Heuristic (Informed) Search

(Where we try to choose smartly)

R&N: Chap. 4, Sect. 4.1–3

Slides from Jean-Claude Latombe at Stanford University
(used with permission)

1

Recall that the ordering of FRINGE defines the search strategy

Search Algorithm #2

SEARCH#2
1. INSERT(initial-node,FRINGE)
2. Repeat:
   a. If empty(FRINGE) then return failure
   b. N ← REMOVE(FRINGE)
   c. s ← STATE(N)
   d. If GOAL?(s) then return path or goal state
   e. For every state s' in SUCCESSORS(s)
      i. Create a node N' as a successor of N
      ii. INSERT(N',FRINGE)

2

Are We Smart Yet?

- So far we've been "blundering about in the dark"!
- Let's try to be smarter
- Informed strategies could find solutions more efficiently than uninformed ones
- We'll consider a new instance of the Tree-Search/Graph-Search called Best-First Search, which chooses nodes for expanson based on an evaluation function

3

Best-First Search

- It exploits state description to estimate how "good" each search node is
- An evaluation function f maps each node N of the search tree to a real number $f(N) \geq 0$
  [Traditionally, f(N) is an estimated cost; so, the smaller f(N), the more promising N]
- Best-first search sorts the FRINGE in increasing f
  [Arbitrary order is assumed among nodes with equal f]

4

Best-First Search

- It exploits state description to estimate how "good" each search node is
- An evaluation function f maps each node N of the search tree to a real number $f(N) \geq 0$
  [Traditionally, f(N) is an estimated cost; so, the smaller f(N), the more promising N]
- Best-first search sorts the FRINGE in increasing f
  [Random order is assumed among nodes with equal f]

"Best" does not refer to the quality of the generated path
Best-first search does not generate optimal paths in general

5

How to construct f?

- Typically, f(N) estimates:
  - either the cost of a solution path through N
    Then $f(N) = g(N) + h(N)$, where
    - g(N) is the cost of the path from the initial node to N
    - h(N) is an estimate of the cost of a path from N to a goal node
  - or the cost of a path from N to a goal node
    Then $f(N) = h(N)$ → Greedy best-search
- But there are no limitations on f. Any function of your choice is acceptable.
  But will it help the search algorithm?

6

How to construct f?

- Typically, f(N) estimates:
  - either the cost of a solution path through N
    Then $f(N) = g(N) + h(N)$, where
    - g(N) is the cost of the path from the initial node to N
    - h(N) is an estimate of the cost of a path from N to a goal node
  - or the cost of a path from N to a goal node
    Then $f(N) = h(N)$

Heuristic function

- But there are no limitations on f. Any function of your choice is acceptable.
  But will it help the search algorithm?

7

Heuristic Function

- The heuristic function $h(N) \geq 0$ estimates the cost to go from STATE(N) to a goal state
  Its value is **independent of the current search tree**; it depends only on STATE(N) and the goal test GOAL?
- Example:

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

$h_1(N)$ = number of misplaced numbered tiles = 6
[Why is it an estimate of the distance to the goal?]

8

Other Examples

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

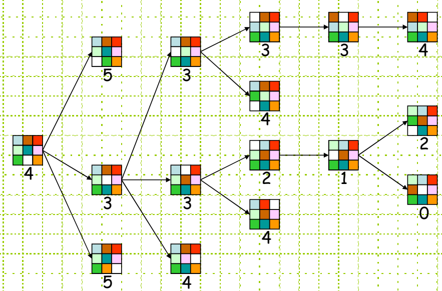
Goal state

- $h_1(N)$ = number of misplaced numbered tiles = 6
- $h_2(N)$ = sum of the (Manhattan) distance of every numbered tile to its goal position
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
- $h_3(N)$ = sum of permutation inversions
  $= n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6$
  = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0
  = 16

9

8-Puzzle

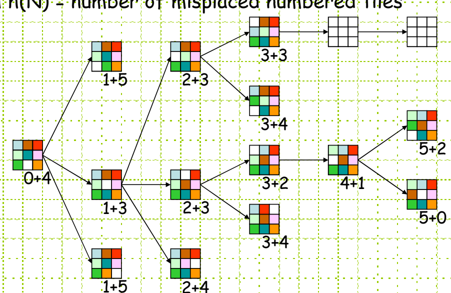
$f(N) = h(N)$ = number of misplaced numbered tiles



The white tile is the empty tile

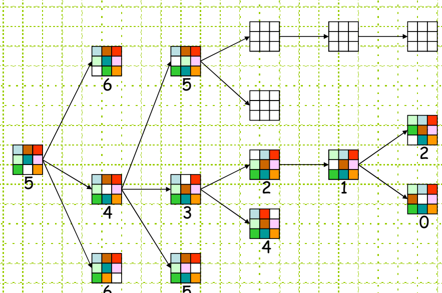
8-Puzzle

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

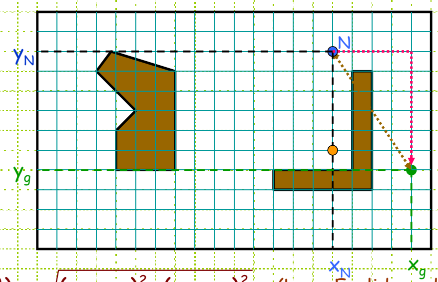


8-Puzzle

$f(N) = h(N) = \Sigma$ distances of numbered tiles to their goals



Robot Navigation

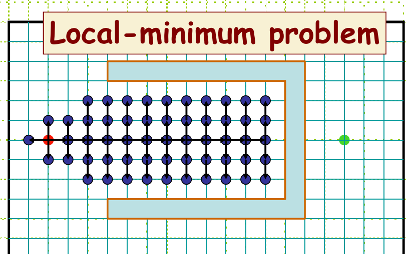


$h_1(N) = \sqrt{(x_N-x_g)^2+(y_N-y_g)^2}$ (L_2 or Euclidean distance)

$h_2(N) = |x_N-x_g| + |y_N-y_g|$ (L_1 or Manhattan distance)

13

Best-First $\not\rightarrow$ Efficiency



f(N) = h(N) = straight distance to the goal

14

How Good is Best-First?

- If the state space is infinite, in general the search is not complete
- If the state space is finite and we do not discard nodes that revisit states, in general the search is not complete
- If the state space is finite and we discard nodes that revisit states, the search is complete, but in general is not optimal

15

Admissible Heuristic

- Let $h^*(N)$ be the cost of the optimal path from N to a goal node
- The heuristic function $h(N)$ is admissible if:

$$0 \leq h(N) \leq h^*(N)$$

- An admissible heuristic function is always optimistic !

16

Admissible Heuristic

- Let $h^*(N)$ be the cost of the optimal path from N to a goal node
- The heuristic function $h(N)$ is admissible if:

$$0 \leq h(N) \leq h^*(N)$$

- An admissible heuristic function is always optimistic !

G is a goal node ➔ $h(G) = 0$

17

8-Puzzle Heuristics

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

- $h_1(N)$ = number of misplaced tiles = 6 is ???

18

8-Puzzle Heuristics

STATE(N)

5		8
4	2	1
7	3	6

Goal state

1	2	3
4	5	6
7	8	

- $h_1(N)$ = number of misplaced tiles = 6
  is admissible
- $h_2(N)$ = sum of the (Manhattan) distances of every tile to its goal position
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
  is ???

19

8-Puzzle Heuristics

STATE(N)

5		8
4	2	1
7	3	6

Goal state

1	2	3
4	5	6
7	8	

- $h_1(N)$ = number of misplaced tiles = 6
  is admissible
- $h_2(N)$ = sum of the (Manhattan) distances of every tile to its goal position
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
  is admissible
- $h_3(N)$ = sum of permutation inversions
  = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16
  is ???

20

8-Puzzle Heuristics

STATE(N)

5		8
4	2	1
7	3	6

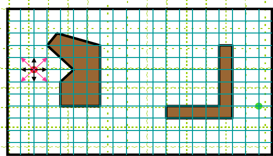
Goal state

1	2	3
4	5	6
7	8	

- $h_1(N)$ = number of misplaced tiles = 6
  is admissible
- $h_2(N)$ = sum of the (Manhattan) distances of every tile to its goal position
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
  is admissible
- $h_3(N)$ = sum of permutation inversions
  = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16
  is not admissible

21

Robot Navigation Heuristics

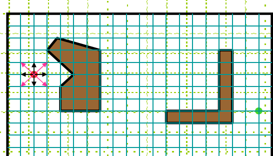


Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$ is admissible

22

Robot Navigation Heuristics

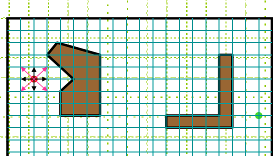


Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_2(N) = |x_N - x_g| + |y_N - y_g|$ is ???

23

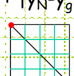
Robot Navigation Heuristics



Cost of one horizontal/vertical step = 1
Cost of one diagonal step = $\sqrt{2}$

$h_2(N) = |x_N - x_g| + |y_N - y_g|$ is admissible if moving along diagonals is not allowed, and not admissible otherwise

$h^*(I) = 4\sqrt{2}$
$h_2(I) = 8$



24

How to create an admissible h?

- An admissible heuristic can usually be seen as the cost of an optimal solution to a relaxed problem (one obtained by removing constraints)
- In robot navigation:
  - The Manhattan distance corresponds to removing the obstacles
  - The Euclidean distance corresponds to removing both the obstacles and the constraint that the robot moves on a grid
- More on this topic later

25

A* Search
(most popular algorithm in AI)

1) $f(N) = g(N) + h(N)$, where:
   - $g(N)$ = cost of best path found so far to N
   - $h(N)$ = **admissible** heuristic function
2) for all arcs: $c(N,N') \geq \varepsilon > 0$
3) SEARCH#2 algorithm is used

→ Best-first search is then called A* search

26

Result #1

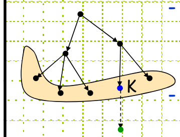
A* is complete and optimal

[This result holds if nodes revisiting states are not discarded]

27

Proof (1/2)

1) If a solution exists, A* terminates and returns a solution

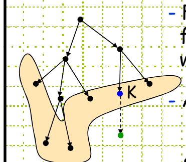


- For each node N on the fringe, $f(N) = g(N)+h(N) \geq g(N) \geq d(N)\times\varepsilon$, where d(N) is the depth of N in the tree
- As long as A* hasn't terminated, a node K on the fringe lies on a solution path

28

Proof (1/2)

1) If a solution exists, A* terminates and returns a solution

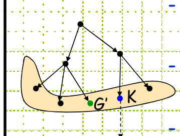


- For each node N on the fringe, $f(N) = g(N)+h(N) \geq g(N) \geq d(N)\times\varepsilon$, where d(N) is the depth of N in the tree
- As long as A* hasn't terminated, a node K on the fringe lies on a solution path
- Since each node expansion increases the length of one path, K will eventually be selected for expansion, unless a solution is found along another path

29

Proof (2/2)

2) Whenever A* chooses to expand a goal node, the path to this node is optimal



- C^*= h*(initial-node)
  [cost of the optimal solution path]
- G': non-optimal goal node in the fringe
  $f(G') = g(G') + h(G') = g(G') > C^*$
- A node K in the fringe lies on an optimal path:
  $f(K) = g(K) + h(K) \leq C^*$
  Optimistic estimate
- So, G' will not be selected for expansion

30

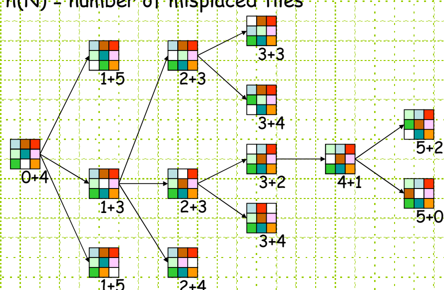
Time Limit Issue

- When a problem has no solution, A* runs for ever if the state space is infinite or states can be revisited an arbitrary number of times. In other cases, it may take a huge amount of time to terminate
- So, in practice, A* is given a time limit. If it has not found a solution within this limit, it stops. Then there is no way to know if the problem has no solution, or if more time was needed to find it
- When AI systems are "small" and solving a single search problem at a time, this is not too much of a concern.
- When AI systems become larger, they solve many search problems concurrently, some with no solution. **What should be the time limit for each of them?**

31

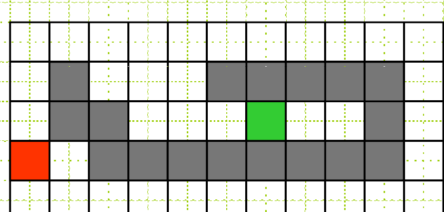
8-Puzzle

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



32

Robot Navigation



33

Robot Navigation

f(N) = h(N), with h(N) = Manhattan distance to the goal
(not A*)

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

34

Robot Navigation

f(N) = h(N), with h(N) = Manhattan distance to the goal
(not A*)

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

35

Robot Navigation

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

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

36

Best-First Search

- An evaluation function f maps each node N of the search tree to a real number $f(N) \geq 0$
- Best-first search sorts the FRINGE in increasing f

37

A* Search

1) $f(N) = g(N) + h(N)$, where:
   - $g(N)$ = cost of best path found so far to N
   - $h(N)$ = **admissible** heuristic function
2) for all arcs: $c(N,N') \geq \varepsilon > 0$
3) SEARCH#2 algorithm is used

→ Best-first search is then called *A* search*

38

Result #1

A* is complete and optimal

[This result holds if nodes revisiting states are not discarded]

39