**HEURISTIC FUNCTIONS**:

A heuristic function is a function that maps from problem state description to measures of desirability, usually represented as numbers. Aspects of problem states are evaluated, and the weights given to individual aspects are chosen in a way that the value of the heuristic function at a given node in the search process gives as good as good an estimate as possible of whether that node is on the desired path to a solution.

Heuristic functions can play an important part in efficiently guiding a search process. Heuristic functions provide a good estimate of a path. The below example shows the heuristic functions for a few problems.

| | |
|---|---|
| Chess | The material advantage of our side over the opponent. |
| Travelling Salesman | The sum of the distances so far. |
| Tic-Tac-Toe | 1 for each row in which we could win and in which we already have one piece plus 2 for each such row in which we have two pieces. |

Sometimes a high value of the heuristic function indicates a good position , low value indicates an advantageous situation. No matter the way function attempt to minimize it or to maximize it as appropriate.

- The Heuristic function (HF) guide the search process in the most profitable direction by suggesting which path to follow first when more than one is available.
- H.F estimates the true merits of each node in the search tree , the more direct solution process.
- H.F is so good that definitely no search would be required.
- For many problems, the cost of computing the value of such a function would outweigh the effort in the search process.
- Compute a perfect heuristic function by doing a complete search from the node in question and determining whether it leads to a good solution.
- Trade –off (Balance) between the cost of evaluating a heuristic function and the savings in search time that the function provides.
- Some heuristics used to define the control structure that guides the application of rules in the search process.

## HILL CLIMBING

It is a variant and test in which feedback from the test procedure is used to help generator decide which direction to move in the search space. The test functions responds yes or no. Test function provides an estimate of how close a given state is to a goal state. The generate procedure that often the computation of the heuristic function is done at almost no cost at the same time that the test for a solution is being performed.

Example:

Suppose we are in an unfamiliar city without a map and want to get downtown. Our ai m is for the tall buildings.

Heuristic function is distance between the current location and the location of the tall buildings and desirable states are those in which this distance is minimized. For this problem, hill climbing can terminate whenever a goal state is reached.

- Absolute solution exist whenever it is possible to recognize a goal state just by examining it.
- Relative solution exists, for maximization problems , such as travelling salesman problem , there is no prior goal state.

### 1.15.1 Simple Hill Climbing:

**Algorithm: Simple Hill Climbing**

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until there are no new operators left to be applied in the current state:
    a. Select an operator for that has not yet been applied to the current state and apply it to produce a new a state.
    b. Evaluate the new state.
        i. If it is a goal state, then return it and quit.
        ii. If it is not a goal state but it is better than the current state, then make it the current state.
        iii. If it is not better than the current state, then continue in the loop.

**Algorithm: Generate and Test:**

1. Generate a possible solution, For some problems, this means generating a particular point in the problem space. For others, it means generating a path from a start state.
2. Test to see if this is actually a solution by comparing the chosen point or the endpoint of the chosen path to the set of acceptable goal state.
3. If a solution has been found, quit. Otherwise, return to step 1.

| Generate and Test | Simple Hill climbing |
|---|---|
| i. Evaluation function is used as a way to inject task-specific knowledge into the control process.<br>ii. Knowledge give heuristic search methods their power to solve intractable problems.<br>iii. Unclear question is asked, "Is one state is better than another?" For the algorithm to execute, definition of better is provided.<br>iv. In some cases, higher value of the heuristic function.<br>v. In some , lower value of the heuristic function. | i. Execution of this algorithm let's take four colored blocks puzzle, to solve this, need to define a heuristic function that describes how close a particular configuration is to being a solution.<br>ii. Function is the sum of the number of different colors on each of the four sides.<br>iii. Solution for this puzzle , is value of 16.<br>iv. A set of rules that describe ways of transforming one configuring into another.<br>Ex: pick a block and rotate it 90 degrees in any direction.<br>v. Next step is to generate a starting configuration.<br>vi. Hill climbing begin, generate a new state by selecting a block and rotating it.<br>vii. Result state is not good, return to previous state and try a different perturbation. |

### 1.15.2 Steepest –Ascent Hill climbing:

Simple hill climbing considers all the moves from the initial state and selects the best onr as the next state. This method is called Steepest –ascent hill climbing or gradient search.

**Algorithm: Steepest- Ascent Hill climbing**

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until a complete iteration produces no change to current state.
    a. Let SUCC be a state such that any possible successor of the current state will be better than SUCC.
    b. For each operator that applies to current state do:
        i. Apply the operator and generate a new state.
        ii. Evaluate the new state. If it is a goal state, then return it and quit. If not, compare it to SUCC. If it is better, then set SUCC to this state. If it is not better, leaves SUCC alone.
    c. If the SUCC is better than current state, then set current state to SUCC.

Applying steepest-ascent hill climbing to the colored blocks problem, consider all perturbations of the initial state and choose the best. There are many moves in a problem. There is a balance between the time required to select a move and the number of moves required to get to a solution that is considered when deciding which method will work better.

Algorithm terminate by not finding a goal state but by getting to a state from which no better states can be generated. (i.e) if the program has reached either a local maxima, a plateau or a ridge.

**Hill climbing Disadvantages**

**Local Maximum**: A local maximum is a state that is better than all its neighbors but is not better than some other states farther away.

**Plateau**: A Plateau is a flat area of the search space in which a whole set of neighboring states have the same value.

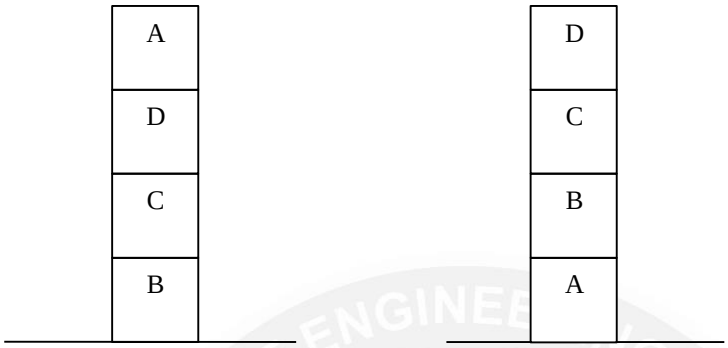
**Ridge**: A ridge is a special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope.

**Ways out**

- Backtrack to some earlier node and try going in a different direction.
- Make a big jump to try to get in a new section.
- Moving in several directions at once.

Hill climbing is a **local method:** Decides what to do next by looking only at the immediate consequence of its choice.

Global information might be encoded in heuristic functions.




**Hill climbing conclusion**

 Can be very inefficient in a large, rough problem space.

 Global heuristic may have to pay for computational complexity.

 Often useful when combined with other methods, getting it started right in the right general neighbourhood.

### 1.15.3 Stimulated Annealing:

It is aviation of hill climbing in which, downhill moves made at the beginning of the climbing in which, downhill moves made at the beginning of the process. The final solution is insensitive to the starting state. This lower the changes of getting caught at a local maximum a plateau , or a ridge.

**Two changes:**

- The objects function is used in place of the term heuristic function.
- Minimize rather than maximize the value of the objective function. We describe a process of valley descending rather than hill climbing.

Stimulated annealing , is a process is patterned after the physical process of annealing physical substances such as metals are melted and then cooled until some solid state is reached. The goal of this process is to produce a minimal energy final state. This process is one of valley descending in which the objects function is the energy level. Physical substances move from

high to low, so the valley descending occurs. There is some probability that a translation to a higher energy state will occur. This probability is given by the function.

This probability is

$$P = e^{-\Delta E/KT}$$

Where E is positive charge in energy level, t is temperature and k is Boltzman constant. As indicated by the equation the probability decreases with badness of the move (evaluation gets worsened by amount -ΔE).

The rate at which -ΔE is cooled is called annealing schedule. The proper annealing schedule is maintained to monitor T.

This process has following differences from hill climbing search:

- The annealing schedule is maintained.
- Moves to worse states are also accepted.
- In addition to current state, the best state record is maintained.

The algorithm of simulated annealing is presented as follows:

**Algorithm: Simulated annealing:**

1. Evaluate the initial state. Mark it as current state. If it is not the goal state, then return it and quit. Otherwise continue with the initial state as the current state.
2. Initialize best state to current state. If the initial state is the best state, return it and quit.
3. Initialize T according to annealing schedule.
4. Loop until a solution is obtained or until there are no new operators left to be applied in the current state.
    a. Select and apply yet unapplied operator to produce a new state
    b. Evaluate the new state compute.
        -ΔE = value of current state – value of new state.
        i. If the new state is the goal state then stop, or if it is better than current state, make it as current state and record as best state.
        ii. If it is not better than the current state, then make it current state with probability P.
    c. Revise T according to annealing schedule

4. Return best state as answer.

**Implementation of Algorithm:**

Select Annealing schedule, which has three components.

a. The first is the initial value user for temperature.

b. The second is the criteria will be used to decide when the temperature of the system should be reduced.
c. The third is the amount by which the temperature will be reduced each time it is changed.

Fourth component of the schedule, when to quit.

- Simulated annealing solve the problems where number of moves from a given state is very large.
- For such problems, it makes no sense to try all possible moves.
- The Best annealing schedule should be observed in a way that the effect on both the quality of the solution that is found and the rate at which the process converges.
    i. T approaches zero, the probability of accepting a move to a worse state goes to zero and simulated annealing becomes identical to simple hill climbing.
    ii. Computing the probability of accepting a move is the ratio - $\Delta E/T$. Thus its important that values of T be scaled so this ratio is meaningful