Name: Kartik Jolapara **SAPID:** 60004200107

DIV: B/B1

A.I.

Exp4

Aim: To study and implement Hill-Climbing Search

Theory:

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 neighbour 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 travelled by the salesman.

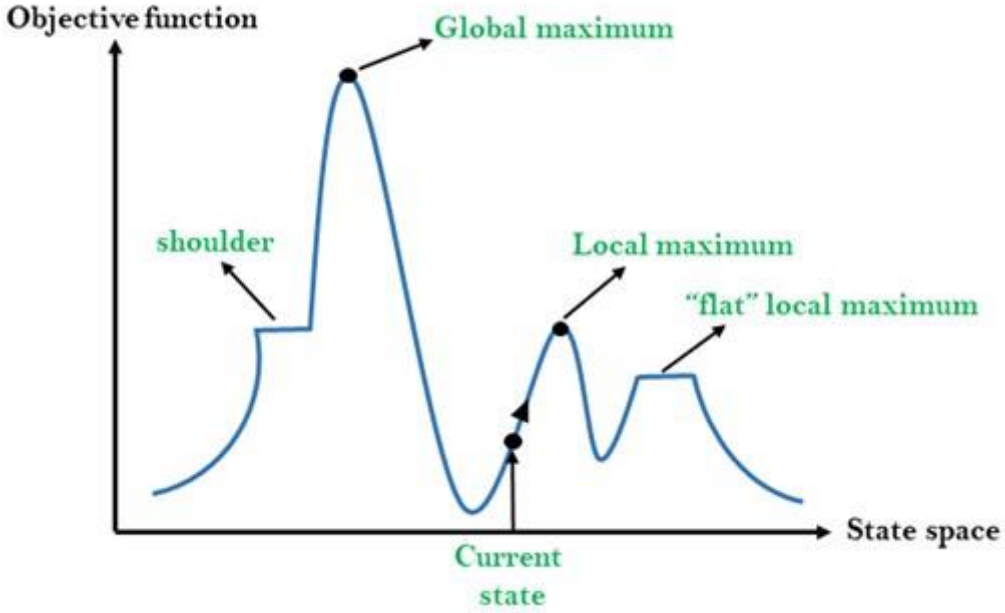
It is also called greedy local search as it only looks to its good immediate neighbour state and not beyond that. A node of hill climbing algorithm has two components which are state and value. Hill Climbing is mostly used when a good heuristic is available. In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Features:

1. 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.

2. Greedy approach: Hill-climbing algorithm search moves in the direction which optimizes the cost.

3. No backtracking: It does not backtrack the search space, as it does not remember the previous state



Problems:

1. **Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighbouring states, but there is another state also present which is higher than the local maximum
2. **Plateau:** A plateau is the flat area of the search space in which all the neighbour states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area
3. **Ridges:** A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Code:

```
import random

def randomSolution(tsp):
cities = list(range(len(tsp)))
solution = []

    for i in range(len(tsp)):
        randomCity = cities[random.randint(0, len(cities) - 1)]
solution.append(randomCity)         cities.remove(randomCity)

    return solution

def routeLength(tsp, solution):
    routeLength = 0     for i in
range(len(solution)):
        routeLength += tsp[solution[i - 1]][solution[i]]
return routeLength

def getNeighbours(solution):
    neighbours = []     for i in
range(len(solution)):         for j in
range(i + 1, len(solution)):
neighbour = solution.copy()
neighbour[i] = solution[j]
neighbour[j] = solution[i]
neighbours.append(neighbour)
return neighbours
```

```
def getBestNeighbour(tsp, neighbours):
    bestRouteLength = routeLength(tsp, neighbours[0])
bestNeighbour = neighbours[0]     for neighbour in
neighbours:
        currentRouteLength = routeLength(tsp, neighbour)
if currentRouteLength < bestRouteLength:
bestRouteLength = currentRouteLength
bestNeighbour = neighbour     return bestNeighbour,
bestRouteLength

def hillClimbing(tsp):
    currentSolution = randomSolution(tsp)     currentRouteLength =
routeLength(tsp, currentSolution)     neighbours =
getNeighbours(currentSolution)     bestNeighbour,
bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

    while bestNeighbourRouteLength < currentRouteLength:
        currentSolution = bestNeighbour         currentRouteLength =
bestNeighbourRouteLength         neighbours =
getNeighbours(currentSolution)         bestNeighbour,
bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

    return currentSolution, currentRouteLength

def main():
tsp = [
        [0, 100, 700, 50],
```

```
        [100, 0, 330, 1200],
        [700, 330, 0, 400],
        [50, 1200, 400, 0]    ]

    print(hillClimbing(tsp))

if __name__ == "__main__":
main()
```

Output:

```
[3, 2, 1, 0], 880)
```

Conclusion:

Thus we successfully studied and implemented Hill-Climbing Search