21CS2213RA
AI for Data Science

Session -7

Contents: *Local Search Algorithms*

- This systematicity is achieved by keeping one or more paths in memory and by recording which alternatives have been explored at each paint along the path.

- When a goal is found, the path to that goal also constitutes a solution to the problem.

- In many problems, how-ever, the path to the goal is irrelevant. For example, in the 8-queens problem . what matters is the final configuration of queens, not the order in which they are added.

- **The same general property holds for many important applications such as integrated-circuit de-sign, factory-flour layout, job-shop scheduling, automatic programming, telecommunications network optimization, vehicle routing. and portfolio management.**

LOCAL SEARCH ALGORITHMS

- Local search algorithms operate using a single current node (**rather than multiple paths**) and generally move only to neighbors of that node.
- Typically, the paths followed by the search are not retained.
- Although local search algorithms are not systematic, they have two key advantages:

(1) they use very little memory—usually a constant amount; and

(2) they can often find reasonable solutions in large or infinite(continuous) state spaces for which systematic algorithms are unsuitable.

- In addition to finding goals, local search algorithms are useful for solving pure optimization problems, in which the aim is to find the best state according to an objective function.
- A complete local search algorithm always finds a goal if one exists;
- An optimal algorithm always finds a global minimum/maximum.

HILL CLIMIBING

- Hill climbing is a simple local optimization method that “climbs” up the hill until a local optimum is found (assuming a maximization goal).

- The method works by iteratively searching for new solutions within the neighborhood of current solution, adopting new solutions if they are better.

- There are several hill climbing variants

- ✓ **Steepest Ascent Hill Climbing:** which searches for upto N solutions in the neighborhood of S and then adops the best one. It is time consuming but gives an optimum result.
- ✓ **Stochastic Hill Climbing:** which replaces the deterministic select function, selecting new solution with a probability of P.

Hill Climbing

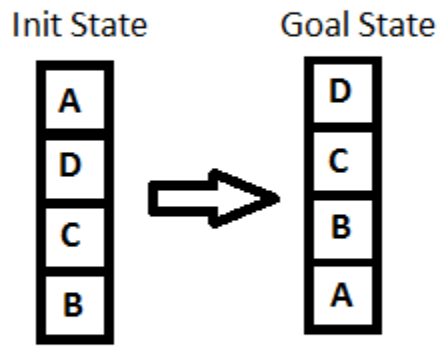
Algorithm:

1. Evaluate the initial state. If it is also 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 that has not yet been applied to the current state and apply it to produce a new state
    b. Evaluate the new state
        i. If it is the 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.

BLOCKS WORLD PROBLEM

Hill Climbing Algorithm can be categorized as an informed search. So we can implement any node-based search or problems like the n-queens problem using it. To understand the concept easily, we will take up a very simple example.

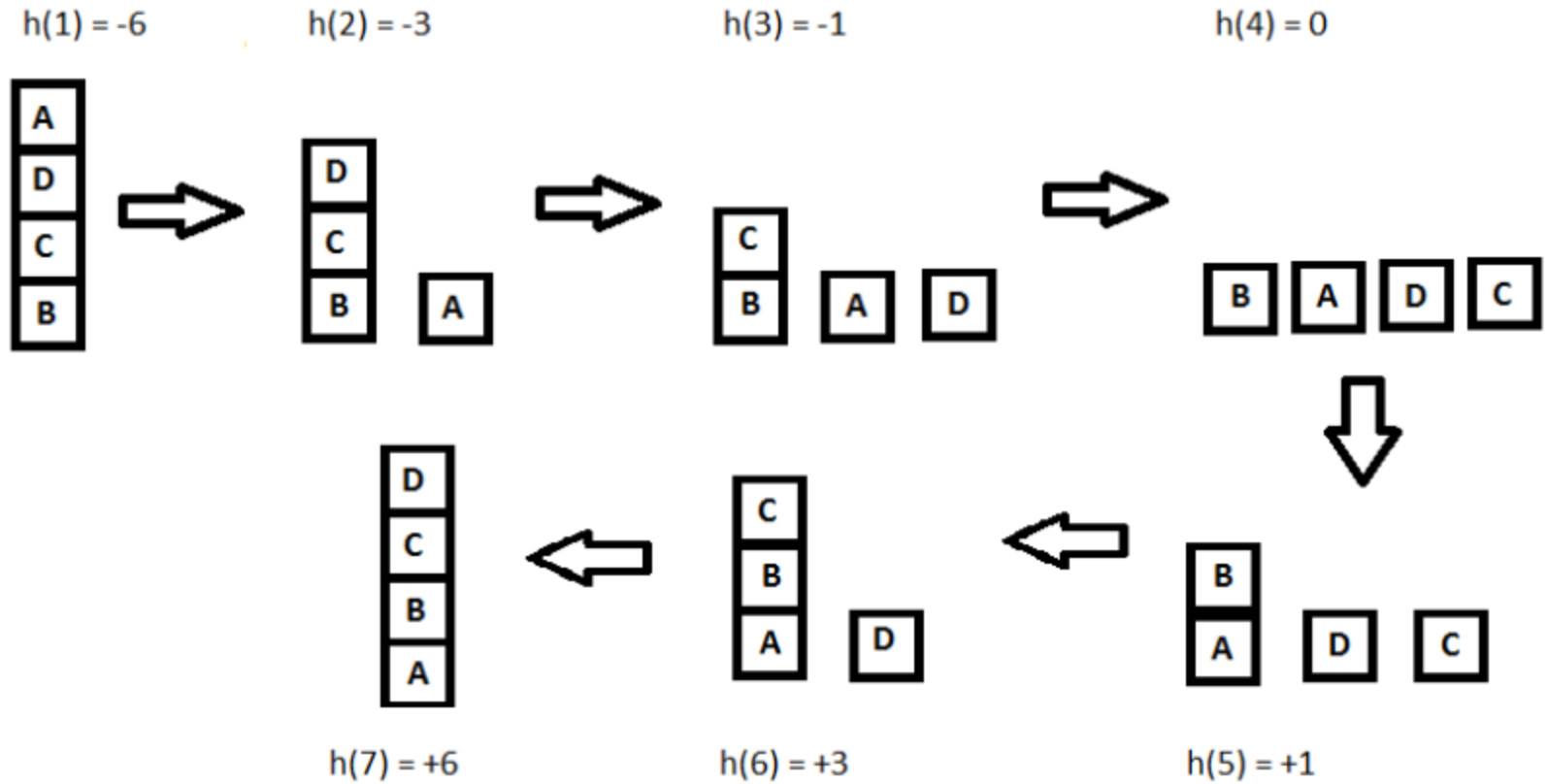
Let's look at the image below:



Key point while solving any hill-climbing problem is to choose an appropriate heuristic function.

Let's define such function *h*:

***h(x)* = +1 for all the blocks in the support structure if the block is correctly positioned otherwise -1 for all the blocks in the support structure.**


h(1) = -6
h(2) = -3
h(3) = -1
h(4) = 0
A
D
C
B
D
C
B
A
C
B
A
D
B
A
D
C
B
A
D
C
D
C
B
A
C
B
A
D
h(7) = +6
h(6) = +3
h(5) = +1

Limitations of Hill-climbing

1. **Local Maxima**: a local maximum as opposed to global maximum.

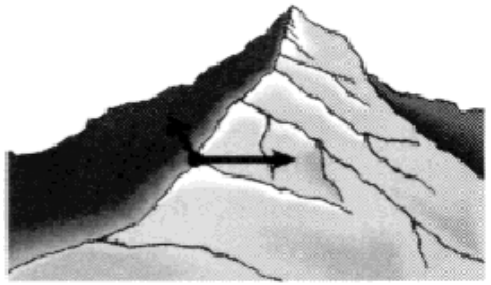
   Way Out: Backtrack to some earlier node and try going in a different direction

2. **Plateaus**: An area of the search space where evaluation function is flat, thus requiring random walk.

   Way out: Make a big jump to try to get in a new section

3. **Ridge**: Where there are steep slopes and the search direction is not towards the top but towards the side.

   Way out: Moving in several directions at once.



Steepest-Ascent Hill Climbing

- This is a variation of simple hill climbing which considers all the moves from the current state and selects the best one as the next state. Also known as 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 the current state do:
      i. Apply the operator and generate a new state
      ii. Evaluate the new state. If is 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, leave SUCC alone.
   c. If the SUCC is better than the current state, then set current state to SUCC,

Simulated Annealing

- Simulated Annealing (SA) is an effective and general form of optimization.

  It is useful in finding global optima in the presence of large numbers of local optima. “Annealing” refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal.

Simulated annealing uses the objective function of an optimization problem instead of the energy of a material. Implementation of SA is surprisingly simple. The algorithm is basically hill-climbing except instead of picking the best move, it picks a random move. If the selected move improves the solution, then it is always accepted. Otherwise, the algorithm makes the move anyway *with some probability* less than 1. The probability decreases exponentially with the “badness” of the move, which is the amount deltaE by which the solution is worsened (i.e., energy is increased.)

Local Beam Search

In this algorithm, it holds k number of states at any given time. At the start, these states are generated randomly. The successors of these k states are computed with the help of objective function. If any of these successors is the maximum value of the objective function, then the algorithm stops.

Otherwise the (initial k states and k number of successors of the states = 2k) states are placed in a pool. The pool is then sorted numerically. The highest k states are selected as new initial states. This process continues until a maximum value is reached.

function BeamSearch(*problem, k*), returns a solution stat

start with k randomly generated states

loop

generate all successors of all k states

if any of the states = solution, then return the state

else select the k best successors

end

Thank you