⇒ Two aspect of search
① ~~Hill climbing~~ Exploitation of gradient (Hill Climbing)
② Exploration (Tabu Search).

Page No.
Date

Let's go towards Stochastic methods or randomized methods

Stochastic / Randomized methods

↳ Focus is still on exploration. How can we make the search to go into newer areas.

↳ Give some randomness to search space.
Allow algo. to go to d/f areas.

eg → Random Walk

just take one step in any dir^n ← n ← random-neighbour (c)

⇒ Not a good way ∵ not systematic

⇒ Hill climbing is extreme of Exploitation & Random walk is extreme of Exploration.

We like algo. which are in-b/w essentially.

①⇒ Make a random move with a probability ∝ movement in eval(n)

for a node (c) choose a random node n.

(c) —random→ (n)

but move from it with prob. how better that move is from (c.)

How better candidate (n) is as compared to (c)

If $n > c$ then greater prob. of move
If $n < c$ then lesser prob. of move

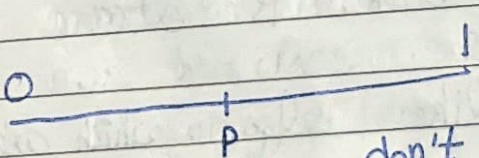
eval(n) - eval(c)

more +ve ↓ better it is ⇒ Greater probability.

even if -ve ⇒ not discarded

∴ I want to bias my search towards better moves but I don't want to stop exploration (bad move ⇒ moves which decrease eval fⁿ value)

⇒ Say probability p
How to make move
choose a random no b/w 0 & 1



If no. > p ⇒ don't make move
If no. < p ⇒ ~~don't~~ make move

I need a way of computing p as a fⁿ of [eval(n) - eval(c)] so that we take this req. behaviour.

Page No.
Date

Optimization - I (Simmulated Annealing)

⇒ Local optima problem
We want global optima
⇒ Hill climbing → exploitation

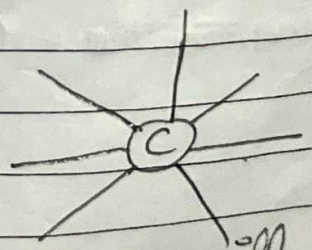
⇒ We want optimization everywhere.
eg → materials → We want them to have minimum energy. (E↓ Properties → good).
This is achieved through controlled annealing.

→ Random Walk → Exploration

⇒ We want Exploration & Exploitation (mix of both)
↓ Leads to Global optima
↓ Leads to local optima

⇒ We want to make a ~~for~~ move with a certain probability i.e it is controlled in such a fashion that if the ~~move~~ move is a good move, the probability is high and if the ~~move~~ move is not a good move, then the probability is low.

eval(c) → c - current node
eval(n) → n - next node



look

⇒ Now, we are taking a neighbour & we will either move to it or not move to it.

Also, In Hill climbing we were evaluating all neighbours but here we will only evaluate one randomly &

$$\frac{1}{1+e^{-\Delta E}}$$

$$\Delta E = eval(n) - eval(c)$$

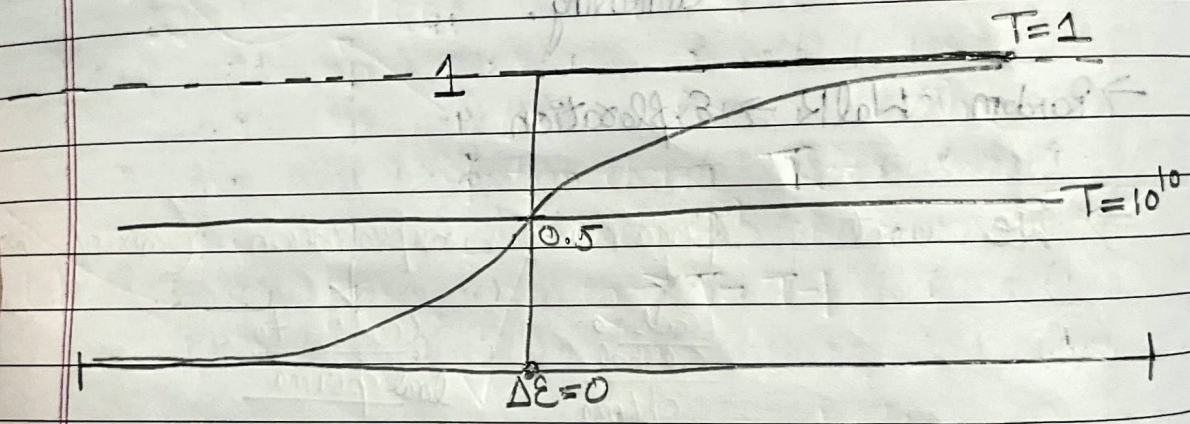
for maximization

$$eval(n) > eval(c)$$

We want our Prob. to be influenced by ΔE

$\Delta E \uparrow \Rightarrow$ Prob of move $\uparrow$

& Vice-Versa



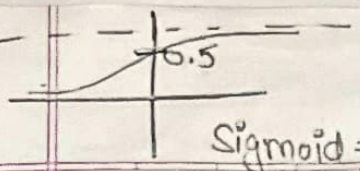
$\Rightarrow$ We will use sigmoid $\neq^n$ here

$$P(c,n) = \frac{1}{1+e^{-\Delta E}}$$

↳ Now, We have that ΔE influences Prob. & How

↳ Now, Question is How much influence?

↳ $$P(c,n) = \frac{1}{1+e^{-\Delta E/T}}$$



Sigmoid fⁿ

Stochastic Hill-climbing

Page No.
Date

n ← random_neighbour(c)
evaluate ΔE
Move with Prob = $\frac{1}{1+e^{-\Delta E/T}}$

This alg has a tendency to go up.

⇒ effect of ΔE ⇒ T = 10, eval(c) = 107

	eval(n)	−ΔE	$e^{-\Delta E/T}$	$P = \frac{1}{1+e^{-\Delta E/T}}$
①	80	27	14.88	0.06
②	100	7	2.01	0.33
③	107	0	1.0	0.5
④	120	−13	0.27	0.78
⑤	150	−43	0.01	0.99

⇒ This table shows how stochastic hill climbing corresponds to diff values of ΔE.

⇒ Consider eval(n) & eval(c)
= 120 = 107
we will see how T affects Prob.

T = Temperature

	T	$e^{-13/T}$	P
Hill climbing ← (i)	1	0.000002	1.0
(ii)	5	0.074	0.93
(iii)	10	0.27	0.78
(iv)	20	0.52	0.66
(v)	50	0.77	0.56
Random Walk ← (vi)	10^{19}	0.9999	0.5

if it sees a better move it will make it

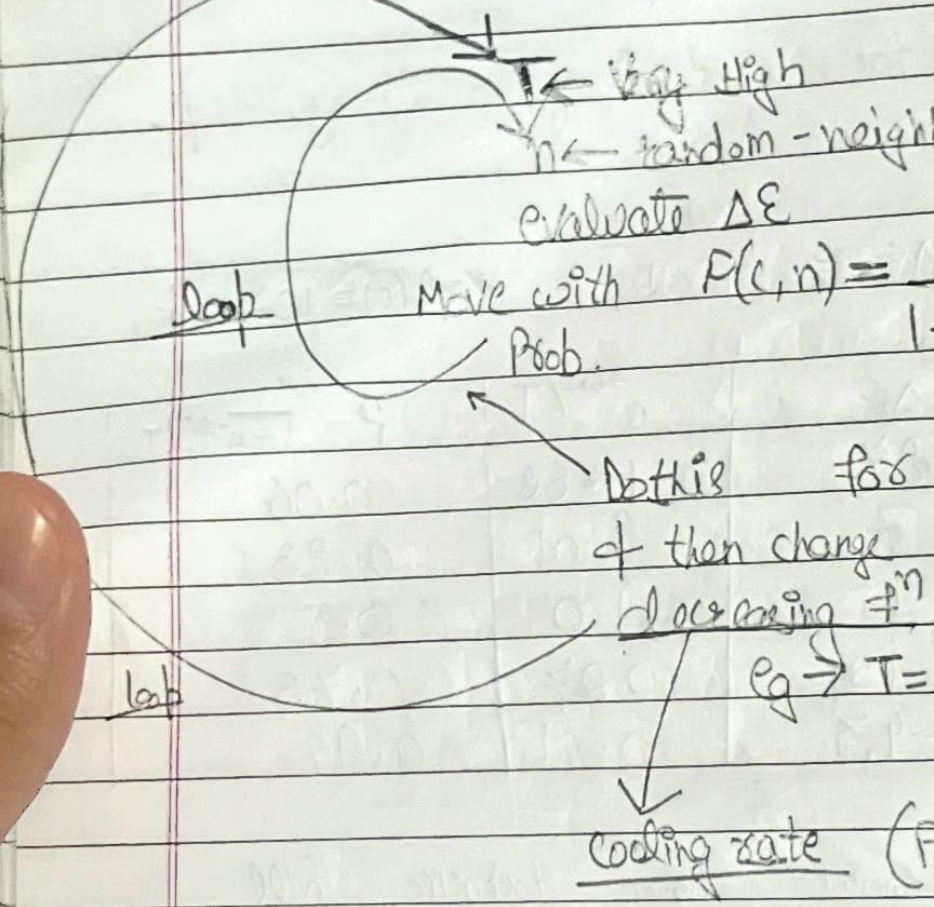
so we can control behavior of our Prob fⁿ by controlling T.

If more random → T↑ (explore more)

If follow gradient → T↓ (exploitation)

⇒ **Simulated annealing algo:-**

⇒ Instead of making a choice of value T, we follow what is done in physical world



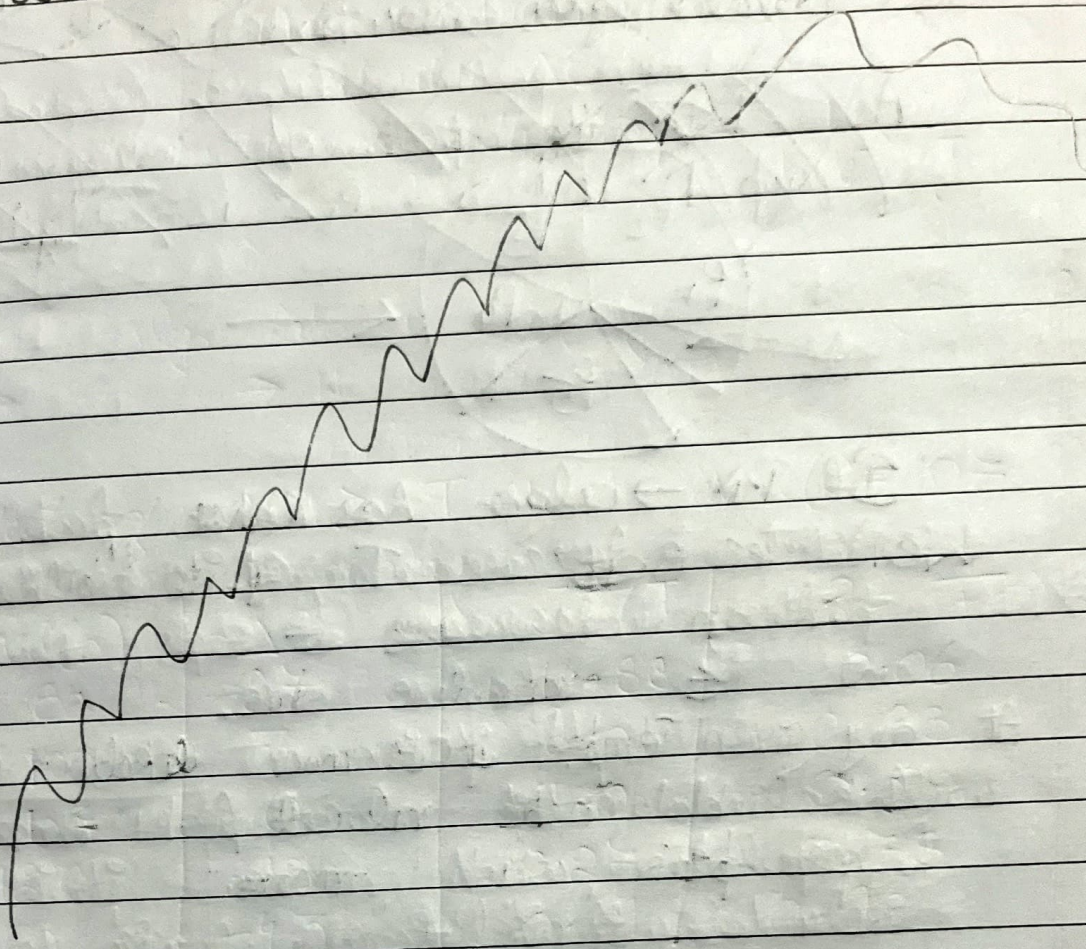
This Algo is known as Simulated annealing.

Here, initially it is behaving randomly & later on follow heuristic fn.

⇒ In many problems this gives close to optimal soln

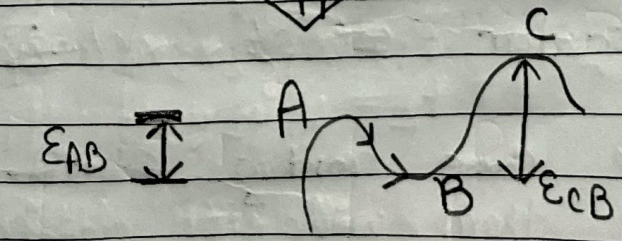
Page No.
Date

Say our surface looks like this :-



H.C would get stuck
Simulated annealing will go V. close to global optima

Snapshot of small reg.



⇒ We don't want to get stuck at A but reach at C

To go from A to C, it has to overcome an energy gap of ε_{AB}.
To go from C to A, it has to overcome an

If T↑ → algo has higher prob to go down
As we decrease T, its ability to go down decreases

At any given Temp, it is more likely that it will climb down of local peak than from a higher peak which means it is more likely to travel from lower to higher peak.

Iterative Hill Climbing
(This works for Solution space)

Choose a random set of starting point
H.C

⇒ If we choose random starting point then one of them will hit global maximal we will get answer
⇒ kind of Hill search.

Page No.
Date

Optimization II

Genetic Algorithms (GAs)

"Whatever persists, persists"

↳ GAs work with population of Candidates.
↓
encode solution as a ~~string~~

We will work with population of randomly generated candidates & then try to mix up the strings to produce new candidates which we hope to be better.

⇒ We devise fitness f^n

$$f(n) \;||\; eval(n) \;||\; h(n)$$

∴ With every candidate we have a fitness value, which tells us how good the candidate is.

⇒ GA algo has 3 steps or

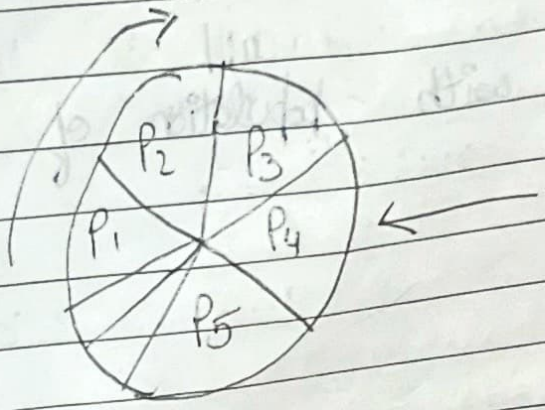
① SELECTION & fitness

(Candidates (Population)) { P_1, P_2, P_3, ..., P_n

We will clone candidates based upon fitness i.e we make more copies of fit candidates & less copies of unfit candidates.

Page No.
Date

Think of it as a wheel consisting of sectors representing each candidate & each sector $\propto$ fitness ($\propto$ angle)



We rotate it randomly & we have some pointer, wherever it stops, that candidate gets to reproduce once.

We do this n times & generate a population of n new candidates
i.e. Say $P_1', P_2', P_3' \dots P_n'$

② Crossover →

We randomly pair up elements of new population & we randomly mix up their genes.

$P_1 = x_1 x_2 x_3 x_4 \dots x_9$
$P_2 = y_1 y_2 y_3 y_4 \dots y_9$

x_i & y_i are genes

We mix up their genes & generate new children.

$C_1 = x_1 x_2 x_3 x_4 y_5 y_6 \dots y_9$

$C_2 = y_1 y_2 y_3 y_4 x_5 x_6 x_7 x_8 x_9$

This process of mixing of genes is called Crossover. (Single point Crossover)

We will choose a pointer (||) & everything from one side will go to one child & else to another child.

We put this in loop [Selection → Crossover]

③ Mutation → (Random rare event) (once in a while) Change one gene randomly in one individual

From this we get new population

P_1'', P_2'' ..... P_n'' (We got this by crossover & mutation)

Say these are sorted by fitness then we will take most fit K element where K<n & replace it in the original population with least fit K elements.

⇒ We will use Genetic algo to solve TSP.

In TSP, we can't do single point crossover or a city might come twice (loop)

eg ⇒ P_1 = (2 3 7 | 1 4 9 8 5)
(1 2 3 4 | 5 8 7 6 9)

⇒ Population based methods for optimization

Here, we work with a population of candidates

⇒ We saw Genetic algos (or Evolutionary algos)

↓

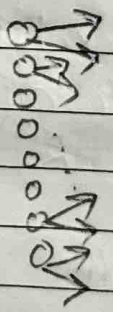
Bottom Up | Emergent Systems

↳ Collection of simple elements ↓ More complex behaviour

In Genetic algos

↳ We have Candidate sol^n → Components → Chromosomes

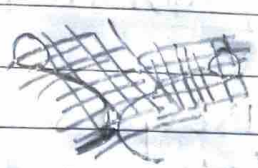
① Selection →



Algo ⇒ Candidates are allowed to reproduce based on some fitness values. (Roulette wheel mechanism to reproduce) ⇒ Crossover (Randomly pair new population & allow crossover operation with)

Page No.
Date

③ Mutation (Rare)

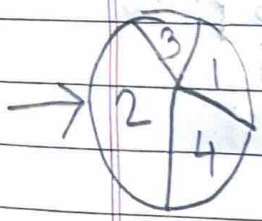


Example :- Our task is to find largest no. that we can represent using 5 bits.

⇒ Small population of size 4.

We have 5 bit vector, (say we interpret as binary #s)

	x	$f(x)=x^2$		Probability value
1	01101	13	169	0.14
2	11000	24	576	0.49
3	01000	8	64	0.06
4	10011	19	361	0.31
			= 293 ← Average fitness	



Here, our fitness ($f(x)$) influences selection process

↳ Chances of selection ∝ fitness

We will spin the wheel 4 times

∴ Expected value = $\mathcal{E} = 4 \times P$

Say after we spin the wheel we get ⇒ values as

1
2
0
1

∴ After selection we have population.

new population ⇒ 1 copy of 1, 2 copies of 2, 1 copy of 4, 0 copy of 3

01101
11000
11000
10011

Page No.
Date

Say we pair [01101 & 11000] & [11000 & 10011]

→ Crossover site

01101
11000
↓
01100
11001

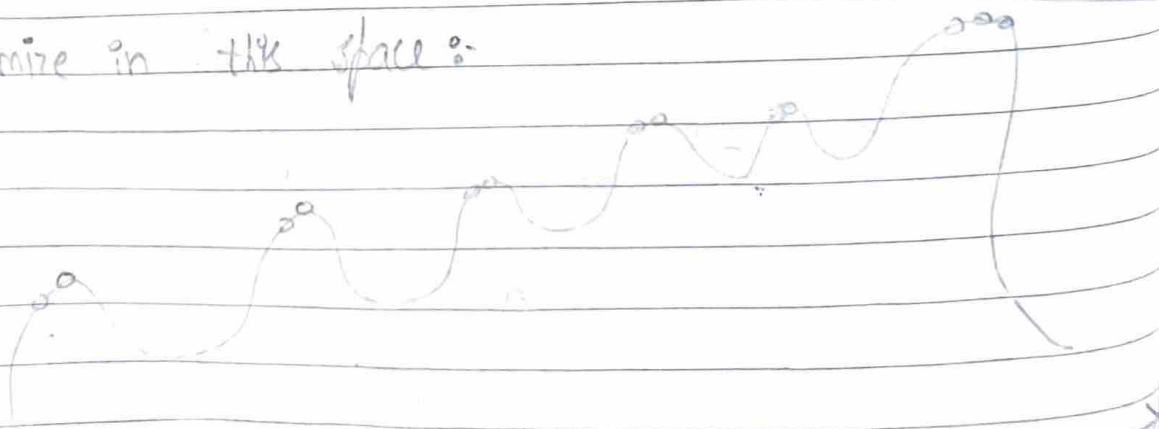
11000
10011
↓
11011
10000

		$f(x)=x^2$	P	E
01100	→ 12	144	0.08	0.3
11001	→ 25	625	0.35	1.4
11011	→ 27	729	0.426	1.66
10000	→ 16	256	0.145	0.58

Avg = 439 fitness

After one cycle average fitness improved 293 → 439

optimize in this space :-



Initially pop. spread over this space
After doing GA the pop will concentrate
peaks (& one of them is sol^n)

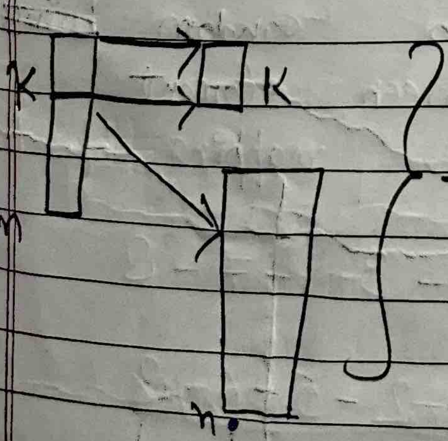
→ observation :-
likely that ① will be left out (when done roulette wheel sel P↓)

& ① is only with middle bit as 1
Hence if we don't consider it, we will never get middle bit as 1.

⇒ How to prevent this from happening?

Here, parameters are encoding & evaluation
↓
fitness f^n

⇒ There is hope that mutation will toggle the middle bit but generally, we would require a large population size.



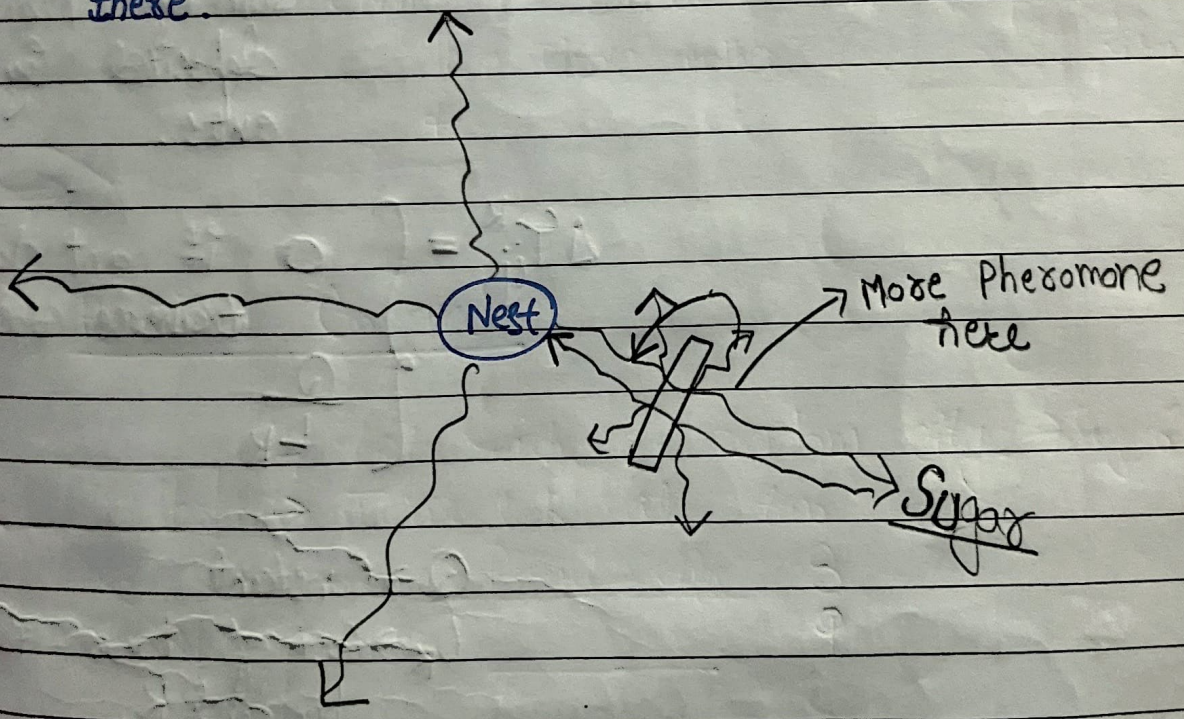
Choose best n.

ANT Colony OPTIMIZATION → (Swarm intelligence) (ACO) (cultural algos).

Ants interact with each other through a process of symbols → memetic interaction
↓
through symbols

They do by process of PHEROMONE TRAILS.

They mark out path and follow from there.



↳ ACO for TSP

Let there be n cities & m ants

Each Ant constructs a tour in greedy fashion.

↳ Also deposit Pheromone on tour it constructs

τ_{ij} → Pheromone on link ith to j.

$$\tau_{ij}(t+n) \leftarrow (1-\lambda)\,\tau_{ij}(t) + \Delta\tau_{ij}$$

λ = Coefficient of evaporation

$(1-\lambda)\,\tau_{ij}(t)$ → old pheromone after evaporation

$\Delta\tau_{ij}$ → new pheromone increment amount

$$\Delta\tau_{ij}(t+n) \rightarrow \sum_{k=1 \text{ to } n} \Delta\tau_{ij}^{k}$$

Sum of pheromone deposited by K ants.

$$\Delta\tau_{ij}^{k} = \begin{cases} 0 & \text{if ant doesnot traverse edge } i,j \\ \dfrac{Q}{L_k} & \end{cases}$$

Q → Constant

L_k → Cost of tour found by ant K.

∴ Amount of Pheromone $\propto \dfrac{1}{\text{Cost of tour}}$

Construction of tour by ants :-

Ant at city i moves to j with prob.

$$P \propto \frac{[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum [\tau_{il}]^{\alpha}\,[\eta_{il}]^{\beta}}$$

Page No.
Date

α, β → Constants for controlling influence of factors.

τ_{ij} = Amount of Pheromone at edge $i-j$.

η_{ij} = Visibility $\propto \frac{1}{\text{cost of edge } ij}$

Δk = allowed cities Set (No cycle)

⇒ Each Ant construct a tour ⇒ Based on tour it deposits Pheromone on every edge it has moved on.

At end we update Pheromone at every edge of graph.

∴ Ants will move towards more pheromone & less cost edge.

so optimal Soln