15/09/25

Lab - 5

Implementing Hill climbing search algorithm to solve N Queens problems

Input :

| | | | |
|---|---|---|---|
| | | | Q |
| | Q | | |
| | | Q | |
| Q | | | |

① - - - Q
- Q - -
- - Q -
Q - - -

(2,2) <-> (3,3) - ①
(1,4) <-> (4,1) - ②
cost = 2

② - - - -
- Q - Q
- - Q -
Q - - -

(2,2) <-> (2,4) - ①
(2,2) <-> (3,3) - ②
(3,3) <-> (2,4) - ③
cost = 3

③ - - Q -
- Q - Q
- - - -
Q - - -

(1,3) <-> (2,2) - ①
(2,2) <-> (2,4) - ②
(1,3) <-> (2,4) - ③
cost = 3

④ - - - -
- - - Q
- Q Q -
Q - - -

(2,4) <-> (3,3) - ①
(3,2) <-> (3,1) - ②
(3,2) <-> (4,1) - ③
cost = 3

⑤ - Q - -
- - - Q
Q - - -
- - Q -

cost = 0

Algorithm of hill climbing

① start with one queen in each column (initial board)
② Calculate cost (G) = no of attacking queen pairs
③ for each column move the queen to every other row

and ~~and~~ compute next cost.

4) choose the move that gives the lowest cost (best neighbour)

5) If best cost < current cost, move queen there and repeat step 2

6) if no neighbour has lowest cost, stop

output:

initial stack

1)
```
- - - Q
- Q - -
- - Q -
Q - - -
```
cost = 2

2)
```
- - - Q
- Q - -
- - - -
Q - Q -
```
cost = 2

3)
```
- - - Q
- Q - -
Q - - -
- - Q -
```
cost = 1

4)
```
- Q - Q
- - - -
Q - - -
- - Q -
```
cost = 1

5)
```
- Q - -
- - - Q
Q - - -
- - Q -
```
cost = 0

sol found in 4 steps

Simulated Annealing

Algorithm

current ← initial state

T ← a large positive value

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 $P = e^{\Delta E/T}$

end if

decrease T
end while
return current

Output
The best position found is = [0 8 5 2 6 3 7 4]
The number of queues that are not attacking each other is : 8

22/5/25