



Ahsanullah University of Science and Technology (AUST)

Department of Computer Science and Engineering

Course No: CSE4108

Course Title: Artificial Intelligence Lab

Term Assignment Number: 02

Name: Devopriya Tirtho

ID: 16.02.04.033

Section: A

Lab Group: A2

Term Assignment 2: First-Choice Hill Climbing.

Introduction: We are living in an age of modern science where '**Artificial Intelligence**' is ruling the whole world. Artificial Intelligence means to make intelligent a machine artificially. The question is how we can make a machine intelligent. For answering that question, the researchers have found various ways. It includes '**Machine Learning**', '**Deep Learning**' etc. By training the machine we can make the machine responsive to events which is not included to machine's memory. The machine gains the ability by training itself with the given set of data.

'Genetic Algorithm' is a part of Artificial Intelligence. It includes '**Hill Climbing**'.

Genetic Algorithm: From the biological term '**Genetics**' we can understand the main theory of '**Genetic Algorithm**' easily. In genetics, we can see that there remains a population which includes some male and female. The male and female reproduce child to carry out the generation. In genetic algorithm, we can find a similar scenario. Where there is a generation containing some result, by crossover we can find a new generation. Before going to that portion, we can define genetic algorithm easily by describing '**Hill Climbing**'.

Hill Climbing: Hill Climbing is the simplest implementation of a Genetic Algorithm. It completely gets rid of the concepts like population and crossover, instead focusing on the ease of implementation. It has faster iterations compared to more traditional genetic algorithms, but in return it is less thorough. In hill climbing, our goal is to reach the highest peak which is called '**Global Maximum**'. Whereas, it is not always reachable, so we will go for a sub-optimal solution named '**Local Maximum**'.

Process of The Execution of Hill Climbing:

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.

Facts about Hill Climbing:

- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Applications of Hill Climbing:

- Integrated Circuit Design
- Job-Shop Scheduling
- Real-World Problems

State Space for Hill Climbing:

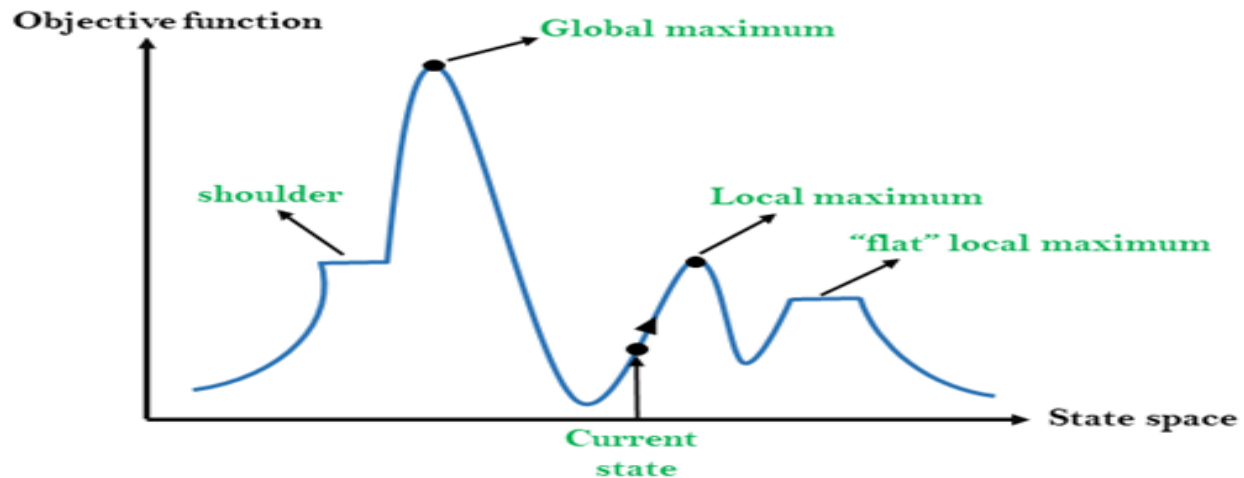


Figure: State Space of Hill Climbing

Global Maximum: This is the peak of the state space. The highest point means this point contains the optimal solution of the problem. The neighboring points of the point must be less than the value of the global maximum.

Local Maximum: The local maximum point is the highest peak in a shorter space of the state space. The neighboring points held less value the local maximum point. By moving sidewise, we can find only lesser values compare to the local maximum point.

Shoulder: It is the point where it seems like there is no much or less value of the particular point. But moving forward or backward for sometimes, we can find lower or higher values.

‘Flat’ Local Maximum/ Plateau Top: It is a flat space in the landscape where all the neighbor states of current states have the same value.

Current State: It is the point of the space where the agent currently stays. For finding a solution, we need to move from the current position. From the current state, by moving in the state space, we can face some problems. These problems are explained below:

- We can fall on a ridge, where both ways have higher values.
- We can be stuck at the local maximum.
- We can be stuck at the plateau
- We can be stuck at the shoulder

To get rid of these problems, we have to adapt some moves. These are:

- Sideways Move
- Random Restart Hill Climbing
- Stochastic Hill Climbing
- First- Choice Hill Climbing

First-Choice Hill Climbing: When we stuck at a position where we need the feel to move but cannot decide where to go, then we can generate some random steps. From the random states, we only choose the first state which has the greater value than the current state.

Working Process of First-Choice Hill Climbing:

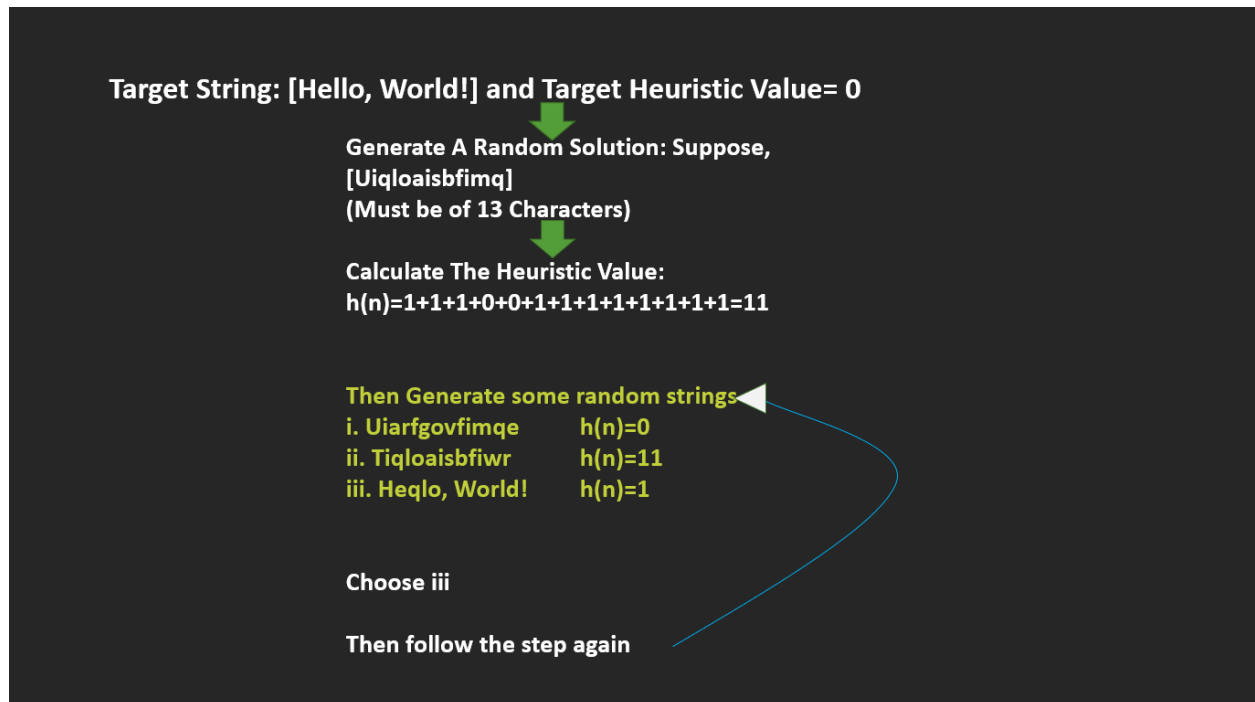
- Firstly, we need to generate a random solution to start in the state space
- Then we have to check it with our goal, if it is the goal or global maximum.
- If not, the we have to generate some random steps.
- Then we have to evaluate those steps.
- After evaluation, we have to compare the evaluated value with the value of the current state.
- Whenever, we find a higher value than the current state, we choose the state and follow the same steps again.

Workflow with an Example:

We have to find **“Hello, World!”** from alphabets.

Process:

- We set a target that we have to find **“Hello, World!”**
- Then we generate a sentence of 13 characters with the help of random function to start the process of finding
- Firstly, we get a string from the random generation.
- We have designed a function which reads and compares character by character of the targeted and generated string.
- From the mismatched value, we evaluate a score for the generated string.
- This score is the heuristic value for the mismatched characters.
- The minimum mismatched value should be zero (0).
- If the evaluation value is greater than zero, that means till now we cannot go to the global maximum or find the optimal solution.
- Then we generate some random strings again.
- From the random strings, we pass each string to calculate the heuristic value of the string.
- From the heuristics, the lesser than the current state's heuristic value is chosen which is called the 'First-Choice Hill Climbing'.
- We follow these steps again and again unless we find the heuristic as zero (0).

Workflow:**Input:**

Input is given internally to the code as a string stored in a list: **“Hello, World!”**

Output:

The output will show the heuristic value (value is calculated by the value of Unicode values) and the newly discovered string.

Solution Code: The code is written in python language.

```
import random
import string

def generate_random_solution(length=13):
    return [random.choice(string.printable) for _ in range(length)]

def evaluate(solution):
    target = list("Hello, World!")
    diff = 0
    for i in range(len(target)):
        s = solution[i]
        t = target[i]
        diff += abs(ord(s) - ord(t))
    return diff

best = generate_random_solution()
heuristic_value = evaluate(best)

while True:

    print('Heuristic Value: ', heuristic_value, 'and the newly found solution',
          "".join(best))

    if heuristic_value == 0:
        break

    new_solution = generate_random_solution()

    score = evaluate(new_solution)

    if evaluate(new_solution) < heuristic_value:
```



```

        best = new_solution
    heuristic_value = score

```

Output:

```

\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09
Heuristic Value: 194 and the newly found solution g`z}{%H
\ir`09

```

Here, the heuristic value (based on Unicode) is 194 and looking for good solution.

```

Console 1/A x
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution 0hpxg!
%Mbhm[&
Heuristic Value: 115 and the newly found solution

```

(Python 3.7.6) Line 28, Col 5 UTF-8 CRLF RW Mem 78%

3:12 AM

The, **heuristic value is decreasing.**

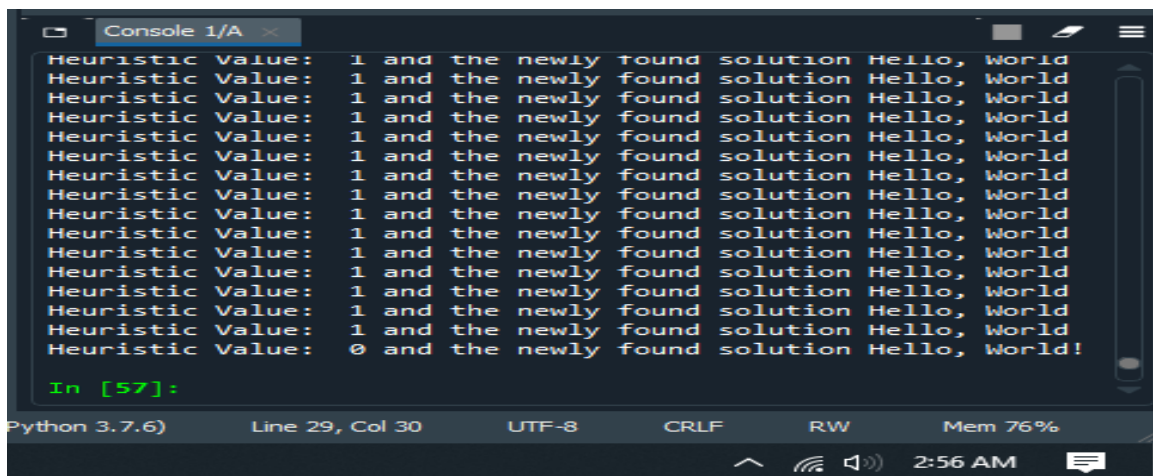
If we introduce a new function:

```
def mutation(solution):  
    index = random.randint(0, len(solution) - 1)  
    solution[index] = random.choice(string.printable)
```

and call the new function with passing the newly generated string:

```
first_choice(new_solution)
```

Then, we can converge to a solution with heuristic 0. Then the output will be like this.



```
Console 1/A  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 1 and the newly found solution Hello, World  
Heuristic Value: 0 and the newly found solution Hello, World!  
In [57]:  
Python 3.7.6) Line 29, Col 30 UTF-8 CRLF RW Mem 76% 2:56 AM
```