SWINBURNE UNIVERSITY OF TECHNOLOGY

COS30019: Introduction to Artificial Intelligence

Heuristic Search

1

In previous lectures…

- Our aim: Building systems that act Rationality
- Intell
  - ☐ P
  - ☐ S
    - ☐ Problem formulation (& abstraction), state space
- Uninformed (blind) search
  - ☐ DFS, BFS, uniform-cost search
  - ☐ Why are they "blind"???

Def: *A rational agent chooses whichever action that maximizes the expected value of the performance measure given the percept sequence to date and prior environment knowledge.* (From Week 2)

2

Search Algorithms

- Blind search – BFS, DFS, uniform cost
  - ☐ no notion of the "right direction"
  - ☐ can only recognize goal once it's achieved
- Heuristic search – we have rough idea of how good various … our se…

Def: *A rational agent chooses whichever action that maximizes the expected value of the performance measure given the percept sequence to date and prior environment know…*

Prior environment knowledge ➔ heuristics

3

Previously: tree-search

function TREE-SEARCH(*problem, frontier*) **return** a solution or failure

frontier ← INSERT(MAKE-NODE(INITIAL-STATE[*problem*]), *frontier*)

loop do

if EMPTY?(*frontier*) **then return** failure

node ← REMOVE-FIRST(*frontier*)

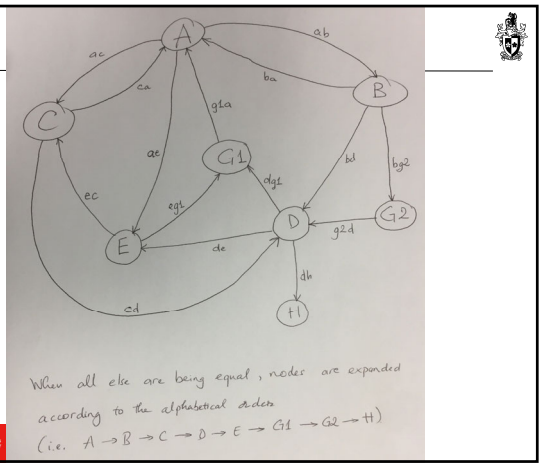
if GOAL-TEST[*problem*] applied to STATE[*node*] succeeds

then return SOLUTION(*node*)

frontier ← INSERT-ALL(EXPAND(*node, problem*), *frontier*)

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

4



5

In this lecture…

- An informed strategy uses problem-specific knowledge to pick the "more promising" node
- Which search strategies?
  - ☐ Best-first search and its variants
- Heuristic functions
- Local search and optimization
  - ☑ Hill climbing,
  - ☐ local beam search,
  - ☐ genetic algorithms,
  - ☐ …

6

Best-first search

- General approach of informed search:
  - Best-first search: node is selected for expansion based on an *evaluation function f(n)*
- Idea: evaluation function measures distance to the goal.
  - Choose node which *appears* best
- Implementation:
  - *frontier* is queue sorted in decreasing order of desirability.
  - Special cases: greedy search, A* search

7

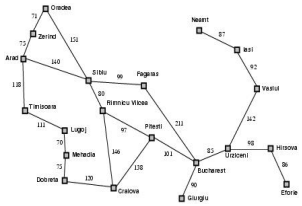
A heuristic function

- [dictionary]*"A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood."*
  - $h(n)$ = estimated cost of the cheapest path from node *n* to goal node.
  - If *n* is goal then $h(n)=0$

8

Example: Route-finding in Romania

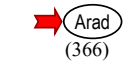
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



- h_{SLD}=straight-line distance heuristic.
- h_{SLD} can **NOT** be computed from the problem description itself
- In this example $f(n)=h(n)$
  - Expand node that is closest to goal
    = Greedy best-first search

9

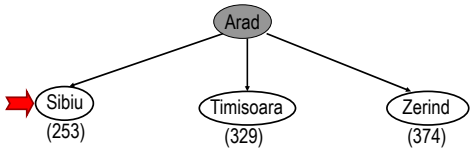
Greedy best-first search example



- Assume that we want to use greedy search to solve the problem of traveling from Arad to Bucharest.
- The initial state=Arad

10

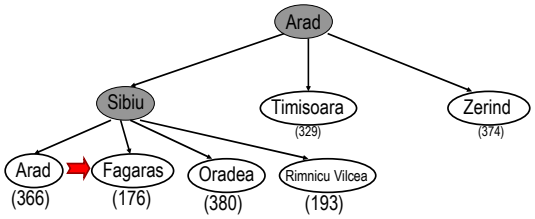
Greedy best-first search example



- The first expansion step produces:
  - Sibiu, Timisoara and Zerind
- Greedy best-first will select Sibiu.

11

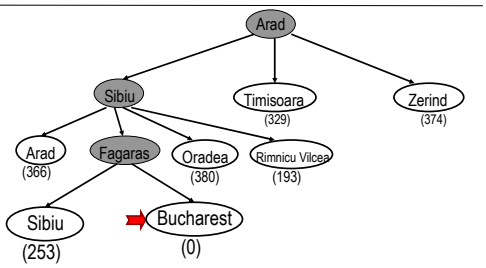
Greedy best-first search example



- If Sibiu is expanded we get:
  - Arad, Fagaras, Oradea and Rimnicu Vilcea
- Greedy best-first search will select: Fagaras

12

Greedy best-first search example

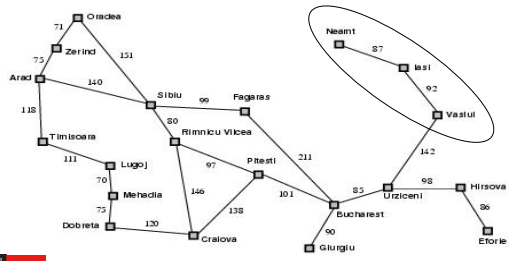


- If Fagaras is expanded we get:
  - Sibiu and Bucharest
- Goal reached !!
  - Yet not optimal (see Arad, Sibiu, Rimnicu Vilcea, Pitesti)

13

Greedy best-first search, evaluation

- Completeness: NO (cfr. DF-search)
  - Check on repeated states
  - Minimizing ***h(n)*** can result in false starts, e.g. Iasi to Fagaras.



14

Greedy best-first search, evaluation

- Completeness: NO (cfr. DF-search)
- Time complexity? $O(b^m)$
  - Cfr. Worst-case DF-search
  - (with m is maximum depth of search space)
  - Good heuristic can give dramatic improvement.

15

Greedy best-first search, evaluation

- Completeness: NO (cfr. DF-search)
- Time complexity: $O(b^m)$
- Space complexity: $O(b^m)$
  - Keeps all nodes in memory

16

Greedy best-first search, evaluation

- Completeness: NO (cfr. DF-search)
- Time complexity: $O(b^m)$
- Space complexity: $O(b^m)$
- Optimality? NO
  - Same as DF-search

17

A* search

- Best-known form of best-first search.
- Important AI algorithm developed by Fikes and Nilsson in early 70s. Originally used in Shakey robot.
- Idea: avoid expanding paths that are already expensive.
- Evaluation function $f(n)=g(n) + h(n)$
  - $g(n)$ the cost (so far) to reach the node.
  - $h(n)$ estimated cost to get from the node to the goal.
  - $f(n)$ estimated total cost of path through n to goal.

18

A* search

- A* search uses an admissible heuristic
  - A heuristic is admissible if it *never overestimates* the cost to reach the goal
  - Are optimistic

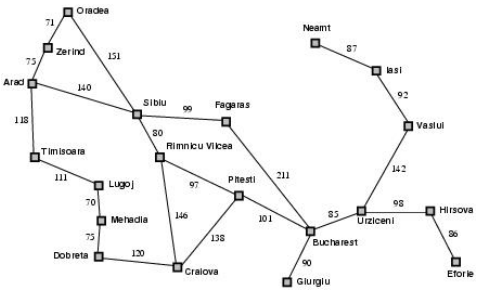
Formally:

1. $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost from n
2. $h(n) >= 0$ so $h(G)=0$ for any goal G.

e.g. $h_{SLD}(n)$ never overestimates the actual road distance

19

Romania example



20

A* search example

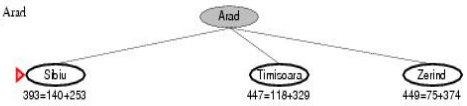
(a) The initial state

- Find Bucharest starting at Arad
  - f(Arad) = c(Arad,Arad)+h(Arad)=0+366=366

21

A* search example

After expanding Arad

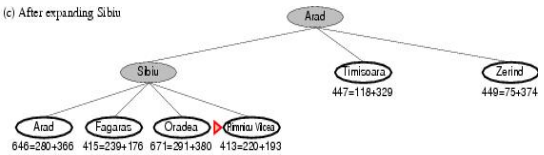


- Expand Arrad and determine *f(n)* for each node
  - f(Sibiu)=c(Arad,Sibiu)+h(Sibiu)=140+253=393
  - f(Timisoara)=c(Arad,Timisoara)+h(Timisoara)=118+329=447
  - f(Zerind)=c(Arad,Zerind)+h(Zerind)=75+374=449
- Best choice is Sibiu

22

A* search example

(c) After expanding Sibiu

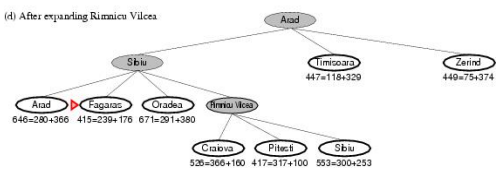


- Expand Sibiu and determine *f(n)* for each node
  - f(Arad)=c(Sibiu,Arad)+h(Arad)=280+366=646
  - f(Fagaras)=c(Sibiu,Fagaras)+h(Fagaras)=239+179=415
  - f(Oradea)=c(Sibiu,Oradea)+h(Oradea)=291+380=671
  - f(Rimnicu Vilcea)=c(Sibiu,Rimnicu Vilcea)+h(Rimnicu Vilcea)=220+192=413
- Best choice is Rimnicu Vilcea

23

A* search example

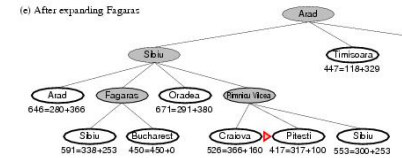
(d) After expanding Rimnicu Vilcea



- Expand Rimnicu Vilcea and determine *f(n)* for each node
  - f(Craiova)=c(Rimnicu Vilcea,Craiova)+h(Craiova)=360+160=526
  - f(Pitesti)=c(Rimnicu Vilcea, Pitesti)+h(Pitesti)=317+100=417
  - f(Sibiu)=c(Rimnicu Vilcea,Sibiu)+h(Sibiu)=300+253=553
- Best choice is Fagaras

24

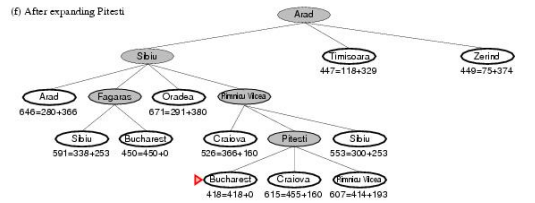
A* search example



- Expand Fagaras and determine *f(n)* for each node
  - f(Sibiu)=c(Fagaras, Sibiu)+h(Sibiu)=338+253=591
  - *f(Bucharest)=c(Fagaras,Bucharest)+h(Bucharest)=450+0=450*
- Best choice is Pitesti !!!

25

A* search example



- Expand Pitesti and determine *f(n)* for each node
  - *f(Bucharest)=c(Pitesti,Bucharest)+h(Bucharest)=418+0=418*
- Best choice is Bucharest !!!
  - Optimal solution (only if *h(n)* is admissible)
- Note values along optimal path !!

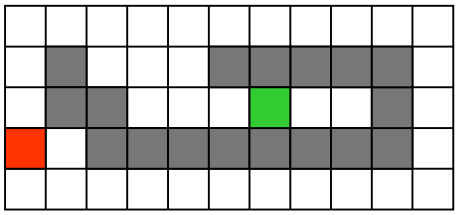
26

A* search, evaluation

- Completeness: YES (Unless there are infinitely many nodes *n* with $f(n) \leq f(G)$)
- Time complexity: (exponential with path length)
- Space complexity: (all nodes are stored)
- Optimality: YES (if the heuristic function *h* is admissible)
- *Formal analyses can be found in AIMA*

27

Another example: Robot Navigation



28

Robot Navigation: Greedy best-first search

f(N) = h(N), with h(N) = Manhattan distance to the goal

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

29

Robot Navigation : Greedy best-first search

f(N) = h(N), with h(N) = Manhattan distance to the goal

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									
8	7	6	5	4	3	2	3	4	5	6

What happened???

30

Robot Navigation : A* search

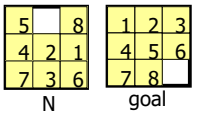
f(N) = g(N)+h(N), with h(N) = Manhattan distance to goal

8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1			3	2+9	1+10	0+11	1	2		4
7+0	6+1									5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

31

Heuristic Function

- Function h(N) that estimates the cost of the cheapest path from node N to goal node.
- Example: 8-puzzle

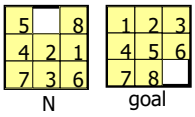


h(N) = number of misplaced tiles
= 6

32

Heuristic Function

- Function h(N) that estimate the cost of the cheapest path from node N to goal node.
- Example: 8-puzzle

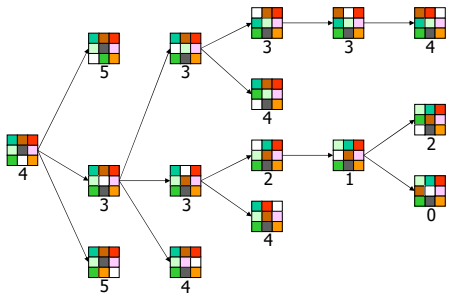


h(N) = sum of the distances of every tile to its goal position
= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1
= 13

33

8-Puzzle

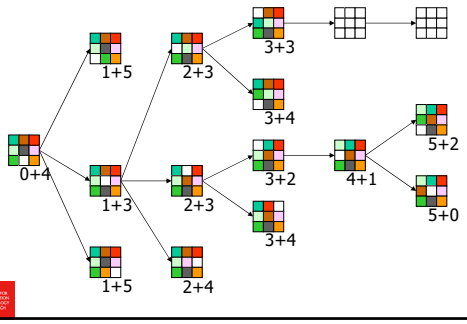
f(N) = h(N) = number of misplaced tiles



34

8-Puzzle

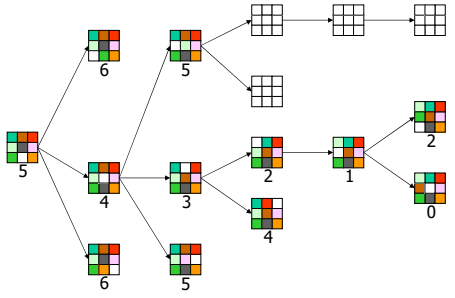
f(N) = g(N) + h(N)
with h(N) = number of misplaced tiles



35

8-Puzzle

f(N) = h(N) = Σ distances of tiles to goal



36

8-Puzzle

5		8
4	2	1
7	3	6

N

1	2	3
4	5	6
7	8	

goal

- $h_1(N)$ = number of misplaced tiles = 6 is admissible
- $h_2(N)$ = sum of distances of each tile to goal = 13 is admissible
- $h_3(N)$ = (sum of distances of each tile to goal) + 3 x (sum of score functions for each tile) = 49 is not admissible

37

About Heuristics

- Heuristics are intended to orient the search along promising paths
- The time spent computing heuristics must be recovered by a better search
- After all, a heuristic function could consist of solving the problem; then it would perfectly guide the search
- Deciding which node to expand is sometimes called meta-reasoning
- Heuristics may not always look like numbers and may involve large amount of knowledge

38

What's the Issue?

- Search is an iterative local procedure
- Good heuristics should provide some global look-ahead (at low computational cost)

39

Another approach…

- for optimization problems
  - rather than constructing an optimal solution from scratch, start with a suboptimal solution and iteratively improve it
- Local Search Algorithms
  - Hill-climbing or Gradient descent
  - Potential Fields
  - Simulated Annealing
  - Genetic Algorithms, others…

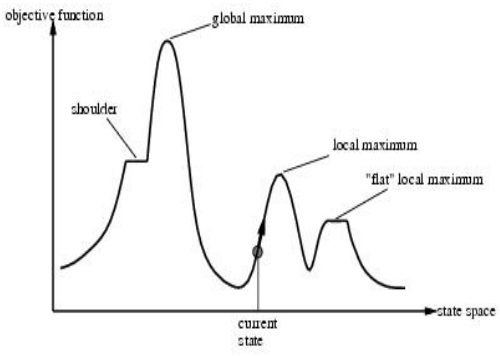
40

Hill-climbing search

- If there exists a successor s for the current state n such that
  - h(s) < h(n)
  - h(s) <= h(t) for all the successors t of n,
- then move from n to s. Otherwise, halt at n.
- Looks one step ahead to determine if any successor is better than the current state; if there is, move to the best successor.
- Similar to Greedy search in that it uses h, but does not allow backtracking or jumping to an alternative path since it doesn't "remember" where it has been.
- Not complete since the search will terminate at "local maxima," "plateaus," and "ridges."

41

Local search and optimization



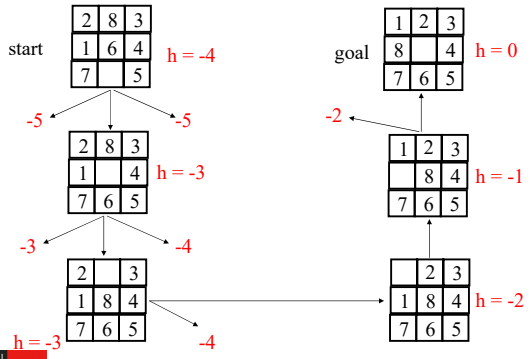
42

Drawbacks of hill climbing

- Problems:
  - **Local Maxima**: peaks that aren't the highest point in the space
  - **Plateaus:** the space has a broad flat region that gives the search algorithm no direction (random walk)
- Remedy:
  - Introduce randomness
    - Random restart.
- Some problem spaces are great for hill climbing and others are terrible.

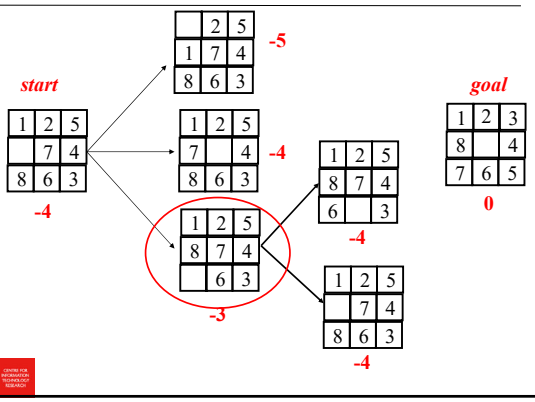
43

Hill climbing example



44

Example of a local maximum



45

When to Use Search Techniques?

- The search space is small, and
  - There is no other available techniques, or
  - It is not worth the effort to develop a more efficient technique
- The search space is large, and
  - There is no other available techniques, and
  - There exist "good" heuristics

46

Summary

- Heuristic function
- Best-first search
  - Greedy best-first search
  - Admissible heuristic and A*
- Heuristic accuracy
- Optimisation problems
  - Hill climbing

47