

Heuristic Search Methods

Methods that use a heuristic function to provide specific knowledge about the problem:

Heuristic Functions
 Hill climbing
 Beam search
 Hill climbing (2)
 Greedy search

© To further improve the quality of the previous methods, we need to include problem-specific knowledge on the problem.

→ How to do this in such a way that the algorithms remain generally applicable ???

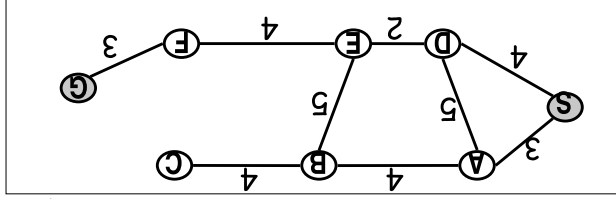
→ HEURISTIC FUNCTIONS:

- ◆ f: States --> Numbers
- ◆ $f(T)$: expresses the quality of the state T
- allow to express problem-specific knowledge, BUT: can be imported in a generic way in the algorithms.

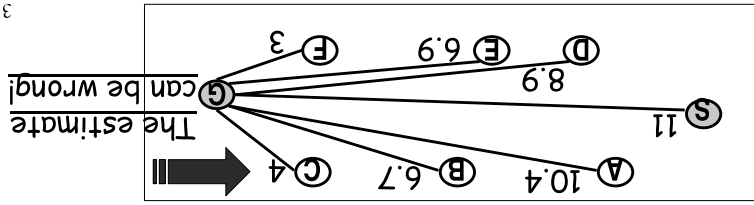
2

Examples (1):

© Imagine the problem of finding a route on a road map and that the NET below is the road map:

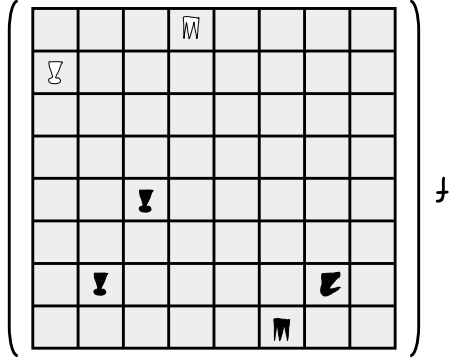


© Define $f(T)$ = the straight-line distance from T to G



3

$$\begin{matrix} \text{v}(\text{M}) + \text{v}(\text{K}) = \\ \text{v}(\text{Q}) + \text{v}(\text{B}) \\ - \text{v}(\text{M}) - \text{v}(\text{Q}) \end{matrix}$$



⊗ $F(T) = (\text{Value count of black pieces}) - (\text{Value count of white pieces})$

Examples (4): Chess!

$$f_3 \begin{pmatrix} 3 & 6 & 7 \\ 8 & & 4 \\ 1 & 5 & 2 \end{pmatrix} = 1 + 4 + 2 + 3 = 10$$

⊗ $f_3(T) =$ the sum of (the horizontal + vertical distance that each tile is away from its final destination):
 → gives a better estimate of distance from the goal node

Examples (3): Manhattan distance

⊗ $f_1(T) =$ the number correctly placed tiles on the board:
 → gives (rough!) estimate of how far we are from goal

$$f_2 \begin{pmatrix} 1 & 3 & 2 \\ 8 & & 4 \\ 5 & 6 & 7 \end{pmatrix} = 4$$

Most often, 'distance to goal' heuristics are more useful!

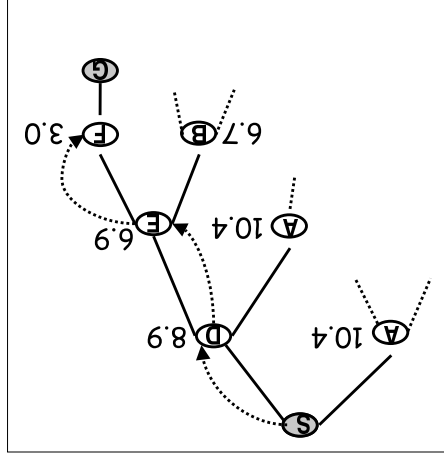
Examples (2): 8-puzzle

Hill climbing

A basic heuristic search method:
depth-first + heuristic

Hill climbing 1

© Example: using the straight-line distance:



- © Perform depth-first, BUT:
- © instead of left-to-right selection,
- © FIRST select the child with the best heuristic value

8

Hill climbing 1 algorithm:

1. QUEUE <-- path only containing the root;
2. WHILE { QUEUE is not empty AND goal is not reached }
 - DO
 - remove the first path from the QUEUE;
 - create new paths (to all children);
 - reject the new paths with loops;
 - sort new paths (HEURISTIC);
 - add the new paths to front of QUEUE;
3. IF goal reached THEN success; ELSE failure;

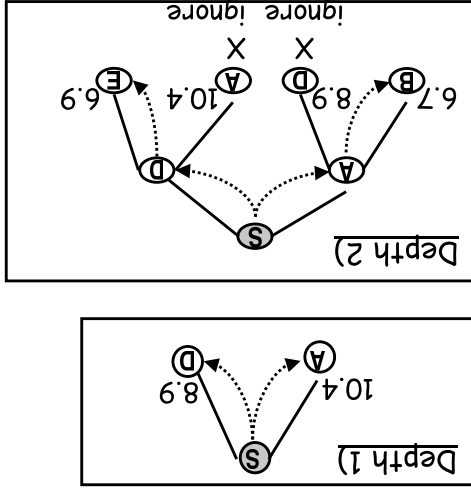
9

Beam search

Narrowing the width of the breadth-first search

Beam search (1):

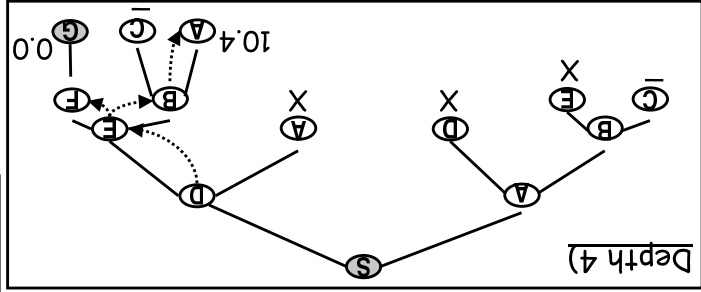
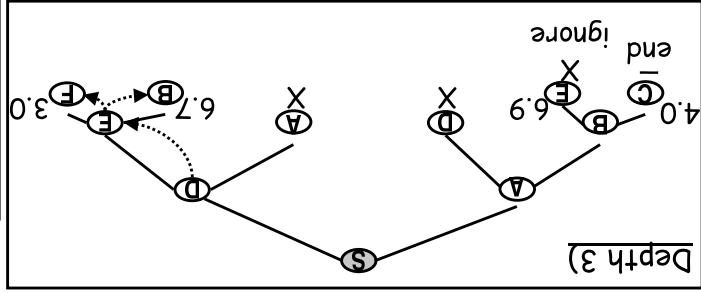
- © Assume a pre-fixed WIDTH (example : 2)
- © Perform breadth-first, BUT:
 - © Only keep the WIDTH best new nodes depending on heuristic at each new level.



11

Beam search (2):

- © Optimization: ignore leafs that are not goal nodes (see C)



12

Steam search algorithm:

© See exercises!

Properties:

- © Completeness: \leftarrow Hill climbing: YES (backtracking), Beam search: NO
- © Speed/Memory: \leftarrow Hill climbing: \blacklozenge same as Depth-first (in worst case)
 \leftarrow Beam search: \blacklozenge QUEUE always has length WIDTH, so memory usage is constant = WIDTH, time is of the order of $WIDTH \times m \times b$ or $WIDTH \times d \times b$ if no solution is found

13

Hill climbing 2

© == Beam search with a width of 1.

© Inspiring Example: climbing a hill in the fog.

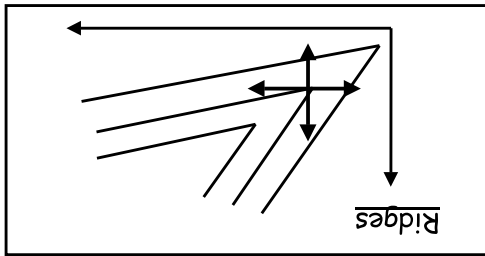
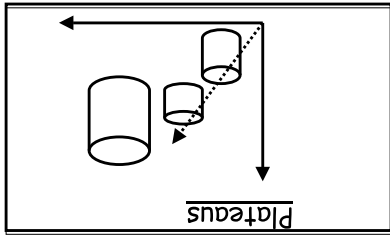
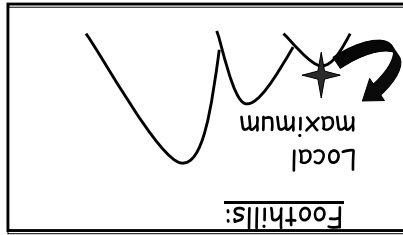
\leftarrow Heuristic function: check the change in altitude in 4 directions: the strongest increase is the direction in which to move next.

© Is identical to Hill climbing_1, except for dropping the backtracking.

© Produces a number of classical problems:

14

Problems with Hill climbing 2:



15

Always expand the heuristically best nodes first.

Greedy search

Another example:
 → MINIMAL COST search:
 ◎ If p is the current path:
 → the next path extends p by adding the node with the smallest cost from the endpoint of p

◎ Hill climbing_2 is an example of local search.
 ◎ In local search, we only keep track of 1 path and use it to compute a new path in the next step.
 → QUEUE is always of the form: ♦ (p)

Local search

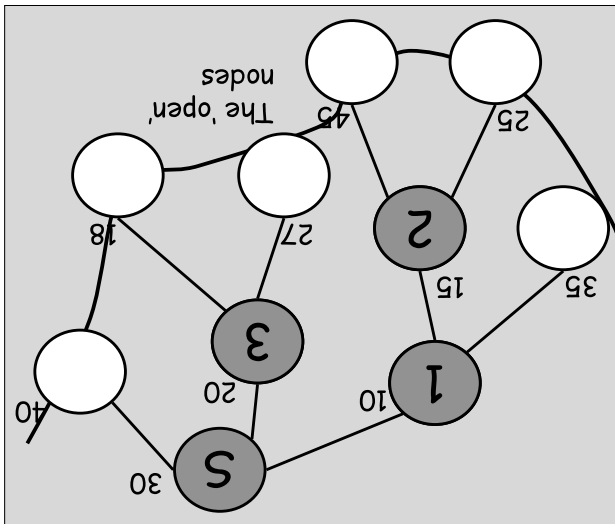
Comments!

◎ Foot hills are local minima: hill climbing_2 can't detect the difference.
 ◎ Plateaus don't allow you to progress in any direction.
 → Foot hills and plateaus require random jumps to be combined with the hill climbing algorithm.
 ◎ Ridges neither: the directions you have fixed in advance all move downwards for this surface.
 → Ridges require new rules, more directly targeted to the goal, to be introduced (new directions to move).

16

Greedy search, or Heuristic best-first search:

© At each step, select the node with the best (in this case: lowest) heuristic value.



19

Greedy search algorithm:

1. QUEUE \leftarrow path only containing the root;
 2. WHILE { QUEUE is not empty AND goal is not reached }
 DO {
 remove the first path from the QUEUE;
 create new paths (to all children);
 reject the new paths with loops;
 add the new paths and sort the entire QUEUE (HEURISTIC)
 }
 3. IF goal reached THEN success; ELSE failure;

20