Lab - 5

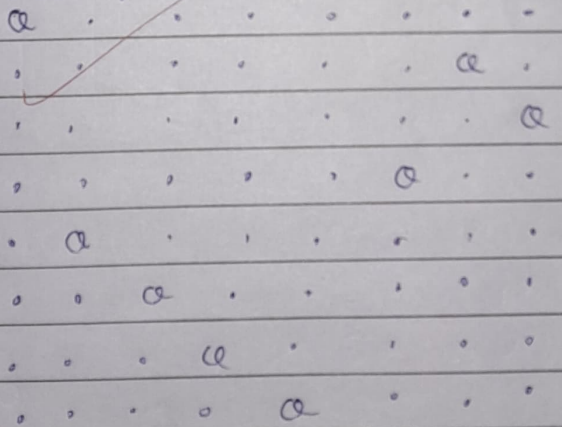
Date 15/11/2024
Page

* N Queens implementation using simulated Annealing

Algorithm:

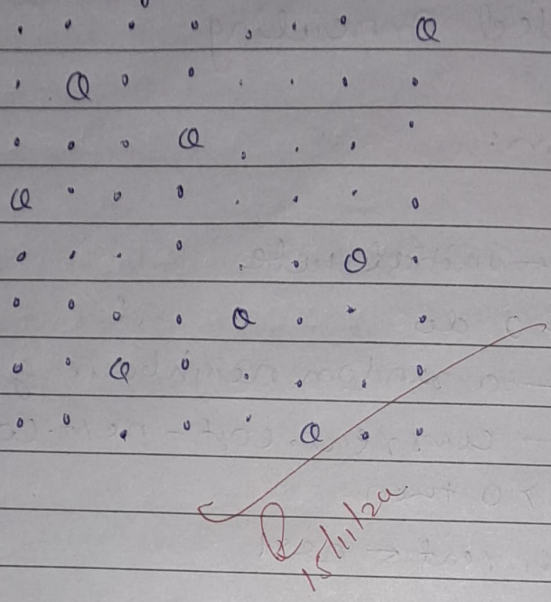
current ← initial state
while T > 0 do
  next ← a random neighbour of current
  ΔE ← current.cost - next.cost
  if ΔE > 0 then
    current ← next
  else
    current ← next with probability $P = e^{\frac{\Delta E}{T}}$
  end if
    decrease T
end while
return current.

For 8 queens by taking random inputs.
Initial state:



Date __/__/__
Page _____

Solution found!



wqdvg6yrc

December 21, 2024

[ ]:
```
print("Name: Sudarshan Komar", "USN: 1BM22CS291", sep="\n")

import random
import math

def count_conflicts(state):
    conflicts = 0
    n = len(state)
    for i in range(n):
        for j in range(i + 1, n):
            if state[i] == state[j]:
                conflicts += 1
            if abs(state[i] - state[j]) == abs(i - j):
                conflicts += 1
    return conflicts

def generate_neighbors(state):
    neighbors = []
    n = len(state)
    for i in range(n):
        for j in range(i + 1, n):
            neighbor = state[:]
            neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
            neighbors.append(neighbor)
    return neighbors

def acceptance_probability(old_cost, new_cost, temperature):
    if new_cost < old_cost:
        return 1.0
    return math.exp((old_cost - new_cost) / temperature)

def simulated_annealing(n, initial_state, initial_temp, cooling_rate,␣
 ↪max_iterations):
    state = initial_state
    current_cost = count_conflicts(state)
    temperature = initial_temp
    iteration = 0
```

```
    while iteration < max_iterations:
        neighbors = generate_neighbors(state)
        random_neighbor = random.choice(neighbors)
        new_cost = count_conflicts(random_neighbor)
        if acceptance_probability(current_cost, new_cost, temperature) > random.
 ↪random():
            state = random_neighbor
            current_cost = new_cost
        temperature *= cooling_rate
        iteration += 1
        if current_cost == 0:
            return state

    return state

def generate_random_state_with_repetition(n):
    #return random.choices(range(n), k=n)
    return random.sample(range(n),k=n)

def print_board(state):
    n = len(state)
    for row in range(n):
        board = ['Q' if col == state[row] else '.' for col in range(n)]
        print(' '.join(board))

n = 8

initial_state = generate_random_state_with_repetition(n)


initial_temp = 1000  # Higher initial temperature
cooling_rate = 0.995  # Slower cooling rate
max_iterations = 10000  # Maximum number of iterations

print("Initial State Board:")
print_board(initial_state)
print()

solution = simulated_annealing(n, initial_state, initial_temp, cooling_rate,␣
 ↪max_iterations)

if count_conflicts(solution) == 0:
    print("Solution found!")
else:
    print("No solution found within the given iterations.")
print_board(solution)
```

```
Name: Sudarshan Komar
USN: 1BM22CS291
Initial State Board:
. . . Q . . . .
. . . . . Q . .
. . . . . . . Q
. Q . . . . . .
. . Q . . . . .
Q . . . . . . .
. . . . . . Q .
. . . . Q . . .

Solution found!
. . . . Q . . .
. . . . . . Q .
. Q . . . . . .
. . . Q . . . .
. . . . . . . Q
Q . . . . . . .
. . Q . . . . .
. . . . . Q . .
```