ASSIGNMENT 3

- ASHISH KUMAR
- 2K18/SE/041

Ques: Write a short note on the following:

1. Hill Climbing

2. Simulated Annealing

3. Intelligent Agents and Intelligent Programs

Ans:

Hill Climbing

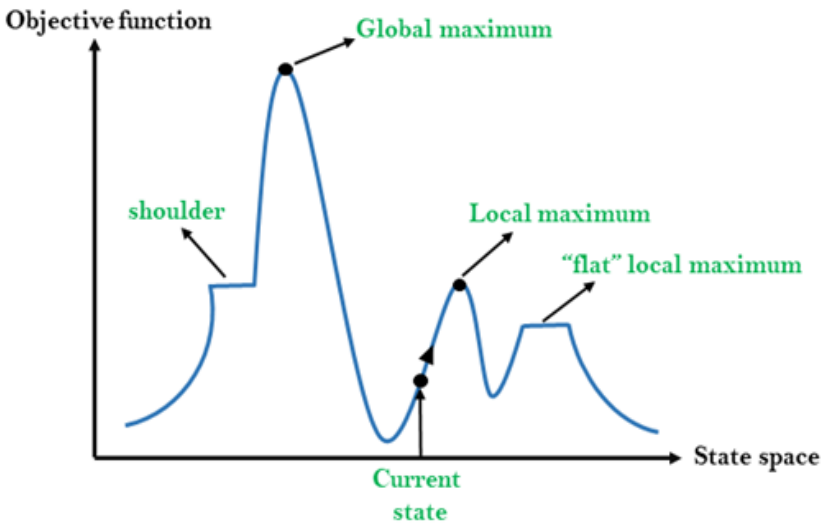
Introduction

- 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.
- 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.

State-space Diagram for Hill climbing

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.

On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis. If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum. If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.



Different regions in the state space landscape:

1. **Local Maximum:** Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.
2. **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
3. **Current state:** It is a state in a landscape diagram where an agent is currently present.
4. **Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value.
5. **Shoulder:** It is a plateau region which has an uphill edge.

Types of Hill Climbing Algorithm:

- Simple hill Climbing:
- Steepest-Ascent hill-climbing:
- Stochastic hill Climbing:

1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill climbing algorithm. It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**.** It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:

- Less time consuming
- Less optimal solution and the solution is not guaranteed

2. Steepest-Ascent hill climbing:

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

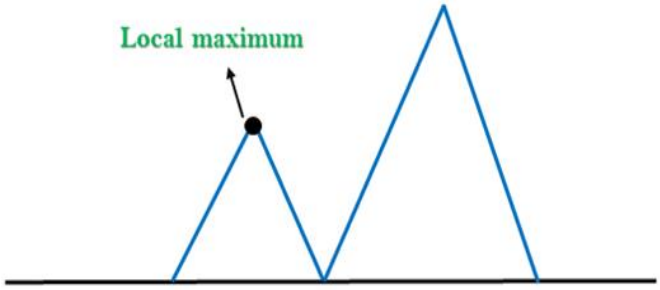
3. Stochastic hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

Problems in Hill Climbing Algorithm:

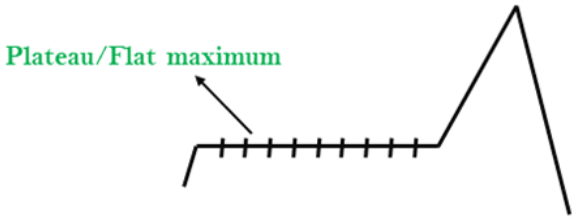
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



2. <u>Plateau:</u> A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



3. <u>Ridges:</u> A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Solution: With the use of bidirectional search, or by moving in different directions, we can improve this problem.

Features of Hill Climbing

Following are some main features of Hill Climbing Algorithm:

- **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
- **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
- **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

Simulated Annealing

A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum. And if algorithm applies a random walk, by moving a successor, then it may complete but not efficient. **Simulated Annealing** is an algorithm which yields both efficiency and completeness.

Simulated Annealing (SA) is motivated by an analogy to annealing in solids. The idea of SA comes from a paper published by Metropolis etc al in 1953 [Metropolis, 1953). The algorithm in this paper simulated the cooling of material in a heat bath. This is a process known as annealing.

In mechanical term **Annealing** is a process of hardening a metal or glass to a high temperature then cooling gradually, so this allows the metal to reach a low-energy crystalline state. The same process is used in simulated annealing in which the algorithm picks a random move, instead of picking the best move. If the random move improves the state, then it follows the same path. Otherwise, the algorithm follows the path which has a probability of less than 1 or it moves downhill and chooses another path.

Implementation of Simulated Annealing

The following algorithm is taken from (Russell, 1995)

Function SIMULATED-ANNEALING (*Problem*, *Schedule*) **returns** a solution state

Inputs : *Problem*, a problem
Schedule, a mapping from time to temperature

Local Variables : *Current*, a node
Next, a node
T, a "temperature" controlling the probability of downward steps

Current = MAKE-NODE (INITIAL-STATE [*Problem*])
For t = 1 **to** ∞ **do**
T = *Schedule*[t]
If T = 0 **then return** C*urrent*
Next = a randomly selected successor of *Current*
ΛE = VALUE[*Next*] – VALUE[*Current*]
if ΛE > 0 **then** *Current* = *Next*
else *Current* = *Next* only with probability exp(-ΛE/T)

We can make several observations about the algorithm.

One of the parameters to the algorithm is the ***schedule***. This is the cooling schedule
This algorithm assumes that the annealing process will continue until the temperature reaches zero. Some implementations keep decreasing the temperature until some other condition is met. For example, no change in the best state for a certain period of time.
That is, a particular phase of the search normally continues at a certain temperature until some sort of equilibrium is reached. This might be a certain number of iterations or it could be until there has been no change in state for a certain number of iterations.

Intelligent Agents and Intelligent Programs

➢ Intelligent Agents

In artificial intelligence, an **intelligent agent** (**IA**) refers to an autonomous entity which acts, directing its activity towards achieving goals (i.e. it is an agent), upon an environment using observation through sensors and consequent actuators (i.e. it is intelligent). Intelligent agents may also learn or use knowledge to achieve their goals.

Intelligent agents are also closely related to software agents (an autonomous computer program that carries out tasks on behalf of users). In computer science, an *intelligent agent* is a software agent that has some intelligence, for example, autonomous programs used for operator assistance or data mining (sometimes referred to as *bots*) is also called "intelligent agents".

Types of intelligent agents

Types of intelligent agents are defined by their range of capabilities and degree of intelligence:

- Reflex agents: These agents function in a current state, ignoring past history. Responses are based on the event-condition-action rule (ECA rule) where a user initiates an event and the agent refers to a list of pre-set rules and pre-programmed outcomes.
- Model-based agents: These agents choose an action in the same way as a reflex agent, but they have a more comprehensive view of the environment. A model of the world is programmed into the internal system that incorporates the agent's history.
- Goal-based agents: These agents expand upon the information model-based agents store by also including goal information, or information about desirable situations.
- Utility-based agents: These agents are similar to goal-based agents but provide an extra utility measurement which rates each possible scenario on its desired result and chooses the action that maximizes the outcome. Rating criteria examples could be the probability of success or the resources required.
- Learning agents: These agents have the ability to gradually improve and become more knowledgeable about an environment over time through an additional learning element. The learning element will use feedback to determine how performance elements should be changed to improve gradually.

Examples of intelligent agents

AI assistants, like Alexa and Siri, are examples of intelligent agents as they use sensors to perceive a request made by the user and the automatically collect data from the internet without the user's help. They can be used to gather information about its perceived environment such as weather and time.

Autonomous vehicles could also be considered intelligent agents as they use sensors, GPS and cameras to make reactive decisions based on the environment to maneuver through traffic.

- **Intelligent Program**

Artificial intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence.

Intelligent Program is a program that is capable of intelligent behavior**.**

Artificial Intelligence (AI) Software is a computer program which mimics human behavior by learning various data patterns and insights.

Today, artificial intelligent programs are at work in applications such as your Smartphone assistant**,** ATMs that read checks**,** voice and image recognition software on your favorite social network**.**

FEW EXAMPLES OF ARTIFICIAL INTELLIGENT PROGRAMS:

1. GOOGLE ASSISTANT:

It is one of the most prominent examples of artificial intelligent program on the market. It can search the Internet, program events and alarms, correct hardware settings on the user's device, and show information from the user's Google account. The Google Assistant can connect in a joint discussion, using Google's normal language processing algorithm. This artificial intelligent program allows Google Assistant to set reminders, gather news, check the weather, generate a shopping list, and even plays games.

2: CORTANA:

Microsoft is depicted as a private digital assistant that helps users systematize their day-to-day actions, along with providing standard web searches for information. Cortana's artificial intelligent program replaces the search function in the Windows Phone and appears as a Live Tile on user's devices.

3: SISENSE:

It is the only industry-focused intelligent program that makes it simple for users to arrange, analyze, and think about complex data. Sisense provides end-to-end explanations and tutorials for tackling the rising numbers of different unstructured and structured data sets from multiple sources that comes out-of-the-box with the capability to chew terabytes of data along with-holding thousands of users – all on a single service server. With Sisense, users can easily create and manage complex data models using a seamless drag-and-join interface that helps to make business decisions simple.