HEURISTIC SEARCH

Heuristics:

Rules for choosing the branches in a state space that are most likely to lead to an acceptable problem solution.

Used when:

- Information has inherent ambiguity
- Computational costs are high
Algorithms for Heuristic Search

Heuristic Search

Hill Climbing  Best first search  A* Algo
Hill Climbing:

If the Node is better, only then you proceed to that Node

Algorithm:

1. Start with current-state (cs) = initial state
2. Until cs = goal-state or there is no change in the cs do:

(a) Get the successor of cs and use the EVALUATION FUNCTION to assign a score to each successor

(b) If one of the successor has a better score than cs then set the new state to be the successor with the best score.
Examples of all climbing algorithms

Tic - Tac - Toe

which one to choose?

Heuristic:
calculate lines and more to that place with most winning lines.
Calculating winning lines

3 winning lines

4 winning lines

3 winning lines

Most winning lines
Example 2

Devise any heuristic to reach goal

<table>
<thead>
<tr>
<th>Start</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>University</td>
</tr>
</tbody>
</table>
Evaluation function:

- Distance between two places
- Adopt the one with minimum distance from goal

Probable route will be

Library ——— Hospital ——— Newsagent ——— University
Suppose $S_2 < S_1$

Then what will happen?

The algorithm will always go to park from hospital instead of going to newsagent. The algo will get stuck here.
Hill Climbing is Good for:

• A limited class of problems where we have an evaluation function that fairly accurately predicts the actual distance to a solution

local maximum/minimum

Hill climbing cannot distinguish between local maximum and global maximum
Local Minima

for any continuous function, gradient search is similar to hill climbing

\[
\frac{d \varnothing(x)}{dx}
\]

The algo gets stuck in the local minima considering in to be the global minimum.
Simulated Annealing

(reading Assignment)

(The method to get rid of the local minima problem.)

It is an optimization method that employ certain techniques to take small steps in the direction indicated by the gradient, but occasionally large steps in the gradient direction / same other directional taken.
Best First Search Method

Algo:

1. Start with agenda = [initial-state]

2. While agenda not empty do:
   
   (a) remove the best node from the agenda

   (b) if it is the goal node then return with success. Otherwise find its successors.

   (c) Assign the successor nodes a score using the evaluation function and add the scored nodes to agenda
A:10

B:5

D:4

G:0

Solution

Breadth - First

F:6

Hill Climbing

Depth First

C:3

E:2

Solution
2. Evaluate A:10; open [C:3,B:5]; closed [A:10]
3. Evaluate C:3; open [B:5,F:6]; closed [C:3,A:10]
4. Evaluate B:5; open [E:2,D:4,F:6]; closed [C:3,B:5,A:10].
5. Evaluate E:2; open [G:0,D:4,F:6];
   closed [E:2,C:3,B:5,A:10]
6. Evaluate G:0; the solution / goal is reached
• If the evaluation function is good, best first search may drastically cut the amount of search requested otherwise.

• The first move may not be the best one.

• If the evaluation function is heavy / very expensive, the benefits may be outweighed by the cost of assigning a score.
Evaluation Function

1. Count the no. of tiles out of place in each state when compared with the goal.

Out of place tiles
2, 8, 1, 6

in place tiles
3, 4, 5, 7
Heuristic Function $f(n) = h(n)$

Drawback:

The heuristic defined does not take into account the distance each tile has to be moved to bring it to the correct place.
Sum the no. of boxes the tile has to be moved to reach the goal

for example:

```
+-----+-----+
|     |     |
| 4   |    1|
+-----+-----+  
```

Total places to be moved = ‘2’

```
+-----+-----+
|     |     |
|     | 1   |
+-----+-----+  
```

No. of places to be moved = ‘3’

```
+-----+-----+
|     |     |
| 1   |     |
+-----+-----+  
```

```
+-----+-----+
|     |     |
| 1   |     |
+-----+-----+  
```

```
+-----+-----+
|     |     |
| 2   |     |
+-----+-----+  
```
Total tiles out of place = 5
Sum of distances out of place = 6

<table>
<thead>
<tr>
<th>Tiles</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

total: 6
Total tiles out of place = 7

Sum of distances

<table>
<thead>
<tr>
<th>Tiles</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

total: 15
Complete Search
Complete Heuristic function (sum of two functions)

\[ f(n) = g(n) + h(n) \]

where \( h(n) = \) Sum of the distances
\( g(n) = \) tiles out of places

Another Heuristic Could be:
\[ g(n) = \text{level of search} \]
\[ h(n) = \text{No. of tiles out of place} \]
**A* Algorithm**

- Problems with Best First Search
  - It reduces the costs to the goal but
    - It is not optimal nor complete
  - Uniform cost
Some Definitions

Admissibility

Heuristics that find shortest path to the goal, whenever it exists are said “ADMISSIBLE”.

e.g.

Breadth - First - Search is ADMISSIBLE
but is too inefficient.
Condition for Admissible Search

\[ h(n) \leq h^*(n) \]

This Condition Ensures Shortest Path

- Estimated Heuristic Cost
- Actual Heuristic Cost
Example:

- Estimated Heuristic
  - Cost = 7 (2+3+2)

- Path for best - fist search
  - (only choose best node)
  - not the path

ABDEF — is it optimal / shortest pat. (N0)
In order to make the search Admissible, Algo. has to be changed.

Need to define heuristic function as

\[ f(n) = g(n) + h(n) \]

where

\[ h(n) \leq h^*(n) \] (under estimation)
\[ g(n) \geq g^*(n) \] (over estimation)
Example for $g(n) \geq g^*(n)$

Goal will never be reached
Some Observations

\[ g = 0 \]

- Choose the Node closest to goal

\[ g = 1 \]

- Path with fewest steps
# Observations

<table>
<thead>
<tr>
<th>h</th>
<th>g</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>h*</td>
<td></td>
<td>Immediate convergence, A* converges to goal (No Search)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Random Search</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Breath - First Search</td>
</tr>
<tr>
<td>&gt;=h*</td>
<td></td>
<td>No Convergence</td>
</tr>
<tr>
<td>&lt;=h*</td>
<td></td>
<td>Admissible Search</td>
</tr>
<tr>
<td>&lt;=g*</td>
<td></td>
<td>No Convergence</td>
</tr>
</tbody>
</table>
Example of \( h^* \)

Path:

\[
\begin{array}{cccccc}
A & B & D & E & F \\
\end{array}
\]

Total cost \( h = 13 \)

Estimated cost = 7

\( 13 > 7 \) (here not admissible)

For A* algorithm

Path:  \ A B C G F \\
as path cost \ B D = 6, \\
and \ C G = 5, \\
hence second choice will be followed
Heuristic Search & Expert Systems

Expert Systems:

Human Expert → Extract Knowledge → Heuristic → Develop Rules

- No certainties are involved
- Knowledge Base
- Manipulates the knowledge to run the system
Problem: How to handle Uncertainty?
Solve: To Heuristics apply level of confidence

e.g.

1. Saving – a/c (adequate) income (adequate)  
   ——— investment (stack)  
   (Level of Confidence (LOC) = 0.7)

2. Headache vomiting fever ——— food poisoning  
   (LOC = 0.8)