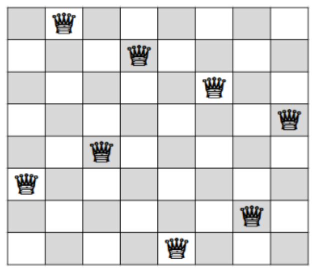
UNC CHARLOTTE
College of Computing and Informatics

PROJECT – 2

SOLVING N-QUEENS PROBLEM USING HILL-CLIMBING AND ITS VARIANTS



DEPARTMENT OF COMPUTER SCIENCE

ITCS 6150 - Intelligent Systems

Submitted To

Dewan T. Ahmed, Ph.D.

Submitted By

Gowtham Bharadwaj
801101552
gbharadw@uncc.edu

Medha Nagaraj
801101751
mnagara1@uncc.edu

N-QUEENS PROBLEM USING HILL-CLIMBING

1. Table of Contents

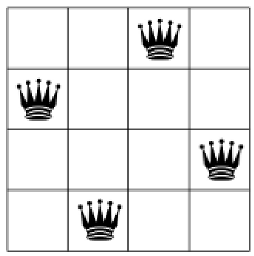
1. PROBLEM FORMULATION	
1.1 Introduction	**3**
1.2 Algorithm Pseudocode	**5**
2. PROGRAM STRUCTURE	
2.1 Procedures	**6**
2.2 Classes and Methods	**7**
2.3 Global Variables	**7**
2.4 Logic	**9**
3. CODE FOR EACH HEURISTIC	
3.1 Code for the N-Queen Problem	**10**
4. SAMPLE OUTPUTS	
4.1 Steepest- ascent hill climbing	**23**
4.2 Hill-climbing search with sideways move	**28**
4.3 Random Restart Hill Climbing without sideways	**32**
4.4 Random Restart Hill Climbing with sideways	**37**
5. CONCLUSIONS	**41**
6. REFERENCES	**42**

PROBLEM FORMULATION

1.1 INTRODUCTION

What is N – Queen Problem?

The N queens problem represent a chessboard whose goal is to place the N-queens on the given N X N chessboard squares, so that no two queens can attack each other. The Queens, like the chessboard, can move horizontally, vertically, and diagonally. Only one queen can be placed per row and per column, and no two queens can be placed on the same diagonal.
The solution for this problem is : NP-complete which makes it a computationally expensive problem to solve. Below provided is a sample example of 4-Queens which have been placed so that no two queens are attacking each other.



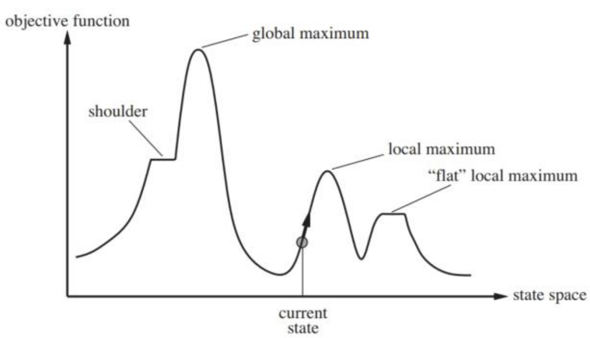
The N-Queens problem can be solved using the famous Hill Climbing Approach which can be further classified into three variants are explained as below:

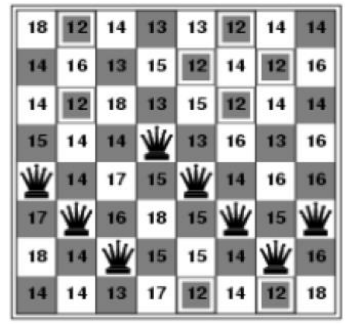
1. Steepest Hill-Climbing Search
2. Hill-climbing search with sideways move
3. Random Restart Hill Climbing without sideways
4. Random Restart Hill Climbing with sideways

What is Hill Climbing Approach? :

The Hill-Climbing Algorithm is the most basic technique used to solve the N-Queen Problem.

- It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by making an incremental change to the solution.
- If the change produces a better solution, the incremental changes continue to be made as the new solution until no further improvements can be done.
- The problem uses a heuristic-search approach which means that it will provide a good solution in a short span of time.
- It might or might not be optimal but always selects the best route out of the possible routes.





h = number of pairs of queens that are attacking each other, either directly or indirectly
h = 17 for the above state

The first figure above shows the one-dimensional state space landscape representation. The second figure shows the heuristic approach for the 8-Queen Problem.

- **Steepest Ascent Hill Climbing**

In this basic hill climbing approach, the algorithm examines all the neighboring nodes and then chooses the best successor closest to the solution state.
This method has a major drawback as it might possibly reach a state from where no subsequent improvement can be done and hence failing to find a solution.

- **Hill Climbing with Sideways move:**

Hill-climbing : how to avoid the shoulders encountered in the graphs?
Starting from a random initial state of 8-queens, the hill-climbing approach gets stuck in a local maximum 86% of the times (with an average of 3 steps) and finds the global maximum only 14% of the times (with an average of 4 steps).
Sideways moves are implemented to avoid getting stuck in a shoulder plateau. We allow hill-climbing to move to neighboring states that have the same value as the current one. However, infinite loops may occur which can be avoided by limiting the number of sideways moves. A 100 sideways moves in the 8-queens problem can increase the percentage of problem instances solved by hill climbing from 14% to 94%. However, each successful instance is solved in 21 steps (on average) and failures (local maxima) are reached after 64 steps (on average).

- **Random Restart Hill Climbing without Sideways moves**

In this method, the hill-climbing algorithm repeats itself from a given initial state, until a solution is found. The solution is guaranteed in this approach as it is set for a finite state space. Here it uses the same logic as the steepest hill climbing algorithm but we do not consider the scenario of all the successors have heuristic greater than or equal to heuristic of current state as a failure, instead a new initial state is generated and algorithm is applied on that newly generated configuration. This is continued until a success is found.

- **Random Restart Hill Climbing with Sideways move**

In this approach we use the same logic as in the sideways hill climbing algorithm with an improvisation that we do not abort in case of below scenarios:

a) When all the successors have heuristic greater than the heuristic of current state.
b) When even after a 100 consecutive sideways move a success is not found nor a successor is found with better heuristic value than current state.

When any of the above scenarios occur, we randomly generate a new initial state and apply the sideways hill climbing on that newly generated configuration. This is continued until a success is found.

1.2 ALGORITHM PSEUDOCODE

```
function HILL-CLIMBING(problem) returns a solution state
        inputs: problem, a problem
        static: current, a node
                next, a node
        current <— MAKE-NODE(INITIAL-STATE[problem])
        loop do
                next— a highest-valued successor of current
                if VALUE[next] < VALUE[current] then return current
                current *—next
        end
```

PROGRAM STRUCTURE

2.1 PROCEDURES

Language Used: The code is developed and implemented in Python 3.7.

Input: The user can choose from the below provided options:

Puzzle Size: The user can choose any value of N for the N-Queen problem.

Iteration Count: The user can choose the default iteration size of 500 or otherwise opt for the 2nd option and provide the iteration count of their interest.

Approach: The user can choose from the below 4 variants

1. Steepest- ascent hill climbing
2. Hill-climbing with sideways move
3. Random-restart hill-climbing without sideways move
4. Random-restart hill-climbing with sideways move

Output:

Based on the approach the user selected the output varies

If the user selects approach 1 or 2 (Steepest- ascent hill climbing, Hill-climbing with sideways move) the output would have the below values.

- Total Number of Runs
- Total Success
- Success Rate
- Total Fail
- Failure Rate
- Average number of steps in success
- Total Steps for Success
- Total Steps for Fail
- Average number of steps when failed

If the user selects approach 3/4 (Random-restart hill-climbing without sideways move, Random-restart hill-climbing with sideways move)

- Total Number of Runs
- Total Success
- Success Rate
- Total Number of random restarts
- Average number of random restarts
- Average number of steps

2.2 CLASSES & METHODS :

Classes:

1) **queenPuzzle–** Contains all the methods which will initialize the hill climbing approach, implement different variants of hill climbing, prints the outputs, calculates the heuristics, gives the successor states.

- **Methods**

a) **__init__:** initializes the queens class objects. Calculates number of iterations, heuristics and create a new board.

b) **hillClimbing**: This method gives a successor with better heuristic and calculate the goal state, otherwise calculate the total steps. When no better successor found, it updates the run as failure (No Solution Found).

c) **sideways:** Calls the method which gives successor and based on the heuristic, it will calculate the goal state, otherwise calculate the total steps. It allows 100 sideways moves post which it updates the run as failure (No Solution Found).

d) **randomRestartWithoutSidemove**: Does the similar function and when no better successor is found, then it will update the run as failure and restart with some random initial state and start the process again.

e) **randomRestartWithSidemove**: Does the similar function of calling a method to calculate goal state with heuristic and find successor and calculate the total steps. It allows 100 sideways moves. If no better successor is found, it will update the run as failure and it will restart with some random initial state and start the process again.

f) **ReportStatistics**: Prints the outputs based on the approach or variant selected.

g) **attackHeuristic**: Calculate the number of attacks, which gives the heuristic. Calculates the horizontal, vertical and diagonal attacks and provides the final heuristic. For the horizontal attacks we select each queen solely present in one column and check for any other queen present in the same row as the selected queen. The attacks are incremented every time for each queen found in the same row as the selected queen. Similarly, for each selected queen all the 4 diagonal squares are checked for any contradicting queen presence and simultaneously the diagonal attacks are incremented for each queen found contradicting the selected queen. The sum of horizontal attacks and diagonal attacks is used as the heuristic function.

h) **optimalBoard**: This method is used by steepest and restart without sideways move approaches. It will check for all the combinations and gives out the best successor with the least heuristic. If it couldn't find any best heuristic it will send back the current heuristic value.

i) **successorBoard**: This method is used by sideways and restart with sideways move approaches. It checks for all the combinations and gives out the best successor with the least heuristic. If it couldn't find any best heuristic it will at-least check for same heuristic successor and gives them. If it couldn't find any of the above it will send back the current heuristic value.

2) **Board –** Contains all the methods which will initialize the chess board

- **Methods**

a) **__init__:** It will initialize the board object. It will create a new random chess board.

Global Methods:
printBoard: It will print the board with the correct spacings between columns and rows.

Main Method:

Starts the application logic and provides choices to the user. The user can select the queen size, no of iterations, hill climbing variant approach etc. The selected method calls the corresponding variant method. It handles the invalid selections, informs the user and select the default values, runs the logic and provides the solution to the user.

2.3 GOLBAL VARIABLES

The local variables within function are:

a) **Iterate**: Number of runs the user can give Eg: 100

b) **solution**: Used this for logic purpose

c) **randomRestart**: Number of restarts Eg:7

d) **stepsClimbed**: Total number of steps in the restart run Eg:22

e) **passedboard**: Used this for logic purpose

2.4 LOGIC

In the code implemented, we have used the priority queues as the data structures. The indexes calculate coordinates of each digit of the input & output and compare the difference and sort them accordingly. We repeat this until our goal is reached by generating the successor nodes and comparing them further. We calculate the heuristics based on these coordinates and the following information are used:

- Goal achieved status
- Total number of nodes generated
- Total number of nodes explored
- Total number of optimized steps
- Total Time Taken

The code works perfectly for states which can be solved, but it goes into a loop for the states which cannot be solved.

CODE FOR N-QUEEN PROBLEM

```
# -*- coding: utf-8 -*-
"""
@author: Gowtham Bharadwaj 801101552
         Medha Nagaraj 801101751
"""

import random,copy
iterate = 0
solution = True
randomRestart = 0
stepsClimbed = 0
passedboard = None

#This method prints the configuration of the N-Queen Puzzle
def printBoard(state):
  for l in range(0,N):
       for m in range(0,N):
          if m < N-1:
             print(state[l][m], end=" ")
          elif(m == N-1):
             print(state[l][m], end="\n")

#Class definition of the Queen
class queenPuzzle:
 #Method definition to intialize the variables required to track the result
 def __init__(self, searchCategory, iterate, solution):
   #Initialize the required variables and counters
```

```
 self.totalRuns = iterate
 self.totalSuccess = 0
 self.totalFail = 0
 self.stepsForSuccess = 0
 self.stepsForFail = 0
 self.sideMove = 0

 self.solution = solution
 for i in range(0,iterate):
  if self.solution == True:
    print ("\n---------------------------")
    print ("Board configuration for the run",i+1)
    print ("---------------------------")
  self.queen_board = board(passedboard)
  self.cost = self.attackHeuristic(self.queen_board)
  #Call to the appropriate hill climbing code
  if (searchCategory == 1):
     self.hillClimbing()
  elif (searchCategory == 2):
     self.sideways()
  elif (searchCategory == 3):
     self.randomRestartWithoutSidemove()
  elif (searchCategory == 4):
     self.randomRestartWithSidemove()

#Method for the Steepest hill climbing code
def hillClimbing(self):
 totalSteps = 0
 while 1:
```

```
  current_attacks = self.cost
  #Breaking if the random initial state itself is a sucess state
  if self.cost == 0:
    break
  #Call the optimal board
  self.optimalBoard()
  if (current_attacks == self.cost):
   self.totalFail += 1
   self.stepsForFail += totalSteps
   if totalSteps == 0:
     self.stepsForFail += 1
   break
  totalSteps += 1
  if self.solution == True:
   print ("\nThe Number of attack pairs is", (int)(self.attackHeuristic(self.queen_board)))
   printBoard(self.queen_board.board)
  if (self.cost == 0):
    break
 #Handle failure - when the solution is not found
 if self.cost != 0:
  if self.solution == True:
   print ("\n*****NO SOLUTION*****")
 #Handle failure - when the solution is found
 else:
  if self.solution == True:
   print ("\n*****SOLUTION FOUND*****")
  self.totalSuccess += 1
  self.stepsForSuccess += totalSteps
 return self.cost
```

```
#Method definition for the Sideways hill climbing Algorithm
def sideways(self):
 totalSteps = 0
 sideMove = 0
 while 1:
  current_attacks = self.cost
  current_board = self.queen_board
  #Breaking if the random initial state itself is a sucess state
  if self.cost == 0:
    break
  #Call to generare the sucessesor board that can return both equal heuristic board or lower heuristic board
  self.successorBoard()
  if current_board == self.queen_board:
    self.stepsForFail += totalSteps
    self.totalFail += 1
    if totalSteps == 0:
     self.stepsForFail += 1
    break
  if current_attacks == self.cost:
   sideMove += 1
   if sideMove == 100:
     self.stepsForFail += totalSteps
     self.totalFail += 1
     break
  elif(current_attacks > self.cost):
    sideMove = 0
  totalSteps += 1
  if self.solution == True:
```

```
            print ("\nThe Number of attack pairs is", (int)(self.attackHeuristic(self.queen_board)))
            printBoard(self.queen_board.board)
        if self.cost == 0:
            break
    if self.cost != 0:
        if self.solution == True:
            print ("\n***** NO SOLUTION *****")
    else:
        if self.solution == True:
            print ("\n***** SOLUTION FOUND *****")
        #Increment the count of the number of success incurred and total steps taken for each successful iteration
        self.totalSuccess += 1
        self.stepsForSuccess += totalSteps
    return self.cost


#Definition for the Random Restart without sideways allowing  hill climbing Algorithm
def randomRestartWithoutSidemove(self):
    while 1:
        current_attacks = self.cost
        current_board = self.queen_board
        #Breaking if the random initial state itself is a sucess state
        if self.cost == 0:
            break
        #Call to generare the sucessesor board
        self.optimalBoard()
        #Check and logic for random Restart
        if (current_board == self.queen_board) or ((current_attacks == self.cost) & (self.cost != 0)):
            self.queen_board = board(passedboard)
            global randomRestart
```

```
        #Increment the Random Restarts counter
        randomRestart += 1
        self.cost = self.attackHeuristic(self.queen_board)
      elif (self.cost < current_attacks):
          if self.solution == True:
              print ("\nThe Number of attack pairs is", (int)(self.attackHeuristic(self.queen_board)))
              printBoard(self.queen_board.board)
      global stepsClimbed
      #Increment the Steps counter in Random restart Hill climbing algorithm
      stepsClimbed += 1
      if self.cost == 0:
        break
    if self.solution == True:
        print ("\n***** SOLUTION FOUND *****")
         #Incrementing the count for number of success incurred
    self.totalSuccess += 1
    return self.cost


#Definition for the Random Restart with sideways allow hill climbing Algorithm
def randomRestartWithSidemove(self):
    sideMove = 0
    while 1:
      current_attacks = self.cost
      current_board = self.queen_board
      #Breaking if the random initial state itself is a sucess state
      if self.cost == 0:
          break
      #Call to generare the sucessesor board
      self.successorBoard()
```

```
  #Check and logic for random Restart
  if current_board == self.queen_board:
   self.queen_board = board(passedboard)
   global randomRestart
   #Increment the Random Restarts counter
   randomRestart += 1
   self.cost = self.attackHeuristic(self.queen_board)
  if current_attacks == self.cost:
   sideMove += 1
   if sideMove == 100:
    self.queen_board = board(passedboard)
    #Increment the Random Restarts counter
    randomRestart += 1
    self.cost = self.attackHeuristic(self.queen_board)
  elif(current_attacks > self.cost):
   sideMove = 0
  global stepsClimbed
  #Increment the Steps counter in Random restart Hill climbing algorithm
  stepsClimbed += 1
  if self.solution == True:
   print ("\nThe Number of attack pairs is", (int)(self.attackHeuristic(self.queen_board)))
   printBoard(self.queen_board.board)
  if self.cost == 0:
   break
 if self.solution == True:
   print ("\n***** SOLUTION FOUND *****")
    #Incrementing the count for number of success incurred
 self.totalSuccess += 1
 return self.cost
```

```
#Print Definition exclsive to each type of Hill climbing algorithm
def ReportStatistics(self):
 print ("\nTotal number of runs is", self.totalRuns)
 print ("Total Success: ", self.totalSuccess)
 print ("Success rate: ", (float(self.totalSuccess)/float(self.totalRuns))*100,"%")

 #Print statements for Steepest Hill climbing Algorithm
 # & Sideways Hill climbing Algorithm
 if(searchCategory == 1) or (searchCategory == 2):
   print ("Total Fail: ", self.totalFail)
   print ("Failure rate: ", (float(self.totalFail)/float(self.totalRuns))*100,"%")
   if(self.totalSuccess >= 1):
    print ("Average number of steps in success: ", float(self.stepsForSuccess)/float(self.totalSuccess))
    print ("Total Steps for Success: ", self.stepsForSuccess)
   if(self.totalFail >= 1):
    print ("Total Steps for Fail: ", self.stepsForFail)
    print ("Average number of steps when failed: ", float(self.stepsForFail)/float(self.totalFail))

 #Print statements for Random Restart Hill climbing Algorithm
 if(searchCategory == 3) or (searchCategory == 4):
   print ("Total number of random restarts:", randomRestart)
   print ("Average number of random restarts: ", float(randomRestart)/float(self.totalRuns))
   print ("Average number of steps: ", float(stepsClimbed)/float(self.totalRuns));

#Definition for calculating the number of attack pairs
def attackHeuristic(self, temp_board):
 #these are separate for easier debugging
 straight_attacks = 0
```

```
diagonal_attacks = 0
for i in range(0,N):
 for j in range(0,N):
  #If the Queen node is encountered then calculate all of the attack pairs
  if temp_board.board[i][j] == "Q":
   #Subtract the total cost by 2 so that we don't count the self state
   straight_attacks -= 2
   for k in range(0,N):
    if temp_board.board[i][k] == "Q":
     straight_attacks += 1
    if temp_board.board[k][j] == "Q":
     straight_attacks += 1
   #Calculate all the diagonal attacks
   k, l = i+1, j+1
   while k < N and l < N:
    if temp_board.board[k][l] == "Q":
     diagonal_attacks += 1
    k +=1
    l +=1
   k, l = i+1, j-1
   while k < N and l >= 0:
    if temp_board.board[k][l] == "Q":
     diagonal_attacks += 1
    k +=1
    l -=1
   k, l = i-1, j+1
   while k >= 0 and l < N:
    if temp_board.board[k][l] == "Q":
     diagonal_attacks += 1
```

```
        k -=1
        l +=1
      k, l = i-1, j-1
      while k >= 0 and l >= 0:
        if temp_board.board[k][l] == "Q":
          diagonal_attacks += 1
        k -=1
        l -=1
  return ((diagonal_attacks + straight_attacks)/2)


#This function tries moving every queen to every spot, with only one move
#and returns the move that has the least number of attacks pairs
def optimalBoard(self):
  least_cost = self.attackHeuristic(self.queen_board)
  most_desirable = self.queen_board
  #We move one queen at a time
  for q_col in range(0,N):
    for q_row in range(0,N):
      if self.queen_board.board[q_row][q_col] == "Q":
        #We get the lowest cost configuration by moving each queen in its respective column
        for m_row in range(0,N):
          if self.queen_board.board[m_row][q_col] != "Q":
            #Queen is placed in empty slot of each column
            test_board = copy.deepcopy(self.queen_board)
            test_board.board[q_row][q_col] = "-"
            test_board.board[m_row][q_col] = "Q"
            test_board_cost = self.attackHeuristic(test_board)
            if test_board_cost < least_cost:
              least_cost = test_board_cost
```

```
        most_desirable = test_board
  self.queen_board = most_desirable
  self.cost = least_cost
#This function tries moving every queen to every spot, with only one move
#and returns the move that has the least number of attacks pairs or if not
#then it will atleast try to send the state with same heuristic
def successorBoard(self):
  equal_h_count = 0
  equi = {}
  presentcost = self.attackHeuristic(self.queen_board)
  least_cost = self.attackHeuristic(self.queen_board)
  most_desirable = self.queen_board
  #move one queen at a time, the optimal single move by brute force
  for q_col in range(0,N):
    for q_row in range(0,N):
      if self.queen_board.board[q_row][q_col] == "Q":
        #get the lowest cost by moving this queen
        for m_row in range(0,N):
          if self.queen_board.board[m_row][q_col] != "Q":
            #try placing the queen here and see if it's any better
            test_board = copy.deepcopy(self.queen_board)
            test_board.board[q_row][q_col] = "-"
            test_board.board[m_row][q_col] = "Q"
            test_board_cost = self.attackHeuristic(test_board)
            if test_board_cost < least_cost:
              least_cost = test_board_cost
              most_desirable = test_board
            if test_board_cost == presentcost:
              equi[equal_h_count] = test_board
```

```
          equal_h_count += 1
  if least_cost == presentcost:
    print("Number of successors with heuristic value same as that of the current state:", equal_h_count)
    if(equal_h_count == 1):
      most_desirable = equi[0]
    elif(equal_h_count > 1):
      rand_ind = random.randint(0,equal_h_count - 1)
      print("Random index chooses one of the successors with same heuristic value:", rand_ind)
      most_desirable = equi[rand_ind]
  self.queen_board = most_desirable
  self.cost = least_cost
#Class for the Board
class board:
 #Intialize Method which will generate a random initial state
 def __init__(self, list=None):
  if list == None:
   self.board = [["-" for i in range(0,N)] for j in range(0,N)]
   #initialize queens at random places
   for j in range(0,N):
    rand_row = random.randint(0,N-1)
    if self.board[rand_row][j] == "-":
     self.board[rand_row][j] = "Q"
   print("\nInitial Configuration:")
   printBoard(self.board)
#Main Method which will call a proper hill climbing variant based on users input
if __name__ == "__main__":
 print ("\n********** N Queen Puzzle Solution *********\n")
 print ("\nEnter the number of queens on the board (N): ")
 N = int(input())
```

```
print ("\nIteration Count Selection: \nChoose \n1. To solve the puzzle for 500 times\n2. To determine how many times you would like to run the code ")
iterationChoice = int(input())
if (iterationChoice == 1):
    iterate = 500
elif (iterationChoice == 2):
    print ("\nPlease Enter the required number of runs: ")
    iterate = int(input())
else:
    iterate = 500
    print ("\nInvalid Choice")
    print ("\nTaking the default value of 500 iterations \n")
print ("\nSearch type: \nChoose \n1. Steepest Ascent Hill Climbing\n2. Hill Climbing with Sideways Move\n3. Random-Restart Hill Climbing without Sidemove\n4. Random-Restart Hill Climbing with Sidemove")
searchStrategy = int(input())
if (searchStrategy == 1):
    searchCategory = 1
elif (searchStrategy == 2):
    searchCategory = 2
elif (searchStrategy == 3):
    searchCategory = 3
elif (searchStrategy == 4):
    searchCategory = 4
else:
    searchCategory = 1
    print ("\nInvalid Choice")
    print ("\nRunning the default approach - Steepest Ascent Hill Climbing\n")
queen_board = queenPuzzle(searchCategory, iterate, solution)
queen_board.ReportStatistics()
```

SAMPLE OUTPUT

4.1 Steepest- ascent hill climbing

```
********** N Queen Puzzle Solution *********

Enter the number of queens on the board (N):

4

Iteration Count Selection:
Choose
1. To solve the puzzle for 500 times
2. To determine how many times you would like to run the code

1

Search type:
Choose
1. Steepest Ascent Hill Climbing
2. Hill Climbing with Sideways Move
3. Random-Restart Hill Climbing without Sidemove
4. Random-Restart Hill Climbing with Sidemove

1
```

-------------------- Board configuration for the run 297 -------------------- Initial Configuration: - Q - Q - - - - - - - - Q - Q - The Number of attack pairs is 1 - Q - Q - - - - Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - *****SOLUTION FOUND*****	-------------------- Board configuration for the run 298 -------------------- Initial Configuration: - Q - - - - Q - - - - Q Q - - - The Number of attack pairs is 2 - Q Q - - - - - - - - Q Q - - - The Number of attack pairs is 1 - - Q - - Q - - - - - Q Q - - - *****NO SOLUTION*****	-------------------- Board configuration for the run 299 -------------------- Initial Configuration: Q Q - Q - - - - - - Q - - - - - The Number of attack pairs is 2 - Q - Q Q - - - - - Q - - - - - The Number of attack pairs is 1 - - - Q Q - - - - - Q - - Q - - *****NO SOLUTION*****	-------------------- Board configuration for the run 300 -------------------- Initial Configuration: - - Q Q - Q - - - - - - Q - - - The Number of attack pairs is 1 - - Q - - Q - - - - - Q Q - - - *****NO SOLUTION*****	**Iteration Count: 300** Total number of runs is 300 Total Success: 118 Success rate: 39.333333333333 % Total Fail: 182 Failure rate: 60.666666666667 % Average number of steps in success: 1.9406779661016949 Total Steps for Success: 229 Total Steps for Fail: 254 Average number of steps when failed: 1.3956043956043955

-------------------- Board configuration for the run 397 -------------------- Initial Configuration: Q - - - - Q - Q - - Q - - - - - The Number of attack pairs is 2 Q - - - - Q - Q - - - - - - Q - The Number of attack pairs is 1 Q Q - - - - - Q - - - - - - Q – The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - *****SOLUTION FOUND*****	-------------------- Board configuration for the run 398 -------------------- Initial Configuration: - - - Q - - - - - - Q - Q Q - - The Number of attack pairs is 1 - - - Q Q - - - - - Q - - Q - - *****NO SOLUTION*****	-------------------- Board configuration for the run 399 -------------------- Initial Configuration: - - - - Q - - Q - - - - - Q Q – The Number of attack pairs is 2 Q - - - - - - Q - - - - - Q Q - The Number of attack pairs is 1 Q Q - - - - - Q - - - - - - Q – The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - *****SOLUTION FOUND*****	-------------------- Board configuration for the run 400 -------------------- Initial Configuration: - - - - - - - - - - Q - Q Q – Q The Number of attack pairs is 2 - Q - - - - - - - - Q - Q - - Q The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - *****NO SOLUTION*****	**Iteration Count: 400** Total number of runs is 400 Total Success: 168 Success rate: 42.0 % Total Fail: 232 Failure rate: 57.99999999999999 % Average number of steps in success: 1.875 Total Steps for Success: 315 Total Steps for Fail: 343 Average number of steps when failed: 1.4784482758620690

-------------------- Board configuration for the run 497 -------------------- Initial Configuration: - - - - Q - - - - - Q Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - *****SOLUTION FOUND*****	-------------------- Board configuration for the run 498 -------------------- Initial Configuration: - - - - - - Q - Q Q - Q - - - - The Number of attack pairs is 2 - - - - - - Q - Q Q - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q *****NO SOLUTION*****	-------------------- Board configuration for the run 499 -------------------- Initial Configuration: - - Q - - - - - - - - - Q Q - Q The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - *****SOLUTION FOUND*****	-------------------- Board configuration for the run 500 -------------------- Initial Configuration: - Q - Q - - - - Q - Q - - - - - The Number of attack pairs is 1 - Q - Q - - - - Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - *****SOLUTION FOUND*****	**Iteration Count: 500 (default)** Total number of runs is 500 Total Success: 203 Success rate: 40.6 % Total Fail: 297 Failure rate: 59.4 % Average number of steps in success: 1.9064039408866995 Total Steps for Success: 387 Total Steps for Fail: 428 Average number of steps when failed: 1.4410774410774412

-------------------- Board configuration for the run 997 -------------------- Initial Configuration: - - Q - - - - Q Q Q - - - - - - The Number of attack pairs is 2 Q - Q - - - - Q - Q - - - - - - The Number of attack pairs is 1 Q - - - - - - Q - Q - - - - Q - *****NO SOLUTION*****	-------------------- Board configuration for the run 998 -------------------- Initial Configuration: - - Q - - - - - Q - - - - Q - Q The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - *****SOLUTION FOUND*****	-------------------- Board configuration for the run 999 -------------------- Initial Configuration: - Q - - Q - Q - - - - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q *****NO SOLUTION*****	-------------------- Board configuration for the run 1000 -------------------- Initial Configuration: - - Q Q - - - - - Q - - Q - - - The Number of attack pairs is 2 - - Q - - - - Q - Q - - Q - - - *****NO SOLUTION*****	**Iteration Count: 1000** Total number of runs is 1000 Total Success: 380 Success rate: 38.0 % Total Fail: 620 Failure rate: 62.0 % Average number of steps in success: 1.9 Total Steps for Success: 722 Total Steps for Fail: 874 Average number of steps when failed: 1.4096774193548387

4.2 Hill-climbing with sideways move

---------------------- Board configuration for the run 297 ---------------------- Initial Configuration: - - - - - Q - - Q - - Q - - Q - The Number of attack pairs is 1 - - - Q - Q - - Q - - - - - Q - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - Q - - - - Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	---------------------- Board configuration for the run 298 ---------------------- Initial Configuration: - - - Q Q Q Q - - - - - - - - - The Number of attack pairs is 2 - - - Q Q - Q - - - - - - Q - - The Number of attack pairs is 1 - - Q Q Q - - - - - - - - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	---------------------- Board configuration for the run 299 ---------------------- Initial Configuration: - - - Q Q - Q - - Q - - - - - - The Number of attack pairs is 2 - - - Q Q - Q - - - - - - Q - - The Number of attack pairs is 1 - - Q Q Q - - - - - - - - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	---------------------- Board configuration for the run 300 ---------------------- Initial Configuration: - Q - - - - - - Q - Q - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - - Q - - - - - Q Q The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	**Iteration Count: 300** Total number of runs is 300 Total Success: 300 Success rate: 100.0 % Total Fail: 0 Failure rate: 0.0 % Average number of steps in success: 2.8266666666666667 Total Steps for Success: 848

------------------------- Board configuration for the run 397 ------------------------- Initial Configuration: - - - - Q Q Q Q - - - - - - - - The Number of attack pairs is 4 Q - - - - Q Q Q - - - - - - - - The Number of attack pairs is 2 Q - - - - - Q Q - Q - - - - - - The Number of attack pairs is 1 Q - - - - - - Q - Q - - - - Q – The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 398 ------------------------- Initial Configuration: - - - Q - Q - - Q - - - - - Q - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - Q - - - - Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 399 ------------------------- Initial Configuration: - - - - - - - - Q Q - Q - - Q - The Number of attack pairs is 2 - - - - - - - Q Q Q - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 400 ------------------------- Initial Configuration: - - - - Q - - - - Q Q - - - - Q The Number of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 400** Total number of runs is 400 Total Success: 400 Success rate: 100.0 % Total Fail: 0 Failure rate: 0.0 % Average number of steps in success: 2.9225 Total Steps for Success: 1169

------------------------- Board configuration for the run 497 ------------------------- Initial Configuration: - - - - Q - - - - Q Q - - - - Q The Number of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 498 ------------------------- Initial Configuration: - - - Q - - - - - - Q - Q Q - - The Number of attack pairs is 1 - - - Q Q - - - - - Q - - Q - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q Q Q - - - - - - - - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 499 ------------------------- Initial Configuration: - - - - - Q - Q Q - Q - - - - - The Number of attack pairs is 2 - Q - - - - - Q Q - Q - - - - - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 500 ------------------------- Initial Configuration: - - - - Q - - - - Q - - - - Q Q The Number of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 500** Total number of runs is 500 Total Success: 500 Success rate: 100.0 % Total Fail: 0 Failure rate: 0.0 % Average number of steps in success: 2.994 Total Steps for Success: 1497

------------------------- Board configuration for the run 997 ------------------------- Initial Configuration: Q - Q - - - - - - Q - - - - - Q The Number of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 998 ------------------------- Initial Configuration: - - - - - - Q - Q Q - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - - Q - - - - - Q Q The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 999 ------------------------- Initial Configuration: - - Q - - Q - Q Q - - - - - - - The Number of attack pairs is 2 - - - - - Q - Q Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 1000 ------------------------- Initial Configuration: Q - - - - Q - Q - - Q - - - - - The Number of attack pairs is 2 Q - - - - Q - Q - - - - - - Q - The Number of attack pairs is 1 Q Q - - - - - Q - - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	**Iteration Count: 1000** Total number of runs is 1000 Total Success: 1000 Success rate: 100.0 % Total Fail: 0 Failure rate: 0.0 % Average number of steps in success: 2.881 Total Steps for Success: 2881

4.3 Random Restart Hill-Climbing Without Sideways Move

--------------------- Board configuration for the run 297 --------------------- Initial Configuration: - - - Q - Q Q - - - - - Q - - - The Number of attack pairs is 2 - - - Q - Q - - - - Q - Q - - - The Number of attack pairs is 2 - - - Q Q - Q - - - - - - Q - - The Number of attack pairs is 1 - - Q Q Q - - - - - - - - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 298 --------------------- Initial Configuration: - - - - Q - - - - - - Q - Q Q - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 299 --------------------- Initial Configuration: - - - - - Q Q - - - - - Q - - Q The Number of attack pairs is 3 - - - - - Q Q - Q - - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q Initial Configuration: - Q - - Q - - Q - - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 300 --------------------- Initial Configuration: Q - Q - - - - - - - - Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 300** Total number of runs is 300 Total Success: 300 Success rate: 100.0 % Total number of random restarts: 477 Average number of random restarts: 1.59 Average number of steps: 5.566666666666666

---------------------- Board configuration for the run 397 ---------------------- Initial Configuration: - - - - Q - Q - - - - - - Q - Q The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	---------------------- Board configuration for the run 398 ---------------------- Initial Configuration: - - - Q - Q - - - - Q - Q - - - Initial Configuration: - Q Q - Q - - Q - - - - - - - - The Number of attack pairs is 2 - Q Q - - - - Q - - - - Q - - - The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - Initial Configuration: - - Q - Q - - Q - Q - - - - - - The Number of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q	---------------------- Board configuration for the run 399 ---------------------- Initial Configuration: Q Q - - - - Q - - - - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q Initial Configuration: Q - - Q - - - - - - - - - Q Q - Initial Configuration: Q - - - - Q - - - - Q Q - - - - The Number of attack pairs is 2 - - - - - Q - - - - Q Q Q - - - The Number of attack pairs is 1 - - Q - - Q - - - - - Q	--------------------------- Board configuration for the run 400 --------------------------- Initial Configuration: Q - - - - - Q - - - - - - Q - Q The Number of attack pairs is 1 Q - - - - - Q - - - - Q - Q - - Initial Configuration: - - - - - - - - Q Q Q Q - - - - The Number of attack pairs is 4 Q - - - - - - - - Q Q Q - - - - The Number of attack pairs is 2 Q - Q - - - - - - Q - Q - - - - The Number of attack pairs is 1 Q - Q - - - - - - - - Q - Q - -	**Iteration Count: 400** Total number of runs is 400 Total Success: 400 Success rate: 100.0 % Total number of random restarts: 592 Average number of random restarts: 1.48 Average number of steps: 5.2975

	Initial Configuration: - - - - Q - - Q - - Q - - Q - - The Number of attack pairs is 1 - - - Q Q - - - - - Q - - Q - - Initial Configuration: Q - - - - Q - - - - - Q - - Q - Initial Configuration: - - - - - - - - - - Q Q Q Q - - The Number of attack pairs is 1 - - Q - - - - - - - - Q Q Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	Q - - - Initial Configuration: - - Q Q - - - - Q - - - - Q - - The Number of attack pairs is 1 - - Q Q Q - - - - - - - - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	

--------------------- Board configuration for the run 497 --------------------- Initial Configuration: - - - - Q - - - - - Q - - Q - Q The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 498 --------------------- Initial Configuration: - - Q - - Q - - Q - - - - - - Q The Number of attack pairs is 2 - Q Q - - - - - Q - - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q Initial Configuration: - - - - - Q - - Q - Q - - - - Q The Number of attack pairs is 2 - Q - - - - - - Q - Q - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q No attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 499 --------------------- Initial Configuration: Q - - - - Q Q - - - - Q - - - - The Number of attack pairs is 1 Q - - - - - Q - - - - Q - Q - - Initial Configuration: Q - - - - Q - - - - Q Q - - - - The Number of attack pairs is 2 - - - - - Q - - - - Q Q Q - - - The Number of attack pairs is 1 - - Q - - Q - - - - - Q Q - - - The Number of attack pairs is 1 - - Q - - - - - - - - Q Q Q - - No attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------------- Board configuration for the run 500 --------------------------- Initial Configuration: - - - - - - - - Q - Q Q - Q - - The Number of attack pairs is 2 - - - - Q - - - - - Q Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 500** Total number of runs is 500 Total Success: 500 Success rate: 100.0 % Total number of random restarts: 686 Average number of random restarts: 1.372 Average number of steps: 5.126

------------------------- Board configuration for the run 997 ------------------------- Initial Configuration: Q - Q - - - - - - Q - - - - - Q The Number of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q - Q - - - - - - - - Q - Q The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 998 ------------------------- Initial Configuration: - - - - - - Q - Q Q - - - - - Q The Number of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - - Q - - - - - Q Q The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	------------------------- Board configuration for the run 999 ------------------------- Initial Configuration: - - Q - - Q - Q Q - - - - - - - The Number of attack pairs is 2 - - - - - Q - Q Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------------- Board for 1000 --------------------------- Initial Configuration: - Q Q - Q - - Q - - - - The No of attack pairs is 2 - Q Q - - - - Q - - - - Q - - - The No of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - Initial Configuration: - Q - - - - - - Q - Q - - - - Q The No of attack pairs is 1 - Q - - - - Q - Q - - - - - - Q The No of attack pairs is 1 - - Q - Q - - - - Q - - - - - Q The No of attack pairs is 2 - - - Q Q Q - - - - - - - - Q - The No of attack pairs is 1 - - - Q - Q - - Q - - - - - Q - The No of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - The No of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 1000** Total number of runs is 1000 Total Success: 1000 Success rate: 100.0 % Total number of random restarts: 1687 Average number of random restarts: 1.687 Average number of steps: 5.77

4.4 Random Restart Hill-Climbing Without Sideways Move

--------------------- Board configuration for the run 297 --------------------- Initial Configuration: - - - - Q - - Q - - - - - Q Q - The Number of attack pairs is 2 Q - - - - - - Q - - - - - Q Q - The Number of attack pairs is 1 Q Q - - - - - Q - - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 298 --------------------- Initial Configuration: - - - - Q - Q Q - Q - - - - - - The Number of attack pairs is 2 Q - - - - - Q Q - Q - - - - - - The Number of attack pairs is 1 Q - - - - - - Q - Q - - - - Q - No of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 Q Q - - - - - Q - - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 299 --------------------- Initial Configuration: - - - - - - - - - - Q - Q Q - Q The Number of attack pairs is 2 - Q - - - - - - - - Q - Q - - Q The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - No of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - Q - - - - Q - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------------- Board configuration for the run 300 --------------------------- Initial Configuration: - Q Q Q - - - - - - - - Q - - - The Number of attack pairs is 2 - Q - Q - - - - - - Q - Q - - - The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - Q - - - - Q - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	**Iteration Count: 300** Total number of runs is 300 Total Success: 300 Success rate: 100.0 % Total number of random restarts: 0 Average number of random restarts: 0.0 Average number of steps: 2.9166666666666665

--------------------- Board configuration for the run 397 --------------------- Initial Configuration: Q - - - - - Q - - Q - Q - - - - The Number of attack pairs is 1 Q - - - - - Q - - - - Q - Q - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 Q - Q - - - - - - - - Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 398 --------------------- Initial Configuration: Q - Q - - Q - - - - - Q - - - - The Number of attack pairs is 1 - - Q - - Q - - - - - Q Q - - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - - Q - - - - - - - - Q Q Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------- Board configuration for the run 399 --------------------- Initial Configuration: - - - - - - Q Q - - - - Q Q - - The Number of attack pairs is 2 Q - - - - - Q Q - - - - - Q - - The Number of attack pairs is 1 Q - - - - - Q - - - - Q - Q - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 Q - Q - - - - - - - - Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	--------------------------- Board configuration for the run 400 --------------------------- Initial Configuration: - - - - Q Q Q - - - - Q - - - - The Number of attack pairs is 2 - - - - Q - Q - - - - Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 400** Total number of runs is 400 Total Success: 400 Success rate: 100.0 % Total number of random restarts: 0 Average number of random restarts: 0.0 Average number of steps: 2.8975

----------------------- Board configuration for the run 497 ----------------------- Initial Configuration: - - - Q - - - - - Q Q - Q - - - The Number of attack pairs is 2 - Q - Q - - - - - - Q - Q - - - The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - Q - - - - Q - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	---------------------- Board configuration for the run 498 ---------------------- Initial Configuration: - - - - - - Q - - - - Q Q Q - - The Number of attack pairs is 1 Q - - - - - Q - - - - Q - Q - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 Q - Q - - - - - - - - Q - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	---------------------- Board configuration for the run 499 ---------------------- Initial Configuration: - Q - - - - Q Q - - - - Q - - - The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - Q - - - - Q - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	---------------------------- Board configuration for the run 500 ---------------------------- Initial Configuration: - - Q - Q - - Q - - - - - Q - - The Number of attack pairs is 0 - - Q - Q - - - - - - Q - Q - - ***** SOLUTION FOUND *****	**Iteration Count: 500** Total number of runs is 500 Total Success: 500 Success rate: 100.0 % Total number of random restarts: 0 Average number of random restarts: 0.0 Average number of steps: 2.876

--------------------- ------ Board configuration for the run 997 --------------------- ------ Initial Configuration: - - - - - - - Q - - Q - Q Q - - The Number of attack pairs is 1 - Q - - - - - Q - - Q - Q - - - Number of successors with heuristic value same as that of the current state: 1 The Number of attack pairs is 1 - Q - - - - - Q - - - - Q - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------- ------ Board configuration for the run 998 --------------------- ------ Initial Configuration: - Q Q Q - - - - Q - - - - - - - The Number of attack pairs is 1 - Q - Q - - - - Q - - - - - Q - The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------- ------ Board configuration for the run 999 --------------------- ------ Initial Configuration: - - - - - - - - Q - - - - Q Q Q The Number of attack pairs is 1 - Q - - - - - - Q - - - - - Q Q The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	--------------------------- Board configuration for the run 1000 --------------------------- Initial Configuration: Q Q - - - - - - - - - - - - Q Q The Number of attack pairs is 1 - Q - - - - - - Q - - - - - Q Q The Number of attack pairs is 0 - Q - - - - - Q Q - - - - - Q - ***** SOLUTION FOUND *****	**Iteration Count: 1000** Total number of runs is 1000 Total Success: 1000 Success rate: 100.0 % Total number of random restarts: 0 Average number of random restarts: 0.0 Average number of steps: 2.875

CONCLUSION

By the implementation of this project we learnt that, finding the solution for N queens =using Steepest-ascent hill climbing method generates success only 14% of the times whereas 86% of the times it gets stuck at the local minimum. However, it takes only 4 steps on average when it succeeds and 3 on average when it gets stuck – (for a state space with 8^8 =~17 million states).

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

- Local maxima: This is a local maximum (a local minimum can be cost h =1); every move of a single queen makes the situation worse.
- Ridges: Ridges result in a sequence of local maxima that makes it very difficult for greedy algorithms to navigate.
- Plateau: A hill climbing problem might get stuck on the plateau.
    - 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.

To escape the shoulders, we implement N queens using Sideways move.

Both the methods – Steepest-ascent and Sideways move Hill climbing are incomplete, i.e., they often fail to find a goal when one exists because they can get stuck on local maxima.

Therefore, we implement the N queen problem using Random restart Hill climbing method. The algorithm conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found. It is complete with probability approaching 1, because it will eventually generate a goal state as the initial state.

REFERENCES

[1] Hill Climbing, Local Search, Wikipedia.

[2] N Queens Problem, Geeks for Geeks