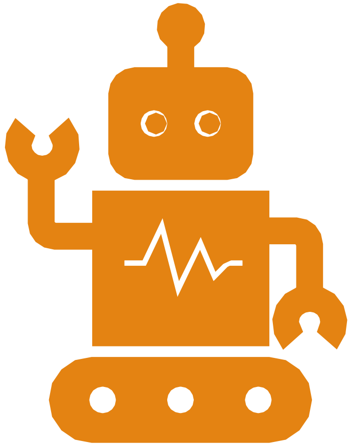


CS341
Artificial Intelligence

Lecture 5

DR. HEBA MOHSEN

Heuristic Search

- On the average they improve the quality of the paths that are explored.
- Using Heuristics, we can hope to get good (though possibly non-optimal) solutions
- There are good general purpose heuristics that are useful in a wide variety of problem domains.
- Special purpose heuristics exploit domain specific knowledge
- Heuristic search uses **Heuristic Function:** This is a function that maps from problem state descriptions to measures of desirability, usually represented as numbers.

Example (1): 8-puzzle

- f1(T) = the number correctly placed tiles on the board:

$$f1 \quad \begin{pmatrix} 1 & 3 & 2 \\ 8 & & 4 \\ 5 & 6 & 7 \end{pmatrix} \quad = 4$$

- f2(T) = number or incorrectly placed tiles on board:
  - **gives (rough!) estimate of how far we are from goal**

$$f2 \quad \begin{pmatrix} 1 & 3 & 2 \\ 8 & & 4 \\ 5 & 6 & 7 \end{pmatrix} \quad = 4$$

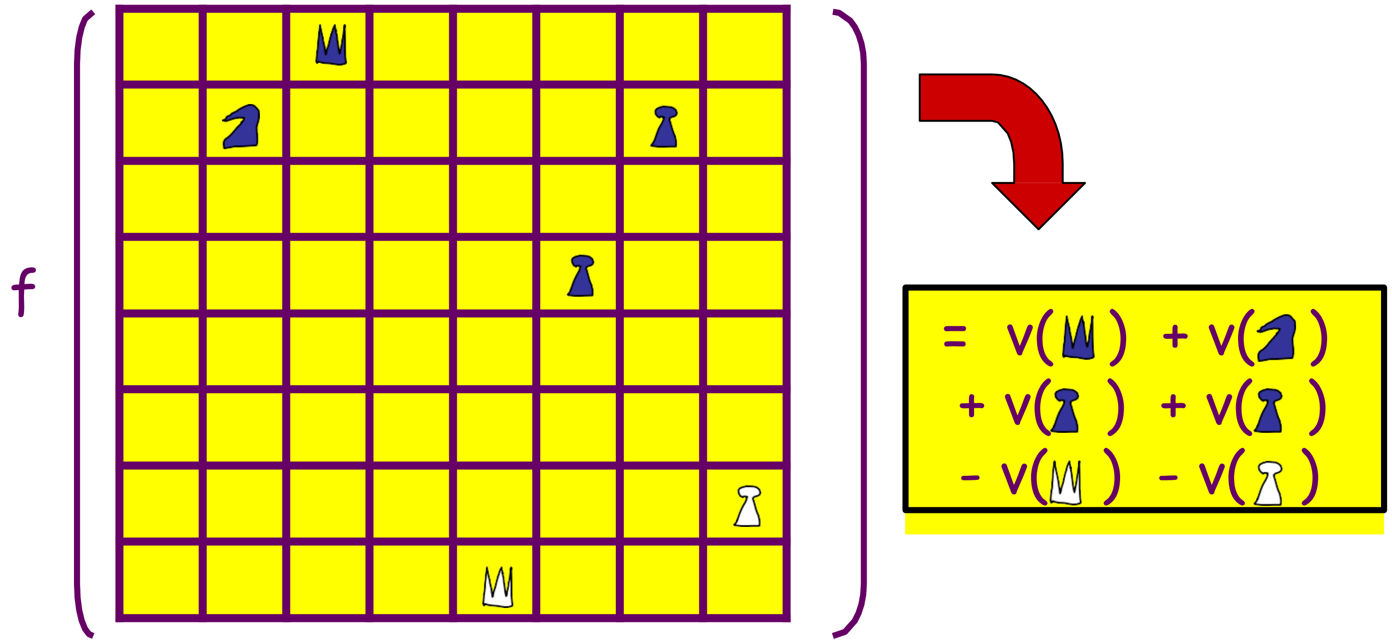
Most often, ‘distance to goal’ heuristics are more useful !

- **f3(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**

$$f3\left(\begin{array}{|c|c|c|}\hline 1 & 3 & 2 \\ \hline 8 & & 4 \\ \hline 5 & 6 & 7 \\ \hline\end{array}\right) = 1 + 1 + 2 + 2 = 6$$

Examples (2): Chess:

- **F(T) = (Value count of black pieces) - (Value count of white pieces)**



Best-first search

Idea: use an evaluation function *f(n)* for each node

- f(n) provides an estimate for the total cost.

→Expand the node n with smallest f(n).

Implementation:

Order the nodes in fringe increasing order of cost.

Special cases:

- greedy best-first search
- A^* search

Example

We start from source "S" and search for goal "I" using given costs and Best First search.

pq (priority queue) initially contains S We remove s from and process unvisited neighbors of S to pq. pq now contains {A, C, B} (C is put before B because C has lesser cost)

We remove A from pq and process unvisited neighbors of A to pq. pq now contains {C, B, E, D}

We remove C from pq and process unvisited neighbors of C to pq. pq now contains {B, H, E, D}

We remove B from pq and process unvisited neighbors of B to pq. pq now contains {H, E, D, F, G}

We remove H from pq. Since our goal "I" is a neighbor of H, we return.