed 2, 3, 4 (9:10-12:00) (B8F40)



Min-Yuh Day
戴敏育
Associate Professor
副教授
Institute of Information Management, National Taipei University
國立臺北大學 資訊管理研究所
https://web.ntpu.edu.tw/~myday

2021-03-09

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

1 2021/02/24 人工智慧概論
(Introduction to Artificial Intelligence)

2 2021/03/03 人工智慧和智慧代理人
(Artificial Intelligence and Intelligent Agents)

3 2021/03/10 問題解決
(Problem Solving)

4 2021/03/17 知識推理和知識表達
(Knowledge, Reasoning and Knowledge Representation)

5 2021/03/24 不確定知識和推理
(Uncertain Knowledge and Reasoning)

6 2021/03/31 人工智慧個案研究 I
(Case Study on Artificial Intelligence I)

課程大綱 (Syllabus)

週次 (Week)　日期 (Date)　內容 (Subject/Topics)

7 2021/04/07 放假一天 (Day off)

8 2021/04/14 機器學習與監督式學習
(Machine Learning and Supervised Learning)

9 2021/04/21 期中報告
(Midterm Project Report)

10 2021/04/28 學習理論與綜合學習
(The Theory of Learning and Ensemble Learning)

11 2021/05/05 深度學習
(Deep Learning)

12 2021/05/12 人工智慧個案研究 II
(Case Study on Artificial Intelligence II)

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

13 2021/05/19 強化學習
(Reinforcement Learning)

14 2021/05/26 深度學習自然語言處理
(Deep Learning for Natural Language Processing)

15 2021/06/02 機器人技術
(Robotics)

16 2021/06/09 人工智慧哲學與倫理，人工智慧的未來
(Philosophy and Ethics of AI, The Future of AI)

17 2021/06/16 期末報告 I
(Final Project Report I)

18 2021/06/23 期末報告 II
(Final Project Report II)

Artificial Intelligence
Problem Solving

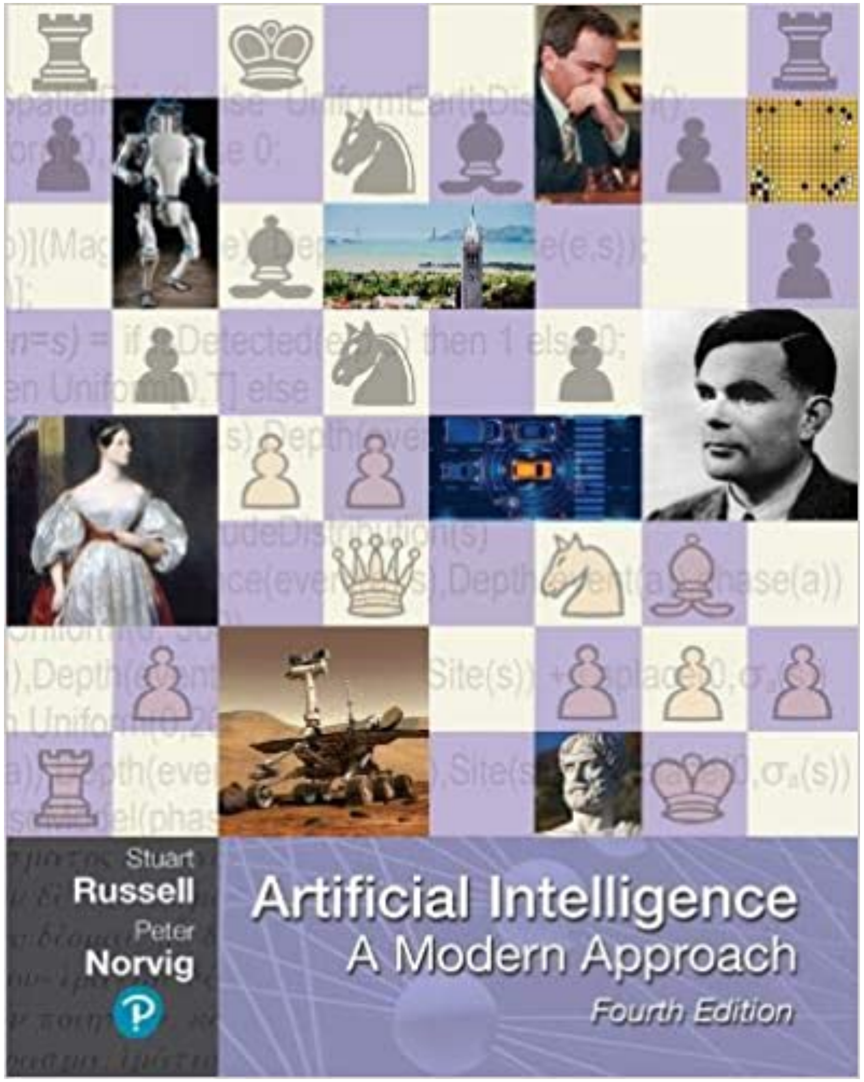
Outline

- **Solving Problems by Searching**
- **Search in Complex Environments**
- **Adversarial Search and Games**
- **Constraint Satisfaction Problems**

Stuart Russell and Peter Norvig (2020),

Artificial Intelligence: A Modern Approach,

4th Edition, Pearson



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

https://www.amazon.com/Artificial-Intelligence-A-Modern-Approach/dp/0134610997/

Artificial Intelligence: A Modern Approach

1. Artificial Intelligence
2. Problem Solving
3. Knowledge and Reasoning
4. Uncertain Knowledge and Reasoning
5. Machine Learning
6. Communicating, Perceiving, and Acting
7. Philosophy and Ethics of AI

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Artificial Intelligence: Problem Solving

Artificial Intelligence:
2. Problem Solving

- Solving Problems by Searching
- Search in Complex Environments
- Adversarial Search and Games
- Constraint Satisfaction Problems

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

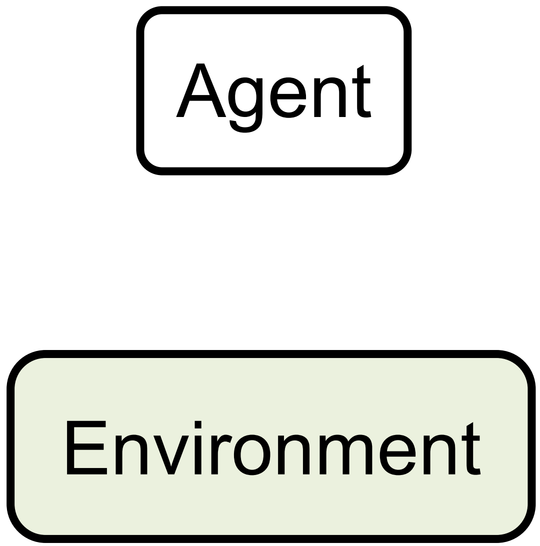
Intelligent Agents

4 Approaches of AI

2. Thinking Humanly: The Cognitive Modeling Approach	3. Thinking Rationally: The “Laws of Thought” Approach
1. Acting Humanly: The Turing Test Approach (1950)	4. Acting Rationally: The Rational Agent Approach

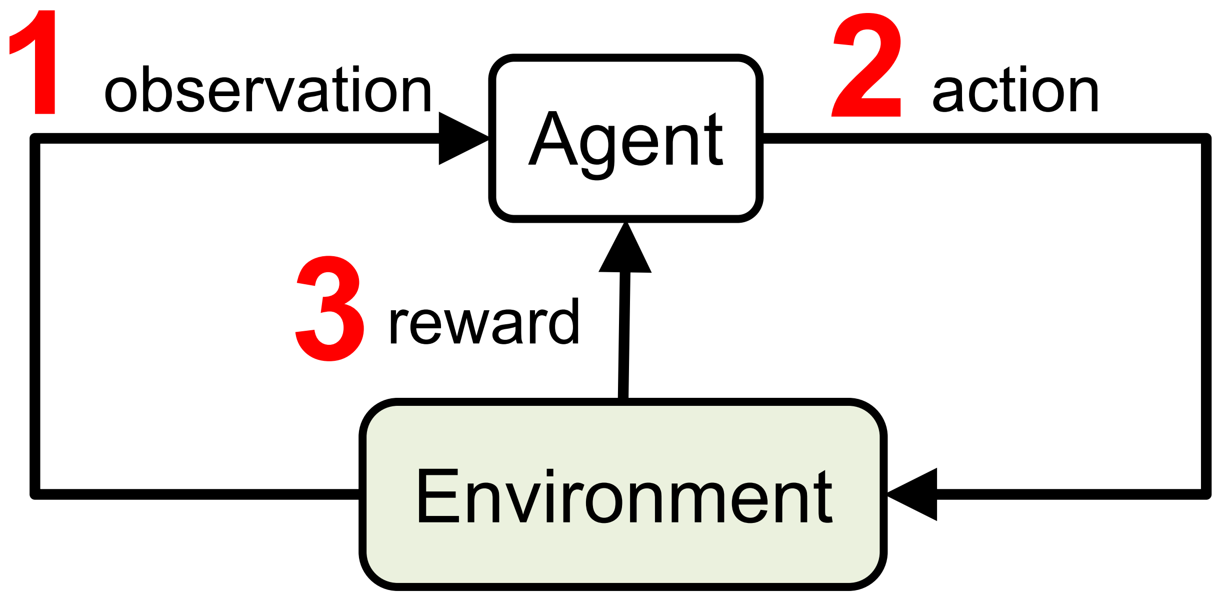
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Reinforcement Learning (DL)



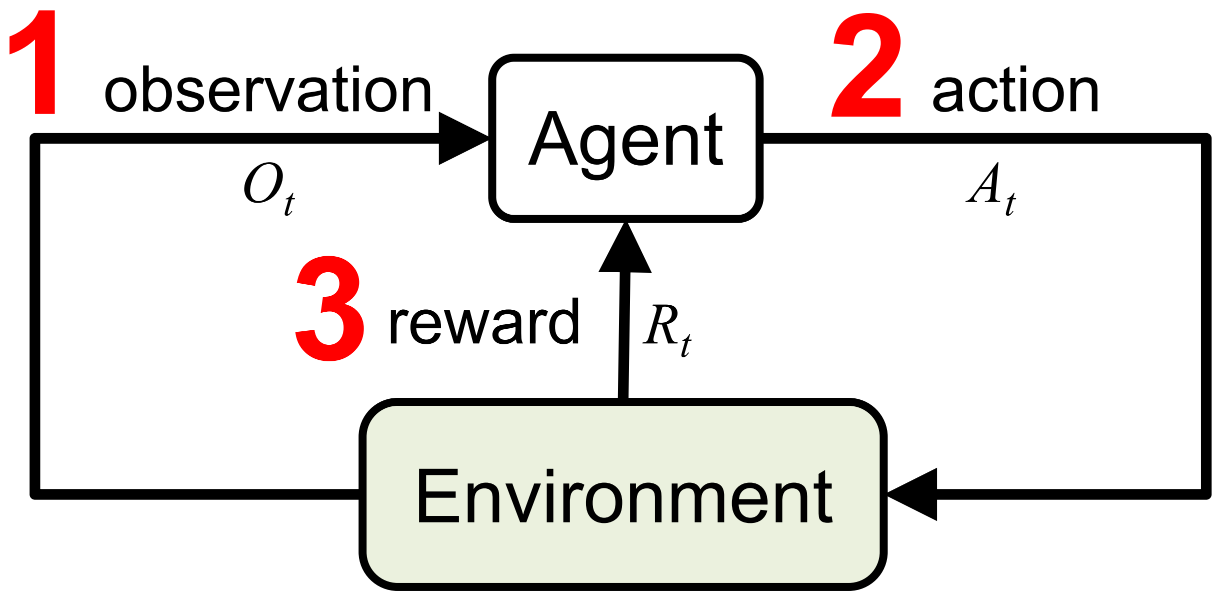
Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford

Reinforcement Learning (DL)



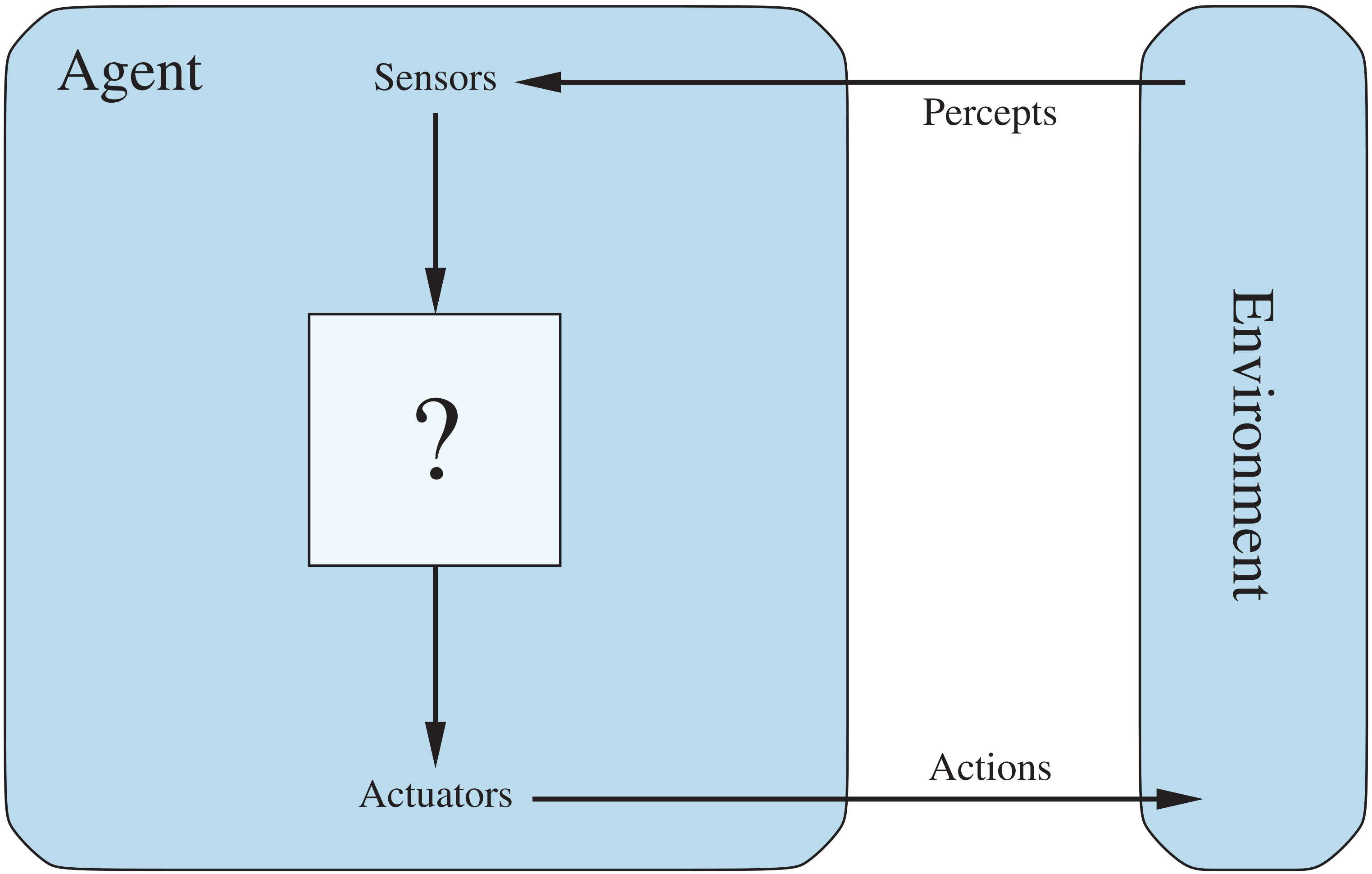
Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford

Reinforcement Learning (DL)



Source: Richard S. Sutton & Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd Edition, A Bradford

Agents interact with environments through sensors and actuators



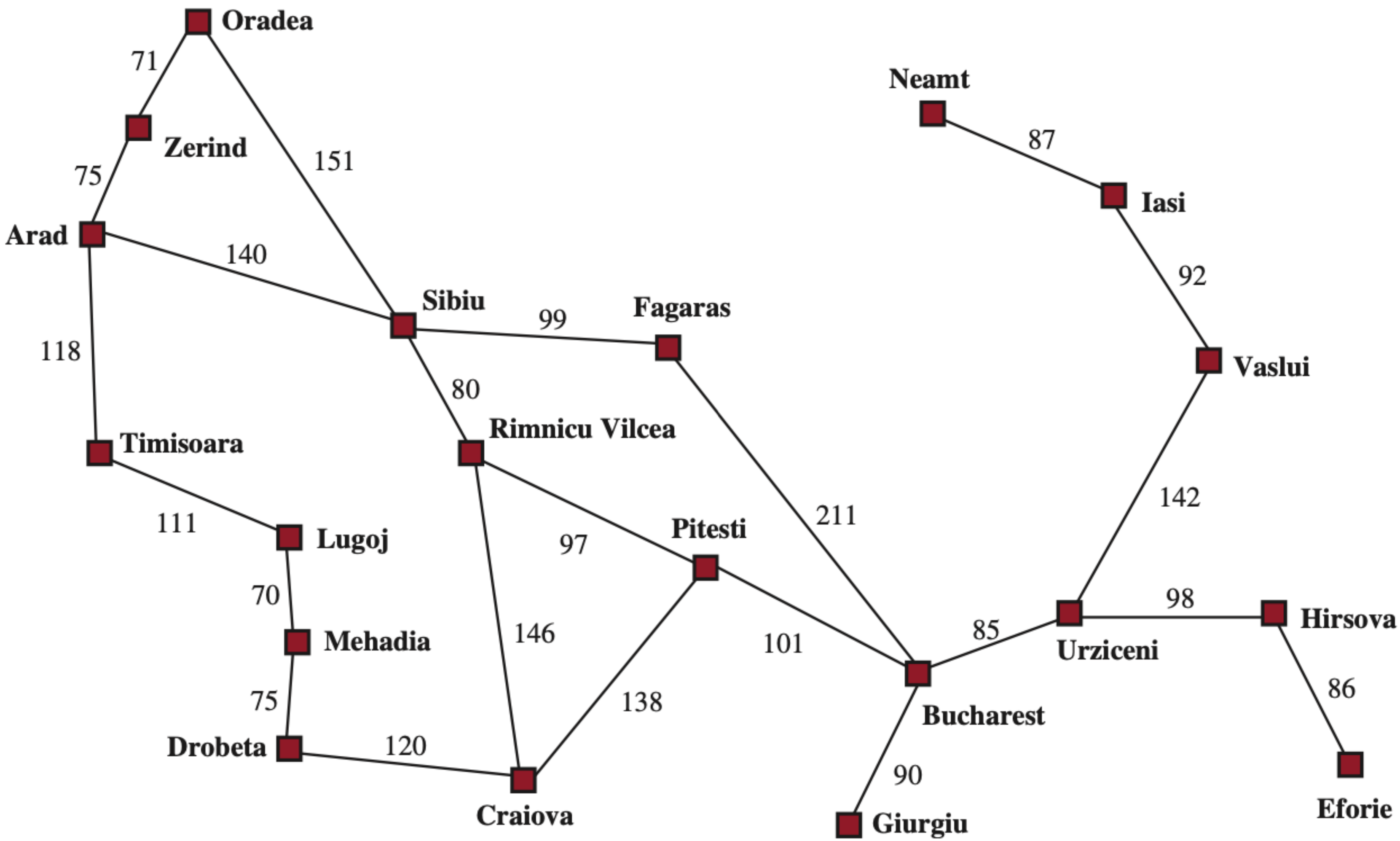
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Solving Problems by Searching

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

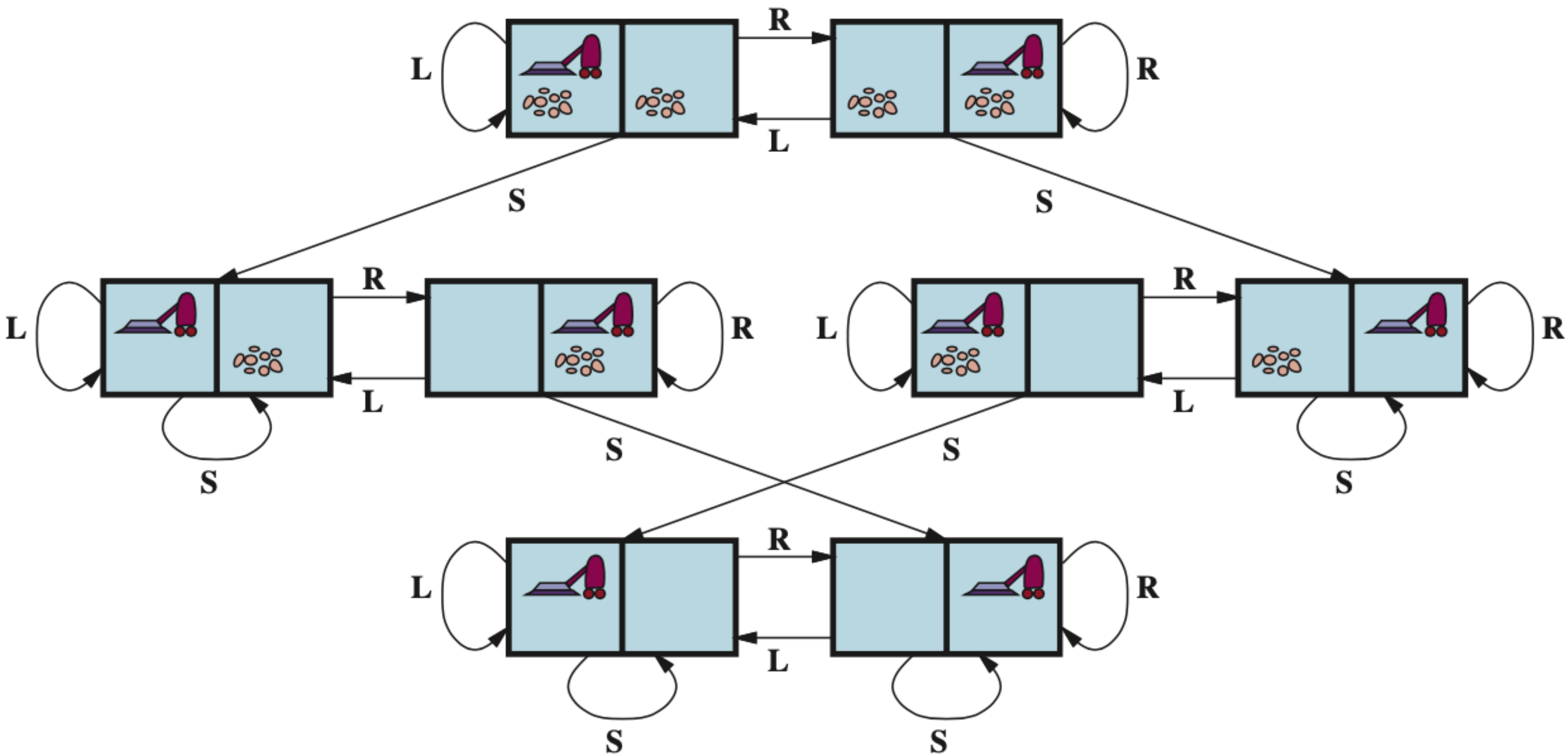
AI: Solving Problems by Searching

A simplified road map of part of Romania, with road distances in miles.



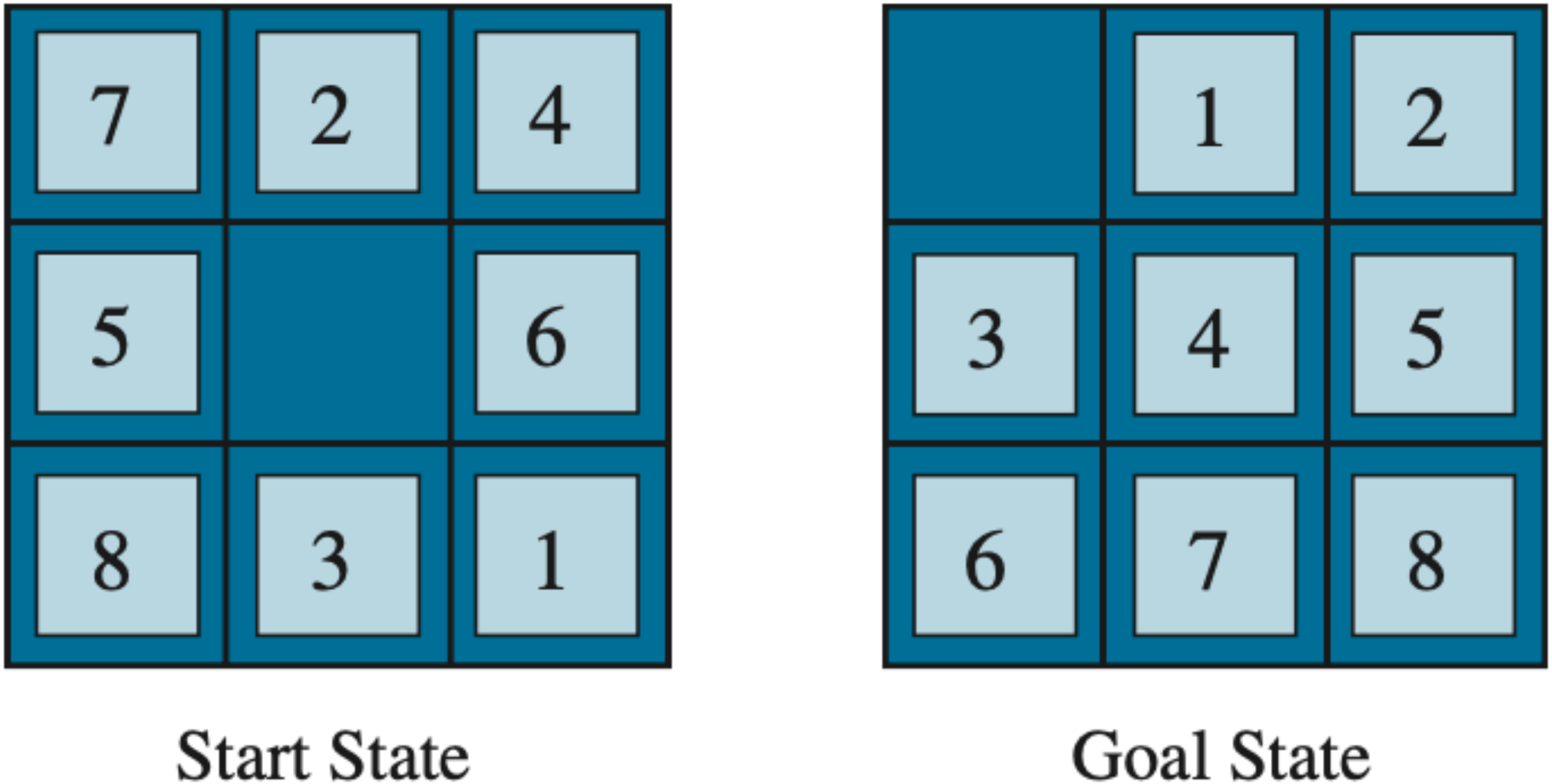
The state-space graph for the two-cell vacuum world

There are 8 states and three actions for each state:
L = *Left*, R = *Right*, S = *Suck*.



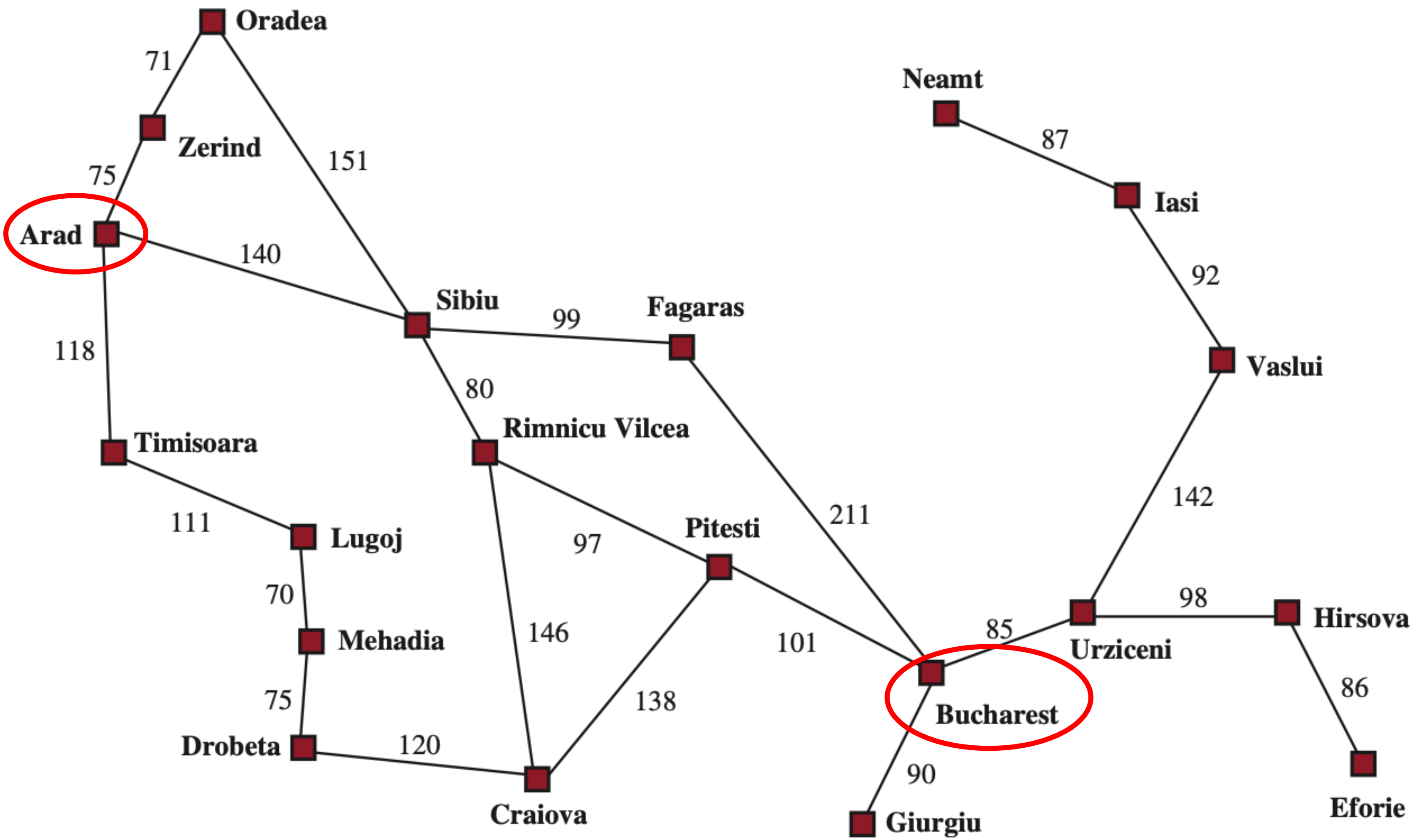
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A typical instance of the 8-puzzle



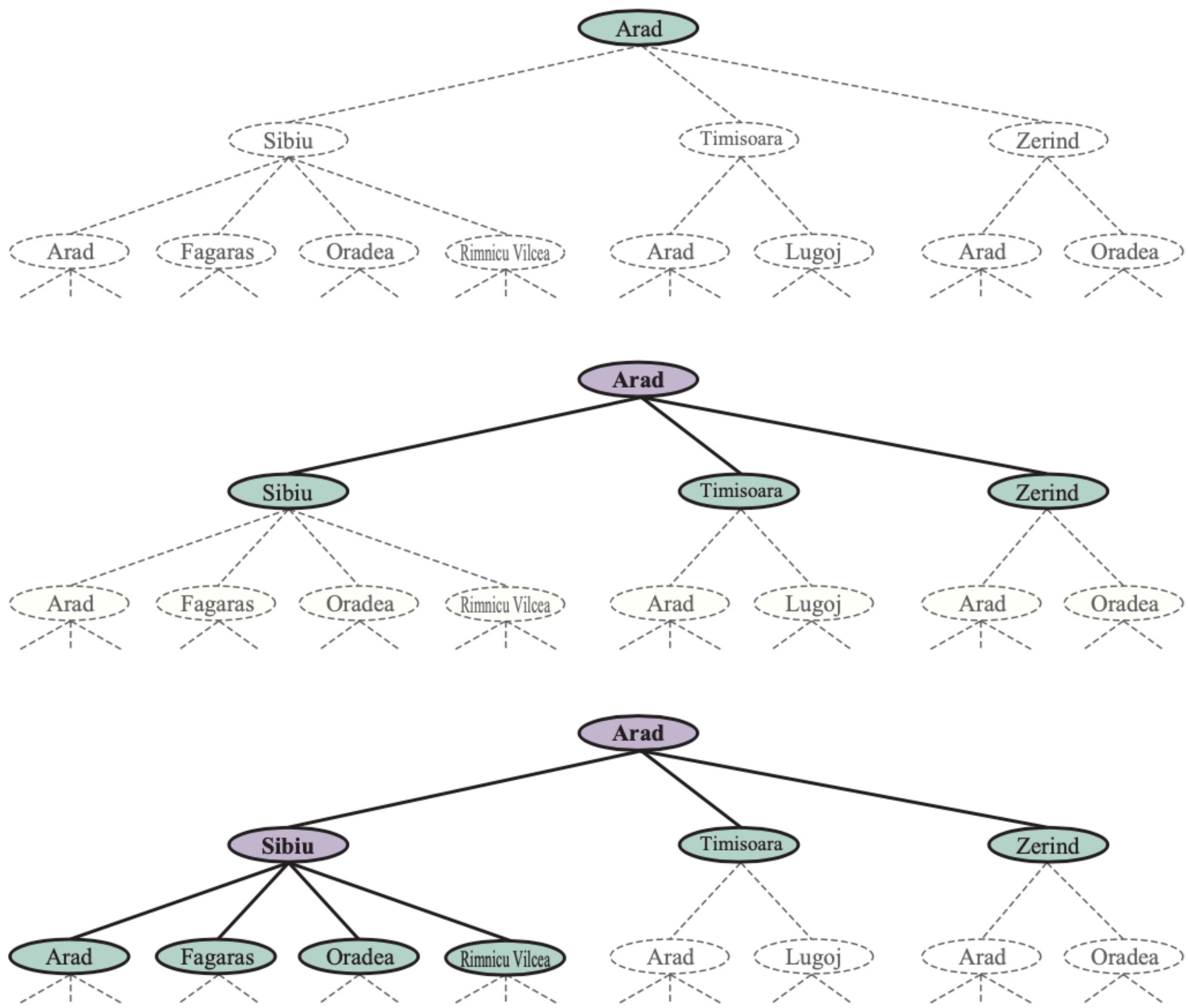
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Arad to Bucharest



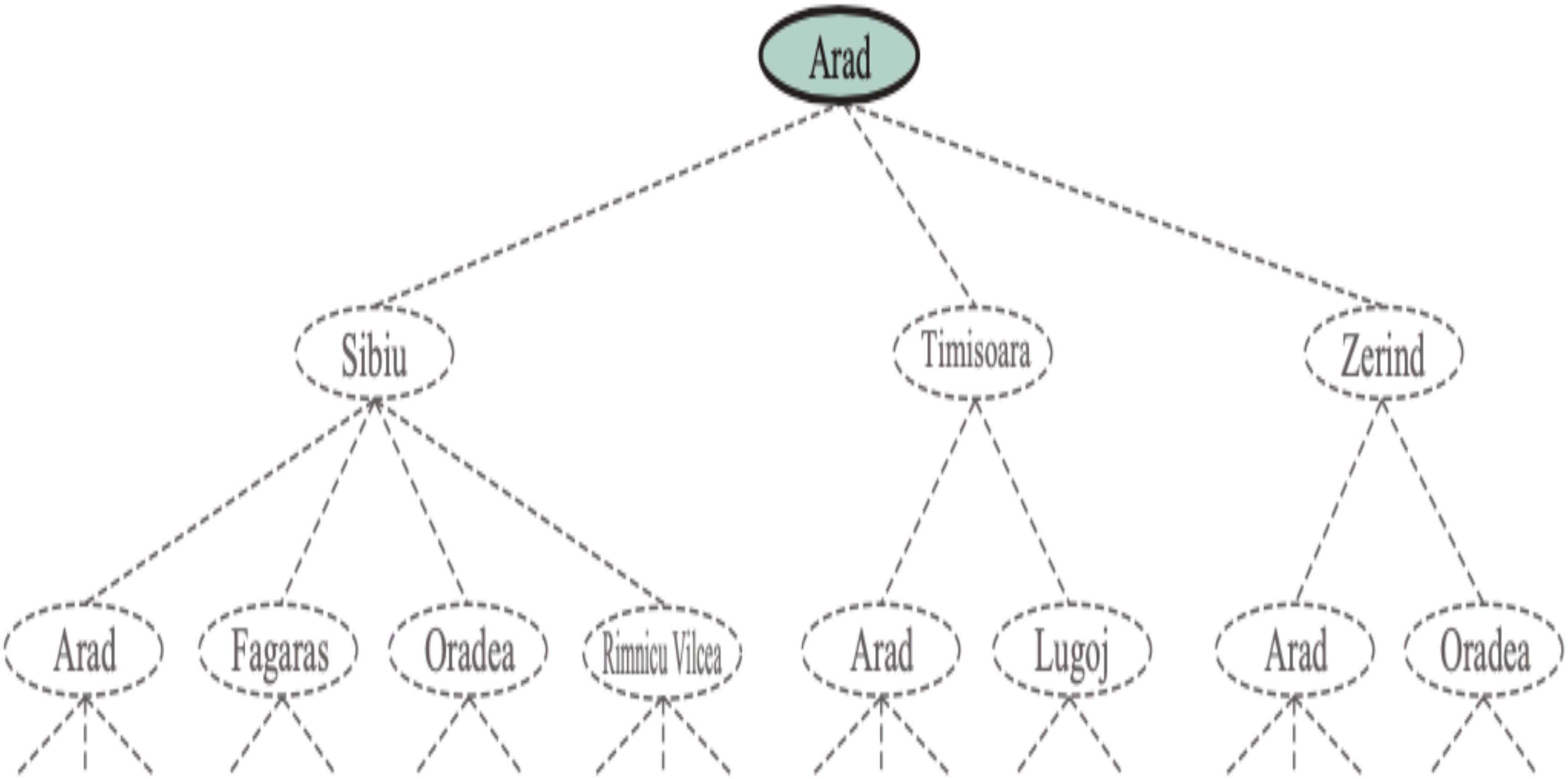
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Three partial search trees for finding a route from Arad to Bucharest



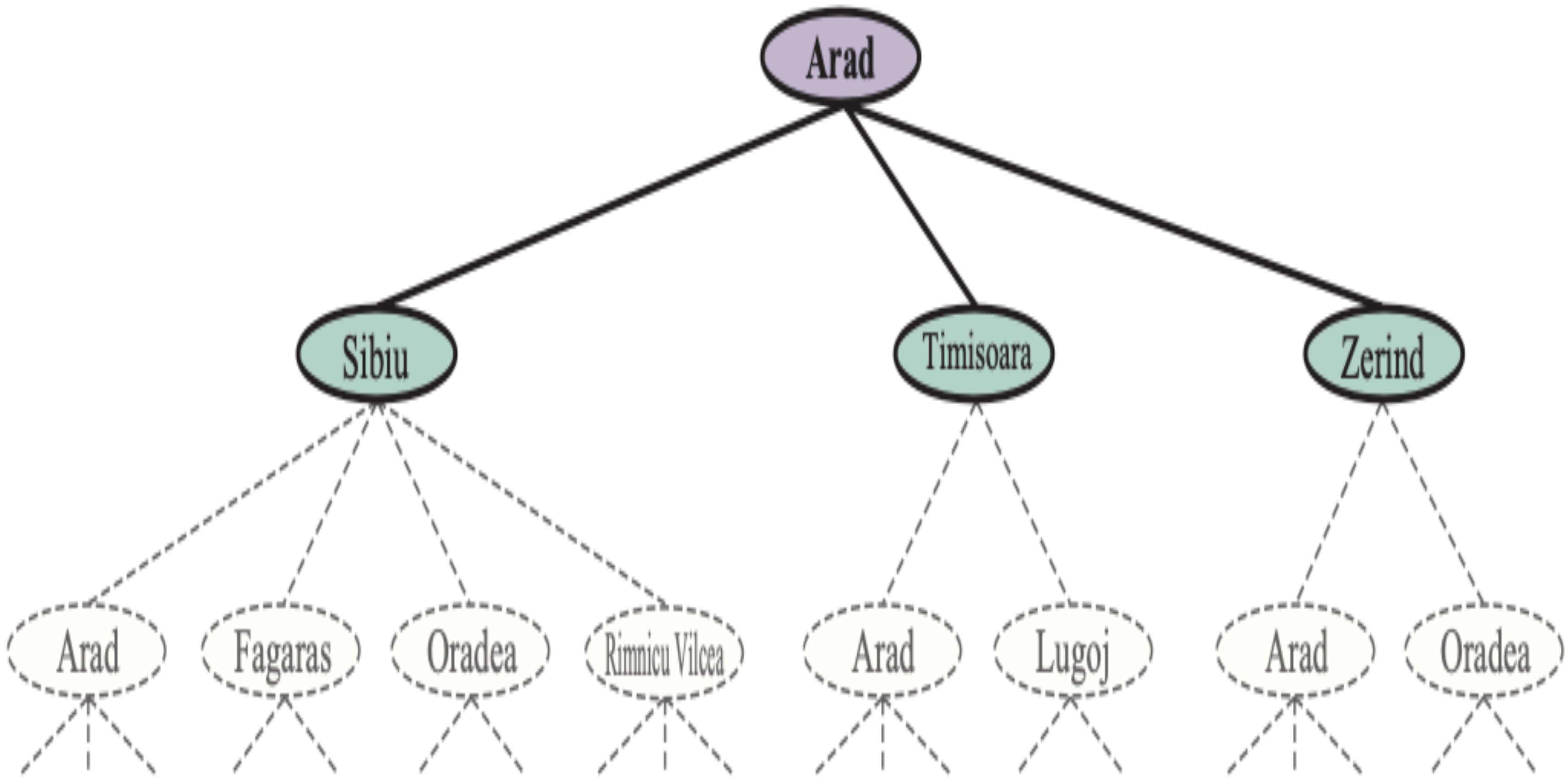
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Three partial search trees for finding a route from Arad to Bucharest



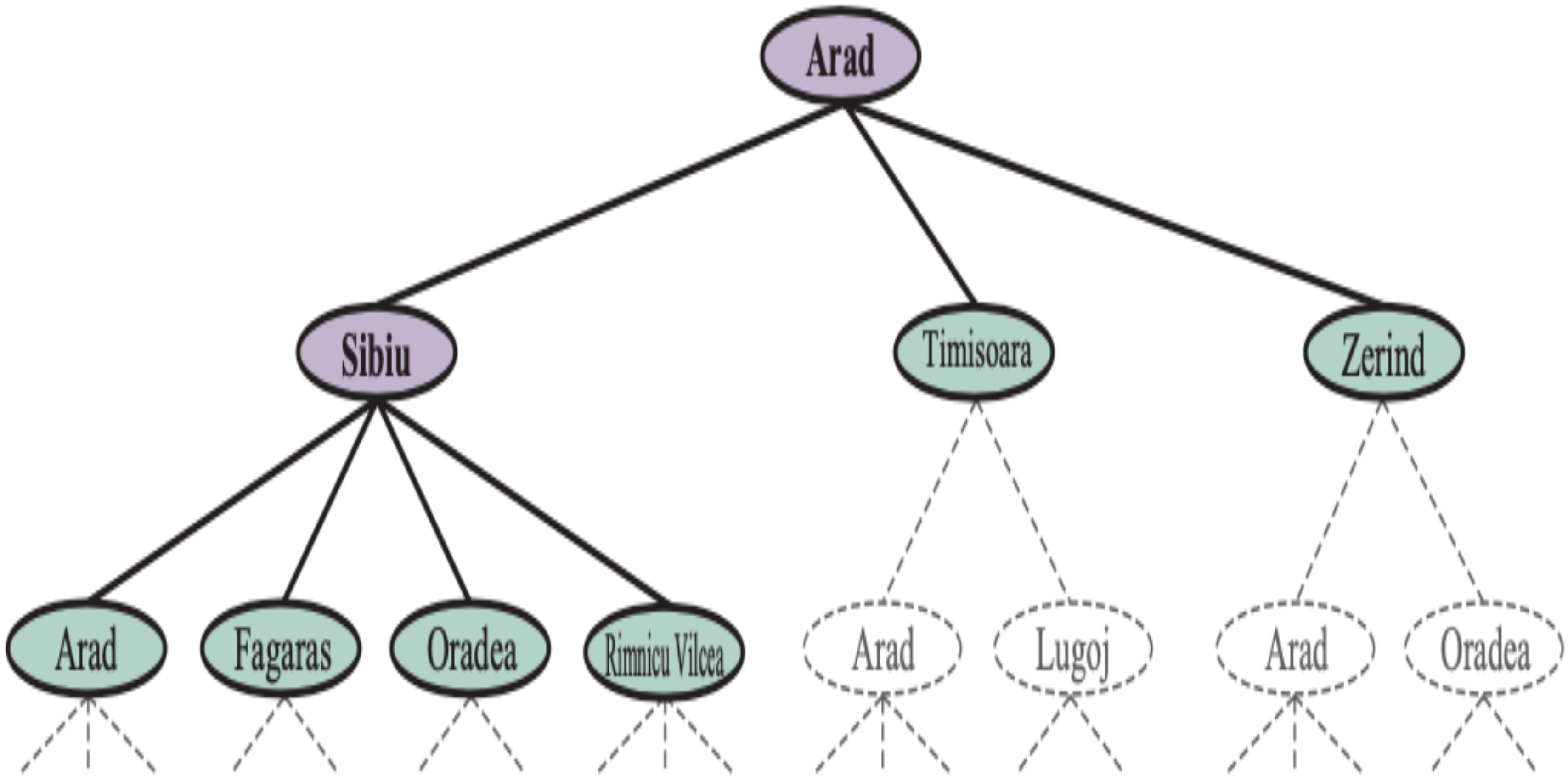
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Three partial search trees for finding a route from Arad to Bucharest



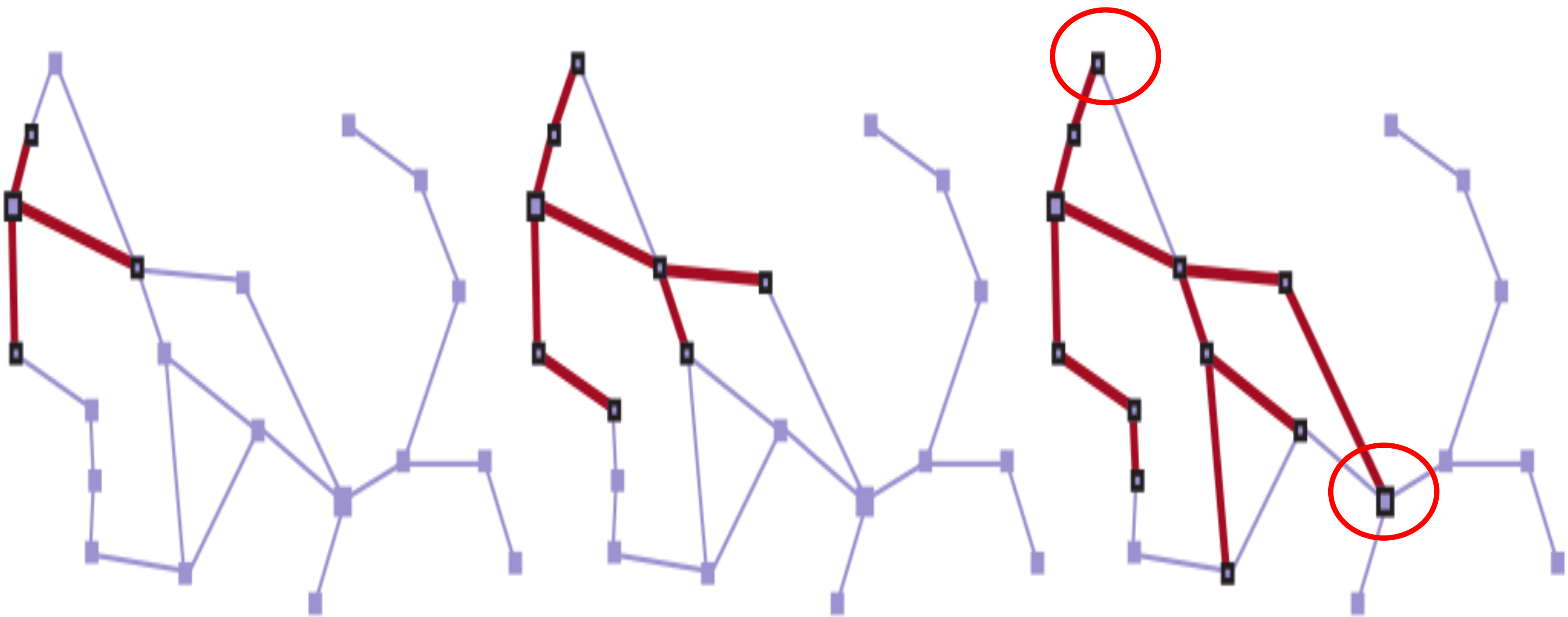
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Three partial search trees for finding a route from Arad to Bucharest



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A sequence of search trees generated by a graph search on the Romania problem

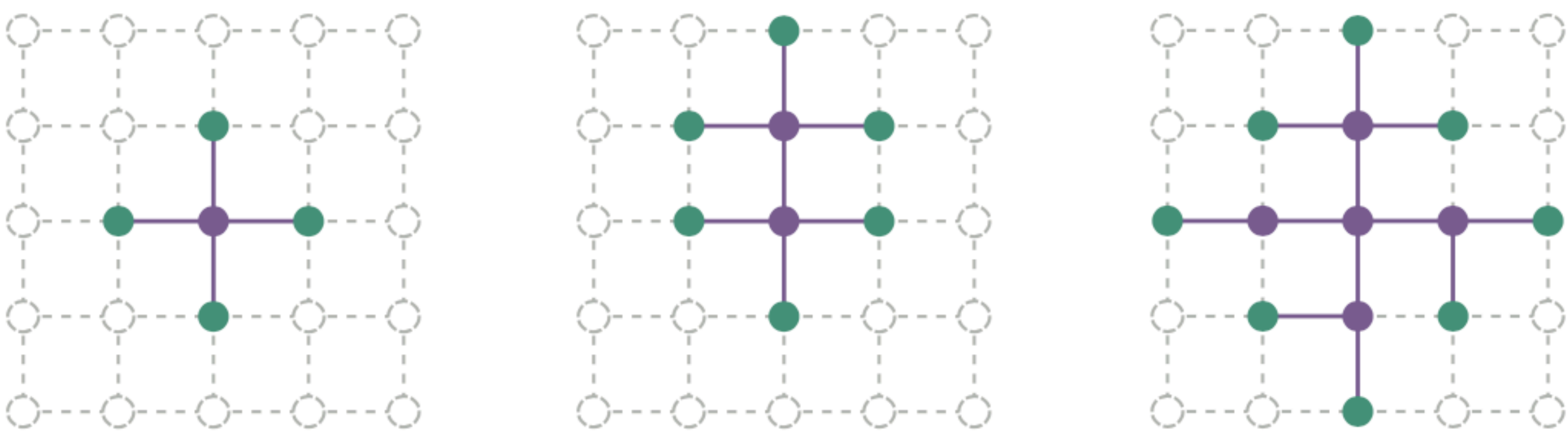


Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

The Separation Property of Graph Search

illustrated on a rectangular-grid problem

The frontier (green) separates
the interior (lavender) from
the exterior (faint dashed)



(a) (b) (c)

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

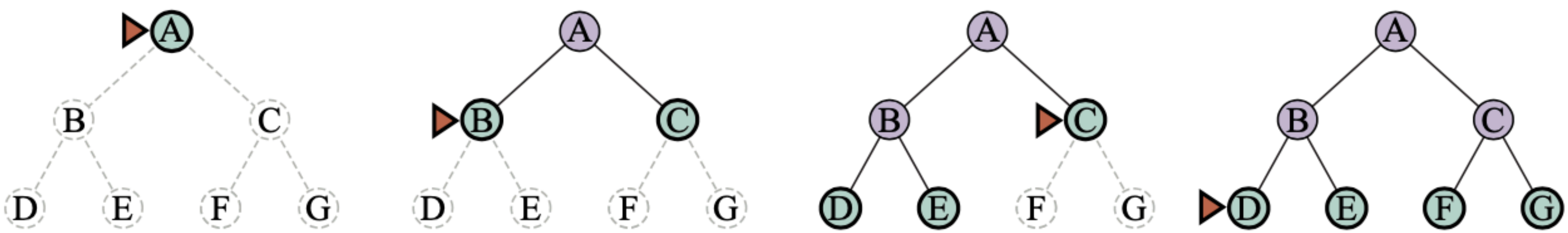
The Best-First Search (BFS) Algorithm

```
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

function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s′ ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s′)
    yield NODE(STATE=s′, PARENT=node, ACTION=action, PATH-COST=cost)
```

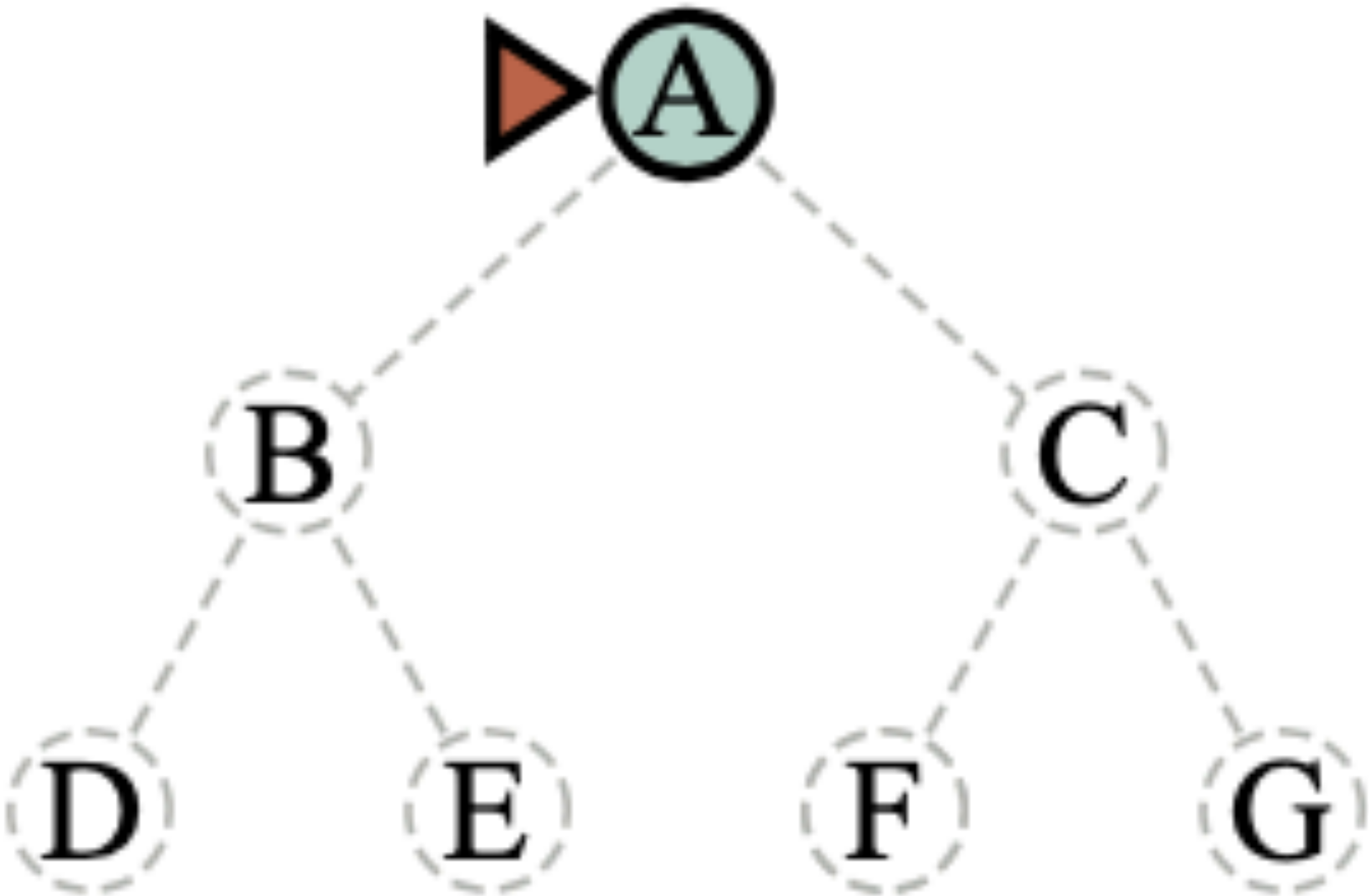
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Breadth-First Search on a Simple Binary Tree



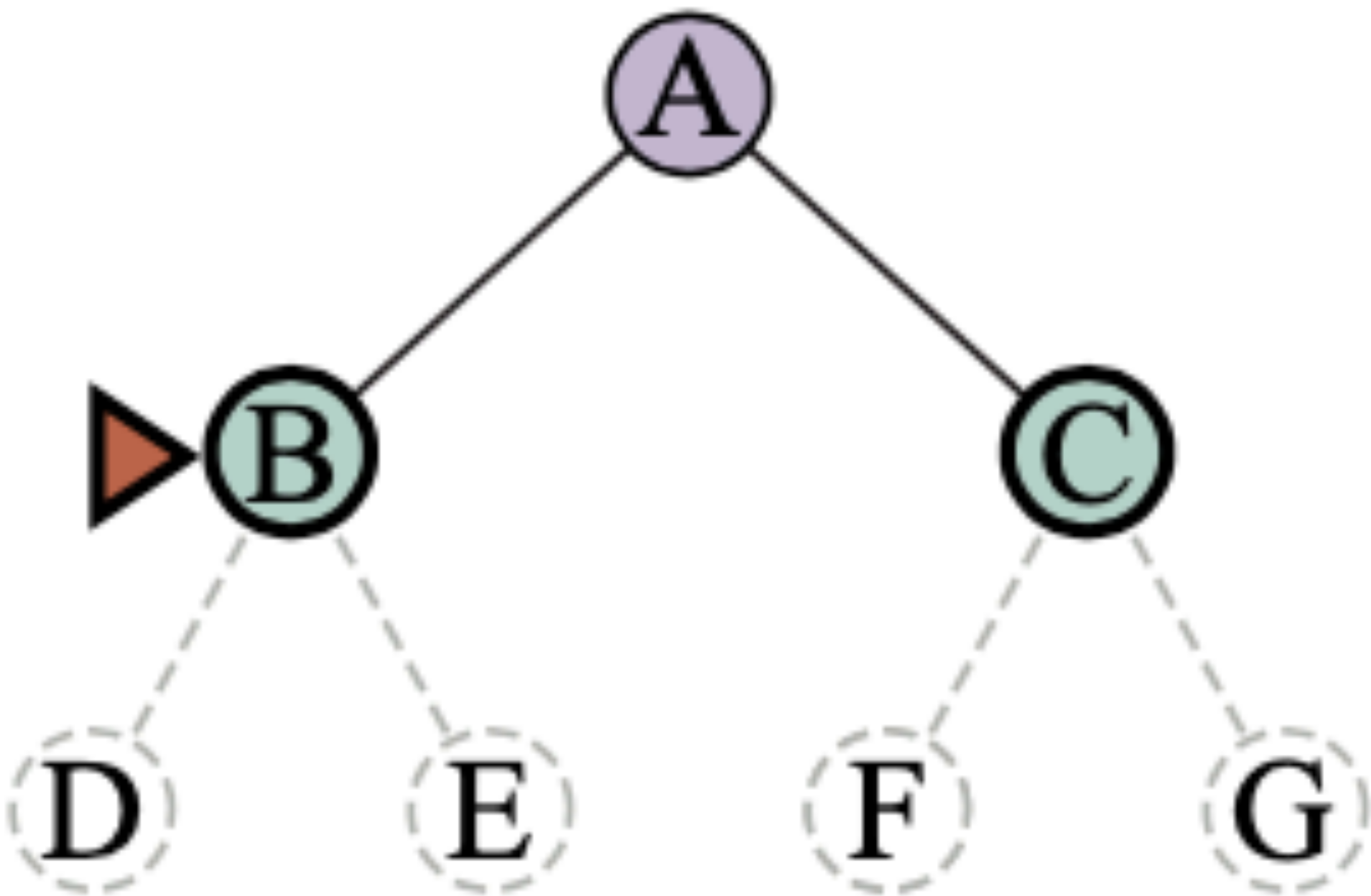
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Breadth-First Search on a Simple Binary Tree



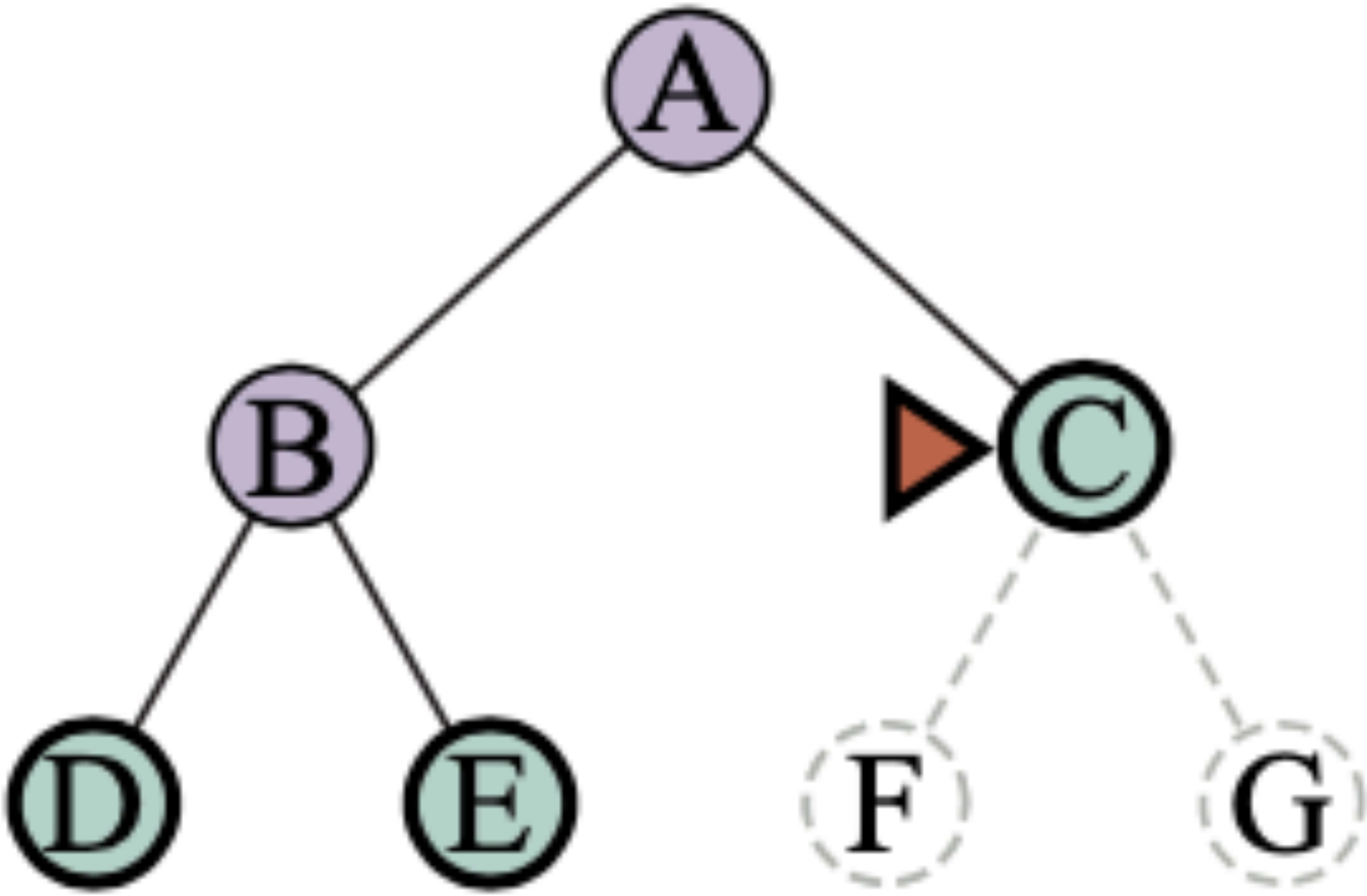
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Breadth-First Search on a Simple Binary Tree



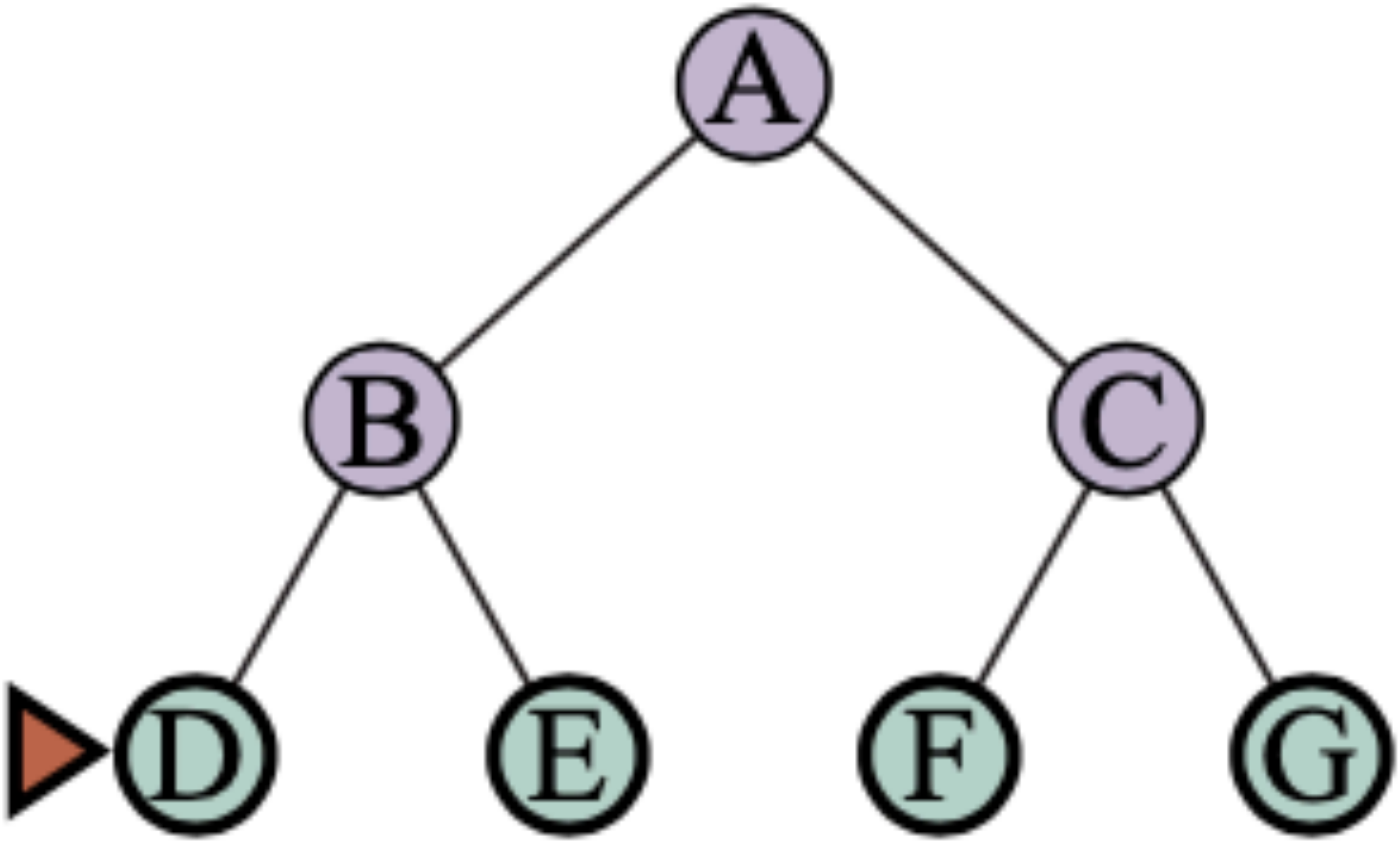
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Breadth-First Search on a Simple Binary Tree



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Breadth-First Search on a Simple Binary Tree



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

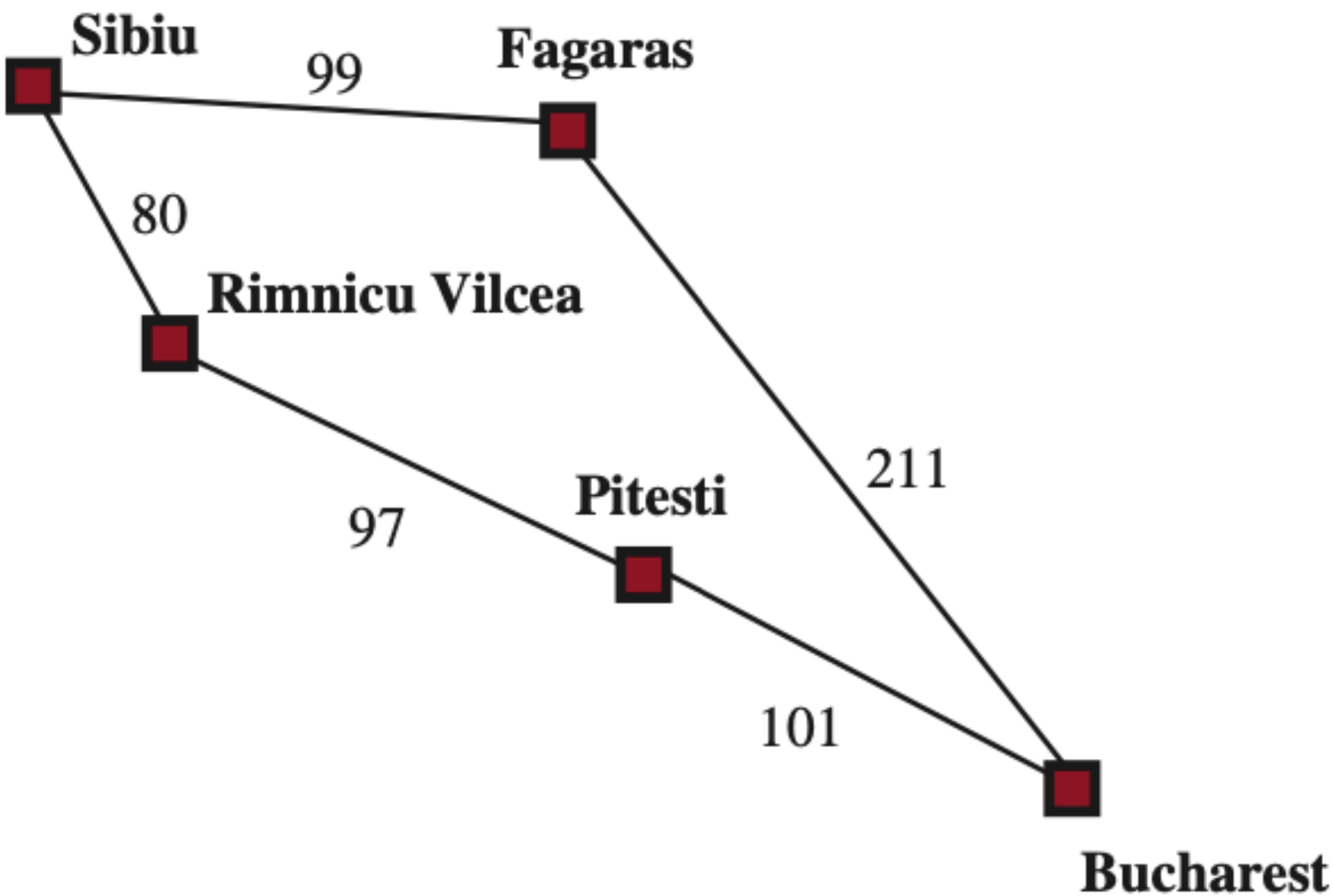
Breadth-First Search and Uniform-Cost Search Algorithms

```
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)
```

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

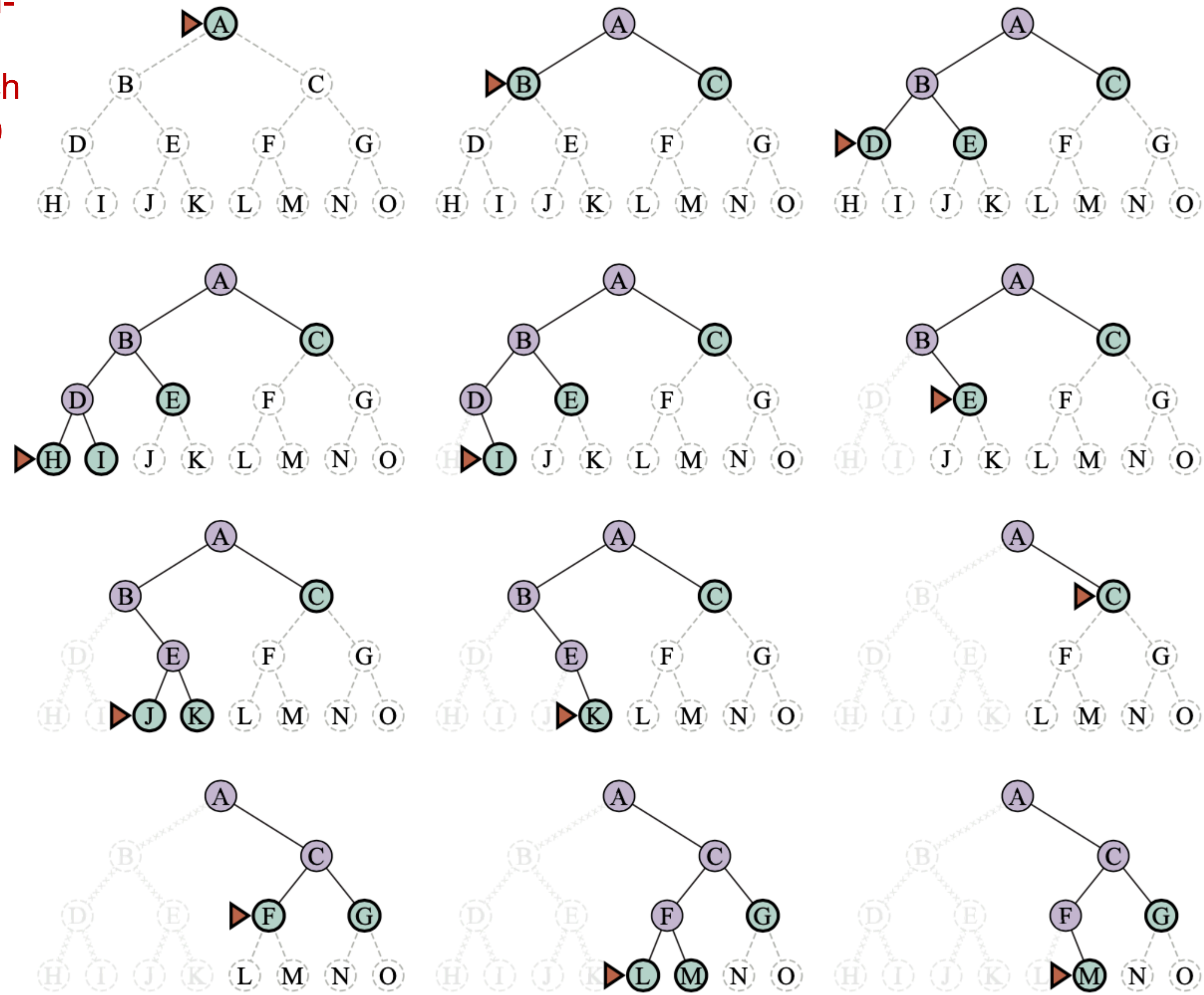
Part of the Romania State Space Uniform-Cost Search



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Depth-First Search (DFS)

Depth-First Search (DFS)



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

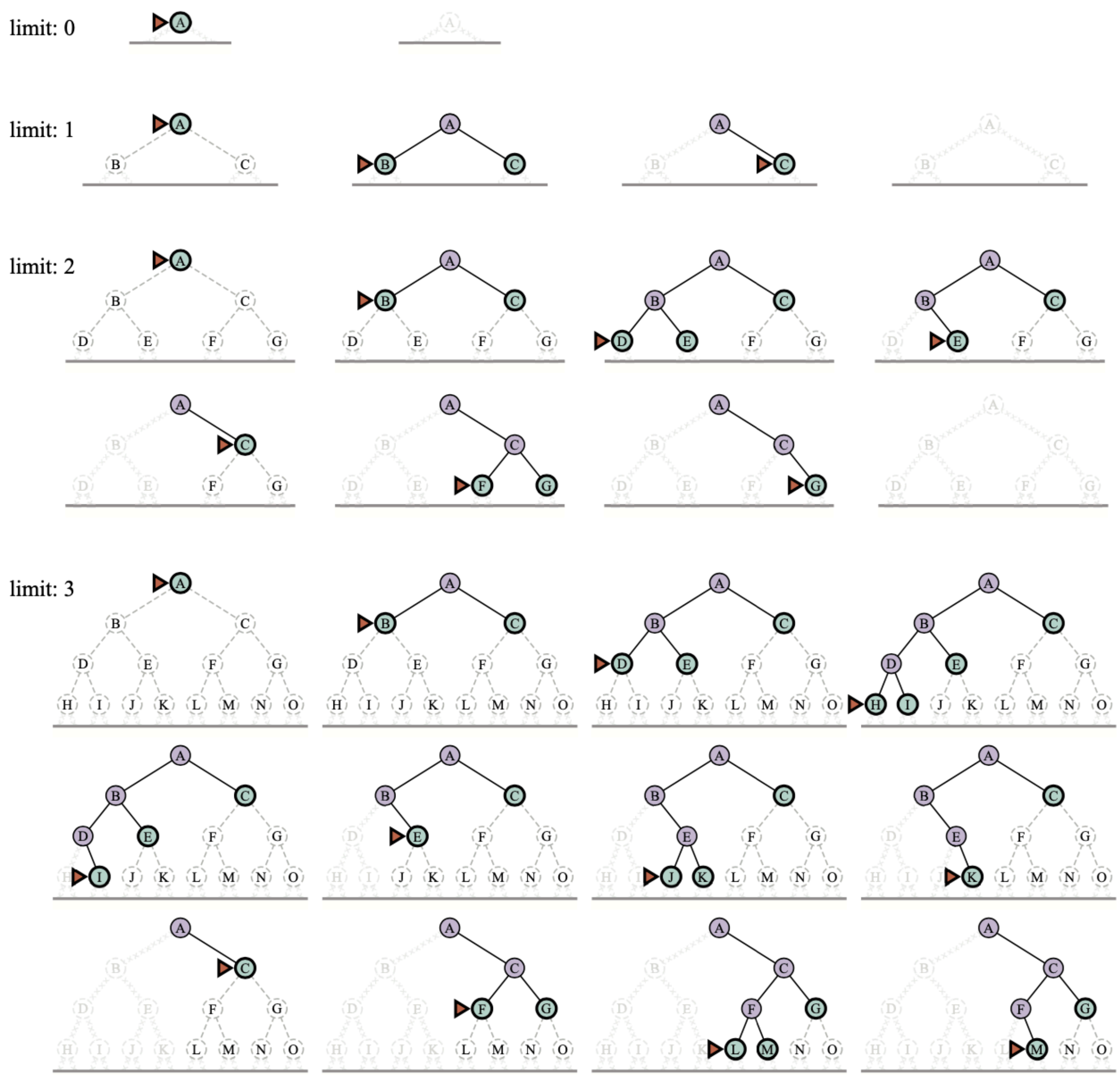
Iterative deepening and depth-limited tree-like search

```
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
```

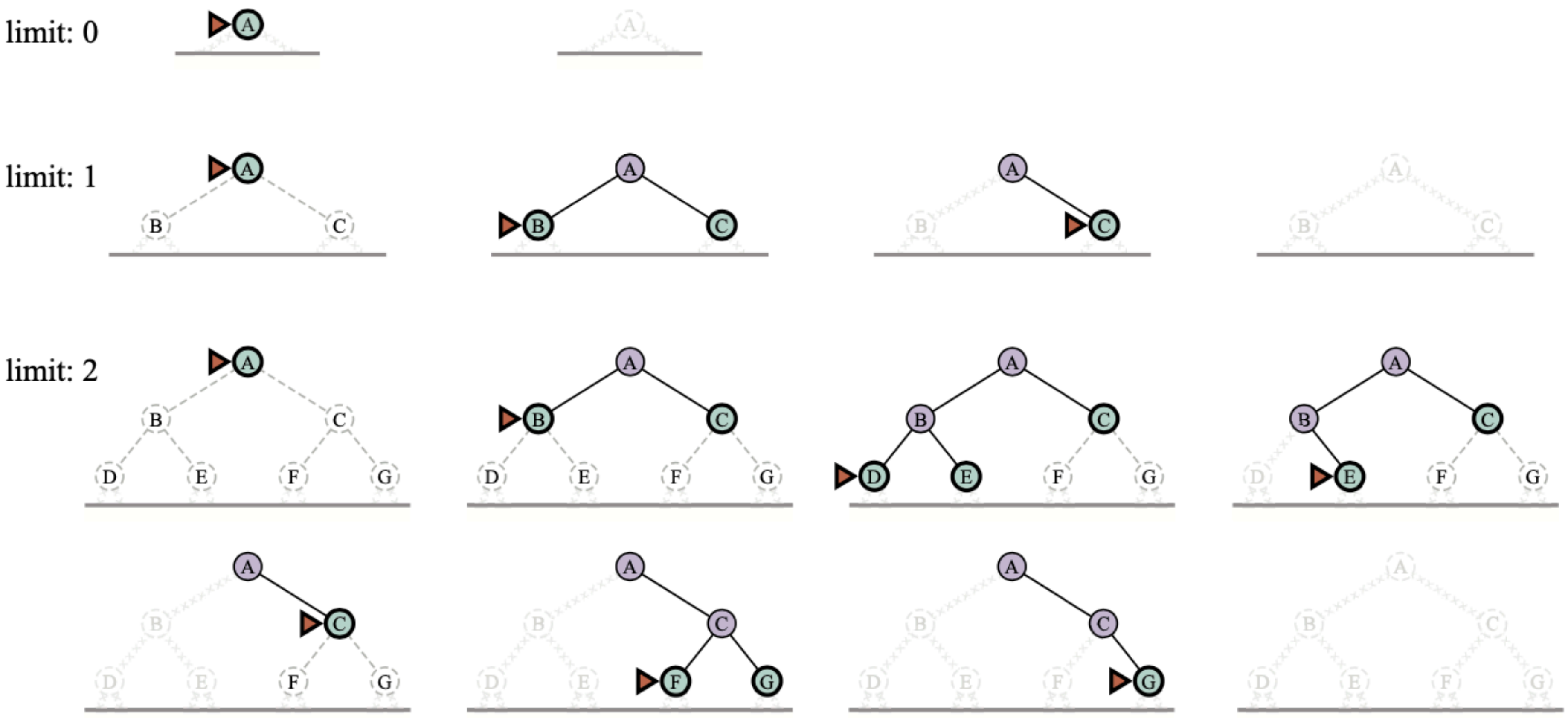
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Four iterations of iterative deepening search



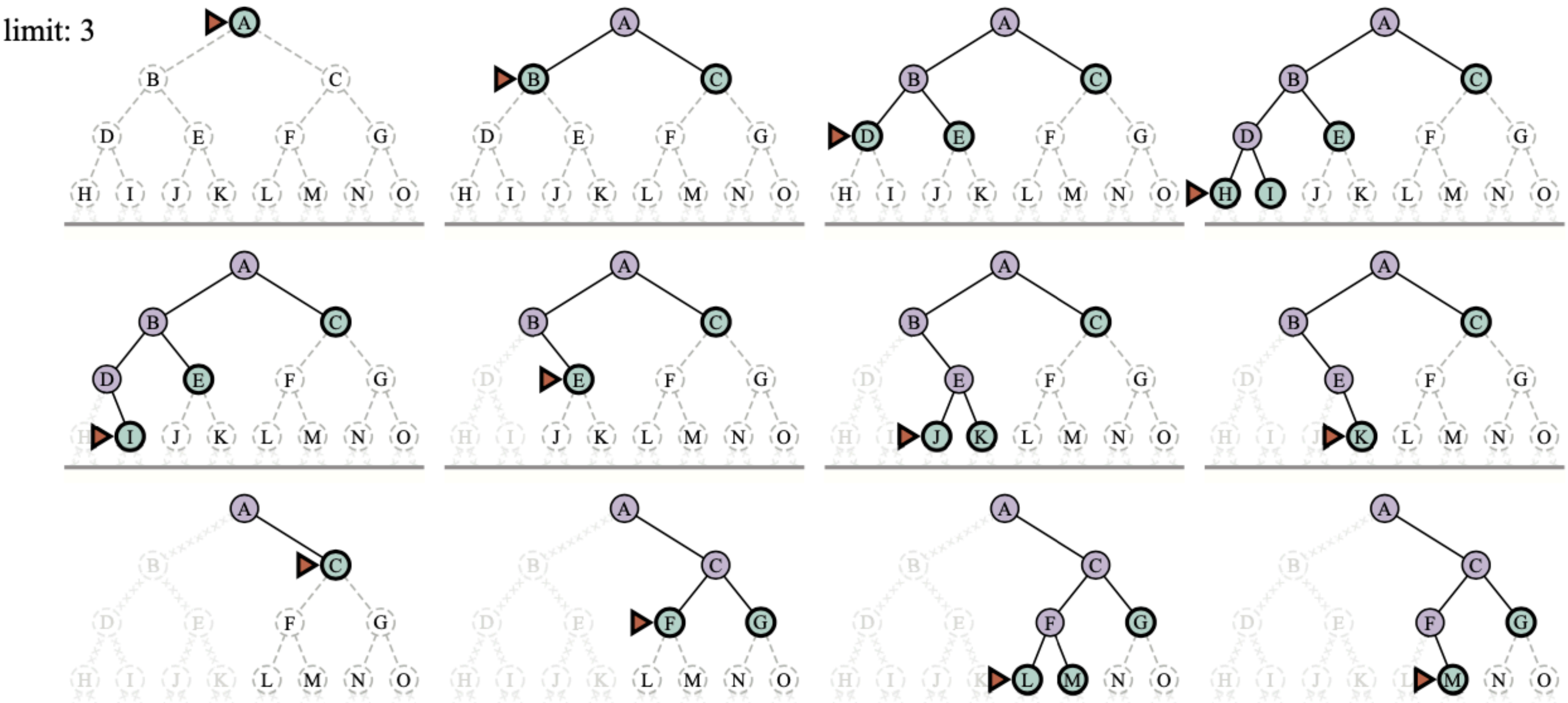
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Four iterations of iterative deepening search



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Four iterations of iterative deepening search



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Bidirectional Best-First Search

keeps two frontiers and two tables of reached states

function BIBF-SEARCH($problem_F$, f_F, $problem_B$, f_B) **returns** a solution node, or *failure*
  $node_F \leftarrow$ NODE($problem_F$.INITIAL) // *Node for a start state*
  $node_B \leftarrow$ NODE($problem_B$.INITIAL) // *Node for a goal state*
  $frontier_F \leftarrow$ a priority queue ordered by f_F, with $node_F$ as an element
  $frontier_B \leftarrow$ a priority queue ordered by f_B, with $node_B$ as an element
  $reached_F \leftarrow$ a lookup table, with one key $node_F$.STATE and value $node_F$
  $reached_B \leftarrow$ a lookup table, with one key $node_B$.STATE and value $node_B$
  $solution \leftarrow failure$
  while not TERMINATED($solution$, $frontier_F$, $frontier_B$) **do**
    if f_F(TOP($frontier_F$)) $<$ f_B(TOP($frontier_B$)) **then**
      $solution \leftarrow$ PROCEED(F, $problem_F$ $frontier_F$, $reached_F$, $reached_B$, $solution$)
    else $solution \leftarrow$ PROCEED(B, $problem_B$, $frontier_B$, $reached_B$, $reached_F$, $solution$)
  return $solution$

Bidirectional Best-First Search

keeps two frontiers and two tables of reached states

```
function PROCEED(dir, problem, frontier, reached, reached₂, solution) returns a solution
        // Expand node on frontier; check against the other frontier in reached₂.
        // The variable "dir" is the direction: either F for forward or B for backward.
  node ← POP(frontier)
  for each child in EXPAND(problem, node) do
    s ← child.STATE
    if s not in reached or PATH-COST(child) < PATH-COST(reached[s]) then
      reached[s] ← child
      add child to frontier
      if s is in reached₂ then
        solution₂ ← JOIN-NODES(dir, child, reached₂[s]))
        if PATH-COST(solution₂) < PATH-COST(solution) then
          solution ← solution₂
  return solution
```

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Evaluation of 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	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

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

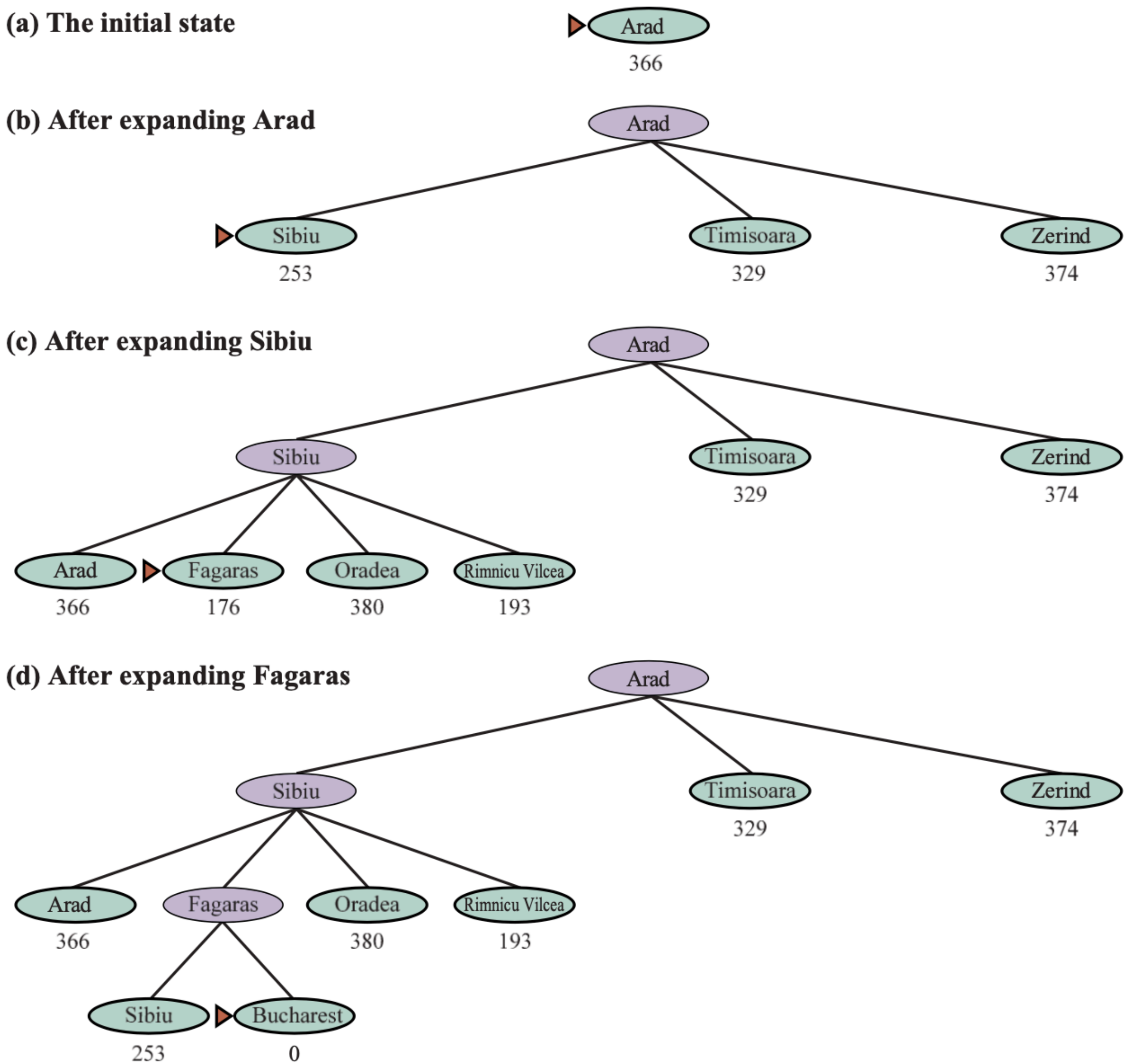
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Values of *hSLD* —straight-line distances to Bucharest.

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

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A* search

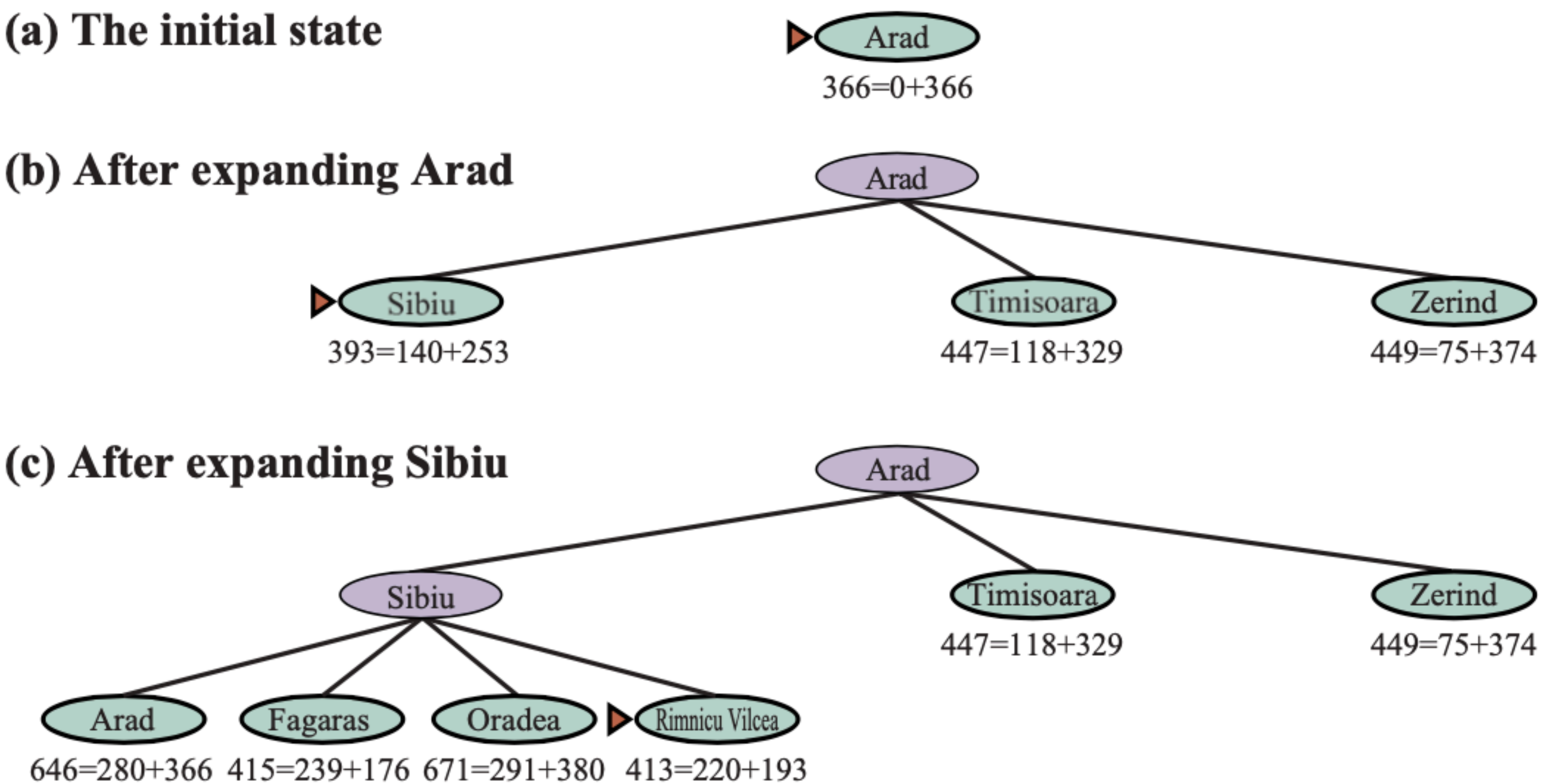


Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A∗ search

Nodes are labeled with $f = g + h$.

The h values are the Straight-Line Distances heuristic h_{SLD}



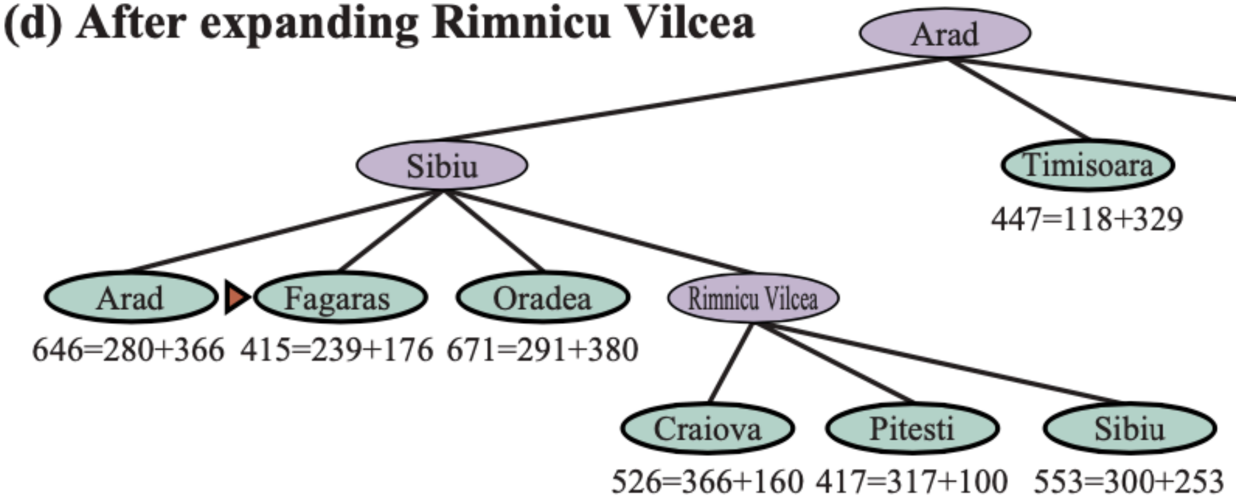
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A∗ search

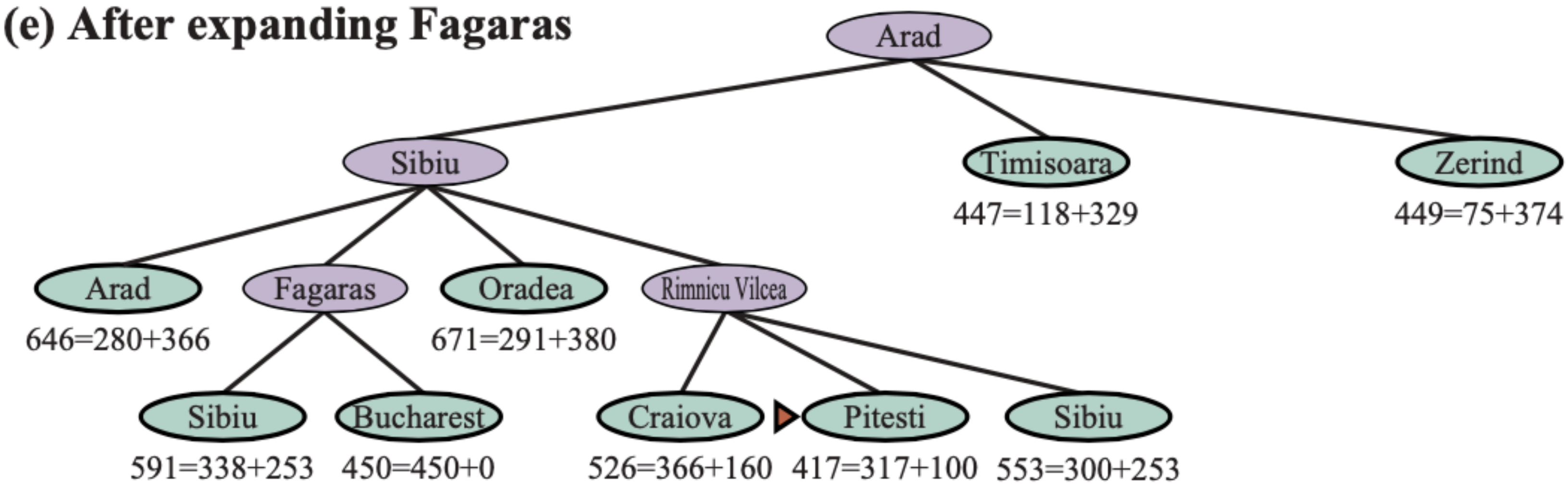
Nodes are labeled with $f = g + h$.

The h values are the Straight-Line Distances heuristic h_{SLD}

(d) After expanding Rimnicu Vilcea

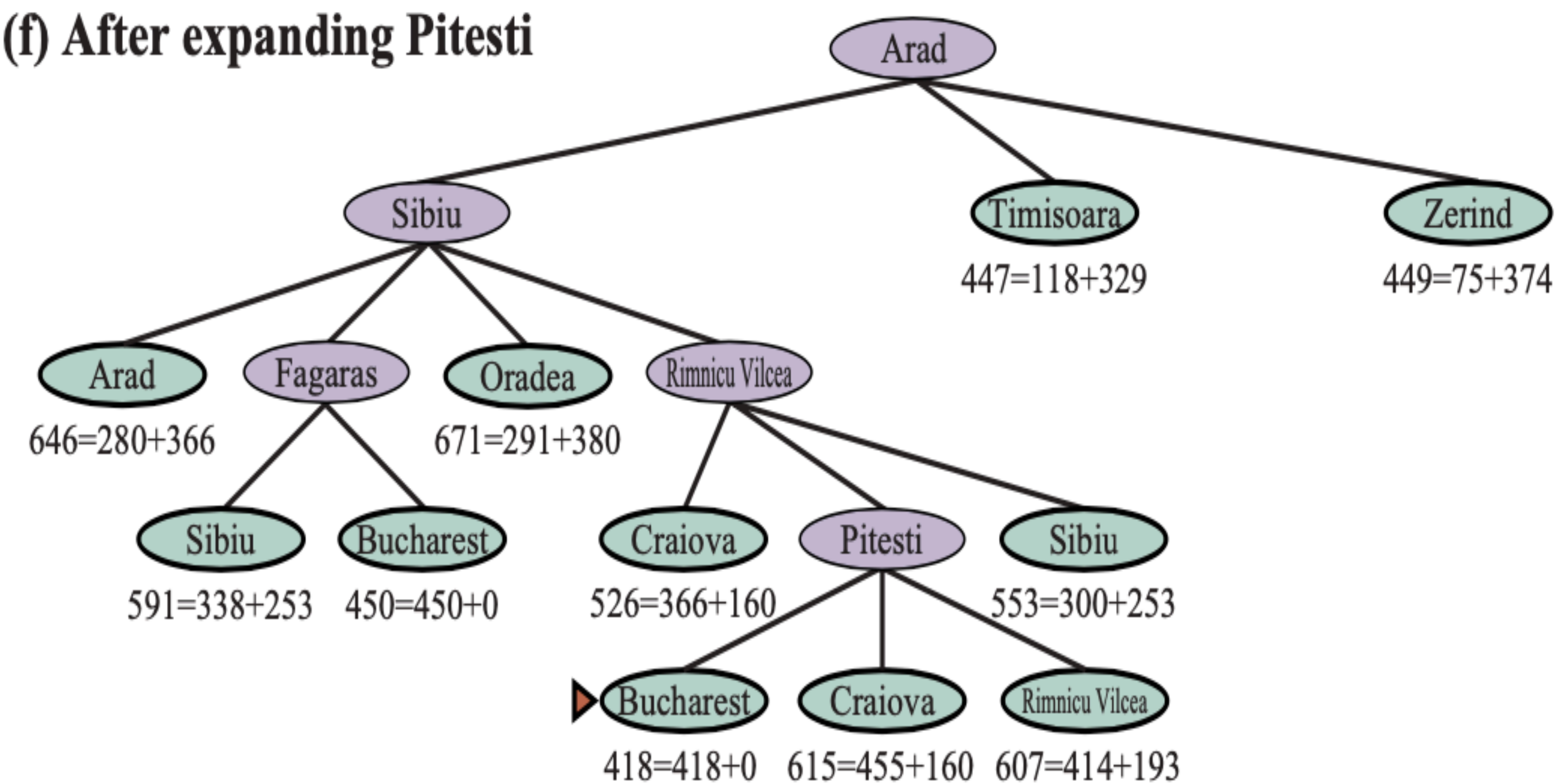


(e) After expanding Fagaras

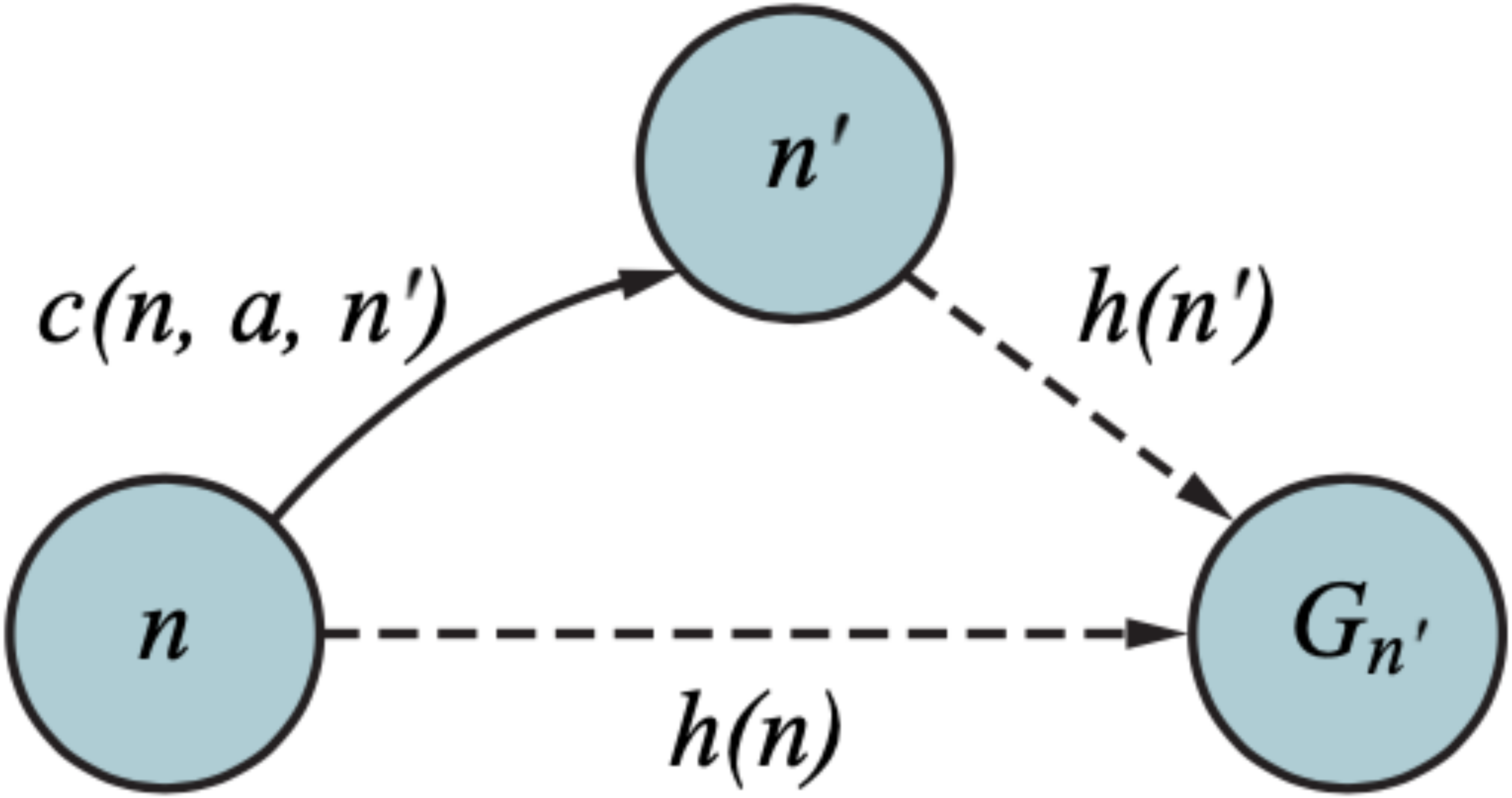


A∗ search

Nodes are labeled with $f = g + h$.
The h values are the Straight-Line Distances heuristic h_{SLD}



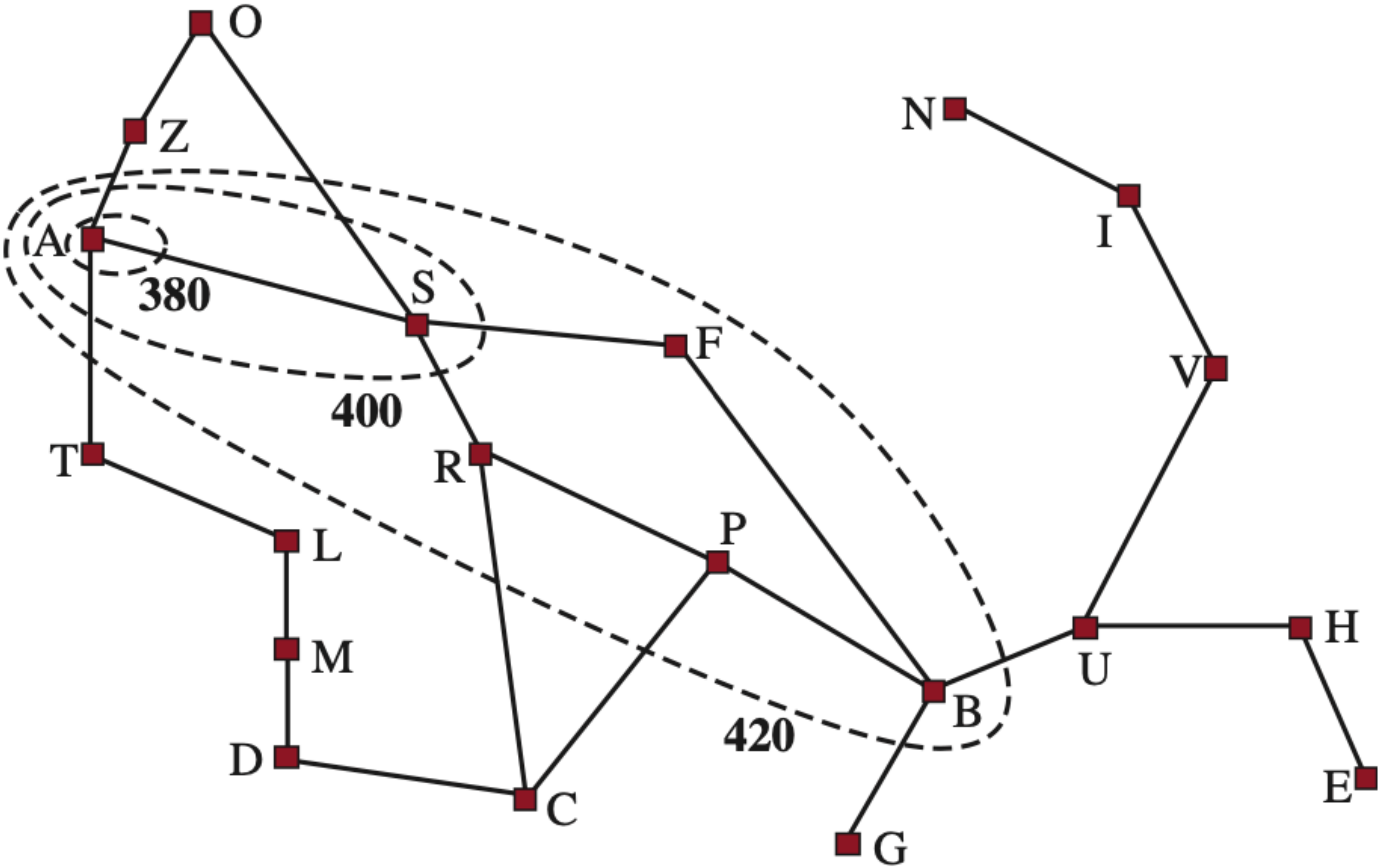
Triangle Inequality



If the heuristic h is consistent, then the single number $h(n)$ will be less than the sum of the cost c(n, a, a′) of the action from n to n′ plus the heuristic estimate h(n′).

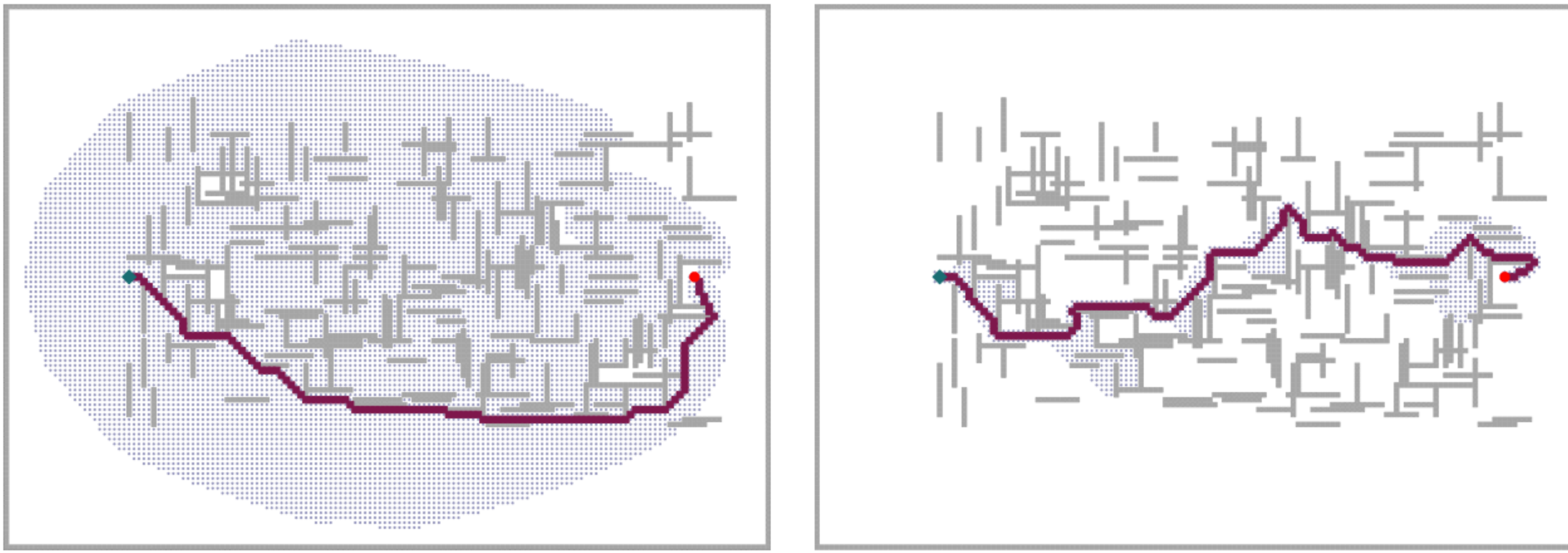
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Map of Romania showing contours *at* *f = 380, f = 400*, and *f = 420,* with Arad as the start state



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

(a) A* Search
(b) Weighted A* Search



(a) (b)

The gray bars are obstacles, the purple line is the path from the green start to red goal, and the small dots are states that were reached by each search.
On this particular problem, weighted A* explores 7 times fewer states and finds a path that is 5% more costly.

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

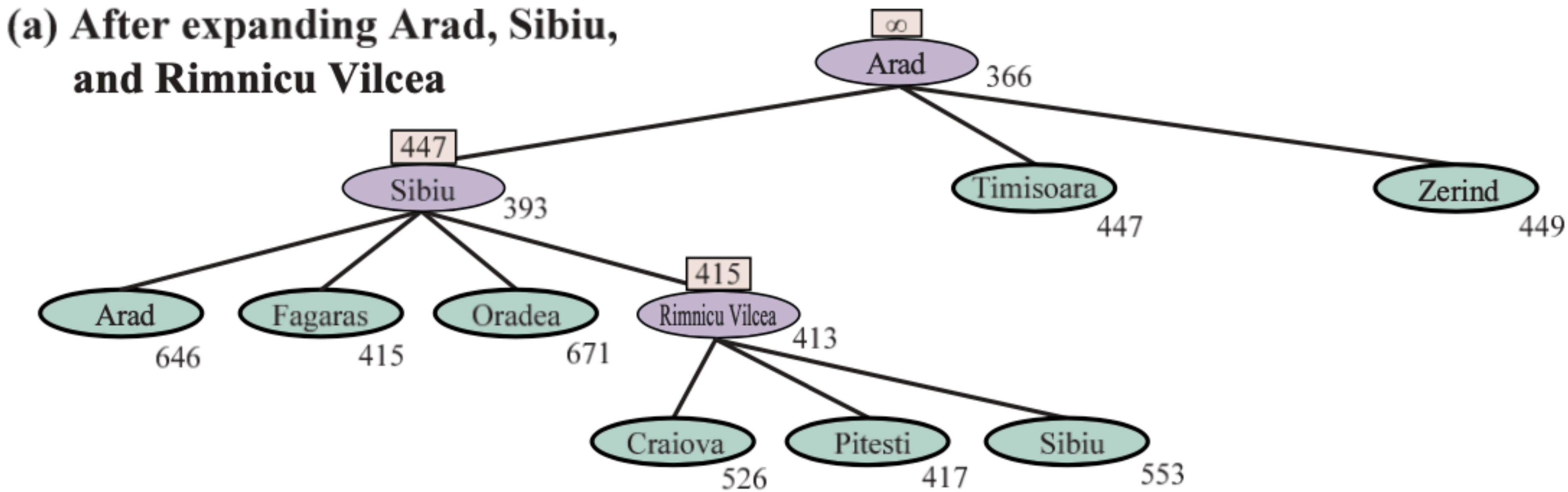
Recursive Best-First Search (RBFS) Algorithm

```
function RECURSIVE-BEST-FIRST-SEARCH(problem) returns a solution or failure
    solution, fvalue ← RBFS(problem, NODE(problem.INITIAL), ∞)
  return solution

function RBFS(problem, node, f_limit) returns a solution or failure, and a new f-cost limit
  if problem.IS-GOAL(node.STATE) then return node
  successors ← LIST(EXPAND(node))
  if successors is empty then return failure, ∞
  for each s in  successors do        // update f with value from previous search
      s.f ← max(s.PATH-COST + h(s), node.f))
  while true do
      best ← the node in successors with lowest f-value
      if best.f > f_limit then return failure, best.f
      alternative ← the second-lowest f-value among successors
      result, best.f ← RBFS(problem, best, min(f_limit, alternative))
      if result ≠ failure then return result, best.f
```

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

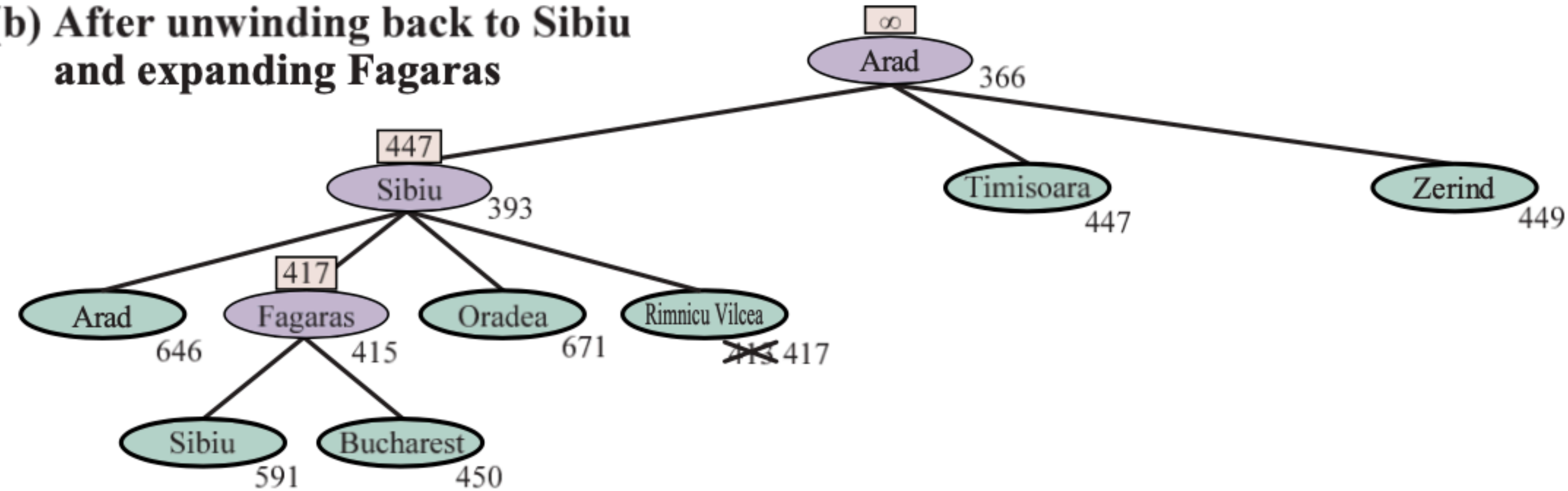
Recursive Best-First Search (RBFS)



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Recursive Best-First Search (RBFS)

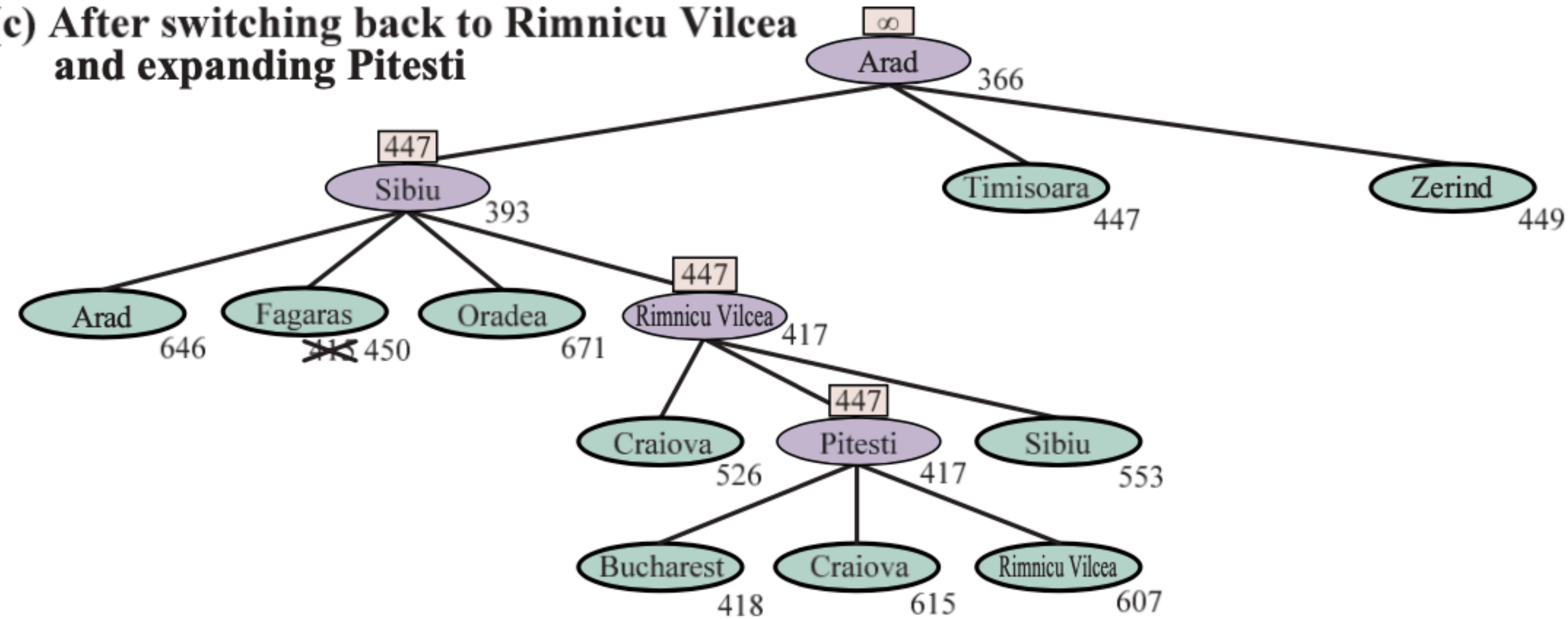
(b) After unwinding back to Sibiu and expanding Fagaras



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

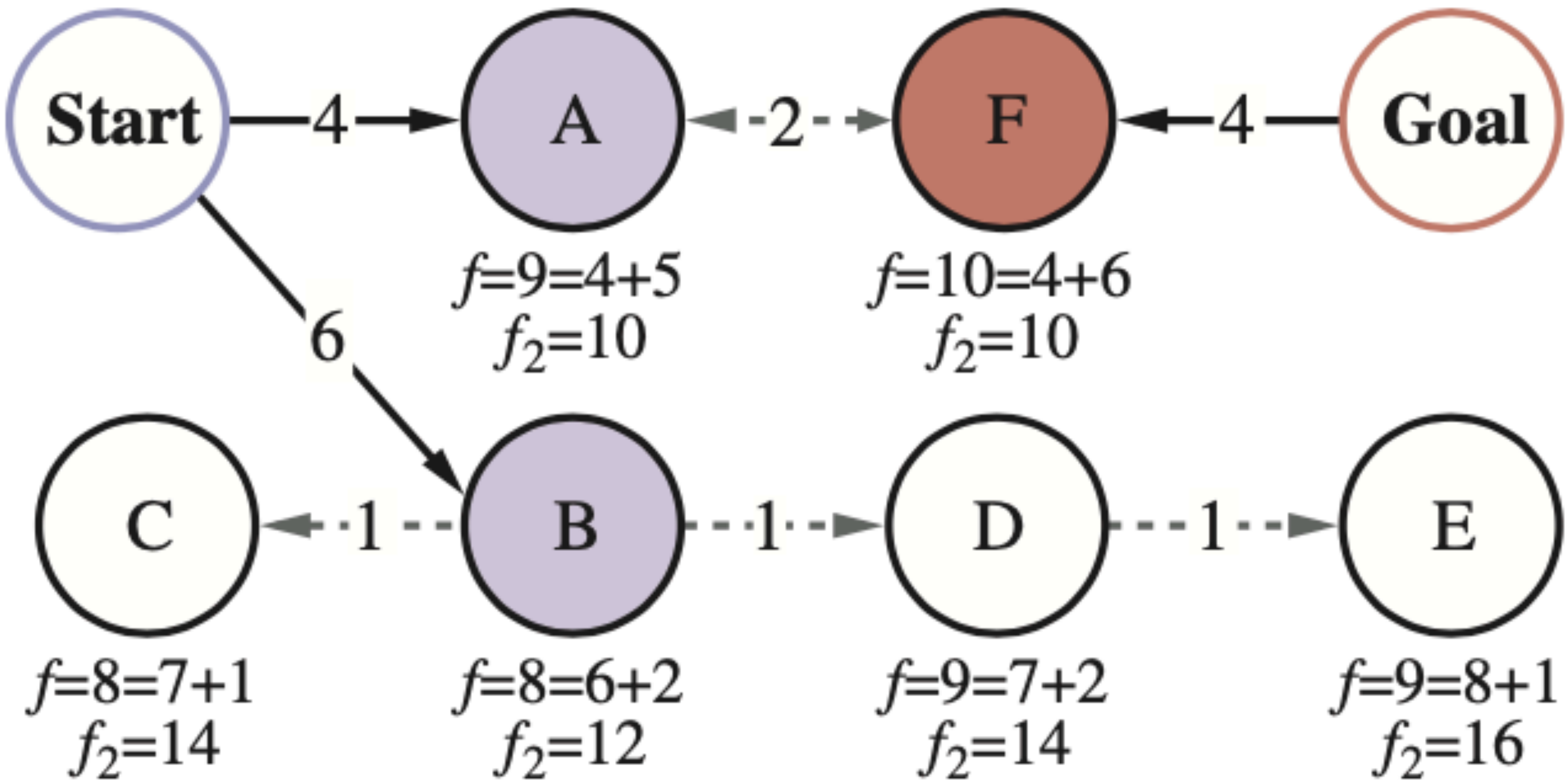
Recursive Best-First Search (RBFS)

(c) After switching back to Rimnicu Vilcea and expanding Pitesti



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Bidirectional Search maintains two frontiers

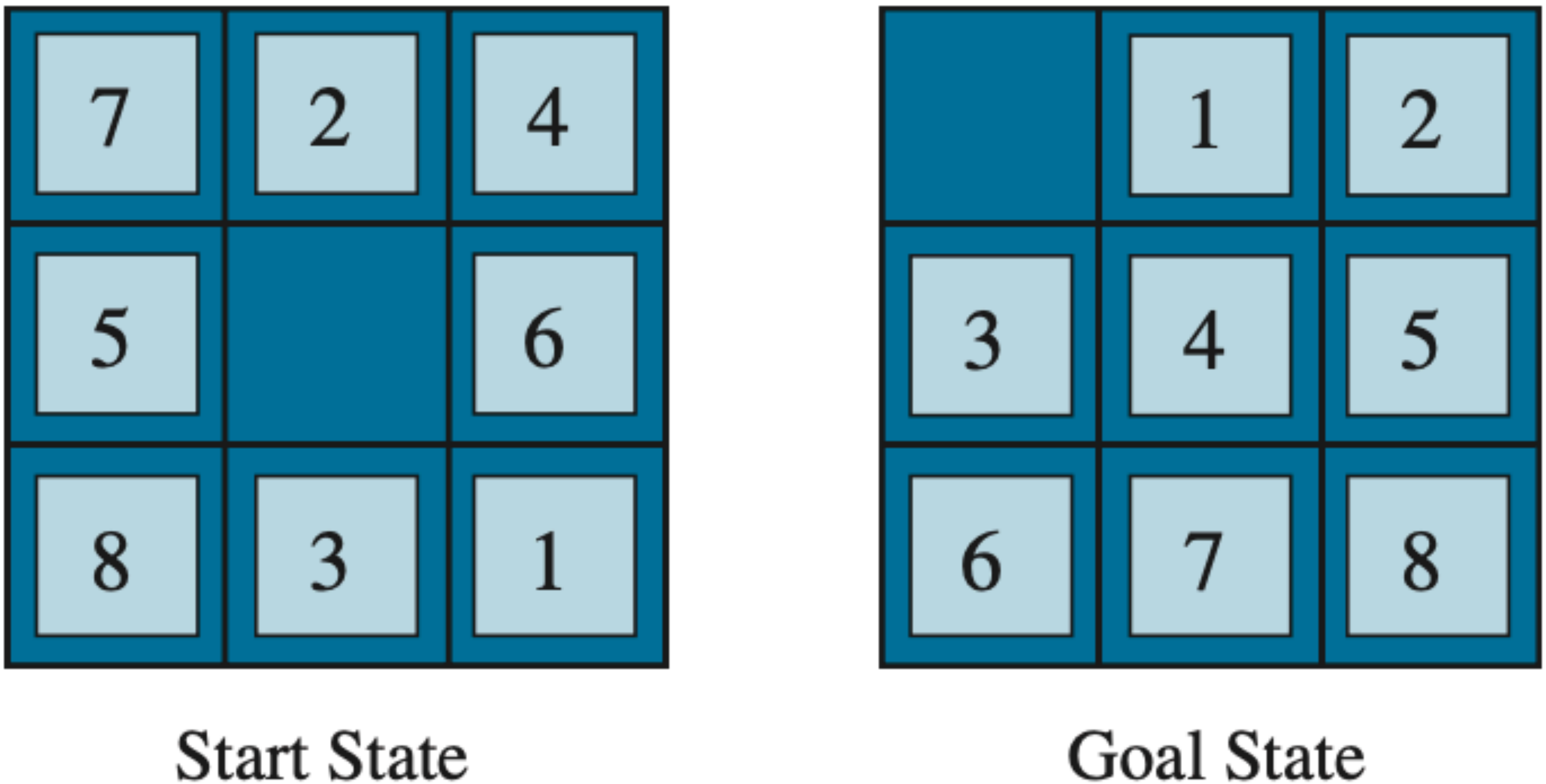


On the left, nodes A and B are successors of Start;
on the right, node F is an inverse successor of Goal

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A typical instance of the 8-puzzle

The shortest solution is 26 actions long



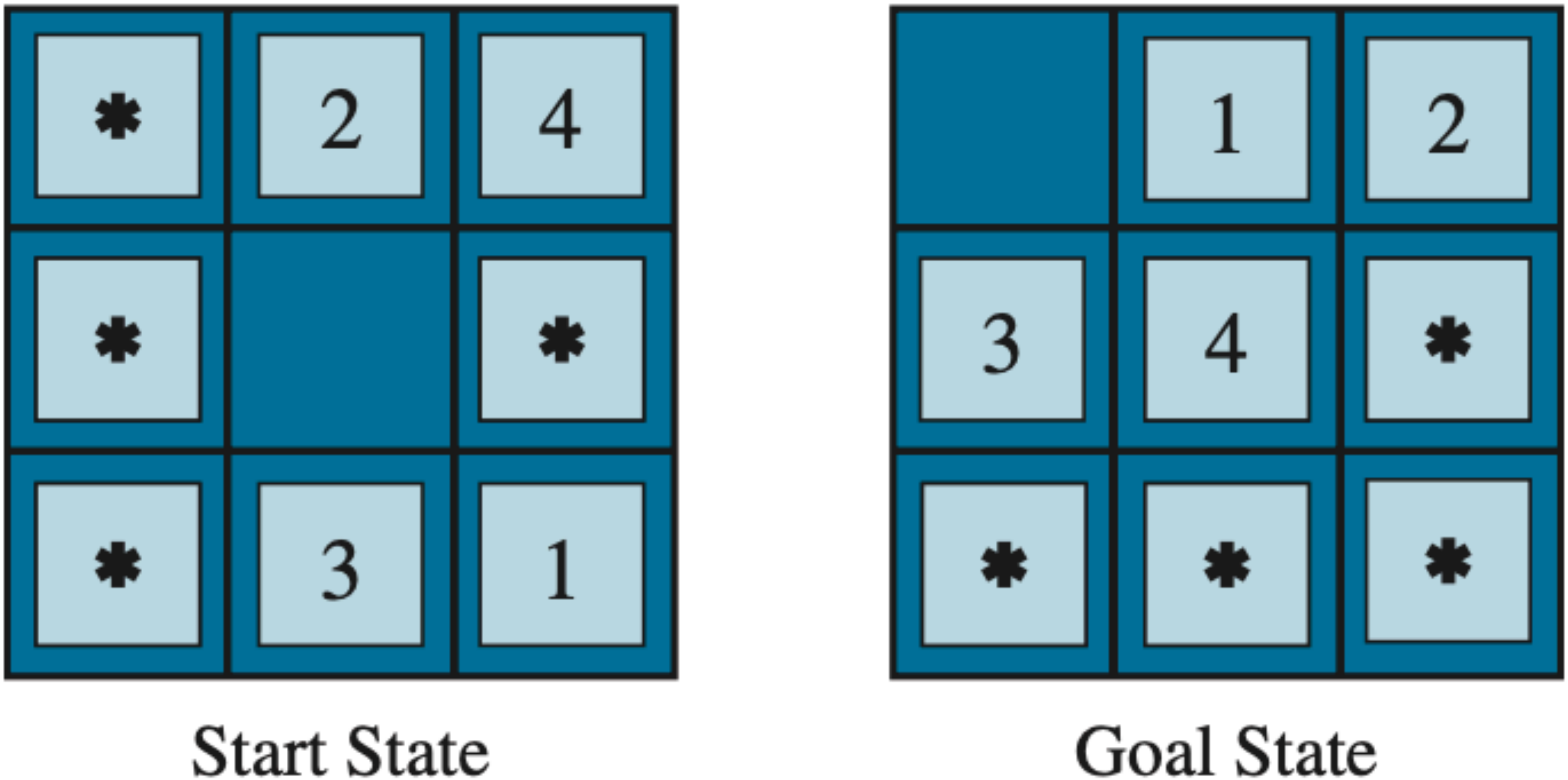
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Comparison of the search costs and effective branching factors for 8-puzzle problems

	Search Cost (nodes generated)			Effective Branching Factor		
d	BFS	$A^*(h_1)$	$A^*(h_2)$	BFS	$A^*(h_1)$	$A^*(h_2)$
6	128	24	19	2.01	1.42	1.34
8	368	48	31	1.91	1.40	1.30
10	1033	116	48	1.85	1.43	1.27
12	2672	279	84	1.80	1.45	1.28
14	6783	678	174	1.77	1.47	1.31
16	17270	1683	364	1.74	1.48	1.32
18	41558	4102	751	1.72	1.49	1.34
20	91493	9905	1318	1.69	1.50	1.34
22	175921	22955	2548	1.66	1.50	1.34
24	290082	53039	5733	1.62	1.50	1.36
26	395355	110372	10080	1.58	1.50	1.35
28	463234	202565	22055	1.53	1.49	1.36

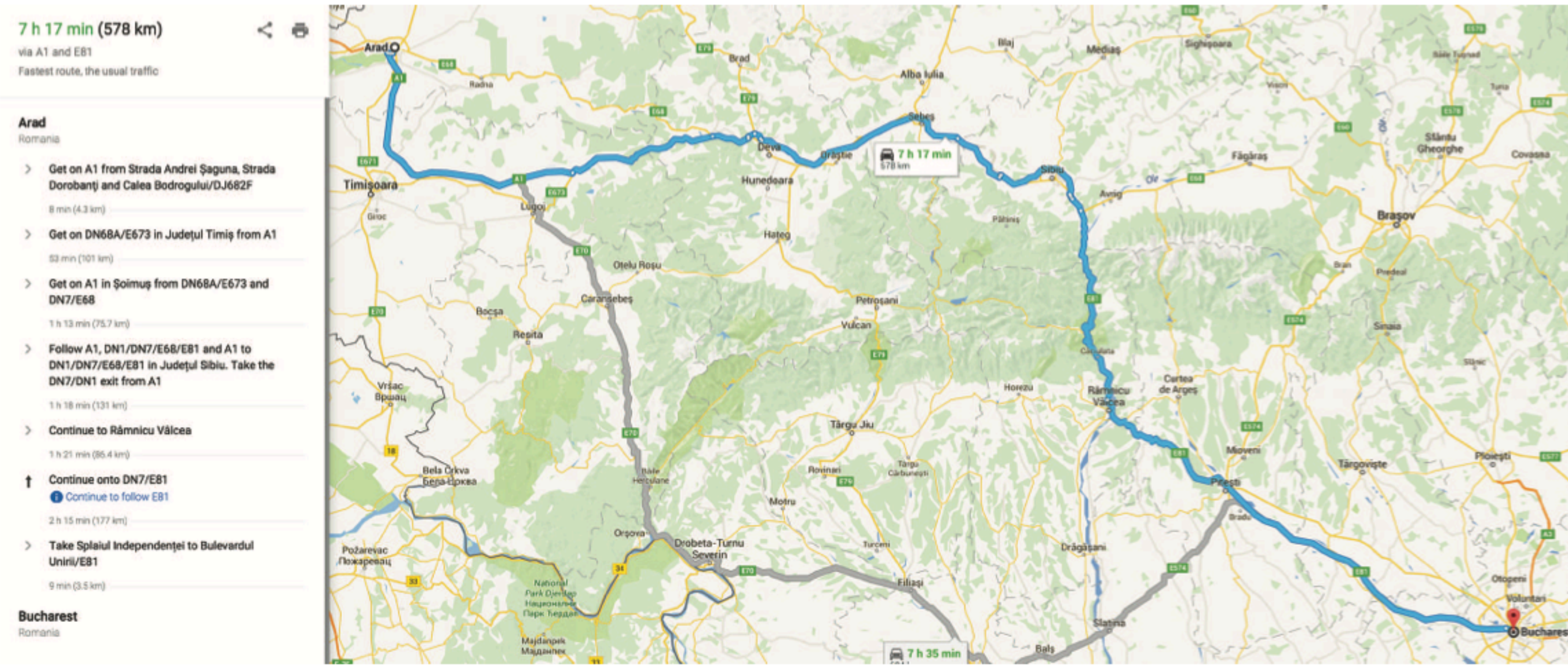
Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

A subproblem of the 8-puzzle



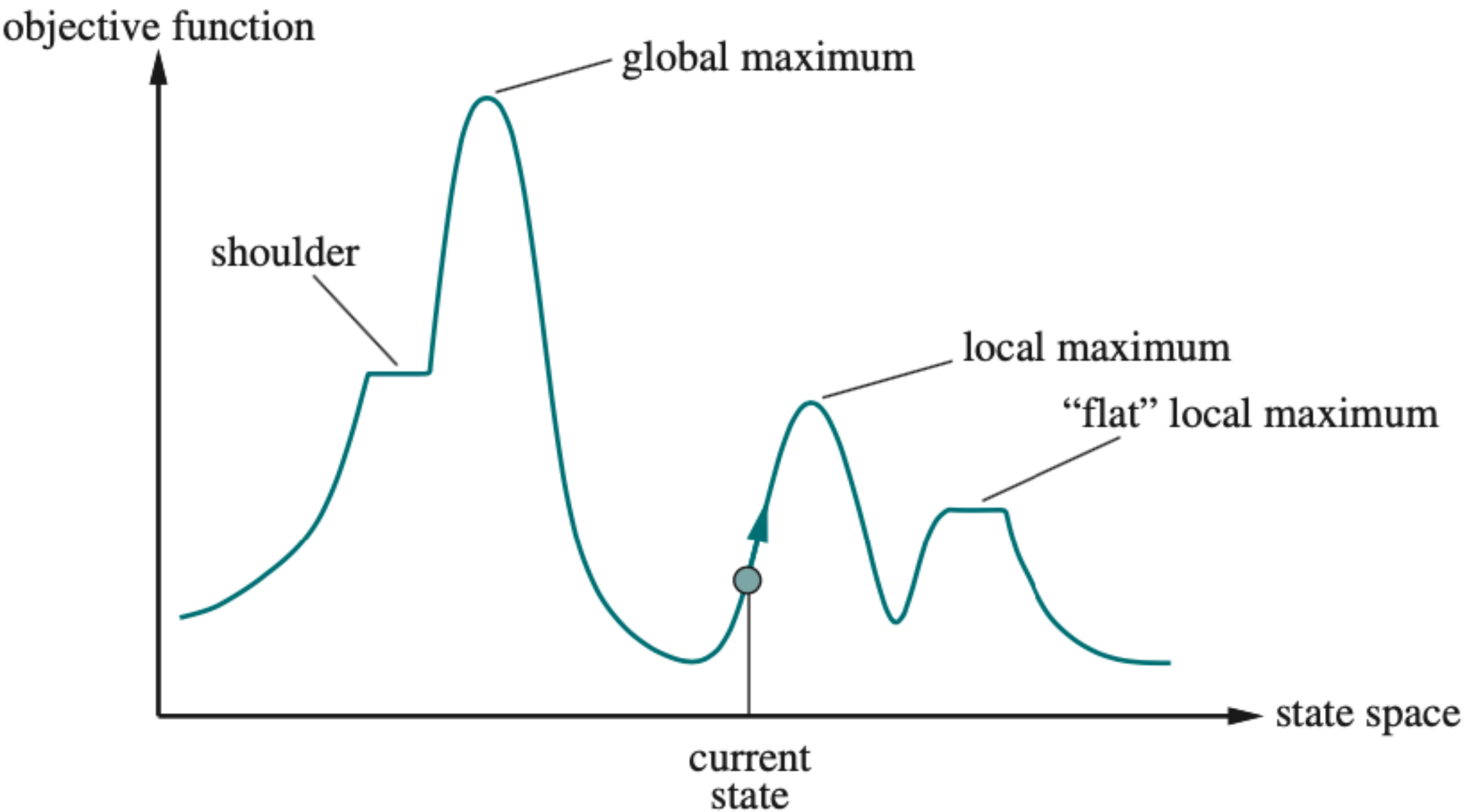
The task is to get tiles 1, 2, 3, 4, and the blank into their correct positions, without worrying about what happens to the other tiles

A Web service providing driving directions, computed by a search algorithm.



Search in Complex Environments

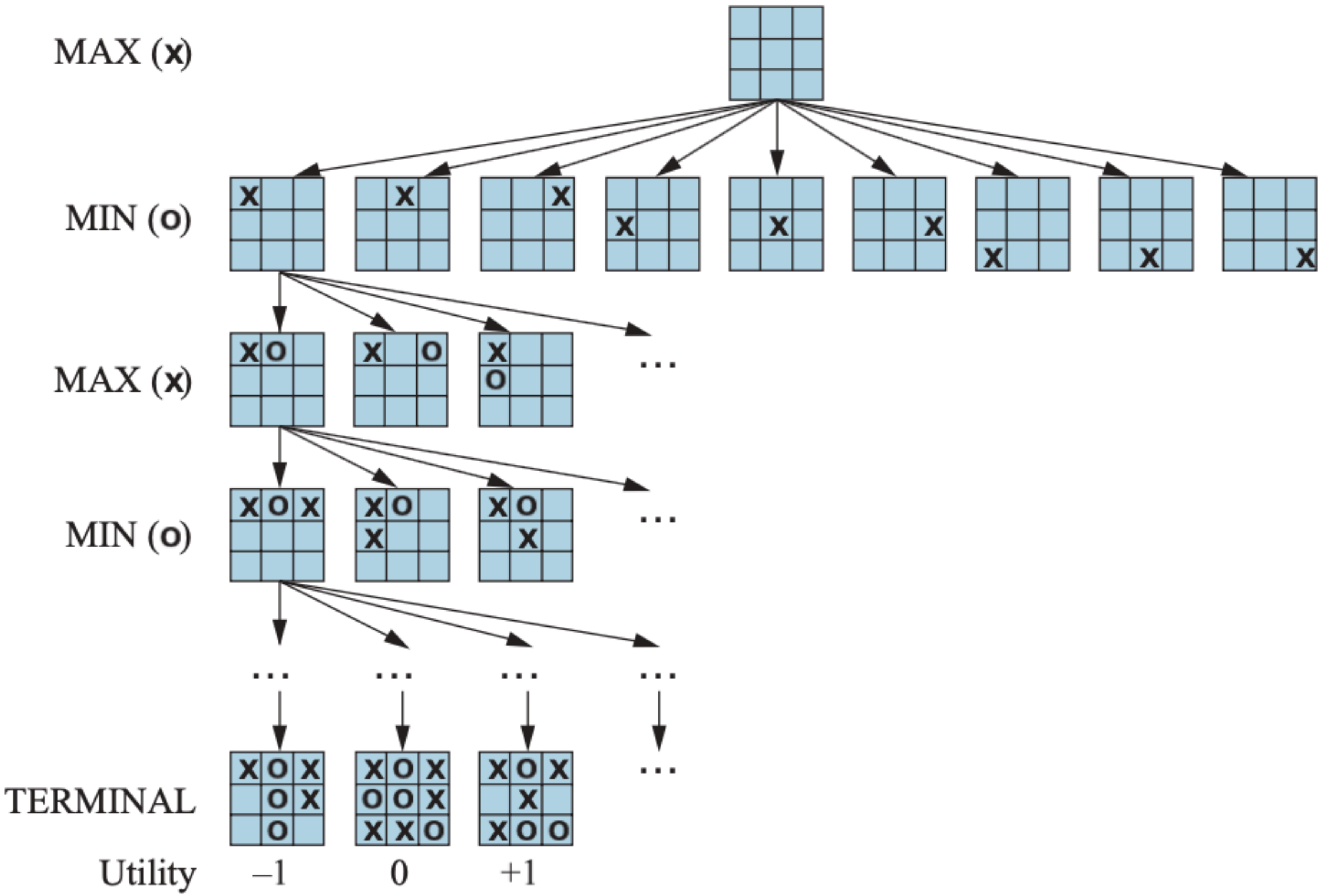
A one-dimensional state-space landscape



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Adversarial Search and Games

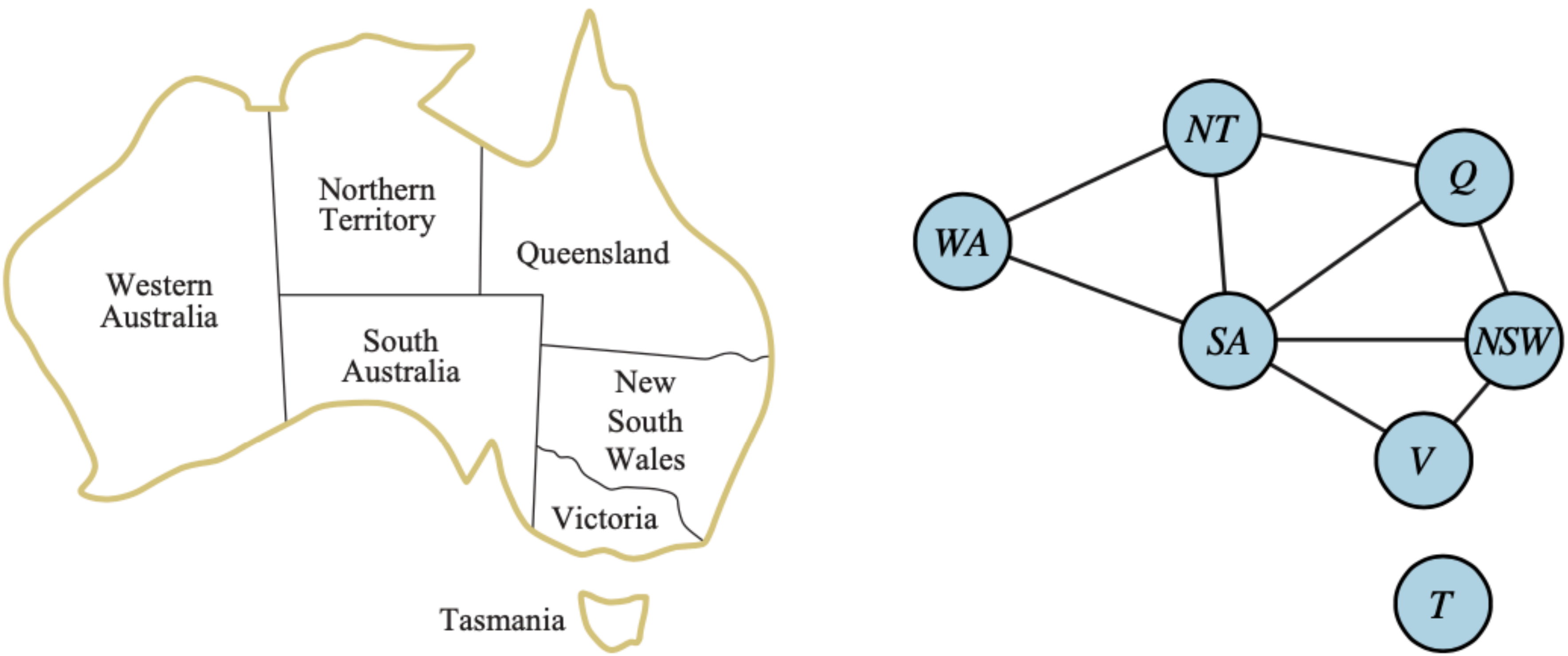
Game Tree for the Game of Tic-tac-toe



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

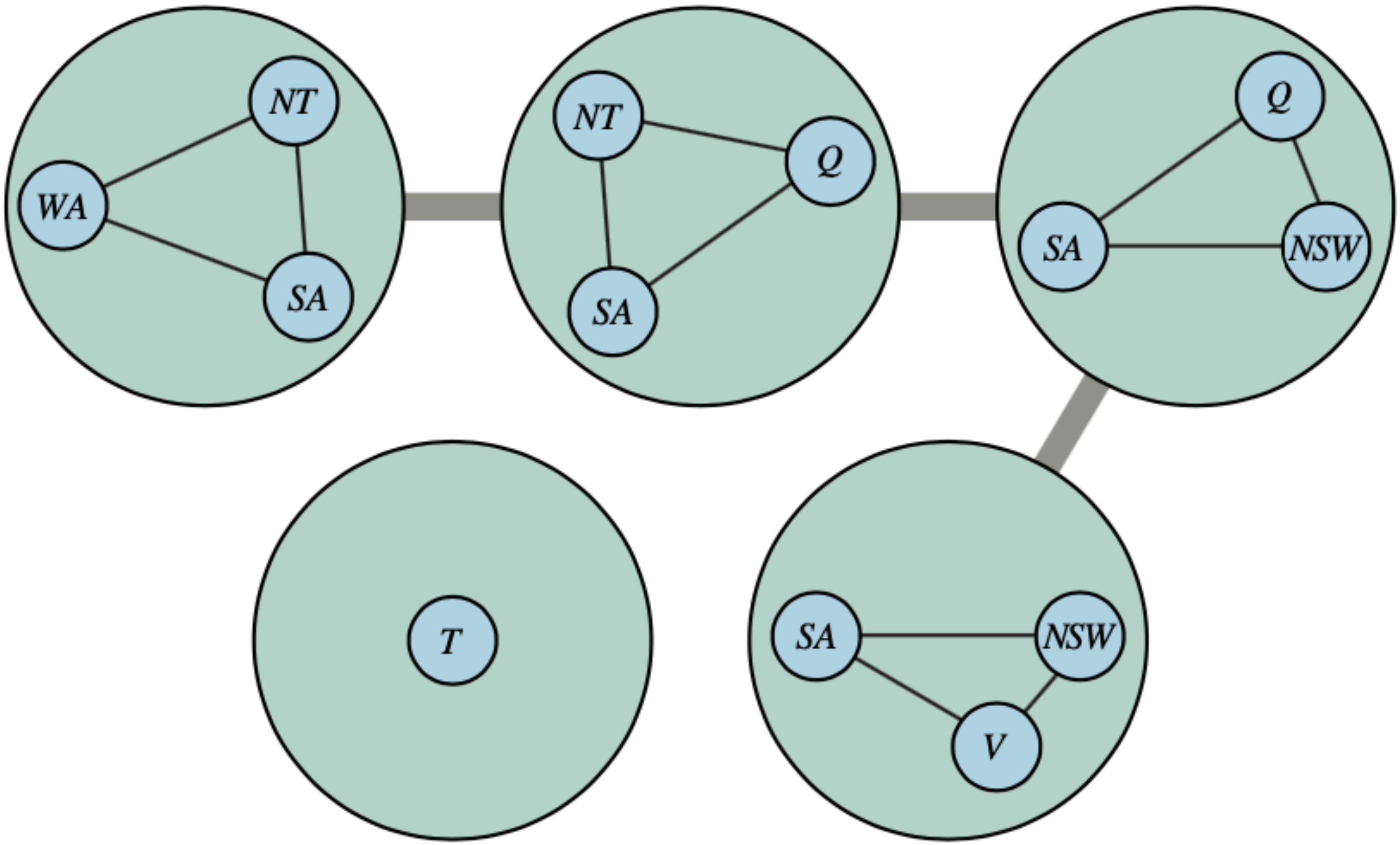
Constraint Satisfaction Problems

The Map-Coloring Problem Represented as a Constraint Graph



(a) (b)

A Tree Decomposition of the Constraint Graph



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

AIMA Python

- Artificial Intelligence: A Modern Approach (AIMA)
  - http://aima.cs.berkeley.edu/
- AIMA Python
  - http://aima.cs.berkeley.edu/python/readme.html
- Search
  - http://aima.cs.berkeley.edu/python/search.html
- Games: Adversarial Search

  http://aima.cs.berkeley.edu/python/games.html
- CSP (Constraint Satisfaction Problems)
  - http://aima.cs.berkeley.edu/python/csp.html

Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

python101.ipynb - Colaboratory

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3

python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

```python
# Future Value
pv = 100
r = 0.1
n = 7
fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

```
194.87
```

[11]
```python
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

```
194.87
```

[12]
```python
# Python Function def
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv
fv = getfv(100, 0.1, 7)
print(round(fv, 2))
```

```
194.87
```

[13]
```python
# Python if else
score = 80
if score >=60 :
    print("Pass")
else:
    print("Fail")
```

```
Pass
```

https://tinyurl.com/aintpupython101

Summary

- **Solving Problems by Searching**
- **Search in Complex Environments**
- **Adversarial Search and Games**
- **Constraint Satisfaction Problems**

References

- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media.