Ishita Handasmalani
C14-2103058.
THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Experiment No. 5.

Aim: Implementing Genetics / Hill climbing algorithm using python.

Theory:

A Genetic algorithm (or GA) is a variant of stochastic beam search in which successor states are generated by combining two parent states, rather than by modifying a single state. The analogy to natural selection is the same as in stochastic beam search, except that now we are dealing with sexual rather than asexual reproduction. Genetic algorithms begin with a set of k randomly generated states, called the population. Each state, or individual, is represented as a string over a finite alphabet - a string of 0's & 1's.

The production of the next generation of states, each state is rated by the objective function or the fitness function. A fitness function should return higher values for better states. The time complexity depends on various factors such as size of the population, length of the chromosomes, complexity of fitness function, no. of generations & implementation, approximated O(N*L*G*F) & the space complexity is O(N*L), N is population size, L is the length of chromose, G is the no. of generations, F is evaluating fitness function t.c.

function GENETIC-ALGORITHM(population, FITNESS-FN)
returns an individual.
inputs: population, a set of individuals.
FITNESS-FN, a function that measures the fitness of an individual.

repeat
new-population ← empty set
for i = 1 to SIZE(population) do
x ← RANDOM-SELECTION(population, FITNESS-FN)
y ← RANDOM-SELECTION(population, FITNESS-FN)
child ← REPRODUCE(x, y)
if (small random probability)
then child ← MUTATE(child)
add child to new-population
population ← new-population
until some individual is fit enough, or enough time has elapsed.
return the best individual in population,
according to FITNESS-FN.

function REPRODUCE(x, y) returns an individual
inputs: x, y, parent individuals

n ← LENGTH(x); c ← random number from 1 to n
return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c+1, n)).

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Hill climbing Algorithm: The hill-climbing search algorithm is simply a loop that continually moves in the direction of increasing value. It terminates when it reaches a "peak" where no neighbor has a higher value. The algorithm does not maintain a search tree, so the data structure for the current node need only record the state & the value of the objective function. Hill climbing does not look ahead beyond the immediate neighbors of the current state.

Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead about where to go next. Hill climbing often makes rapid progress toward a solution because it is usually quite easy to improve a bad state.

Unfortunately, hill climbing often gets stuck for the following reasons:

- Local maxima: a local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum. Hill-climbing algorithms that reach the vicinity of a local maximum will be drawn upward towards the peak but will then be stuck with nowhere else to go.

- Ridges: Ridges result in a sequence of local maxima that is very difficult for greedy algorithms to navigate.
- Plateaux: A plateau is a flat area of the state-space landscape. It can be a flat local maximum, from which no uphill exit exists, or a shoulder, from which progress is possible. A hill-climbing search might get lost on the plateau.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← MAKE-NODE(problem.INITIAL-STATE)
    loop do
        neighbor ← a highest-valued successor of current
        if neighbor.VALUE ≤ current.VALUE then return current.STATE
        current ← neighbor
```

Hill climbing is neither complete nor optimal, time complexity O(∞) & space complexity O(b)