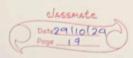
29-10-2024 WEEK- 6

Analyze and implement N-Queens problem using stimulated Annealing technique

ALGORITHM / PSEUDOCODE -

Analyse and implement in queens problem using Analyse and implement in queens problem using Analyse and implement in queens between a simulated Annualing technique. ALGORITHM: FUNCTION GenerateTisHalstake(n): FUNCTION (acculate(o)t (state): n=length of state cost = 0 por from 0 to n-1; for if from it i to n-1: 1f state [i] == state [i] or abs(state(i) - state(j) == abs(i) - i); Increment cost by if return cost FUNCTION Generate Neighboure (state): n = length of state new-state = copy of state row-to-state = random integer between 0 and n-1 new-state (row-to-change 7 = random integer between 0 and n-1 return new-state FUNCTION Simulated Annualing (n, integer temperature, cooling-rate): current-state = Generate Initial state (n) werent - cost = Calculate (ost current-state)	Date 29/10/24 Page 16	113
ALGORITHM: FUNCTION GENERALE CON: RETURN A SILL OF N ANDOM INTEGERS BETWEEN D OND N-1 RETURN A SILL OF N ANDOM INTEGERS BETWEEN D OND N-1 FUNCTION CALCULATECOST (STATE): N=LENGTH OF STATE COST = 0 FOR FORM PH 1 TO N-1: 1 for j from PH 1 TO N-1: 1 f STATE F: 1= STATE F' j OR abs (STATE PT) == abs (P-f): INCREMENT COST by 3 return cost FUNCTION GENERALE Neighbour (STATE): N=LENGTH OF STATE N=LENGTH OF STATE TOW_TO_STATE = COPY OF STATE TOW_TO_STATE = TONDOM INTEGER BETWEEN O AND N-1 NEW_STATE COPY TO CHANGE T= TONDOM PRIEGER BETWEEN O and N-1 return new-STATE FUNCTION Simulated Annualing (N, INTEAL-temperature, cooling-rate): CUrrent_STATE = Generale Initial STATE (N)		
ALGORITHM: FUNCTION GENERALE CON: RETURN A SILL OF N ANDOM INTEGERS BETWEEN D OND N-1 RETURN A SILL OF N ANDOM INTEGERS BETWEEN D OND N-1 FUNCTION CALCULATECOST (STATE): N=LENGTH OF STATE COST = 0 FOR FORM PH 1 TO N-1: 1 for j from PH 1 TO N-1: 1 f STATE F: 1= STATE F' j OR abs (STATE PT) == abs (P-f): INCREMENT COST by 3 return cost FUNCTION GENERALE Neighbour (STATE): N=LENGTH OF STATE N=LENGTH OF STATE TOW_TO_STATE = COPY OF STATE TOW_TO_STATE = TONDOM INTEGER BETWEEN O AND N-1 NEW_STATE COPY TO CHANGE T= TONDOM PRIEGER BETWEEN O and N-1 return new-STATE FUNCTION Simulated Annualing (N, INTEAL-temperature, cooling-rate): CUrrent_STATE = Generale Initial STATE (N)	WEEK-5 Nowens problem using	
RETURN GENERATE THAT STAKE CON: RETURN A STATE COST = 0 FUNCTION CALCULATECOST (STAKE): N = Length of state COST = 0 FOR I from it 1 to n-1: 1f State [i] == state [i] or abs(state [i] - State [i]) == abs(f-f): Increment cost by 3 return cast FUNCTION Generate Neighboure (state): n = Length of state new_state = copy of state row_to_state = copy of state row_to_state [row_to_change] = random integer between 0 and n-1 return new_state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): Current_state = Generate Initial state (n)	Analyse and impurious technique.	//
FUNCTION Generate Interest interest between a and n-1 FUNCTION (alculatecost (state): n=length of state cost = 0 for i from it 1 to n-1: 1f state ril== stateril or abs(stateril-stateril)== abs(r-1): therement cost by i return cast FUNCTION Generate Neighbour (state): n=length of state new_state = capy of state row_to_stage = random integer between 0 and n-1 new_state (row_to_change 7= random frieger between 0 and n-1 return new_state FUNCTION Simulated Annealing (n, inteal-temperature, cooling-rate): Current_state = Generate Initial state (n)	shmularea minetary	1
FUNCTION Generate Interest Entres between 0 and n-1 FUNCTION (adulatecost (state): n=length of state cost = 0 for if from it 1 to n-1: 1f state ril== state ril or abs(state ril-state ril)== abs(r-1): for increment cost by i return cast FUNCTION Generate Neighbour (state): n= length of state new_state = copy of state row_to_stage = random integer between 0 and n-1 new_state (row_to_change 7= random integer between 0 and n-1 return new_state FUNCTION Simulated Annealing (n, inteal-temperature, cooling-rate): Current_state = Generate Initial state (n)	ALGORITHM:	
FUNCTION CALCULATECOST (STATE): n=length of state cost = 0 for if from let to n-1: for if state ril== state ril or abs (state ril-state ril)== abs cl-f): increment cost by i return cost FUNCTION Generate Neighbour (state): n=length of state new-state = copy of state row_to_state = copy of state row_to_state (row_to_change l= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, instal-temperature, cooling-rate): current_state = Generale Initial state (n)	and the CM):	1
FUNCTION CALCULATECOST (STATE): n=length of state cost = 0 for if from it 1 to n-1: for if state ril== state ril or abs (state ril-state ril)== abs (r-f): increment cost by if return cost FUNCTION Generate Neighbour (state): n=length of state new-state = copy of state row_to_state = copy of state row_to_state (row_to_change l= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, initial-temperature, cooling-rate): current_state = Generale Initial state (n)	Return a list of a random integers between o oran-	1
n=length of state cost = 0 for ifrom 0 to n-1: for j from it 1 to n-1: 1f state r:1== state r; 1 or abs (state r:1-state r; 1)== abs (P-j): Increment cost by i return cost FUNCTION Generate Neighbour (state): n= length of state new-state = copy of state row_to_state = random integer between 0 and n-1 new-state (row-to-change 7= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, inteal-temperature, cooling-rate): Current_state = Generale Initeal state (n)		1
cost = 0 for i from 0 to n-1: for j from it 1 to n-1: 1f state ril== state ril or abs (state ril-state ril)== abs (r-j): Increment cost by i return cost FUNCTION Generate Neighbour (state): n= length of state new_state = copy of state row_to_stage = vandom integer between 0 and n-1 new_state (row_to_change l= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current_state = Generale Inirfal state (n)	FUNCTION Calculatecost (State):	1
for i from it 1 to n-1: for i from it 1 to n-1: if state ril== state ril or abs (state ril-state ril)== abs (r-1): increment cost by i return cast function Generate Neighbous (state): n = length of state new-state = copy of state row_to_state [row_to_change l= random integer between 0 and n-1 new-state (row_to_change l= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): wrent-state = Generale Inital state (n)	n=length of state	1
for i from it 1 to n-1: If state ril== state ril or abs(state(il-state(il)== abs(il-i)): Increment cost by i return cost FUNCTION Generate Neighbous (state): n= length of state new-state = copy of state row_to_state = random integer between 0 and n-1 new-state (row_to_change l= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, integer temperature, cooling-rate): current-state = Generate Initial state (n)		-
if state [:] = state [:] or abs (state[:] - State[:]) == abs (!] - [:]: Increment cost by 3 return cost FUNCTION Generate Neighbous (state): n = length of state new_state = copy of state row_to_state = rondom integer between 0 and n-1 new_state (row_to_change 7 = random integer between 0 and n-1 return new_state FUNCTION Simulated Annealing (n, integer temperature, cooling_rate): current_state = Generate Initial state (n)		1
return cost return cost return cost FUNCTION Generate Neighbous (state): n = length of state new-state = copy of state row_to_stage = vandom integer between 0 and n-1 new-state (row-to-change 7= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, integer temperature, cooling-rate): where the state = Generate Initial state (n)	for g from it 1 to n-1:	+
return cost return cost return cost FUNCTION Generate Neighboux (state): n = Length of state new-state = copy of state row_to_state = copy of state row_to_state (row_to_change 7= random integer between o and n-1 return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current_state = Generate Initial state (n)	if state ril== state ril or abs (state [] - state []	+
return cost FUNCTION Generate Neighboux (state): n = Length of state new_state = copy of state row_to_state = random integer between 0 and n-1 new_state (row_to_change 7= random integer between 0 and n-1 return new_state FUNCTION Simulated Annealing (n, initial_temperature, cooling_rate): current_state = Generate Initial state (n)	abs(F-f).	+
FUNCTION Generate Neighbour (state): n = length of state new-state = copy of state row_to_stage = random integer between 0 and n-1 new-state (row_to_change 7 = random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, inteal-temperature, cooling-rate): current_state = Generate Initeal state (n)		+
n= length of state new-state = copy of state row_to_stage = random integer between 0 and n-1 new-state (row_to_change 7= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current_state = Generale Initral state (n)	return cast	+
n= length of state new-state = copy of state row_to_stage = random integer between 0 and n-1 new-state (row_to_change 7= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current_state = Generale Initral state (n)	0 702 (8.8-9.11	-
new-state = copy of state row_to_stage = random integer between 0 and n-1 new-state (row-to-change 7= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, integer temperature, cooling-rate): current_state = Generate Initeal state (n)	FUNCTION Generale Neighbour (state):	-
row_to_stage = random integer between 0 and n-1 new_state (row_to_change 7= random integer between 0 and n-1 return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current_state = Generale Initral state (n)	n = length of state	-
new-state (row-to-change 7= random enteger between o and n-1 return new-state FUNCTION Simulated Annealing (n, in Hal-temperature, cooling-rate): current-state = Generate Initeal state (n)	new-state - copy of state	
return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current-state = Generate Initral state (n)	row_to_stage = random integer between 0 and n-	-1
return new-state FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current-state = Generate Initral state (n)	new-state (row-to-change 7-random integer	11-10
FUNCTION Simulated Annealing (n, intral-temperature, cooling-rate): current-state = Generate Initral state (n)	between o and n-1	
cooling_rate): current_state = Generale Initial state (n)	return new-state	
cooling_rate): current_state = Generale Initial state (n)	N-100 [80 2 12 12 12 12 12 12 12 12 12 12 12 12 1	. 1
current-state = Generale Initral state (n)	FUNCTION Simulated Annealing (n, intral-temperatu	re,
	cooling-rate):	
wrient-cost = calculate (ost current-state)	current-state = Generate Initial state (n)	
	wrient-cost = calculate (ost current-state)	E.
temperature = intial - temperature	temperature = intial - temperature	
The state of the s	The state of the s	



STATE SPACE TREE. neighbour_cost < current-cost: IF current-state = neighbour-state current- obst = neighbour-cost ELSE : delta-cost= resqubour-cost - current-cost Probability = EXPC-delta-cost I temperature) If random number between 0 and 12 probability: current - state = neighbour - state current-cost = nelghbour-cost temperature = temperature × cooling-rate return aureint-state, current-cost n= 8 Initial-temperature = 100 cooling-rate= 0.95 solution, cost = Simulated Annealing (n, initial-temperature, (ooung-rare) It cost == 00 .PRINT " Solution found:" PRINT SOLUTION ELSE! PRINT Lould not find a perfect solution. PRINT "Best solution found:" PRINT solution PRINT " COST: ", COST

CODE-

```
import random
import math
def generate initial state(n):
 """Generates a random initial state for the N-Queens problem."""
 return [random.randint(0, n - 1) for in range(n)]
def calculate cost(state):
 """Calculates the number of conflicts in the current state."""
 n = len(state)
 cost = 0
 for i in range(n):
  for j in range(i + 1, n):
   if state[i] == state[j] or abs(state[i] - state[j]) == abs(i - j):
     cost += 1
 return cost
def generate neighbor(state):
 """Generates a neighboring state by randomly moving a queen."""
 n = len(state)
 new state = state[:]
 row to change = random.randint(0, n - 1)
 new state[row to change] = random.randint(0, n - 1)
 return new state
def simulated annealing(n, initial temperature, cooling rate):
 """Solves the N-Queens problem using simulated annealing."""
 current state = generate initial state(n)
 current cost = calculate cost(current state)
 temperature = initial temperature
 while temperature > 1:
  neighbor state = generate neighbor(current state)
  neighbor cost = calculate cost(neighbor state)
  if neighbor cost < current cost:
   current state = neighbor state
   current cost = neighbor cost
  else:
```

```
delta cost = neighbor cost - current cost
   probability = math.exp(-delta cost / temperature)
   if random.random() < probability:
     current state = neighbor state
     current cost = neighbor cost
  temperature *= cooling_rate
 return current state, current cost
n = 8
initial temperature = 100
cooling rate = 0.95
solution, cost = simulated annealing(n, initial temperature, cooling rate)
if cost == 0:
 print("Solution found:")
 print(solution)
else:
 print("Could not find a perfect solution.")
 print("Best solution found:")
 print(solution)
 print("Cost:", cost)
```

OUTPUT -

```
Could not find a perfect solution.

Best solution found:

[0, 7, 4, 2, 5, 6, 4, 0]

Cost: 3
```

STATE SPACE TREE -

																(5	Page _)वी ।त ।ह		0	
1	STAT	€ .	SPA	CE	т																	-
						_		_		-	-			170	1.9.5	15/2	1.76		88:			1
						-9	-	Q			-	162	-		1,11		-					
					143		9	200	4 10	Q	+	200	37	_ 1		- 40						
	ller.					9	+				+	-					12	413	3			
	10.00	100	2 4 1 1			100		1	0	110	10	- 1	1000	3								
	1	212	-		1	-		1		T	-		-	-	1	7				I	-	
1000		150	-	3		3	1	75	2	1	3	1	T	2	2	1	3	3	2	0:	2 1	-
Q.	2	Q			R				-	10		0	-	1		+	-	+		0		
	100	01					Q	120		2		0	100	0	1	1	Q		-	1		0
		~ 3		0				100		39		-		a	Q	1	100	1 2.5	0.		0	
cost =	19			Q	30		-1	1	31	_	101	10	MAG	144		10	tu	100				
	-			031	t = 3	5			60	St= C	1			C	054	- 4	-1		co	7 =	3	
013	3 /	-	V	_							_	T				8	J	+		>		
0		3	1 .	5	1	-2	10	2		300	2	-	-	The second	7 7		No		0			
		2			0				0	91	1	+	JE		1	3.15	Þ9		2			
I Q	0		Q		Q		Q		Q		_		9	Q			0	2				
	12/1	11/	0		40	Q	A.	Q	3.	20	113	4	72	20	14	Q	7.4	100				
101118	13/0	2	_	2							1			13	· ·					1		
cost=1.		Cos	+ = 2	2		cos	+ 1	4			cos	かっ	3		3	103	3=	H			M	
1	4		J	,	133	16	CC h	Jol	1	on	100	03			di	54	4					
023	1	1	0 1	2	1		Į,	0	1	3	0	20	12	-	243	à	1		19.11			, Ti
2		6	5			1		0			0				HI		32	13		13 1		
mand	2	1	0	34	0	1			0	OE ON	0	0.4		-32	ir.					i d		
8			11	Q		400				200			F					27 mil		1		
10							-	1												37-1		A
		_					-	1			11					_	194			4,		
ost=2		C	teost	= (-			a	25+	- 2	13	4.9	1	1	PU	01	9.9					
		- 7																		7		
kerul	t =	2	2 !	3	1		ce	25	+=	0.	H			()						1		
											1 -				-					81		