CS3243: Introduction to Artificial Intelligence

Semester 2, 2018/2019

Previously…

- Uninformed search strategies use only the information available in the problem definition
  - Breadth-first search
  - Uniform-cost search
  - Depth-first search
  - Depth-limited search
  - Iterative deepening search
- This class –exploit additional information!

Property	BFS	UCS	DFS	DLS	IDS
Complete	Yes[1]	Yes[2]	No	No	Yes[1]
Optimal	No[3]	Yes	No	No	No[3]
Time	$\mathcal{O}(b^d)$	$\mathcal{O}\left(b^{1+\left\lfloor\frac{C^*}{\varepsilon}\right\rfloor}\right)$	$\mathcal{O}(b^m)$	$\mathcal{O}(b^\ell)$	$\mathcal{O}(b^d)$
Space	$\mathcal{O}(b^d)$	$\mathcal{O}\left(b^{1+\left\lfloor\frac{C^*}{\varepsilon}\right\rfloor}\right)$	$\mathcal{O}(bm)$	$\mathcal{O}(b\ell)$	$\mathcal{O}(bd)$

1. BFS and IDS are complete if b is finite.
2. UCS is complete if b is finite and step cost $\geq \varepsilon$
3. BFS and IDS are optimal if step costs are identical.

Q1. Consider the following parameters $b = 3, d = 3$, what is the overhead of IDS as compared to DLS?

Q2. Consider the following parameters $b = 3, d = 3$, what is the number of nodes to keep track in DFS?

Choosing a Search Strategy

- Depends on the problem:
  - Finite/infinite depth of search tree
  - Known/unknown solution depth
  - Repeated states
  - Identical/non-identical step costs
  - Completeness and optimality needed?
  - Resource constraints (e.g., time, space)

Can We Do Better?

- Yes! Exploit problem-specific knowledge; obtain heuristics to guide search
- Today:
  - Informed (heuristic) search
  - Expand "more promising" nodes.

INFORMED SEARCH

AIMA Chapter 3.5.1 – 3.5.2, 3.6.1 – 3.6.2

Outline

- Best-first search
- Greedy best-first search
- A* search
- Heuristics

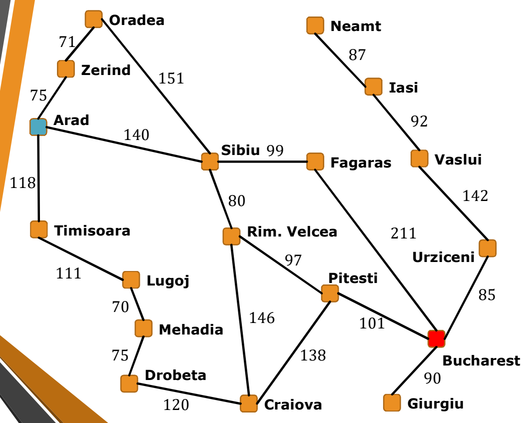
Best-First Search

- Idea: use an evaluation function $f(n)$ for each node n
  - Cost estimate → Expand node with lowest evaluation/cost first
- Implementation:

Frontier = priority queue ordered by nondecreasing cost f

- Special cases (different choices of f):
  - Greedy best-first search
  - A* search

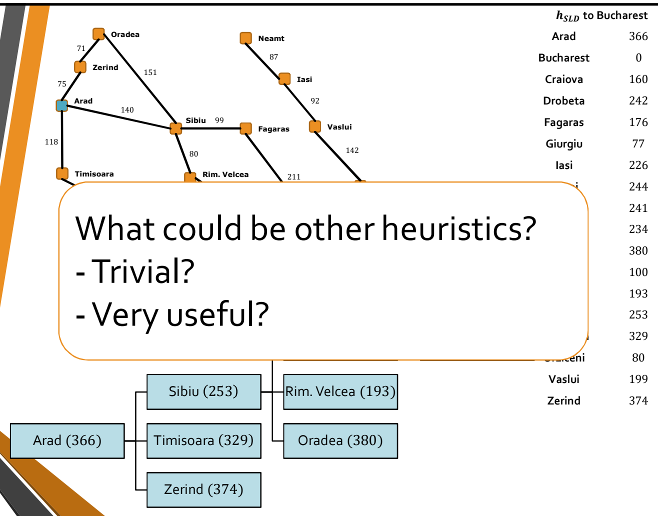
Romania with Step Costs (km)



h_{SLD} to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Fagaras	176
Giurgiu	77
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
R. Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First Search

- Evaluation function $f(n) = h(n)$ (heuristic function) = estimated cost of cheapest path from n to goal
  - Restriction: $h(goal) = 0$
- e.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal



What could be other heuristics?
- Trivial?
- Very useful?

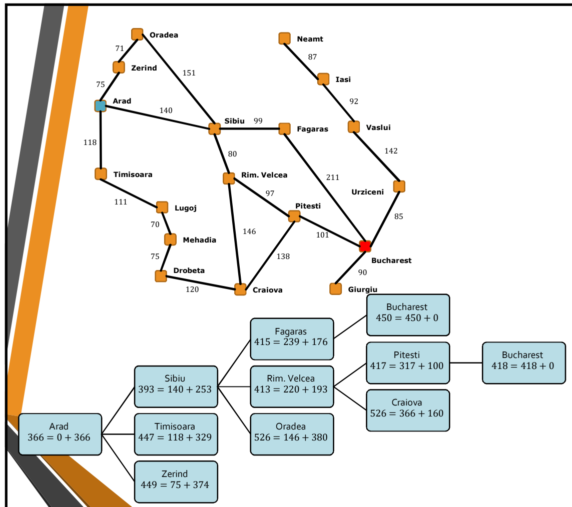
Properties of Greedy Best-First Search

Property	
Complete?	Yes (if b is finite)
Optimal	No (shortest path to Bucharest: $418km$)
Time	$\mathcal{O}(b^m)$, but a good heuristic can reduce complexity substantially
Space	Max size of frontier $\mathcal{O}(b^m)$

What Important Information Does the Algorithm Ignore?

A^* Search

- Idea: Avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost of reaching n from start node
- $h(n)$ = cost estimate from n to goal
- $f(n)$ = estimated cost of cheapest path **through** n to goal



h_{SLD} to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Fagaras	176
Giurgiu	77
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
R. Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Admissible Heuristics

- $h(n)$ is admissible if, $\forall n$, $h(n) \leq h^*(n)$
- $h^*(n) =$ true cost to reach the goal state from n.
- Never overestimates cost to reach goal
- Example: $h_{SLD}(n)$ never overestimates the actual road distance (roads are at best straight!)

Admissible Heuristics

Theorem: If $h(n)$ is admissible, then A^* using TREE-SEARCH is optimal

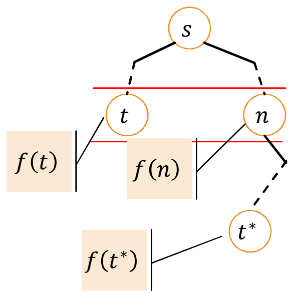
Optimality of A* using TREE-SEARCH

t - a suboptimal goal in the frontier.

n - an unexpanded node in the frontier; n is on a shortest path to an optimal goal t^*.

It would be **very** bad if suboptimal goal node t gets checked before n !!

=> $f(t)$ is lower than $f(n)$



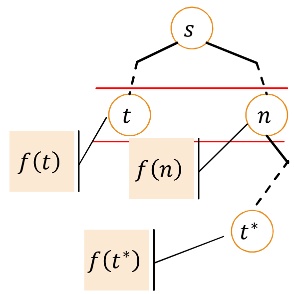
Optimality of A* using TREE-SEARCH

t - a suboptimal goal in the frontier.

n - an unexpanded node in the frontier; n is on a shortest path to an optimal goal t^*.

t gets checked after n if $f(t) > f(n)$

We need to show that t gets checked after n



Optimality of A* using TREE-SEARCH

t - a suboptimal goal in the frontier.

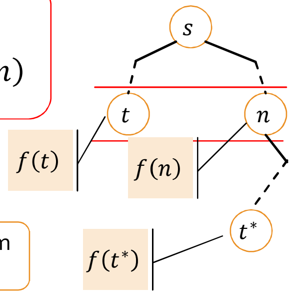
n - an unexpanded node in the frontier; n is on a shortest path to an optimal goal t^*.

t gets checked after n if
$$g(t) + h(t) > g(n) + h(n)$$

In A^*, $f(v) = g(v) + h(v)$

Cost to get to v from s

Est. dist. from v to goal



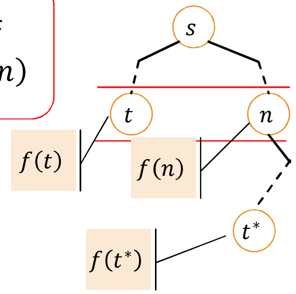
Optimality of A* using TREE-SEARCH

t - a suboptimal goal in the frontier.

n - an unexpanded node in the frontier; n is on a shortest path to an optimal goal t^*.

t gets checked after n if
$$g(t) + h(t) > g(n) + h(n)$$

This is 0



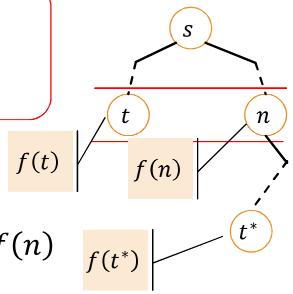
Optimality of A* using TREE-SEARCH

t - a suboptimal goal in the frontier.

n - an unexpanded node in the frontier; n is on a shortest path to an optimal goal t^*.

t gets checked after n if
$$g(t) > g(n) + h(n)$$

$$\begin{aligned} f(t) = g(t) &> g(t^*) \\ &= g(n) + d(n, t^*) \\ &\geq g(n) + h(n) = f(n) \end{aligned}$$

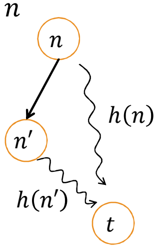


Consistent Heuristics

- A heuristic is consistent if, for every node n and every successor n' of n generated by any action a,
$$h(n) \leq c(n, n') + h(n')$$

Lemma: if h is consistent,

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, n') + h(n') \\ &\geq g(n) + h(n) = f(n) \end{aligned}$$



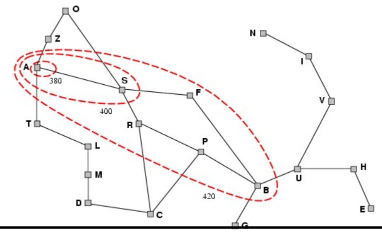
$f(n)$ is non-decreasing along any path.

Consistent Heuristics

Theorem: If $h(n)$ is consistent, then A^* using GRAPH-SEARCH is optimal

Optimality of A^* using GRAPH-SEARCH

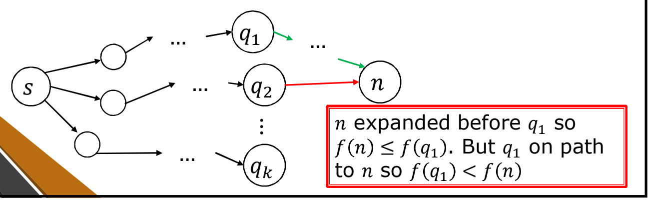
- $f(n)$ is non-decreasing along any path
- A^* expands nodes in non-decreasing order of f value
- Gradually adds "f-contours" of nodes
- Contour i has all nodes with $f = f_i$ where $f_i < f_{i+1}$



Optimality of A^* using GRAPH-SEARCH

Stronger claim: when A^* selects a node n for expansion, **the shortest path to n has been found** (proof by induction on distance from s).

Assume otherwise; let n be the first node reached by suboptimal path ⇒ all nodes before n reached by optimal path.



Properties of A^* Search

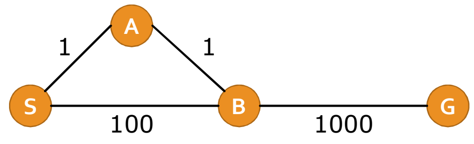
Property	
Complete?	Yes (if there is a finite no. of nodes with $f(n) \le f(G)$)
Optimal	Yes
Time	$O\left(b^{h^*(s_0)-h(s_0)}\right)$ where $h^*(s_0)$ is actual cost of getting from root to goal.
Space	Max size of frontier $O(b^m)$

Admissible vs. Consistent Heuristics

- Why is consistency a stronger sufficient condition than admissibility?
  - Consistent $\Rightarrow$ admissible
  - Admissible $\nRightarrow$ consistent
- $k(n)$ be the cost of cheapest path from n to goal
- To prove, $h(n) \leq k(n)$
  - If n is the goal, $h(n) = 0 \leq k(n)$
  - Induction hypothesis: $h(n') \leq k(n')$
  - Let, path from n to goal has i steps, n' is the successor of n
  - $n' \rightsquigarrow$ goal, $i - 1$ steps
- h is consistent $\Rightarrow$ we have: $h(n) \leq c(n, n') + h(n')$
  - $\leq c(n, n') + k(n') = k(n)$

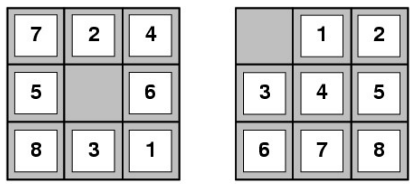
Admissible vs. Consistent Heuristics

- An admissible but inconsistent heuristic cannot guarantee optimality of A* using GRAPH-SEARCH
  - GRAPH-SEARCH discards new paths to a repeated state. May discard the optimal path.
  - Consistent heuristic: always follows optimal path (that lemma was important!)



Admissible Heuristics

- Let's revisit the 8-puzzle
  - Branching factor is about 3
  - Average solution depth is about 22 steps
  - Exhaustive tree search examines 3^{22} states
- How do we come up with good heuristics?

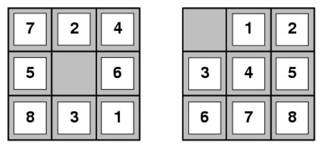


Start State Goal State

Admissible Heuristics

E.g., 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

- $h_1(s) = 8$
- $h_2(s) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible), then h_2 dominates h_1. It follows that h_2 incurs lower search cost than h_1.

Average search costs (nodes generated):

$d = 12$

Algorithm	# Nodes
IDS	3,644,035
$A^*(h_1)$	227
$A^*(h_2)$	73

$d = 24$

Algorithm	# Nodes
IDS	Galactic Number
$A^*(h_1)$	39,135
$A^*(h_2)$	1,641

Deriving Admissible Heuristics

- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem

Deriving Admissible Heuristics

- Rules of 8-puzzle:

  A tile can move from square A to square B if A is horizontally or vertically adjacent to B and B is blank

- We can generate three relaxed problems

  1. A tile can move from square A to square B if A is adjacent to B
  2. A tile can move from square A to square B if B is blank
  3. A tile can move from square A to square B

Deriving Admissible Heuristics

- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ is the resulting heuristic.
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ (Manhattan Dist.) is the resulting heuristic

LOCAL SEARCH

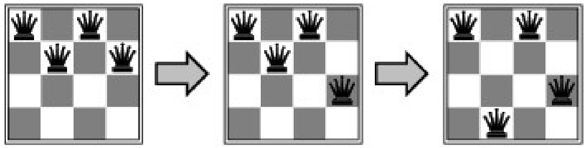
AIMA Chapter 4.1

Local Search Algorithms

- The path to goal is irrelevant; the goal state itself is the solution
- State space = set of "complete" configurations
- Find final configuration satisfying constraints, e.g., n-queens
- Local search algorithms: maintain single "current best" state and try to improve it
- Advantages:
  - very little/constant memory
  - find reasonable solutions in large state space

Example: n-queens

- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal



Hill-Climbing Search

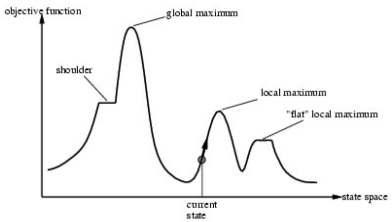
```
function HILL-CLIMBING(problem) returns a state that is a local maximum

  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
      neighbor ← a highest-valued successor of current
      if neighbor.VALUE ≤ current.VALUE then return current.STATE
      current ← neighbor
```

"Like climbing Mt. Everest in thick fog with amnesia"

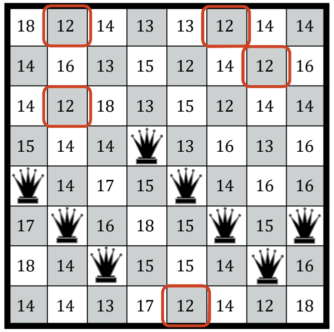
Hill-Climbing Search

- Problem: depending on initial state, can get stuck in local maxima



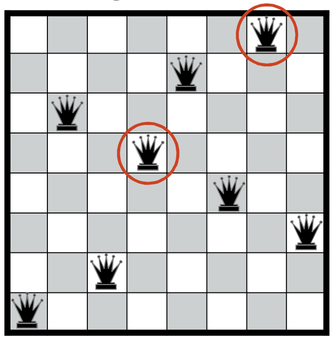
- Non-guaranteed fixes: sideway moves, random restarts

Hill-Climbing Search: 8-Queens



- $h =$ number of pairs of queens that are attacking each other, either directly or indirectly
- $h = 17$ for the above state

Hill-Climbing Search: 8-Queens



Local Minimum with $h = 1$

Hill-climbing

- Use hill-climbing if we are OK with approximate solutions
- If accuracy matters, need to be more careful

Local search strategies

- Hill-climbing search: use of heuristic function to improve "current" state