Indian Institute of Technology Patna

CS571: ARTIFICIAL INTELLIGENCE

Assignment #3 Hill Climb

Submission Date: 15th April 2024

Baskar Natarajan - 2403res19(IITP001799)

Jyotisman Kar – 2403res35(IITP001751)

SEMESTER-1

MTECH AI & DSC

IIT PATNA.

8-Puzzle Solver Using Hill Climbing Algorithm with Various Heuristic Functions

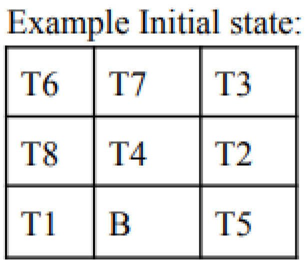
1. Problem Description:

A local search algorithm tries to find the optimal solution by exploring the states in the local region. Hill climbing is a local search technique that constantly looks for a better solution in its neighborhood.

1. Implement the Hill Climbing Search Algorithm for solving the 8-puzzle problem.
2. Check the algorithm for the following heuristics:
   a. h1(n) = number of tiles displaced from their destined position.
   b. h2(n) = sum of the Manhattan distance of each tile from the goal position.

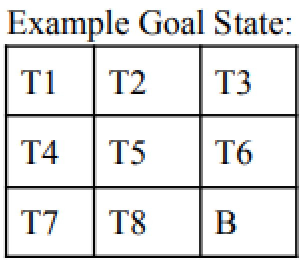
1. Instructions:

1. Take the input and store the information in a matrix. Configuration of the start and goal states are random (try 100 times, stop on 1st success). For example, T1, T2, and T8 are tile numbers, and B is blank space. (Success is when the goal is reachable from initial state).



Example Initial state:

T6	T7	T3
T8	T4	T2
T1	B	T5



Example Goal State:

T1	T2	T3
T4	T5	T6
T7	T8	B

The output should have the following information

for each h1 and h2:

a. On success:

i. Success Message

ii. Start State / Goal State

iii. Total number of states explored

iv. Total number of states to the optimal path using h1 h2

v. Optimal Path

b. On failure:

i. Failure Message

ii. Start State / Goal State

iii. Total number of states explored before termination.

2. Overview

This Python script is designed to solve the classic 8-puzzle problem using the Hill Climbing algorithm. The 8-puzzle problem is a sliding puzzle consisting of a 3x3 grid of numbered tiles, with one tile missing (represented as blank space). The goal is to rearrange the tiles from an initial scrambled state to a goal state by sliding tiles into the empty space.

3. Class: Hill_Search

This class represents the solver for the 8-puzzle problem. It provides methods to manage the puzzle state and perform operations related to solving the puzzle using following 2 Heuristic functions.

1. H1, Total Number of displaced tiles between initial state to goal State.
2. H2, Sum of the Manhattan distance of each tile from the goal position

1. *Function: __**init**__(self, state=None):*

Initializes the puzzle state. If no state is provided, it defaults to a predefined initial state.

1. *Function: __**str**__(self):*

Converts the puzzle state to a string representation for display.

2. *Function: **goal_test**(self):*

Checks if the current puzzle state matches the goal state.

3. *Function: **get_blank_position**(self):*

- Determines the position of the blank tile (empty space) in the puzzle grid.
- It's using generative function to find the blank tile position in given state

4. *Function: **get_neighbors**(self):*

Generates neighboring states by moving the blank tile in all possible directions (Up, Down, Left, Right).

4. Heuristic Functions

1. displaced_heuristics(state)

This heuristic function calculates the number of tiles that are displaced from their correct positions in the current state compared to the goal state.

2. ***manhattan_heuristics(state)***

   This heuristic function calculates the sum of Manhattan distances for each tile from its current position to its goal position. The Manhattan distance is the sum of the horizontal and vertical distances between two points on a grid.

5. Helper Functions

1. ***is_solvable****(state):*

   Checks if a given puzzle state is solvable. It uses the inversion count method to determine solvability.

2. ***create_random_state****():*

   Generates a random initial state for the puzzle by shuffling the tiles of the goal state.

6. Hill Climbing Search Function

1. ***hill_climbing_search****(initial_state, heuristic)*

   This function implements the Hill Climbing algorithm to solve the 8-puzzle problem. It iteratively explores neighboring states and moves to the one with the lowest heuristic value until it reaches a local minimum or finds the goal state.

2. ***Main Function***

   The main function of the script generates a random initial state for the puzzle and attempts to solve it using both heuristic functions. It prints the results of the search, including the initial and goal states, the status (success/failure), and the number of states explored.

3. ***Goal State***

   The goal state of the 8-puzzle problem is predefined as follows:

```
[['T1', 'T2', 'T3'],
 ['T4', 'T5', 'T6'],
 ['T7', 'T8', 'B']]
```

7. Heuristic Functions Comparison

1. ***Displaced Tiles Heuristic (H1)***

Description:

- This heuristic calculates the number of tiles that are currently displaced from their correct positions in the puzzle grid compared to the goal state.

Advantages:

- Simple to compute: It only involves counting the number of misplaced tiles.
- Provides a quick estimate of how far the current state is from the goal state.

Limitations:

- Lack of precision: It does not consider the actual distance of the displaced tiles from their goal positions.
- Not always optimal: It may underestimate the effort required to reach the goal state, especially in cases where tiles are far from their correct positions but have a low displacement count.

2. ***Manhattan Distance Heuristic (H2):***

Description:

- This heuristic calculates the sum of Manhattan distances for each tile from its current position to its goal position in the puzzle grid.

Advantages:

- **More accurate estimation**: It considers the distance each tile needs to move to reach its goal position.
- Provides a better measure of the "closeness" to the goal state.

Limitations:

Computationally intensive: It involves calculating the Manhattan distance for each tile, which requires more computational effort compared to counting displaced tiles.

May overestimate: In some cases, it may overestimate the distance to the goal state if certain tiles are close to their correct positions but require multiple moves to reach them.

3. *Comparison*

Accuracy: The Manhattan Distance Heuristic (H2) is generally more accurate than the Displaced Tiles Heuristic (H1) because it considers the actual distance each tile needs to move.

Computational Cost: H1 is computationally less expensive than H2 since it involves simple counting operations. H2 requires calculating the Manhattan distance for each tile, which is more computationally intensive.

Optimality: H2 tends to provide more optimal solutions compared to H1 because it offers a better estimate of the distance to the goal state. However, H1 may still find acceptable solutions in many cases.

8. Output:

- Prepared the sample success output as well, since random generated states are mostly finding the local minima always.

<u>We have collected multiple Inputs for the metrics.</u>

```
*************************************************************************************

Hill Climbing Search started, Initial State is,

[['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8']]

Solving using h1 (number of tiles displaced from their destined position)

State Number: 1

Number of displaced tiles :  0

Number of displaced tiles :  3

Number of displaced tiles :  3

Number of displaced tiles :  2

Optimal path using h1: [[['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8']], [['T1', 'T2', 'T3'], ['T4',
'T5', 'T6'], ['T7', 'T8', 'B']]]
```

```
Start State:

T1 T2 T3

T4 T5 T6

T7 B T8

Goal State:

T1 T2 T3

T4 T5 T6

T7 T8 B
```

Status is Success,Total number of states explored: 1

```
------------------------------------------------------------------------------------------

Solving using h2 (sum of the Manhattan distance of each tile from the goal position)

State Number: 1

Manhattan distance is: 0

Manhattan distance is: 2

Manhattan distance is: 2

Manhattan distance is: 1

Optimal path using h2: [[['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8']], [['T1', 'T2', 'T3'], ['T4',
'T5', 'T6'], ['T7', 'T8', 'B']]]

Start State:

T1 T2 T3

T4 T5 T6

T7 B T8

Goal State:

T1 T2 T3

T4 T5 T6

T7 T8 B
```

```
Status is Success, Total number of states explored: 1

*****************************************************************************************

Hill Climbing Search started, Initial State is,

[['T1', 'T2', 'T3'], ['T4', 'B', 'T6'], ['T7', 'T5', 'T8']]

Solving using h1 (number of tiles displaced from their destined position)

State Number: 1

Number of displaced tiles :  4

Number of displaced tiles :  4

Number of displaced tiles :  2

Number of displaced tiles :  4

Number of displaced tiles :  3

Number of displaced tiles :  3

Number of displaced tiles :  3

State Number: 2

Number of displaced tiles :  0

Number of displaced tiles :  3

Number of displaced tiles :  3

Number of displaced tiles :  2

Optimal path using h1: [[['T1', 'T2', 'T3'], ['T4', 'B', 'T6'], ['T7', 'T5', 'T8']], [['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8']], [['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'T8', 'B']]]

Start State:

T1 T2 T3

T4 B T6

T7 T5 T8

Goal State:
```

```
T1 T2 T3

T4 T5 T6

T7 T8 B
```

Status is Success,Total number of states explored: 2

```
-----------------------------------------------------------------------------------------

Solving using h2 (sum of the Manhattan distance of each tile from the goal position)

State Number: 1

Manhattan distance is: 3

Manhattan distance is: 3

Manhattan distance is: 1

Manhattan distance is: 3

Manhattan distance is: 2

Manhattan distance is: 2

Manhattan distance is: 2

State Number: 2

Manhattan distance is: 0

Manhattan distance is: 2

Manhattan distance is: 2

Manhattan distance is: 1

Optimal path using h2: [[['T1', 'T2', 'T3'], ['T4', 'B', 'T6'], ['T7', 'T5', 'T8']], [['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'B', 'T8']], [['T1', 'T2', 'T3'], ['T4', 'T5', 'T6'], ['T7', 'T8', 'B']]]

Start State:

T1 T2 T3

T4 B T6

T7 T5 T8

Goal State:
```

```
T1 T2 T3

T4 T5 T6

T7 T8 B
```

Status is Success, Total number of states explored: 2

```
*****************************************************************************************

Hill Climbing Search started, Initial State is,

[['B', 'T2', 'T5'], ['T4', 'T3', 'T8'], ['T7', 'T1', 'T6']]

Solving using h1 (number of tiles displaced from their destined position)

State Number: 1

Number of displaced tiles :  7

Number of displaced tiles :  7

Number of displaced tiles :  6

Number of displaced tiles :  6

Number of displaced tiles :  6

Local Minima reached with heuristic Value 6 at state:

B T2 T5

T4 T3 T8

T7 T1 T6

Optimal path using h1: [[['B', 'T2', 'T5'], ['T4', 'T3', 'T8'], ['T7', 'T1', 'T6']]]

Start State:

B T2 T5

T4 T3 T8

T7 T1 T6

Goal State:

T1 T2 T3
```

T4 T5 T6

T7 T8 B

Total number of states explored before termination: 1

Status is Failed, Termination Reason: Local Minima Reached

Solving using h2 (sum of the Manhattan distance of each tile from the goal position)

State Number: 1

Manhattan distance is: 11

Manhattan distance is: 11

Manhattan distance is: 10

Manhattan distance is: 10

Manhattan distance is: 10

Local Minima reached with heuristic Value 10 at state:

B T2 T5

T4 T3 T8

T7 T1 T6

Optimal path using h2: [[['B', 'T2', 'T5'], ['T4', 'T3', 'T8'], ['T7', 'T1', 'T6']]]

Start State:

B T2 T5

T4 T3 T8

T7 T1 T6

Goal State:

T1 T2 T3

T4 T5 T6

T7 T8 B

Total number of states explored before termination: 1

Status is Failed,Termination Reason: Local Minima Reached