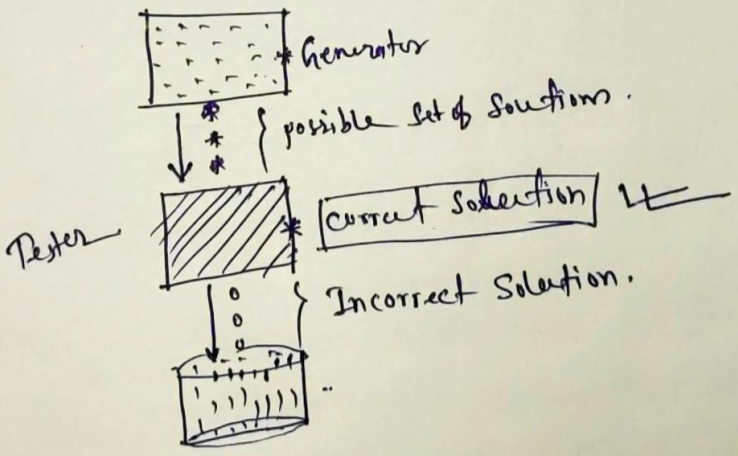
Generate & Test

Generate & Test search algorithm is a very simple Algorithm which guarantees to find a solution if there exists a Solution & the process applied systematically.

Algorithm:-

1. Define current state as initial state
2. Apply any possible operation on the current state & generate a possible solution.
3. Compare newly generated solution with the goal state.
4. If the goal is acheived or no new state can be created, quit. Otherwise, return to the step 2.



Generating complete Solution & generating random solutions are two extremes there exists another approach that lies in between. The approach is that the search process proceeds systematically but some path that unlikely to lead the solution are ignored. This evaluation is performed by a heuristic function. Deapth-first-Search tree with backtracking can be used to implement systematic generate-&-test procedure

Hill Climbing

Hill climbing is heuristic search used for mathematical optimization problem in the field of Artificial Intelligence.

→ Given a large set of inputs & a good heuristic function, it tries to find out a sufficiently good solution to the problem. This solution may not be the global optimal maximum.

→ Hill climbing solves the problem where we need to maximize or minimize a given real function by choosing values from a given input.

→ It's heuristic, which implies that the search algorithm may not find the optimal solution.

→ A heuristic function can rank all the possible solutions at any branching step in search algorithm based on the available information.

Variants of Hill Climbing

⇒ It's a variant of generate & Test algorithm we have seen earlier.

⇒ Uses the Greedy Approach. At any point in the state space the search moves in that direction only which optimizes the cost of function with the hope of finding the optimal solution at the end.

Types of Hill Climbing:

1) **Simple Hill Climbing:** It examines the neighboring nodes one by one & select the first neighboring node which optimizes the current cost as next node.

Algorithm for Simple Hill climbing:

Step1: Evaluate the initial state. ~~If it is the initial state~~
if it is the goal state
then stop & return
else make initial state as the current state.

Step 2: Loop until the solution state is found or there are no new operator present which can be applied to current state.

a) Select a state that has not been yet applied to the current state.
& apply it to to generate a new state.

b) perform these to evaluate new state

i. If the current state is a goal state then stop & return success.

ii. If it is better than the current state then make it current & proceed further

iii. If it is not better than the current state.
then continue in the loop until a solution is found.

Step 3: Exit.

2. Steepest-Ascent Hill climbing: It first examines all the neighbouring nodes and then selects the node closest to the solution state as next node.

Algorithm

Step1: Evaluate the initial state.
If it is goal state then exit
else make the current state as initial state

Step2: Repeat these steps until a solution is found or current state does not change.

i. Let target be a state such that any successor of the current state will be better than it.

ii. for each operator that applies to the current state

a. apply the new operator & create a new state

b. evaluate the new state

c. If there state is a goal state
then quit
else compare with target

d. If this state is better than target
Set this state as target

e. If target is better than current state
Set current state to target

Step3: Exit.

3. Stochastic Hill Climbing: It does not examine all the neighbouring nodes before deciding which node to select. It just selects a neighbouring node at random & decide (based on the amount of improvement in that neighbour) whether to move to that neighbour or to examine another.

Algorithm:

Step1: Set current state to a randomly generated state.

Step2: Repeat the following steps until a solution is found or current state does not change.

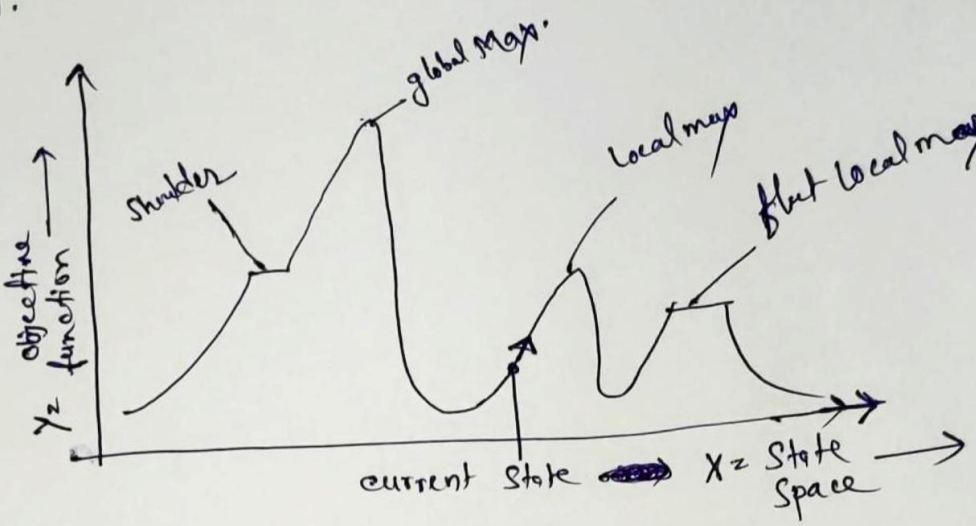
i. Set the target state to a random state neighbouring to the current state

ii. if target state is better than the current state
then
set the target state as the current state

Step3: Exit.

State Space Diagram for hill climbing:

It's a graphical representation of the set of states our search algorithm can reach vs the value of our objective function.



1. **Local Maxima:** At local maxima all neighbouring states have values which is worse than the current state. Since hill climbing uses a greedy approach, it will not explore other options & terminate itself. The process will end even though a better solution may exist.

2. **Overcoming local maxima:** We should utilize back-tracking technique. Maintain a list of visited state if search reaches a undesirable point, it can back-track to the previous configuration & explore new path.

2. **Global maxima:** It's the best possible state in the state space diagram. at this state the objective function has highest value.

3. Plateau / flat local maxima: It is a flat region of state space where neighbouring state have the same value.
⇒ On plateau all neighbouring has the same value. Hence it is not possible to select the best direction

To overcome plateau: Make a big jump. Randomly select a state far away from current state. chances are that we will land at a non-plateau region.

4. Ridge: It is a region which is higher than the neighbours but itself has a slope. It is a special kind of local maxima.
⇒ Any point on the ridge can look like peak because movement in all possible direction is downward. Hence the Algorithm stops when it reaches this state.

To overcome Ridge: In this kind of obstacle, use two or more rules before testing. It implies moving in several direction at once.

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 uphil edge.