**Lab 05**

**Search Problems in Artificial Intelligence**

**Local Search Algorithms**

In many optimization problems the path to a goal state is irrelevant. The goal state itself is the solution. They start from a prospective solution and then move to a neighboring solution. They can return a valid solution even if it is interrupted at any time before they end.

There are mainly Three types of local search Algorithms

* Hill Climbing Search
* Local Beam **Search**
* **Simulated Annealing**

### 1) **Hill Climbing Search**

Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence.

So, given a large set of inputs and a good heuristic function, the algorithm tries to find the best possible solution to the problem in the most reasonable time. This solution may not be the absolute best (global optimal maximum), but it is sufficiently good considering the time allotted.

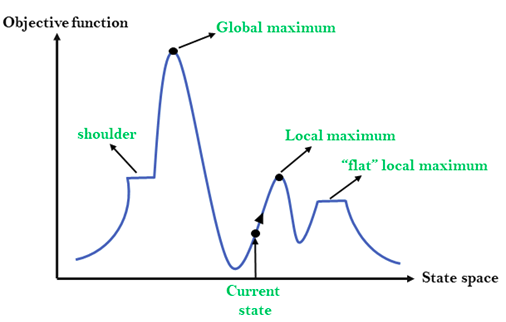
* Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
* Hill climbing algorithm is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

## **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.

**State Diagram for hill Climbing**



Different regions in the State Space Diagram:

1. Local maximum: It is a state which is better than its neighboring state however there exists a state which is better than it(global maximum). This state is better because here the value of the objective function is higher than its neighbors.
2. Global maximum: It is the best possible state in the state space diagram. This because at this state, objective function has highest value.
3. Plateau/flat local maximum: It is a flat region of state space where neighboring states have the same value.
4. Ridge: It is region which is higher than its neighbors but itself has a slope. It is a special kind of local maximum.
5. Current state: The region of state space diagram where we are currently present during the search.
6. Shoulder: It is a plateau that has an uphill edge.

Algorithm for Simple Hill Climbing

* Step 1: Evaluate the initial state, if it is goal state then return success and Stop.
* Step 2: Loop Until a solution is found or there is no new operator left to apply.
* Step 3: Select and apply an operator to the current state.
* Step 4: Check new state:
  1. If it is goal state, then return success and quit.
  2. else if it is better than the current state then assigns new state as a current state.
  3. else if not better than the current state, then return to step 2.
* Step 5: Exit.

**Types of Hill Climbing Algorithm:**

* Simple **hill Climbing**:
  + Above mentioned Algorithm.
* 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.
* 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.

1. Local Beam **Search**

**Introduction:**

A heuristic technique is a set of criteria for determining which of multiple options will be the most effective in achieving a particular goal. This strategy increases the efficiency of a search process by surrendering claims of systematic and completeness of the best. We can hope to achieve a good solution to difficult problems (such as the traveling salesman problem) in less than exponent time if we use appropriate heuristics.

**Beam Search:**

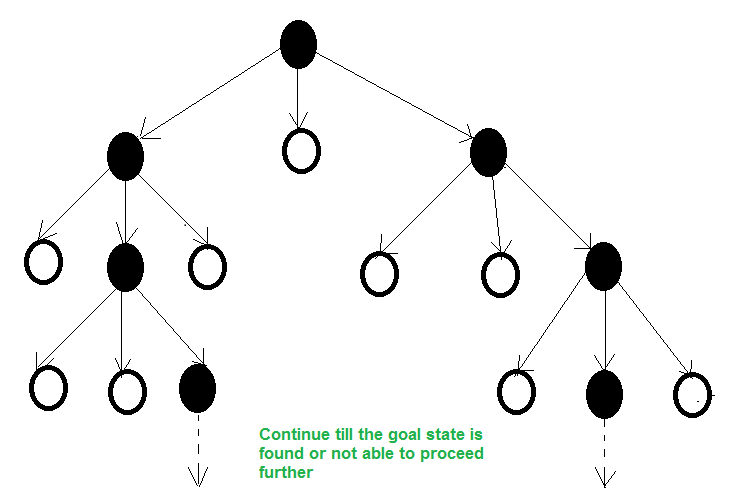
A heuristic search algorithm that examines a graph by extending the most promising node in a limited set is known as beam search. Beam search is a heuristic search technique that always expands the W number of the best nodes at each level. It progresses level by level and moves downwards only from the best W nodes at each level. Beam Search uses breadth-first search to build its search tree. Beam Search constructs its search tree using breadth-first search. It generates all the successors of the current level’s state at each level of the tree. However, at each level, it only evaluates a W number of states, other nodes are not considered.

The heuristic cost associated with the node is used to choose the best nodes. The width of the beam search is denoted by W. If B is the branching factor, at every depth, there will always be W × B nodes under consideration, but only W will be chosen. More states are trimmed when the beam width is reduced.

When W = 1, the search becomes a hill-climbing search in which the best node is always chosen from the successor nodes. No states are pruned if the beam width is unlimited, and the beam search is identified as a breadth-first search.

The beamwidth bounds the amount of memory needed to complete the search, but it comes at the cost of completeness and optimality (possibly that it will not find the best solution). The reason for this danger is that the desired state could have been pruned.

**Example:** The search tree generated using this algorithm with W = 2 & B = 3 is given below:



The black nodes are selected based on their heuristic values for further expansion.

The algorithm for beam search is given as:

**Input:** Start & Goal States.  
**Local Variables:** OPEN, NODE, SUCCS, W\_OPEN, FOUND  
**Output:** Yes or No (yes if the search is successfully done)

**Algorithm**:

Start

Take the inputs

NODE = Root\_Node & Found = False

If : Node is the Goal Node,

Then Found = True,

Else :

Find SUCCs of NODE if any, with its estimated cost&

store it in OPEN List

While (Found == false & not able to proceed further), do

{

Sort OPEN List

Select top W elements from OPEN list and put it in

W\_OPEN list and empty the OPEN list.

for each NODE from W\_OPEN list

{

if NODE = Goal,

then FOUND = true

else

Find SUCCs of NODE. If any with its estimated

cost & Store it in OPEN list

}

}

If FOUND = True,

then return Yes

else

return No

Stop

### 3) Simulated Annealing

Annealing is the process of heating and cooling a metal to change its internal structure for modifying its physical properties. When the metal cools, its new structure is seized, and the metal retains its newly obtained properties. In simulated annealing process, the temperature is kept variable.

We initially set the temperature high and then allow it to ‘cool' slowly as the algorithm proceeds. When the temperature is high, the algorithm is allowed to accept worse solutions with high frequency.

**Start**

* **Initialize k = 0; L = integer number of variables;**
* **From i → j, search the performance difference Δ.**
* **If Δ <= 0 then accept else if exp(-Δ/T(k)) > random(0,1) then accept;**
* **Repeat steps 1 and 2 for L(k) steps.**
* **k = k + 1;**

Repeat steps 1 through 4 till the criteria is met.

**End**

Simulated Annealing is an algorithm which yields both efficiency and completeness.

**Lab Task**

**8-Queen Problem using Hill Climbing**

The 8-queens problems asks us to place 8 queens on a chessboard so that no two can capture one another; that is, no two are on the same row, column, or diagonal. One solution (source: Wikipedia) is shown below:

Chart, treemap chart

Description automatically generated

The first observation is that, in any solution, no two queens can occupy the same column, and consequently no column can be empty. At the start we can therefore assign a column to each queen and reduce the problem to the simpler task of finding an appropriate row.

**Statement of the Problem:**

Consider an 8 × 8 chessboard. Place In queens on the board such that no two queens are attacking each other. The queen is the most powerful piece in chess and can attack from any distance horizontally, vertically, or diagonally. Thus, a solution must place the queens such that no two queens are in the same row, the same column, or along the same diagonal.

**To do:**

* Create a 2D array to represent chessboard and initialize it with zero.
* Write a function to randomly place Queen in the array (represent the position index with 1)
* Write a function to calculate the objective function i.e., Number of queens attacking each other.
* Write a function to search for the solution using Hill climbing (Algorithm given below).

**Hill Climbing**

• Start with a random configuration.

• Calculate h(board) as the number of pairs of attacking queens.

• Consider all 56 (for N = 8) possible next boards formed by moving any one queen to any other row in its column.

• Evaluate h(board) for each of these possibilities.

• If none is better than the current h, quit (local min). Else, take the best as the new current board and repeat. This will fail 86% of the time for N = 8.

Submit the Notebook file named after your RollNo.