Lab

Date 29/10/24
Page

Simulated Annealing
Solving N-Queens problem

Pseudocode:

```
function EVALUATE (state) :
    n ← length of state
    attacks ← 0
    for i from 0 to n-1:
        for j from i+1 to n-1:
            if state[i] = = state[j] or abs(state[i] - state[j])
== abs(i-j):
                attacks ← attacks + 1
return attacks  //bcoz we are maximizing

function RANDOM_NEIGHBOR(state) :
    n ← length of state
    new_state ← copy of state
    col ← random integer b/w 0 and n-1
    row_conflicts ← [ ]
    for each row from 0 to n-1 :
        if row ≠ state[col] :
            temp_state ← copy of new_state with new_state
[col] = row
            row_conflicts.append((row, EVALUATE (temp_state)))

    if row_conflicts is not empty :
        best_row ← row with max evaluation in
        row_conflicts
        new_state[col] ← best_row

    return new_state
function SCHEDULE(t) :
    return max(0.01, min(1, 1-0.005*t))
```

Date __/__/__

```
function SIMULATED-ANNEALING (state, max_iterations):
    for t from 1 to max_iteration:
        T ← SCHEDULE(t)
        if T==0:
            return state, t

        Candidate ← RANDOM_NEIGHBOR (state)
        E ← EVALUATE(Candidate) - EVALUATE(state)
        if E > 0 :
            state ← candidate
        else:
            prob ← exp(E * 1.5))
            if random() < prob:
                state ← candidate
        :
        if EVALUATE(state) == 0
            print("Global max found at, t)
            return state, t

    print("Reached local maximum found at iteration",
        max_iteration)
    return state, max_iterations

function SOLVE-N_QUEENS(n):
    initial_state ← array of n random integer
    b/w 0 and n-1
    result, iteration ← SIMULATED_ANNEALING
    (initial_state, max_iteration = 5000)
    return result, iterations

Function PRINT_BOARD (state):

    for row from 0 to n-1
        line ← " "
```

```
for col from 0 to n-1:
    if state[col] == row:
        line = line + "Q "
    else:
        line = line + ". "
    print(line)

// Main code
N = user input "Enter no of queens (N): "
SOLUTION, iteration = SOLVE_N_QUEENS(N)
print("sol for" N, "queen found at", iteration,
    "iteration")
PRINT_BOARD(solution)
print("Final evaluation", EVALUATE(solution))
```

Output

Enter number of queens (N): 8

Global maximum (no attacks) found at iteration 902

Solution for 8 queens in 902 iterations:

. . . . Q . . .

Q . . . .

. . . Q . . . .

. . . . . Q . .

. . . . Q

. . Q . . . . .

. . . . . Q . .

final evaluation (objective function value): 0