

# Computational Intelligence

Unit # 7

# Swarm Intelligence and Computer Science

- **Ant Colony Optimization (ACO)**



Inspired by food foraging behavior of ants/bees

- **Particle Swarm Optimization (PSO)**



Inspired by bird flocking and fish schooling

# Swarms in Nature

- Lens of Time: Secrets of Schooling
- [https://www.youtube.com/watch?v=Y-5ffl5\\_7AI&t=107s](https://www.youtube.com/watch?v=Y-5ffl5_7AI&t=107s)
- Starlings Murmuration  
<https://www.youtube.com/watch?v=eakKfY5aHmY&t=145s>

# Particle Swarm Optimization

- Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.
- The system is initialized with a population of random solutions and searches for optima by updating generations.
- Unlike EA, PSO has no evolution operators such as crossover and mutation.

# Concept

- As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario:
  - A group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration.
- **So what's the best strategy to find the food?**

# Concept

- As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario:
  - A group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration.
- **So what's the best strategy to find the food?**
- The effective one is to follow the bird which is nearest to the food.

# Particle

- Particle
  - Current position
  - Velocity

The heart of the classic PSO algorithm is in the step which calculates a new position for the particle based on its updated velocity.

# Basic Theme

- The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its *pbest* and *lbest* locations (local version of PSO).
- Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.



# Velocity Update

There are three factors that influence the velocity update:

- **Current velocity:** the particle's current velocity (obviously);
- **Personal Best:** the particle remembers the fittest position it has yet encountered, called the personal best. A component of its updated velocity is the direction from its current position to the personal best;
- **Global Best:** every particle in the swarm is aware of the best position that any particle has yet discovered (i.e. the best of the personal bests). The final component of velocity update, shared by all particles, is a vector in the direction from its current position to this globally best known position.

# Algorithm Description (Cont'd)

- At each step of the algorithm, particles are displaced from their current position by applying a velocity (gradient) vector to them.
- The magnitude and direction of their velocity at each step is influenced by their velocity in the previous iteration of the algorithm, simulated momentum, and the location of the a particle relative to the location of its pbest and the gbest.
- Therefore, at each step, the size and direction of each particle's move is a function of its own history (experience), and the social influence of its peer group.

# Algorithm Description (Cont'd)

- After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).
- $$v[i] = \varphi v[i] + c1 * \text{rand}() * (pbest[i] - present[i]) + c2 * \text{rand}() * (gbest[i] - present[i]) \quad (a)$$
$$present[i] = present[i] + v[i] \quad (b)$$

$v[i]$  is the particle velocity,  $present[i]$  is the current particle (solution).  $pbest[i]$  and  $gbest[i]$  are defined as stated before.  $\text{rand}()$  is a random number between (0,1).  $c1$ ,  $c2$  are learning factors, also known as acceleration constants (usually  $c1 = c2 = 2$ ).  $\varphi$  is inertia weight.

# PSO Algorithm

- ▶ Swarm: a set of particles (S)
- ▶ Particle: a potential solution
  - Position:  $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n}) \in \mathbb{R}^n$
  - Velocity:  $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \dots, v_{i,n}) \in \mathbb{R}^n$
- ▶ Each particle maintains
  - Individual best position (PBest)
- ▶ Swarm maintains its global best (GBest)

<https://slideplayer.com/slide/13389026/>

# PSO Algorithm

1. Randomly initialize the swarm to form the solution space
2. Evaluate the fitness of each particle
3. Update individual and global bests
4. Update velocity and position of each particle
5. Go to step2, and repeat until termination condition

<https://slideplayer.com/slide/13389026/>

# PSO Algorithm

- ▶ Original velocity update equation

$$\mathbf{v}_i(k+1) = \text{Inertia} + \text{cognitive} + \text{social}$$

$$\mathbf{v}_i(k+1) = \omega \times \mathbf{v}_i(k) + c_1 \times \text{random}_1() \times (PBest_i - \mathbf{x}_i(k)) \\ + c_2 \times \text{random}_2() \times (GBest - \mathbf{x}_i(k))$$

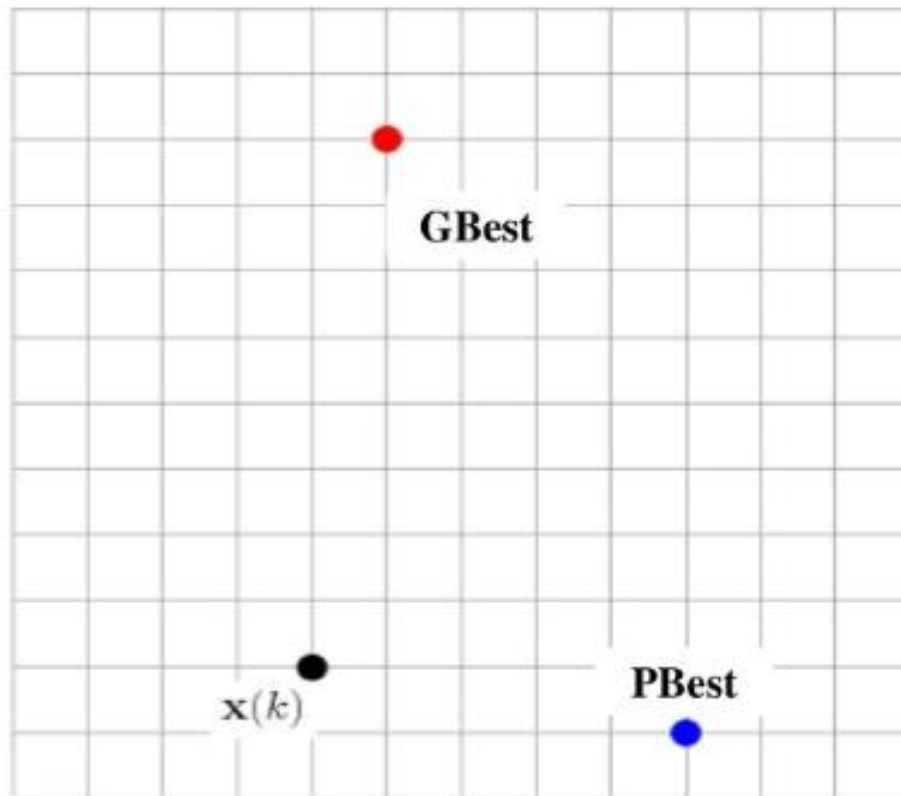
- $\omega, c_1, c_2$ : Constant
- $\text{random}_1(), \text{random}_2()$ : random variable

- ▶ Position update

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1)$$

<https://slideplayer.com/slide/13389026/>

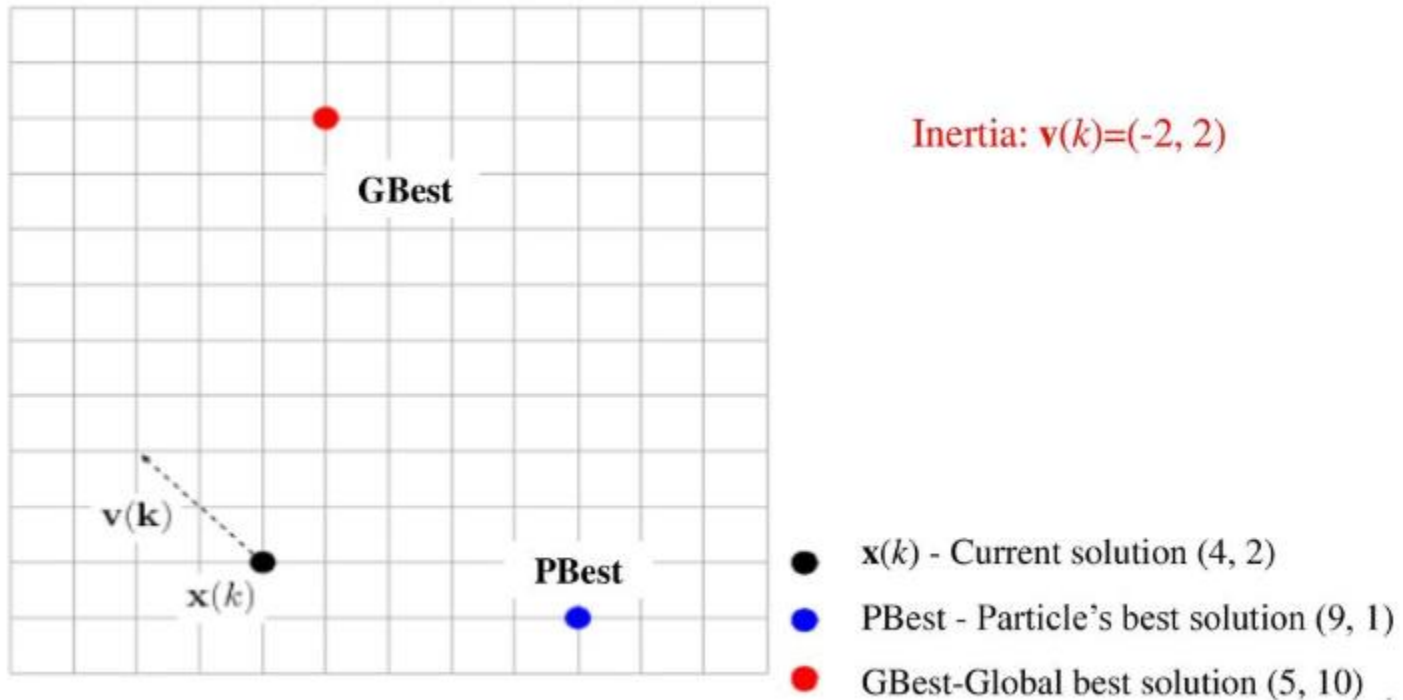
# PSO Algorithm



- $x(k)$  - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

<https://slideplayer.com/slide/13389026/>

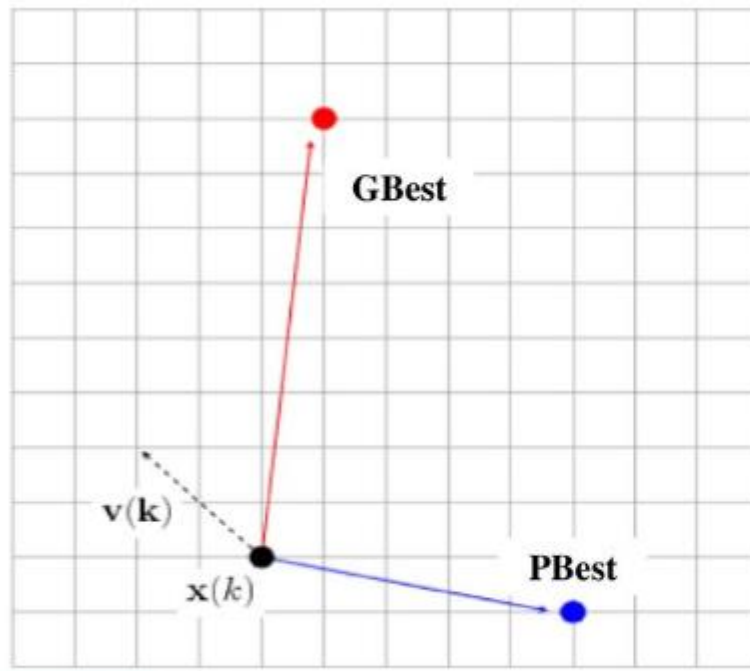
# PSO Algorithm



<https://slideplayer.com/slide/13389026/>



# PSO Algorithm

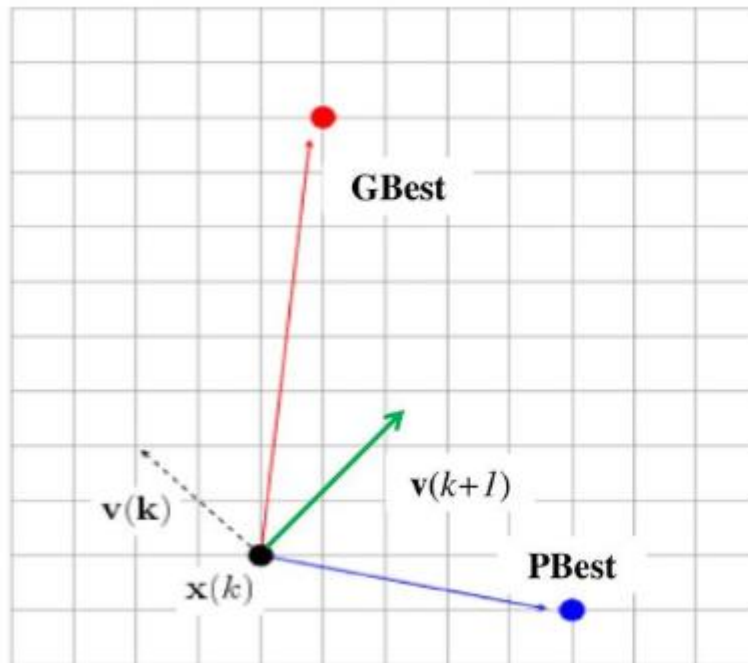


- Inertia:  $v(k)=(-2,2)$
- Cognitive:  
 $PBest-x(k)=(9,1)-(4,2)=(5,-1)$
- Social:  
 $GBest-x(k)=(5,10)-(4,2)=(1,8)$

- $x(k)$  - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

<https://slideplayer.com/slide/13389026/>

# PSO Algorithm



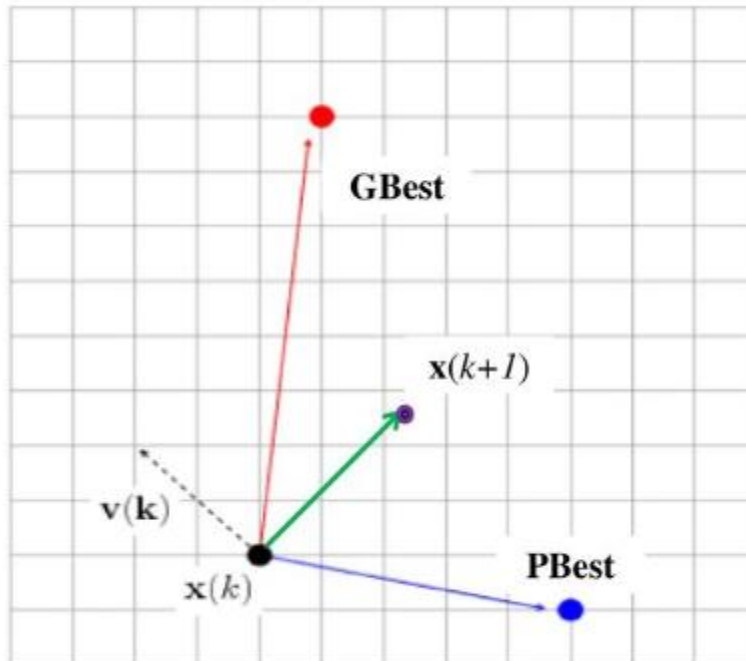
- Inertia:  $v(k)=(-2,2)$
- Cognitive:  
 $PBest-x(k)=(9,1)-(4,2)=(5,-1)$
- Social:  
 $GBest-x(k)=(5,10)-(4,2)=(1,8)$

$$v(k+1)=(-2,2)+0.8*(5,-1) \\ +0.2*(1,8) = (2.2,2.8)$$

- $x(k)$  - Current solution (4, 2)
- $PBest$  - Particle's best solution (9, 1)
- $GBest$  - Global best solution (5, 10)

<https://slideplayer.com/slide/13389026/>

# PSO Algorithm



- Inertia:  $\mathbf{v}(k) = (-2, 2)$
- Cognitive:  
 $\text{PBest} - \mathbf{x}(k) = (9, 1) - (4, 2) = (5, -1)$
- Social:  
 $\text{GBest} - \mathbf{x}(k) = (5, 10) - (4, 2) = (1, 8)$
- $\mathbf{v}(k+1) = (2.2, 2.8)$

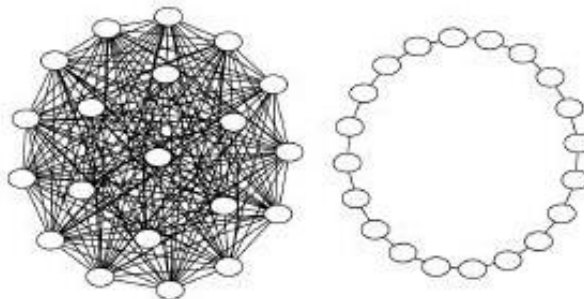
$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{x}(k) + \mathbf{v}(k+1) = \\ (4, 2) &+ (2.2, 2.8) = (6.2, 4.8)\end{aligned}$$

- $\mathbf{x}(k)$  - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest - Global best solution (5, 10)

<https://slideplayer.com/slide/13389026/>

# Neighborhood Topologies

- **Star:** In the gbest swarm, all the particles are neighbors of each other; thus, the position of the best overall particle in the swarm is used in the social term of the velocity update equation. It is assumed that gbest swarms converge fast, as all the particles are attracted simultaneously to the best part of the search space. However, if the global optimum is not close to the best particle, it may be impossible to the swarm to explore other areas; this means that the swarm can be trapped in local optima.
- **Ring:** In the lbest swarm, only a specific number of particles (neighbor count) can affect the velocity of a given particle. The swarm will converge slower but can locate the global optimum with a greater chance.



# Role of pbest and gbest

- At the start of the algorithm, the pbest for each particle is set at its initial location, and gbest is set to the location of the best of the pbests.
- In each iteration of the algorithm, a particle is stochastically accelerated towards its previous best position and towards a neighborhood (global) best position, thereby forcing particles to continually search in the most-promising regions found so far in the solution space.

# Role of $c_1$ and $c_2$

- The weight coefficient  $c_1$  and  $c_2$  control the relative impact of the pbest and gbest locations on the velocity of a particle.
- Low values for  $c_1$  and  $c_2$  allow each particle to explore far away from already uncovered good points, high values of the parameters encourage more intensive search of regions close to these points.

# Binary PSO

- This setting works very well for a continuous problem. The inherent parameters, Particle's position and Particle's velocity assume a number from the set of real numbers,  $\mathbb{R}$ . But, there are numerous optimization problems that are discrete in nature and are basically classified as Combinatorial Optimization Problems. A "Continuous PSO" would not work here. We have to build a "Binary PSO".
- So what are the changes we should make to the above algorithm that it works for these newer kinds of problems too?

# Binary PSO

- The answer is to make the Particles a bit string and to constrain the velocity to be a value in the interval  $[0, 1]$ . The velocity is defined as the probability of a bit  $X_{ij}$  (  $i$  th particle and  $j$  th bit) to attain a value of 1. To constrain the velocity the following limiting transformation function is used. It uses the Sigmoid function.

$$S(v_{ij}) = 1 / (1 + e^{-v_{ij}})$$

$$x_{ij} = \begin{cases} 1 & \text{if } \text{rand}() \leq S(v_{ij}) \\ 0 & \text{otherwise} \end{cases}$$



# Comparing PSO with the EA

- Like the EA, PSO is population-based, it is typically initialized with a population (swarm) of random encodings of solutions, and search proceeds by updating these encodings over a series of generations (iterations).
- Both systems do not guarantee success.
- Both algorithms start with a group of a randomly generated population, both have fitness values to evaluate the population.
- Unlike the EA, PSO has no explicit selection process as all particles persist over time. Instead a *memory* in the form of gbest/lbest is substituted for selection.
- PSO also does not have genetic operators like crossover and mutation.

# Comparing PSO with the EA (Cont'd)

- Compared with evolutionary algorithms (EAs), the information sharing mechanism in PSO is significantly different.
- In EAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area.
- In PSO, only gBest (or lBest) gives out the information to others. It is a one-way information sharing mechanism. The evolution only looks for the best solution.
- Compared with EA, all the particles tend to converge to the best solution quickly even in the local version in most cases.


# Applications

- Shortest Path Problems
- Structural Optimization
- Scheduling
- Vehicle Routing
- Strategy Building
- Computer Games
- Industrial Problems
- Operation Research
- Bioinformatics

# Recent Developments in SI

- US Military is applying SI techniques to control unmanned vehicles
- NASA is applying SI techniques for planetary mapping
- Medical Research is trying SI based controls for nanobots to fight Cancer.
- Load balancing in telecommunication
- Entertainment industry is applying SI techniques for battle and crowd scenes

# Pentagon's DARPA seeks massive drone swarms to swamp enemy defense bubbles

Bruce Crumley - Feb. 13th 2023 12:02 pm PT  @BDroneDJ

MILITARY DRONES

DRONE SWARM

DEPARTMENT OF DEFENSE (DOD)

DARPA



Comments



[DARPA seeks massive drone swarms to swamp enemy defenses \(dronedj.com\)](https://dronedj.com)

# Crowd Simulation

- **Crowd simulation** is the process of simulating the movement (or dynamics) of a large number of entities or characters. It is commonly used to create virtual scenes for visual media like films and video games, and is also used in crisis training, architecture and urban planning, and evacuation simulation.

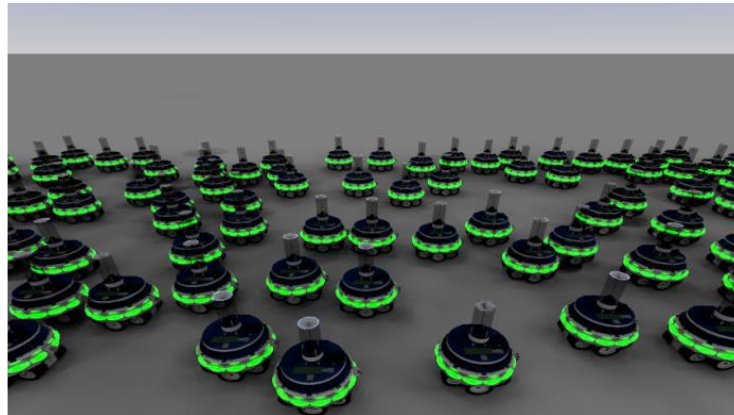
<https://firemissionalpha.wordpress.com/2016/04/26/crowd-simulation-in-movies/>

- Massive: Software extensively used for crowd simulation in movies

<https://www.youtube.com/watch?v=cr5Cwz-5Wsw>

# Swarm Robotics

- **Swarm robotics** is a field of multi-robotics in which large number of **robots** coordinate in a distributed and decentralized way to solve a complex task.



- <https://www.youtube.com/watch?v=jwu9SX3YPSk>
- <https://www.youtube.com/watch?v=G1t4M2XnIhI>

# Swarm Robotics



# Swarm of drones thinking for itself

- <https://www.youtube.com/watch?v=3TYqTtmWhx4>

# Meet Kilobots!

- [Self-Assembling Robot Swarm KiloBot - YouTube](#)