CHAPTER 6

Unit VI

Swarm Intelligence

Syllabus Topic : Particle Swarm Optimization(PSO) Algorithm

6.1 Particle Swarm Optimization Algorithm

- Particle Swarm Optimization (PSO) is inspired from the nature, social behavior and dynamic movements with communication of insects, birds and fish.
- In 1986, Craig Reynolds described this process in three simple behaviors: separation (avoid crowding local flockmates), alignment (move towards the average heading of local flockmates) and cohesion (move toward the average position of local flockmates).
- The idea of PSO Algorithms is to solve optimization problems defined over a large search space. Each point in the search space has an associated numerical value called the fitness value and the goal is to choose the best fitness value called the global optimization point.
- For doing the same, PSO makes use of agents termed **"particles"**, which move step by step in search of the best solution (global optimum). Accordingly, the flying experience of itself along with the flying experience of the other particles is taken into consideration by each particle to adjust its own flying. The collection of flying particles is called a **"swarm".**
- Each particle of the swarm maintains a record of two things: its best solution, **personal best,** ***pbest*** and the best value among all the particles, **global best,** ***gbest.***
- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues. Each particle modifies its position according to: its current position, its current velocity, the distance between its current position and *pbest and* the distance between its current position and *gbest.*

Syllabus Topic : Formulations

6.1.1 Formulations

- In PSO, particles move towards the global optimum step by step in what is known as an iteration.
- A particle has the following:
  - (i) A position inside the search space
  - (ii) The fitness value at this position
  - (iii) A velocity (in fact a displacement), which is used to compute the next position
  - (iv) A memory, that contains the best position (called the previous best) found so far by the particle.
  - (v) The fitness value of this previous best
- In each iteration, every particle in the swarm gets one chance to move towards the global optimum by a magnitude equal to the velocity of the particle.
- Inside the swarm a topology is defined: it is a set of links between particles, saying who informs whom. When a particle is informed by another one, it means that the particle knows the previous best.
- The particle is informed by a set of particles which is called its neighborhood. The neighborhood also includes the particle under consideration. There are two stages in which the search is performed: initialization followed by a number of iterations.

6.1.1.1 Initialisation of the Swarm

For each particle :

1. Choose a random position within the search space. Compute the fitness value at this position. Set the initial value of the previous best to this initial position.
2. Choose a random velocity

6.1.1.2 Iteration

- The new velocity (displacement) is computed by combining the given elements :
  - the current position
  - the current velocity
  - the previous best
  - the best previous best in the neighbourhood

Move, by applying the new velocity to the current position

A confinement method is applied to ensure that the new position lies within the search space and the new fitness is computed. If the new fitness is better than the fitness of the previous best position, the previous best position and the previous best fitness are replaced by the new fitness and the new position respectively.

For the iterations, there are two stopping criteria :

- When the fitness value on the optimum point is known, a maximum admissible error is used as stopping criteria. If the absolute difference between the global best and the best fitness is smaller than this error, then the algorithm stops.
- A maximum number of fitness evaluations, given in advance.

Syllabus Topic : Pseudo-code

6.1.2 Pseudo-code

Let

A : Population of agents

p_i : Position of agent a_i in the solution space

f : Objective function

v_i : Velocity of agent's a_i

$V(a_i)$: Neighborhood of agent a_i

Pseudo code

1. [x*] = PSO()
2. P = Particle_Initialization();
3. While stopping criteria is not met
   (i) For each particle p in P
      (a) $fp = f(p)$;
      (b) If fp is better than f($pBest$)
         $pBest = p$;
      End
   (ii) $gBest = p$ corr. to best fp in P;
   (iii) For each particle p in P d
      (a) $v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)$;

(b) $p = p + v;$

Note :

Particle update rule

$$p = p + v$$

with

$$v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)$$

Where,

p : particle's position

v : path direction

c_1 : weight of local information

c_2 : weight of global information

pBest : best position of the particle

gBest : best position of the swarm

rand : random variable

Syllabus Topic : Parameters

6.1.3 Parameters

- Number of particles is chosen usually between 10 and 50.
- c_1 is the importance of personal best value.
- c_2 is the importance of neighborhood best value.
- Usually $c_1 + c_2 = 4$ (empirically chosen value)
- Particle's velocity is given by :

$$v_i^{t+1} = \underbrace{v_i^t}_{\text{Diversification}} + \underbrace{c_1 U_1^t (pb_i^t - p_i^t) + c_2 U_2^t (gb^t - p_i^t)}_{\text{intensification}}$$

  - **Intensification :** explores the previous solutions, finds the best solution of a given region.

- **Diversification :** searches new solutions, finds the regions with potentially the best solutions.
- If velocity is too low, then the algorithm becomes too slow and if velocity is too high, the algorithm becomes too unstable.

Syllabus Topic : Premature Convergence of PSO

6.1.4 Premature Convergence of PSO

- Although the speed of convergence is very fast, many experiments have shown that once PSO traps into local optimum, it is difficult for PSO to jump out of the local optimum. This leads to premature convergence of PSO.
- The lack of population diversity in PSO is understood to be a factor in its convergence on local optima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance.
- There are different ways to avoid the premature convergence of PSO.
- Using a quantum model (QPSO), the traditional position and velocity equations are replaced with a wave function which can give optimal solutions.
- The gbest (the global best particle) and *mbest* (the mean value of all particles' previous best position) can be mutated with Cauchy distribution. This happens because the expectation of Cauchy distribution does not exist and so the variance of Cauchy distribution is infinite. This helps to introduce diversification in the PSO and thus prevent premature convergence.

Syllabus Topic : Topology

6.1.5 Topology

The neighborhood of each particle can be decided depending upon the chosen topology to pass on the information. Popular topologies include the ring and adaptive random topologies which are discussed below.

6.1.5.1 The Ring Topology

The ring topology is a very common topology used for years because of its simplicity. As per this topology, the neighborhood of a particle i is given by:

$$i - 1 \ mod(S), i, i + 1 \ mod(S)$$

For example, if S = 30 the neighbourhood of the particle 0 is {29, 0, 1}, and the neighbourhood of the particle 29 is {28, 29, 0}.

6.1.5.2 The Adaptive Random Topology

- At the very beginning and after each unsuccessful iteration (no improvement of the best known fitness value), the graph of the information links is modified: each particle informs at random k particles (the same particle may be chosen several times), and informs itself.
- The parameter *k* is usually set to 3. It means that each particle informs at least one particle (itself), and at most k + 1 particles (including itself). It also means that each particle can be informed by any number of particles between 1 and S.

Syllabus Topic : Real Valued and Binary PSO

6.1.6 Real Valued and Binary PSO

- In regular (real valued) PSO, everything is in terms of a velocity. Generally the velocity is defined in terms of a probability of the bit changing.
- In Binary PSO, each solution in the population is a binary string. Each binary string is of dimension *n* which is evaluated to give parameter values.
- In the binary PSO, each binary string represents a particle. Strings are updated bit-by-bit based on its current value, the value of that bit in the best (fitness) of that particle to date, and the best value of that bit to date of its neighbors.
- In BPSO, bit-by-bit updates are done probabilistically. In other words, for a chosen bit *b* in a chosen string *s* it is changed to a 1 with a probability *p* that is a function of its predisposition to be a 1, the best value of itself to date, and the best value of its neighbors. (1 – p) is the probability of changing to a zero. Once *p* is determined, a random number *r* is generated. If r < p, then the bit becomes a 1; otherwise it becomes a zero.

Syllabus Topic : Ant Colony Optimization (ACO)

6.2 Ant Colony Optimization (ACO)

- Ant Colony Optimisation is a technique to solve optimisation problems based on the way that ants indirectly communicate directions to each other.

- The main idea is inspired by the natural behavior of ants when they set out in search of food.
- The ants find the shortest paths between their nest and the food by means of a chemical substance they secret called the **pheromone.**
- This pheromone acts as a signal to other ants. If an ant decides with some probability to follow the pheromone trail, it itself lays more pheromone, thus reinforcing the trail.
- If more number of ants follows the trail, the pheromone becomes stronger and ants are more likely to follow it.

Syllabus Topic : Formulations

6.2.1 Formulations

- Ants are *agents* that move along between nodes in a graph.
- They choose where to go based on pheromone strength (and maybe other things)
- An ant's path represents a specific candidate solution.
- When an ant has finished a solution, pheromone is laid on its path, according to quality of solution.
- This pheromone trail affects behaviour of other ants.
- An ACO algorithm can be used typically to solve graph based problems like the Travelling Salesperson Problem (TSP).
- Consider, for example, a 4-city TSP instance.

Initially, random levels of pheromone are scattered on the edges.

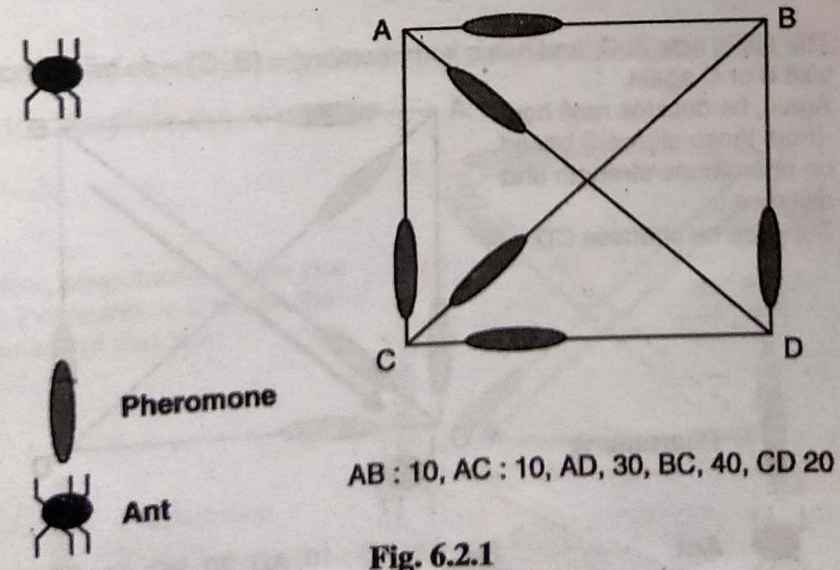
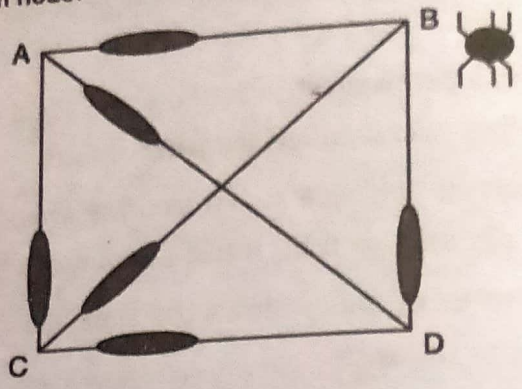


Fig. 6.2.1

An ant is placed at a random node.



Pheromone

Ant

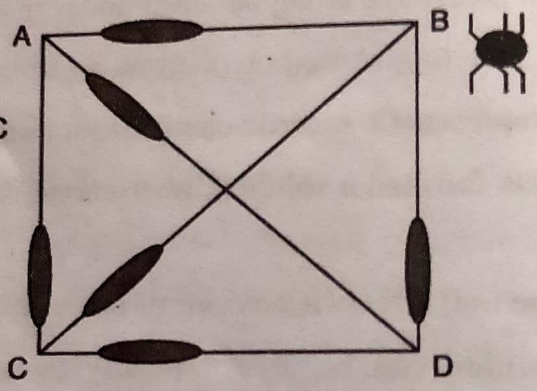
AB : 10, AC : 10, AD, 30, BC, 40, CD 20

Fig. 6.2.2

The ant decides where to go from that node, based on probabilities calculated from :

- pheromone strengths,
- next-hop distances.

Suppose this one chooses BC



Pheromone

Ant

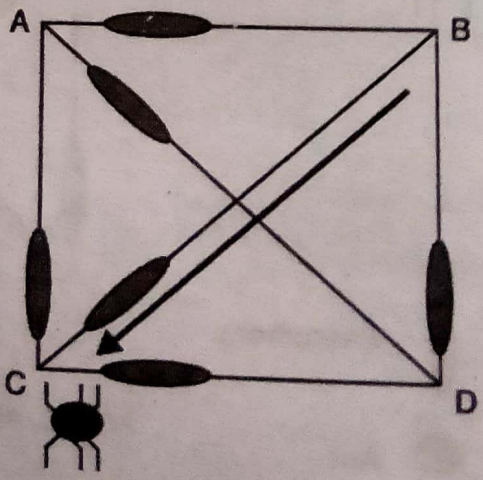
AB : 10, AC : 10, AD, 30, BC, 40, CD 20

Fig. 6.2.3

The ant is now at C, and has a 'tour memory' = {B, C} – so he cannot visit B or C again.

Again, he decides next hop (from those allowed) based on pheromone strength and distance ;

Suppose he chooses CD.



Pheromone

Ant

AB : 10, AC : 10, AD, 30, BC, 40, CD 20

Fig. 6.2.4

The ant is now at D, and has a 'tour memory' = {B, C, D}
There is only one place he can go now :

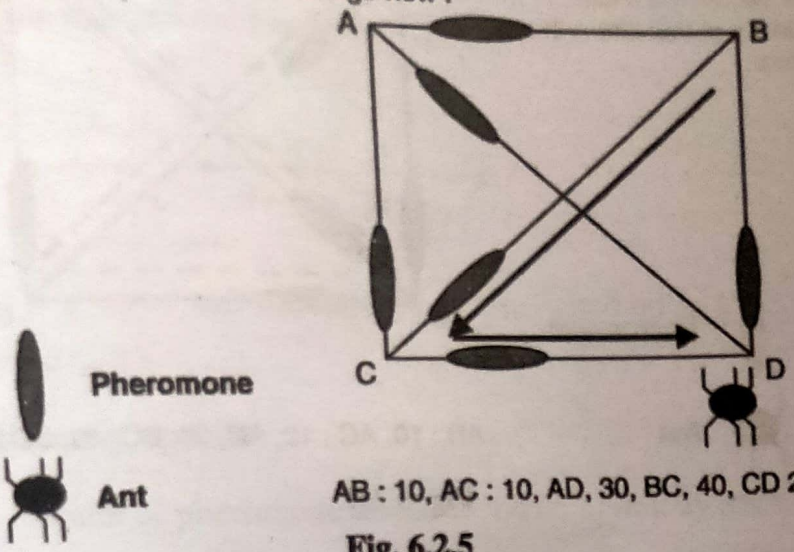


Fig. 6.2.5

So, he has nearly finished his tour, having gone over the links : BC, CD and DA.

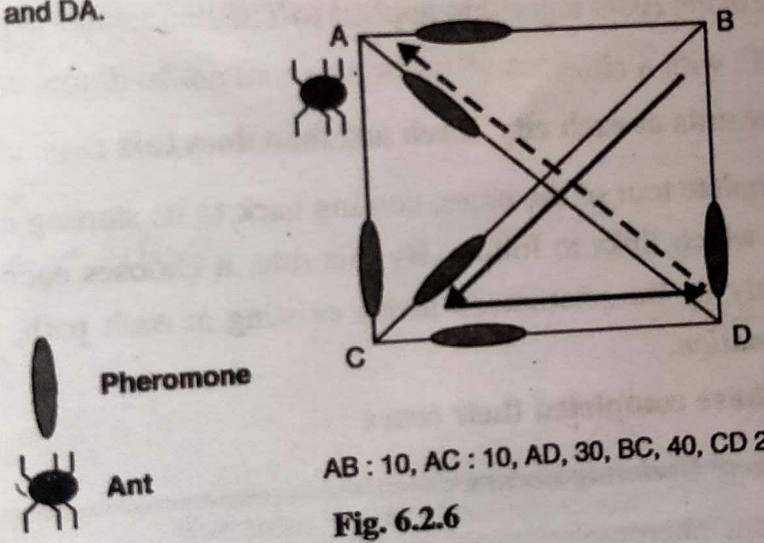


Fig. 6.2.6

So, he has nearly finished his tour, having gone over the links : BC, CD, and DA.
AB is added to complete the round trip.

Now, pheromone on the tour is increased, in line with the fitness of that tour.

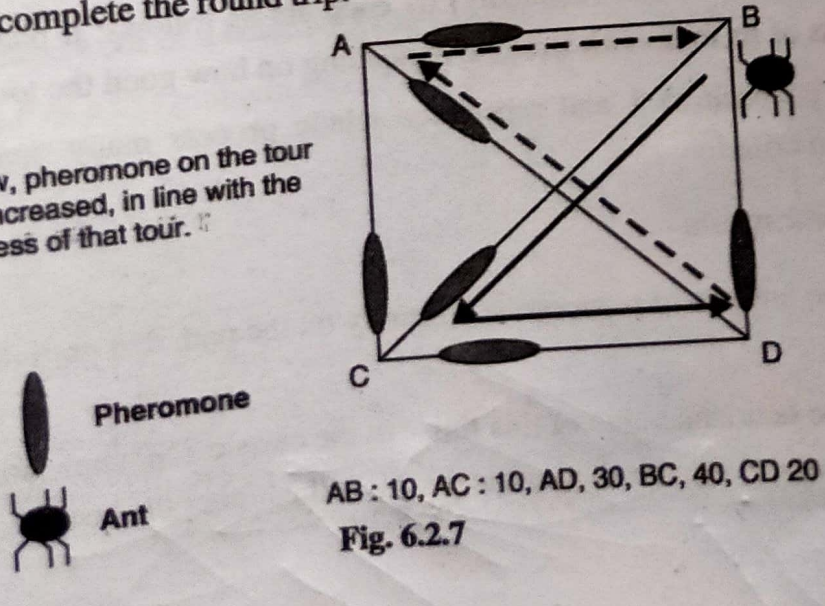


Fig. 6.2.7

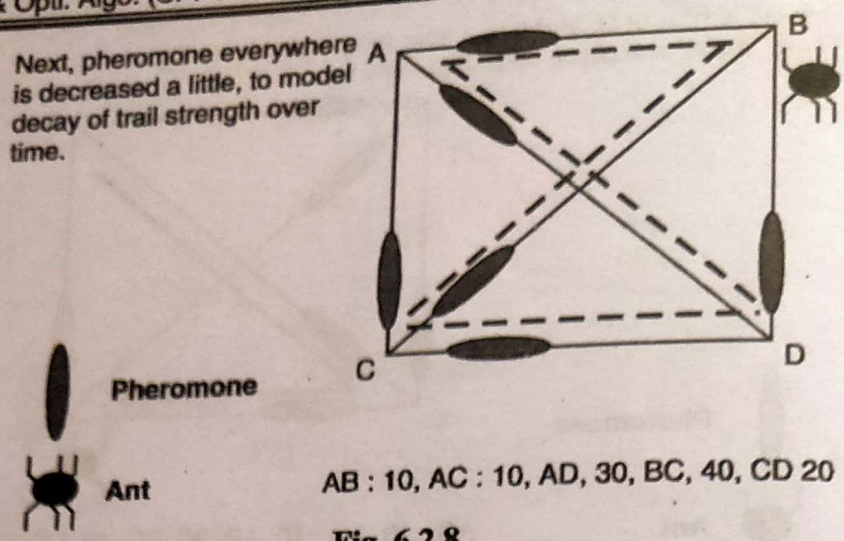


Fig. 6.2.8

We repeat with the same procedure by placing an ant at another random location.

The following is the ACO algorithm applied to TSP.

We have a TSP, with n cities.

1. **We place some ants at each city. Each ant then does this :**

   It makes a complete tour of the cities, coming back to its starting city, using a *transition rule* to decide which links to follow. By this rule, it chooses each next-city at random, but biased partly by the pheromone levels existing at each path, and biased partly by *heuristic information.*

2. **When all ants have completed their tours**

   Global Pheromone Updating occurs.

   - The current pheromone levels on all links are reduced (i.e. pheromone levels decay over time).
   - Pheromone is lain (belatedly) by each ant as follows: it places pheromone on all links of its tour, with strength depending on how good the tour was.

   Then we go back to 1 and repeat the whole process many times, until we reach a termination criterion.

The transition rule

- $T(r, s)$ is the amount of pheromone currently on the path that goes directly from city r to city s.
- $H(r, s)$ is the heuristic value of this link - in the classic TSP application, this is chosen to be 1/distance(r, s) i.e. the shorter the distance, the higher the heuristic value.

$p_k(r, s)$ is the probability that ant k will choose the link that goes from r to s is a parameter that we can call the *heuristic strength.*

The rule is :

$$p_k(r, s) = \frac{T(r, s) \cdot H(r, s)^{\beta}}{\sum_{\text{unvisited cities } c} T(r, c) \cdot H(r, c)^{\beta}}$$

Where our ant is at city r and s is a city as yet unvisited on its tour, and the summation is over all of k's unvisited cities.

Global pheromone update

$A_k(r, s)$ is amount of pheromone added to the (r, s) link by ant k.

m is the number of ants.

ρ is a parameter called the pheromone decay rate.

L_k is the length of the tour completed by ant k.

$T(r, s)$ at the next iteration becomes :

$$\rho \cdot T(r, s) + \sum_{k=1}^{m} + A_k(r, s)$$

Where,

$$A_k(r, s) = \frac{1}{L_k}$$

Syllabus Topic : Pseudo-code

6.2.2 Pseudo-code

Algorithm 1 : The framework of a basic ACO algorithm

Input : An instance P of a CO problem model $P = (S, f, \Omega)$.

```
InitializePheromoneValues(T)
S_bs ← NULL
while termination conditions not met do
    S_iter ← ∅
    for j = 1, ...., n_a do
        S ← ConstructSolution (T)
        if S is a valid solution then
        S ← LocalSearch(S)        {optional}
        if (f(S) < f (S_bs)) or (S_bs = NULL) then S_bs ← S
```

$S_{iter} \leftarrow S_{iter} \cup \{S\}$
end if
end for
ApplyPheromoneUpdate (T, S_{iter}, S_{bs})
end while

Output : The best-so-far solution S_{bs}

Syllabus Topic : Applications of PSO and ACO

6.3 Applications of PSO and ACO

- PSO is used in most optimization problems.
- ACO is mostly used for graph based problems :
  - Travelling Salesman Problem
  - Quadratic Assignment Problem
  - Network Model Problem
  - Vehicle routing

Review Questions

Q. 1 Explain Particle Swarm Optimisation along with its pseudo code? *(Ans. : Refer section 6.1)*

Q. 2 What are the prevalent topologies of PSO? Explain in short. *(Ans. : Refer section 6.1.5)*

Q. 3 What is the velocity update equation of PSO? Comment on how the PSO Parameters affect the working of the algorithm. *(Ans. : Refer section 6.1.3)*

Q. 4 Explain the Ant Colony Optimisation Algorithm with the help of an example. *(Ans. : Refer section 6.2)*

Q. 5 What is the Global Pheromone Update Rule and Transition Rule with respect to ACO? *(Ans. : Refer section 6.2.1)*

Q. 6 Write the pseudo code for Ant Colony Optimisation. *(Ans. : Refer section 6.2.2)*

□□□