Artificial Intelligence *CS361*

Prof. Abdel-Rahman Hedar

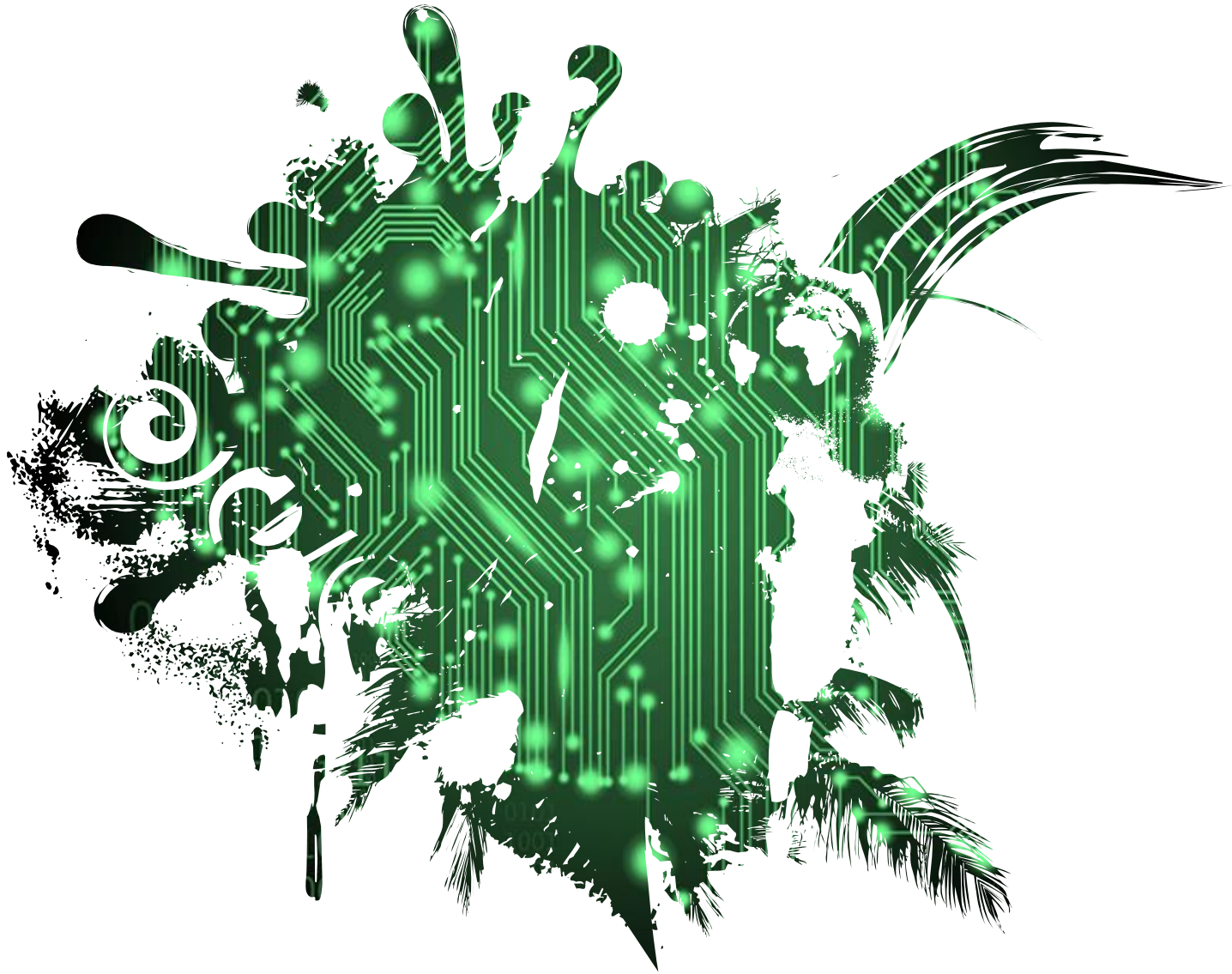
Solving Problems by Searching

Chapter 3 – Part II

Contents

» Informed search
- Best-first search
- Greedy search
- A* search
- Heuristics
- Iterative improvement algorithms

Best-First Search
Revisited



Review: Tree search

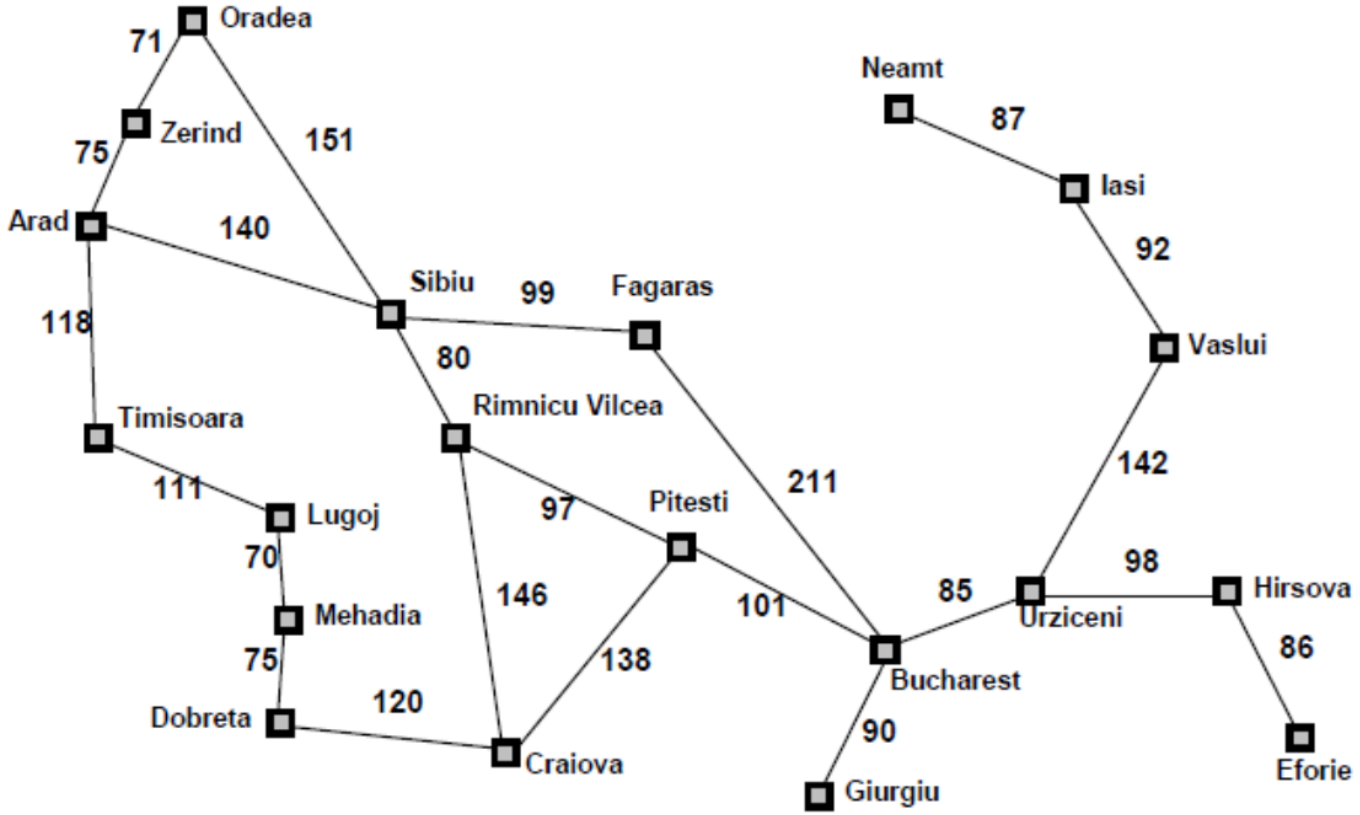
```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem] applied to STATE(node) succeeds return node
        fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

» A strategy is defined by picking **the order of node expansion**.

Best-First Search

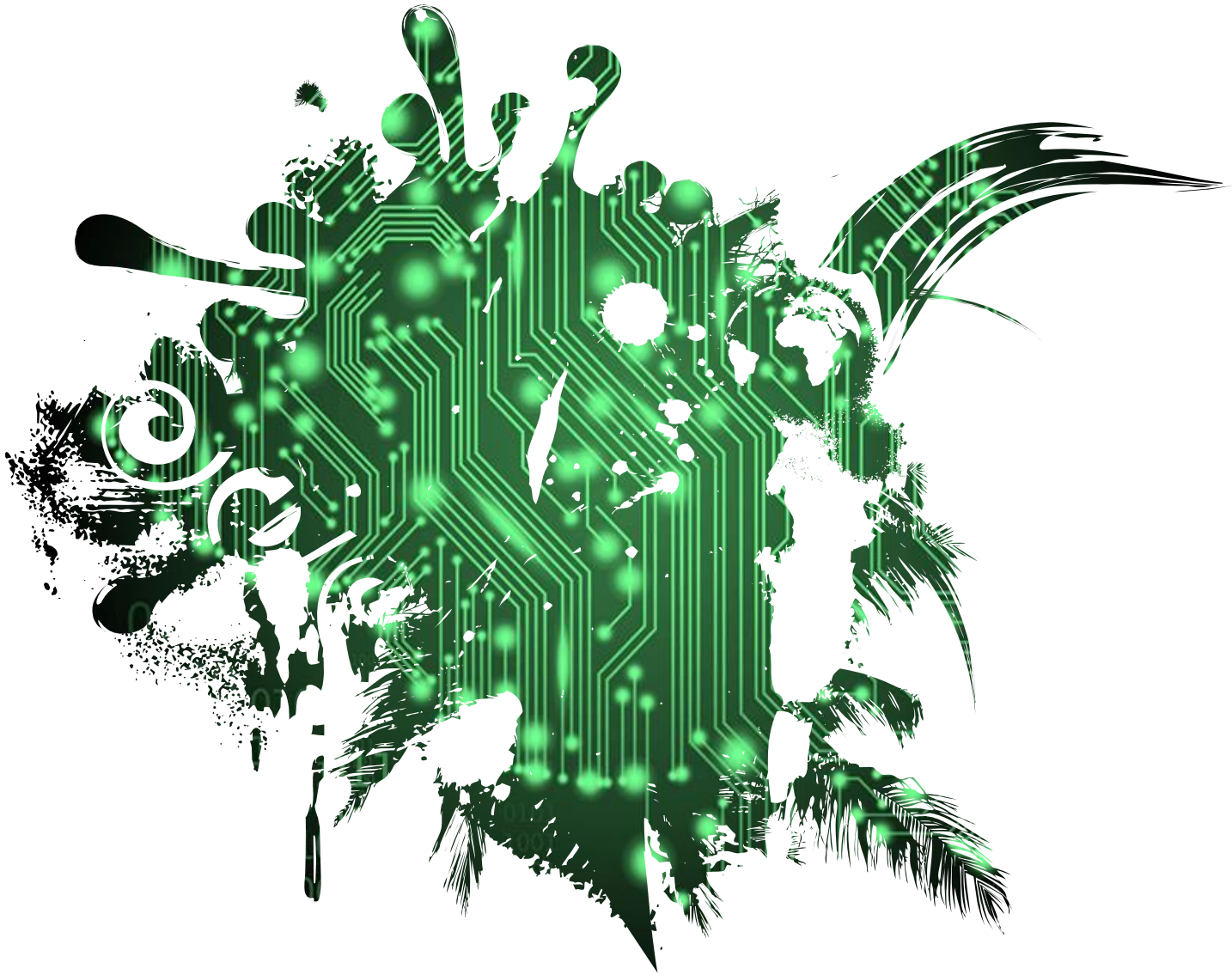
- Idea: use an evaluation function for each node:
  - estimate of "desirability."
  - Expand most desirable unexpanded node.
- Implementation:
  - *frontier* is a queue sorted in decreasing order of desirability.
- Special cases:
  - Greedy best-first search.
  - A* search.

Romania with Step Costs



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

Greedy Search



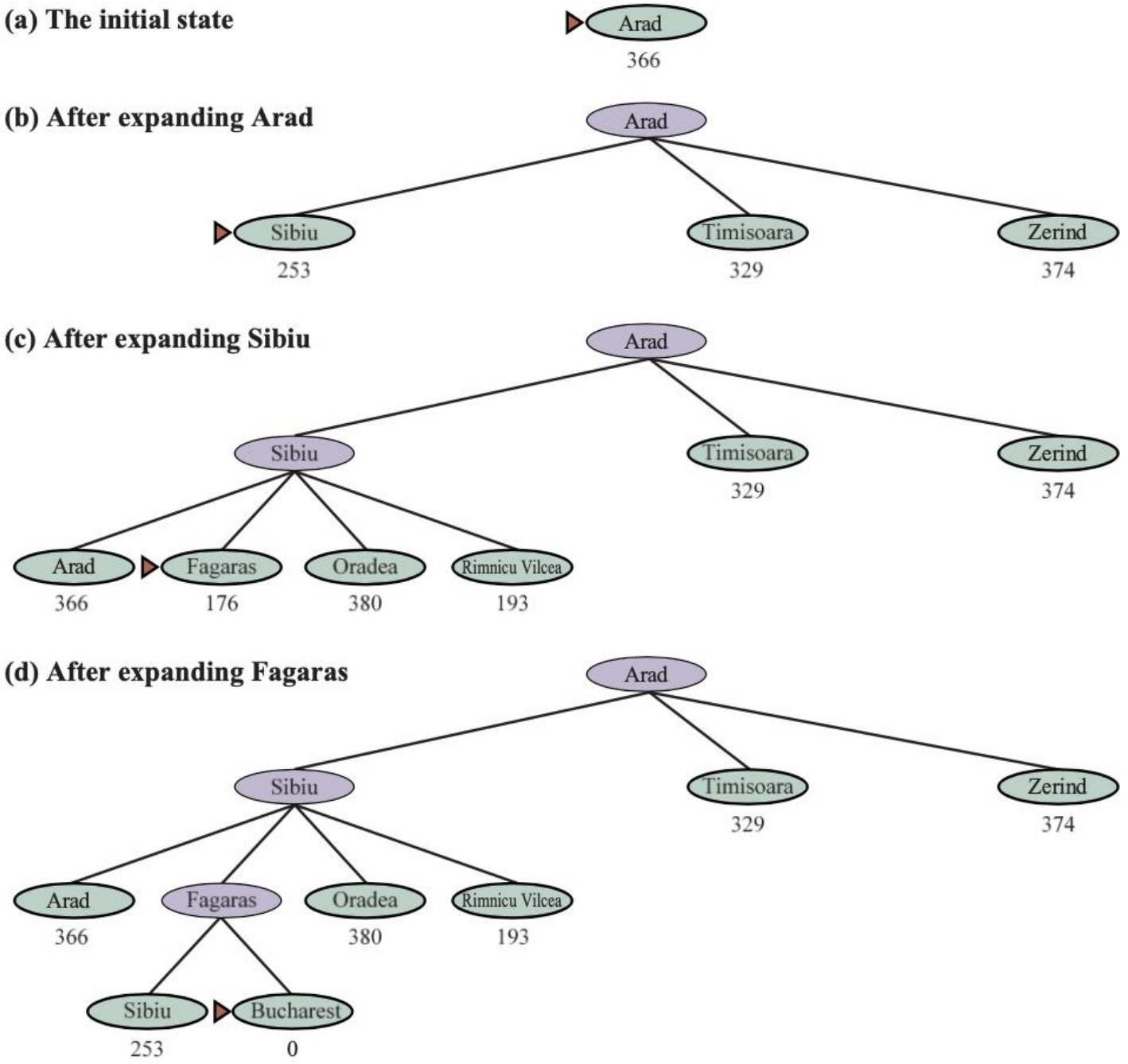
Greedy Search

- Evaluation function $h(n)$ (heuristic)

  = estimate of cost from n to the closest goal.

- E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest.
- Greedy search expands the node that appears to be closest to goal.

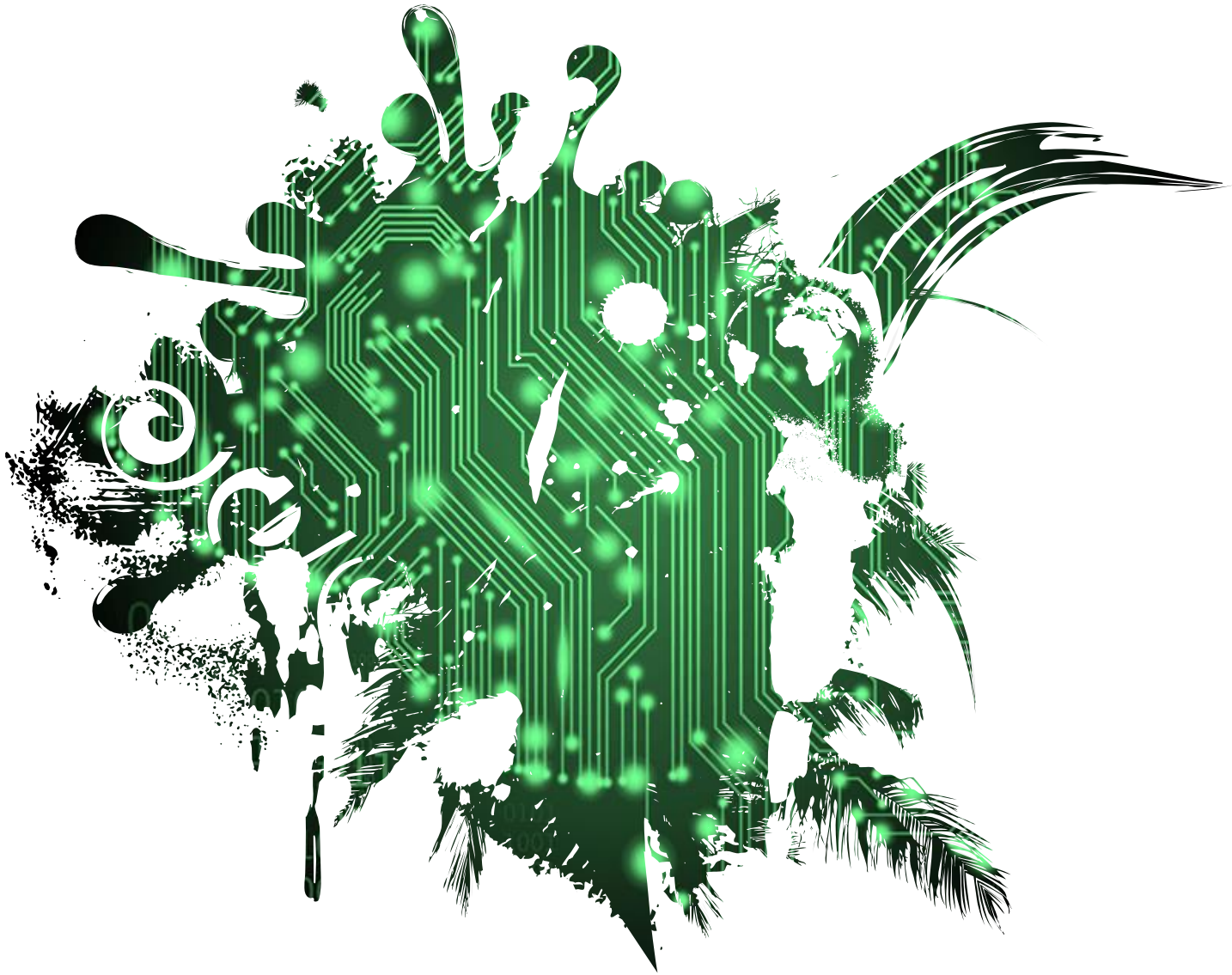
Example


(a) The initial state
Arad
366
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
253
329
374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
329
374
Arad
Fagaras
Oradea
Rimnicu Vilcea
366
176
380
193
(d) After expanding Fagaras
Arad
Sibiu
Timisoara
Zerind
329
374
Arad
Fagaras
Oradea
Rimnicu Vilcea
366
380
193
Sibiu
Bucharest
253
0

Greedy Search Analysis

- **Complete:** No (Can get stuck in loops).
- **Time:** $O(b^m)$, but a good heuristic can give dramatic improvement.
- **Space:** $O(b^m)$, keeps all nodes in memory.
- **Optimal:** No.

A* Search



A* Search

» **Idea:** avoid expanding paths that are already expensive.

» Evaluation function $f(n) = g(n) + h(n)$,

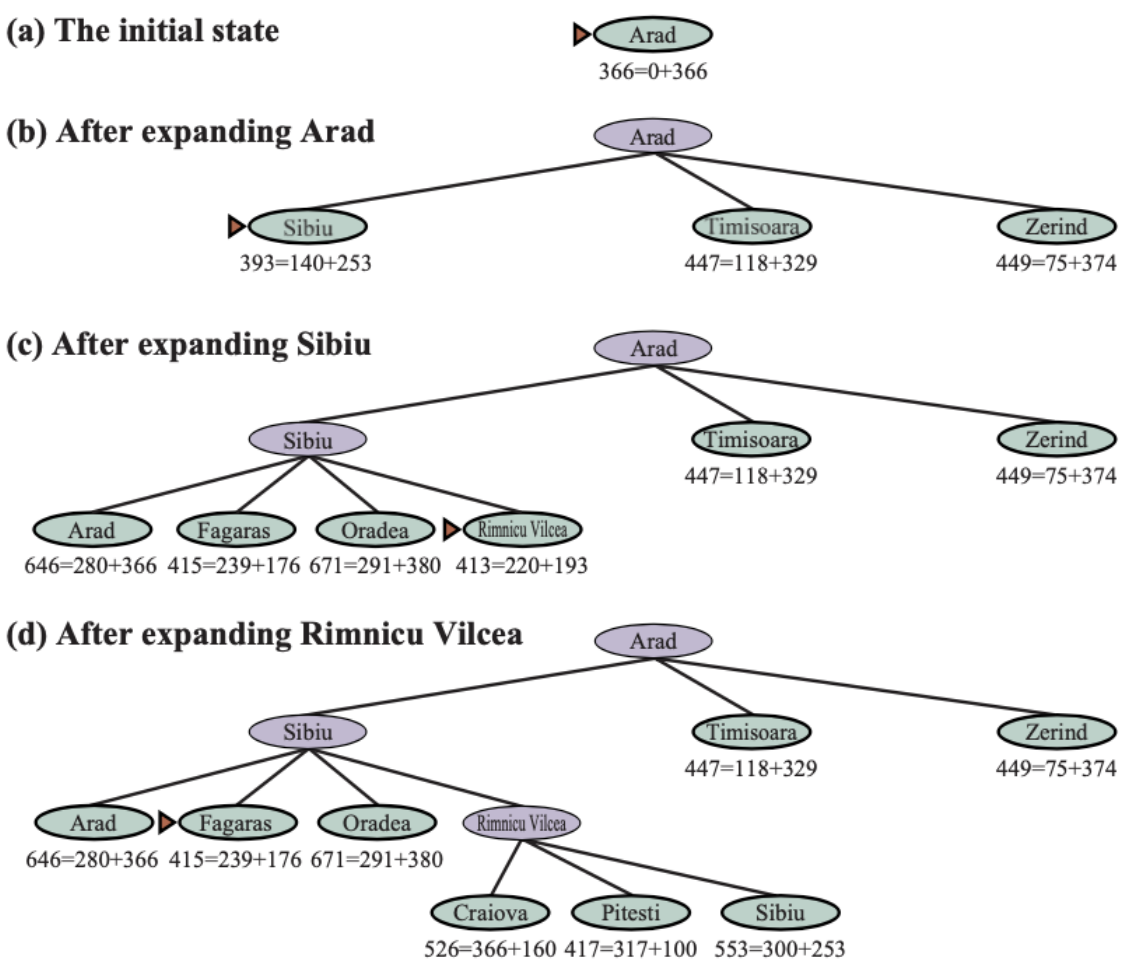
- $g(n)$: cost so far to reach n.
- $h(n)$: estimated cost to goal from n.
- $f(n)$: estimated total cost of path through n to goal.

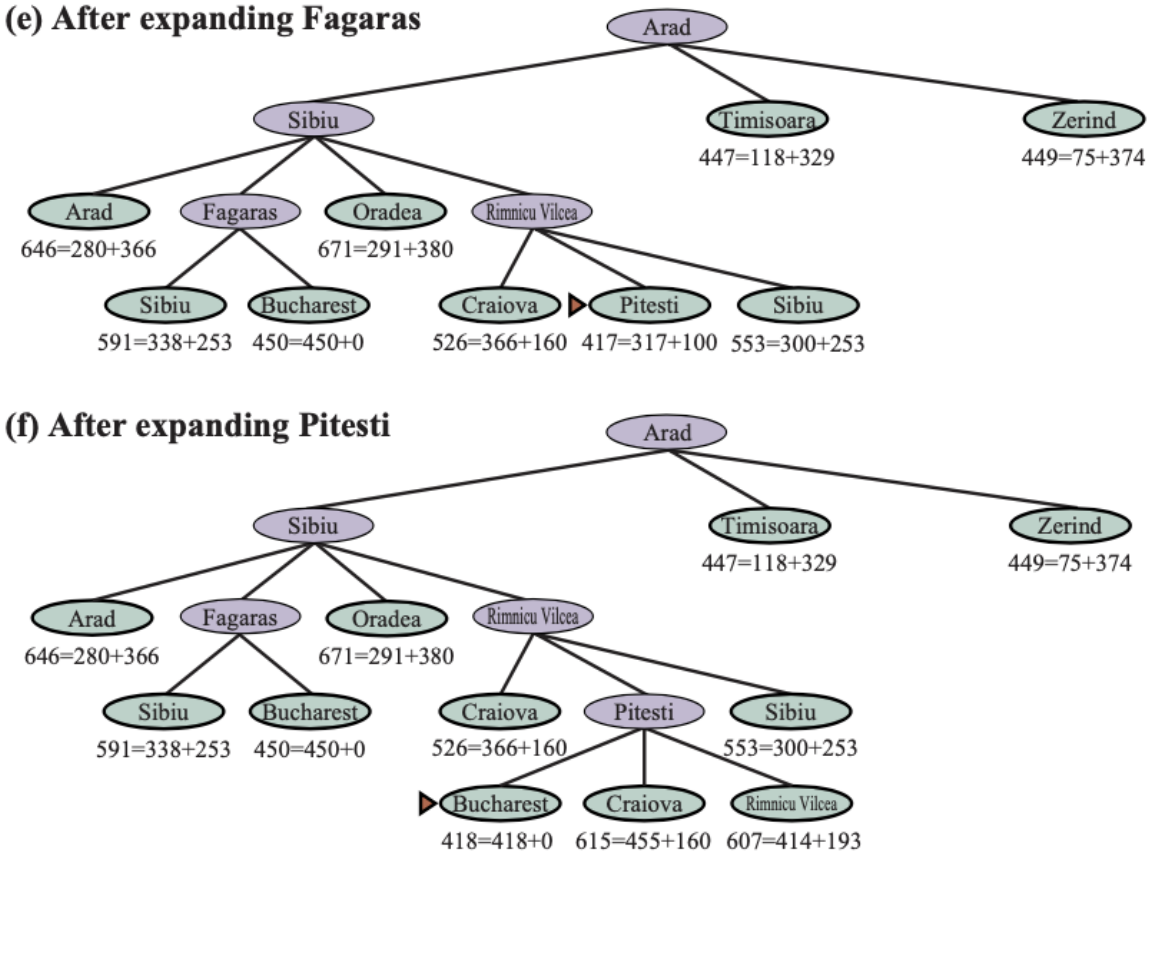
» A* search uses an admissible heuristic (***never overestimate***):

i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the **true** cost from n.
(Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G.)

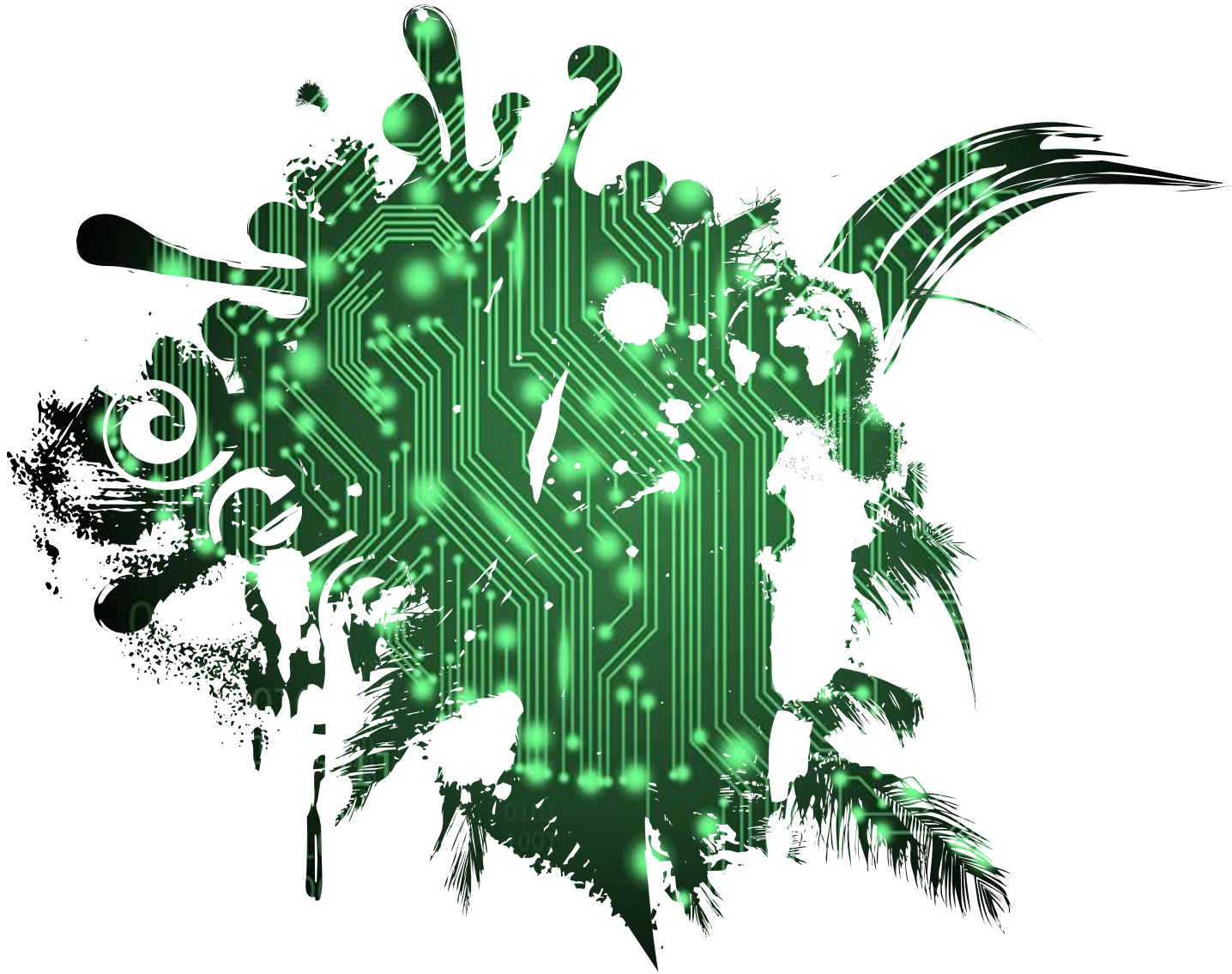
» **Theorem:** A* Search is optimal.

Example


(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
393=140+253
447=118+329
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
413=220+193
(d) After expanding Rimnicu Vilcea
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
Craiova
Pitesti
Sibiu
526=366+160
417=317+100
553=300+253


(e) After expanding Fagaras
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
417=317+100
553=300+253
(f) After expanding Pitesti
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
553=300+253
Bucharest
Craiova
Rimnicu Vilcea
418=418+0
615=455+160
607=414+193

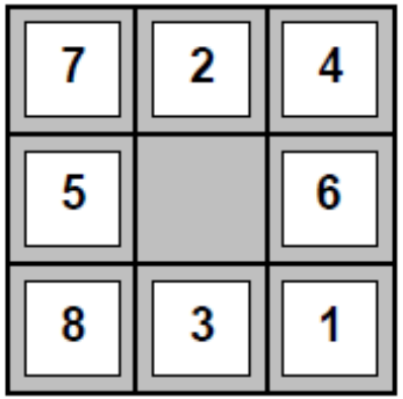
Heuristics



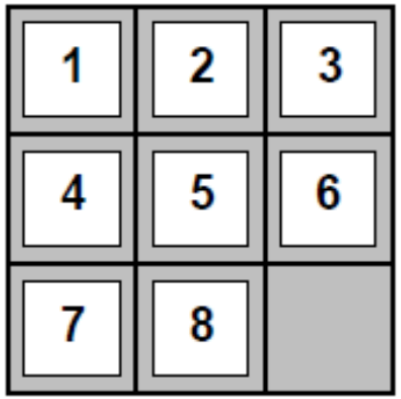
Admissible Heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles
$h_2(n)$ = total Manhattan distance
(i.e., no. of squares from desired location of each tile)



Start State



Goal State

$h_1(S) =$?? 6
$h_2(S) =$?? 4+0+3+3+1+0+2+1 = 14

Dominance

If $h_2(n) \geq h_1(n)$ for all n (both admissible)
then h_2 dominates h_1 and is better for search

Typical search costs:

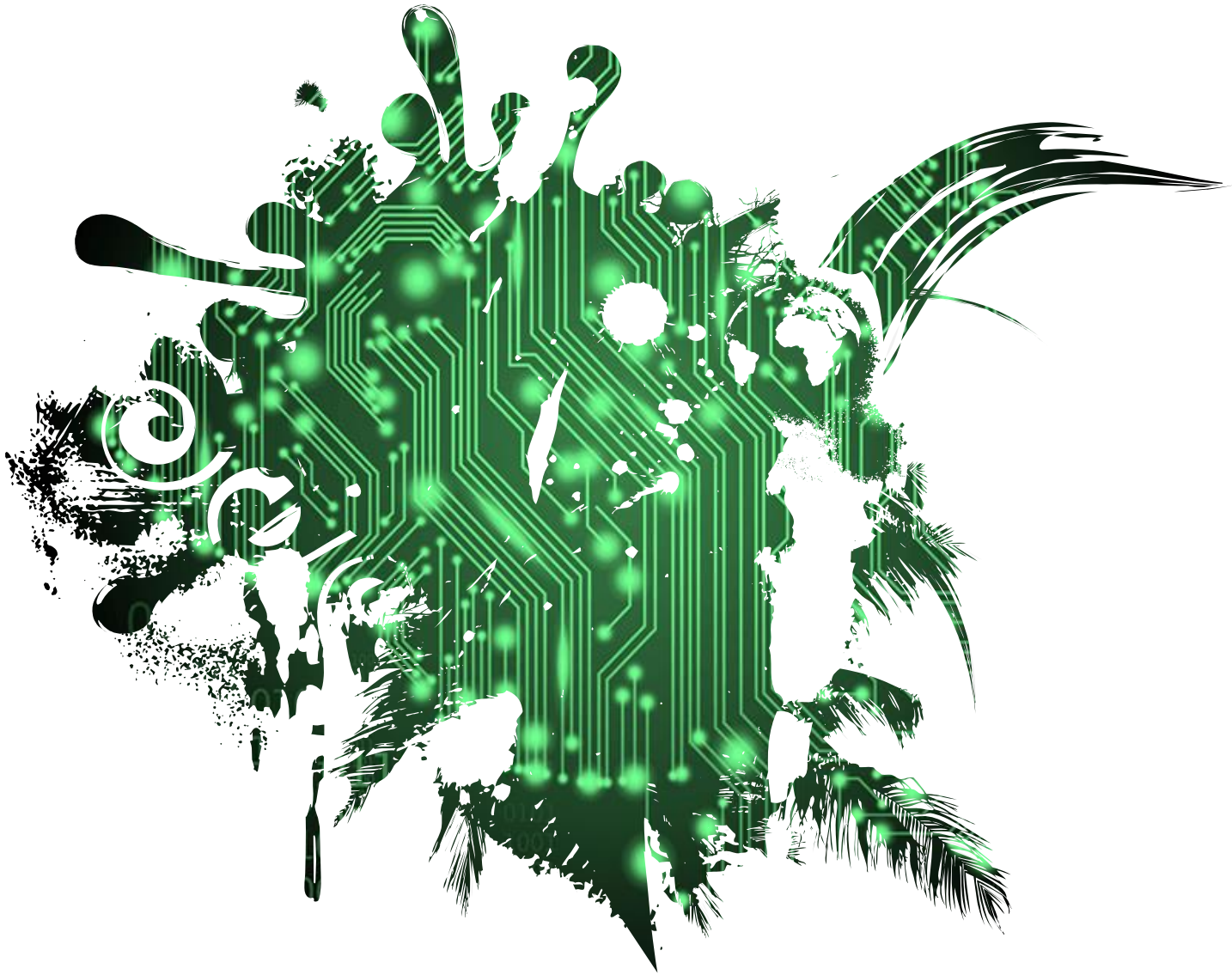
$d = 14$ IDS $= 3{,}473{,}941$ nodes
A$^*(h_1) = 539$ nodes
A$^*(h_2) = 113$ nodes
$d = 24$ IDS $\approx 54{,}000{,}000{,}000$ nodes
A$^*(h_1) = 39{,}135$ nodes
A$^*(h_2) = 1{,}641$ nodes

Given any admissible heuristics h_a, h_b,

$$h(n) = \max(h_a(n), h_b(n))$$

is also admissible and dominates h_a, h_b

Iterative Improvement Algorithms

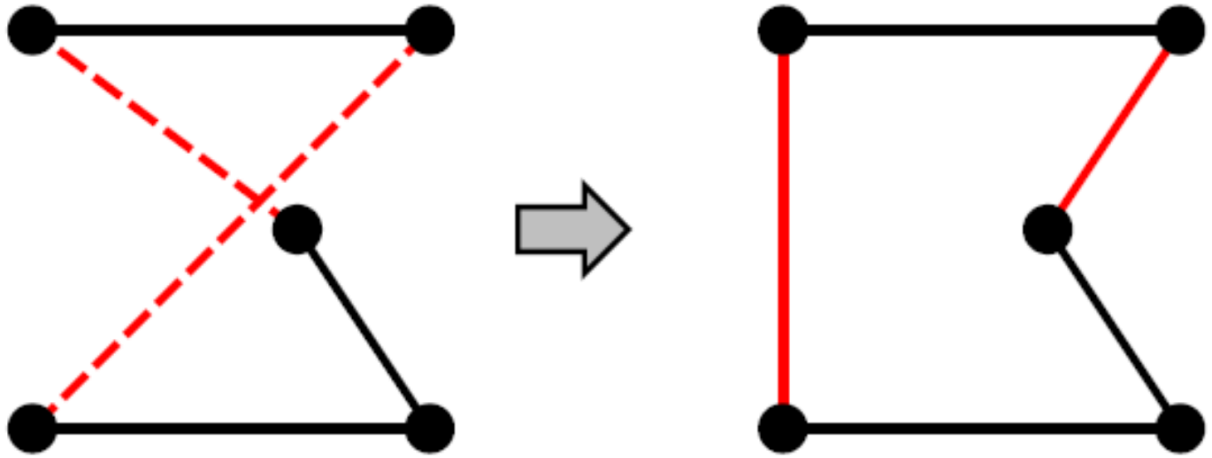


Iterative Improvement Algorithms

- In many optimization problems, path is irrelevant; the goal state itself is the solution.
- Then state space = set of "complete" configurations;
  - find optimal configuration, e.g., TSP or,
  - find configuration satisfying constraints, e.g., timetable.
- In such cases, can use **iterative improvement** algorithms; keep a single "current" state, try to improve it.
- Constant space, suitable for online as well as offline search.

Example: Travelling Salesperson Problem

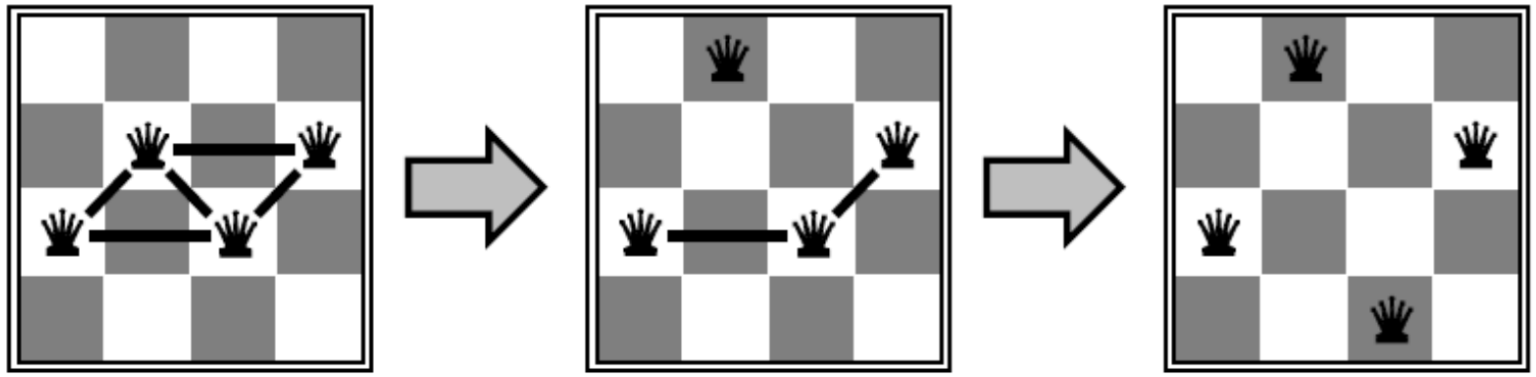
» Start with any complete tour, perform pairwise exchanges



» Variants of this approach get within 1% of optimal very quickly with thousands of cities

Example: *n*-queens

- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal.
- Move a queen to reduce number of conflicts:



- Almost always solves n-queen problems almost instantaneously for very large n, e.g., n = 1 million.

Conclusion

- Heuristic functions estimate costs of shortest paths.
- Good heuristics can dramatically reduce search cost.
- Greedy best-first search expands lowest h (incomplete and not always optimal).
- A* search expands lowest g + h (complete and optimal).
- Admissible heuristics can be derived from exact solution of relaxed problems.

Questions & Comments

Abdel-Rahman Hedar

hedar@au.edu.eg

https://www.aun.edu.eg/fci/