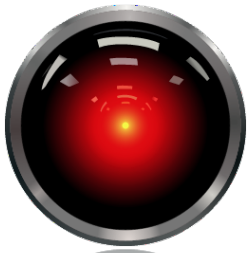


INTELLIGENT COMPUTING

(COMPUTATIONAL INTELLIGENCE)
Particle Swarm Optimisation (PSO)

S. Sheridan

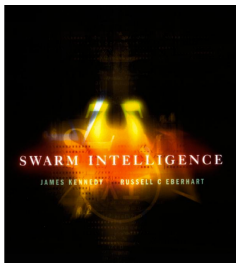
PARTICLE SWARM OPTIMISATION (PSO) DEFINITION

- In computer science, particle swarm optimisation (PSO) is a computational method that optimises a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.
- PSO optimises a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity.
- Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PARTICLE SWARM OPTIMISATION (PSO) BACKGROUND

Russell C. Eberhart

James Kennedy



April 9, 2001 ISBN-10: 1558605959

Kennedy, J. Eberhart, R. (1995). ***"Particle Swarm Optimization"****. Proceedings of IEEE International Conference on Neural Networks IV. pp. 1942–1948.*

"The authors of this paper are a social psychologist and an electrical engineer. The particle swarm optimiser serves both of these fields equally well. Why is social behaviour so ubiquitous in the animal kingdom? Because it optimizes. What is a good way to solve engineering optimisation problems? Modeling social behavior."

PARTICLE SWARM OPTIMISATION (PSO) BACKGROUND

- Inspired by the flocking and schooling patterns of birds and fish, Particle Swarm Optimisation (PSO) was invented by *Russell Eberhart* and *James Kennedy in 1995*. Originally, these two started out developing computer software simulations of birds flocking around food sources, then later realised how well their algorithms worked on optimisation problems.
- Imagine a flock of birds circling over an area where they can smell a hidden source of food. The one who is closest to the food chirps the loudest and the other birds swing around in his direction. If any of the other circling birds comes closer to the target than the first, it chirps louder and the others veer over toward him.
- This tightening pattern continues until one of the birds happens upon the food. PSO is an algorithm that's simple to understand and easy to implement.

https://www.youtube.com/watch?v=0uqsRGFLM20

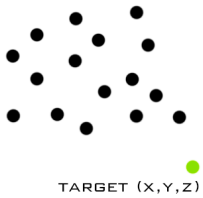
https://www.youtube.com/watch?v=rUSTXUis_ys

PARTICLE SWARM OPTIMISATION (PSO) OVERVIEW

- The algorithm usually keeps track of three global variables:
  - Target value or condition
  - **Global best** (gBest) value indicating which particle's data is currently closest to the Target
  - Stopping value indicating when the algorithm should stop if the Target isn't found
- Each particle usually consists of:
  - Data representing a possible solution
  - A Velocity value indicating how much the Data can be changed
  - A **personal best** (pBest) value indicating the closest the particle's Data has ever come to the Target

PARTICLE SWARM OPTIMISATION (PSO) OVERVIEW

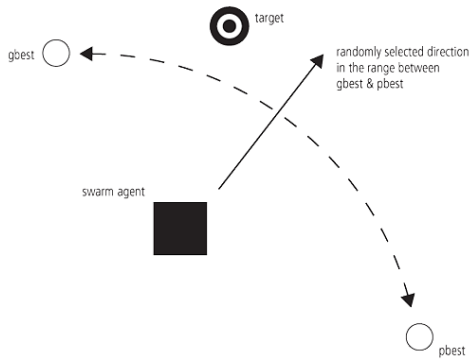
- The particles' data could be anything. In the flocking birds example, the data would be the X, Y, Z coordinates of each bird. The individual coordinates of each bird would try to move closer to the coordinates of the bird which is closer to the food's coordinates (gBest).
- If the data is a pattern or sequence, then individual pieces of the data would be manipulated until the pattern matches the target pattern.



NINGO = 1	47 + -3 + 8 = 52
BINGO = 0	27 + 3 + 10 = 40
NINGOGG = 3	41 + 6 + 31 = 78
NINXKN = 4	47 + 40 + -4 = 83
BIGOIN = 3	41 + -3 + 30 = 68
IIGGOG = 3	6 + 14 + 35 = 55
IJNSOI = 4	3 + 11 + 36 = 50
IGNJNI = 5	47 + 2 + 9 = 58
NINIONNN = 5	40 + -3 + 29 = 66
GGNNNCNN = 7	1 + 8 + 11 = 20

PARTICLE SWARM OPTIMISATION (PSO) OVERVIEW

- The velocity value is calculated according to how far an individual's data is from the target. The further it is, the larger the velocity value.
- In the birds example, the individuals furthest from the food would make an effort to keep up with the others by flying faster toward the gBest bird. If the data is a pattern or sequence, the velocity would describe how different the pattern is from the target, and thus, how much it needs to be changed to match the target.



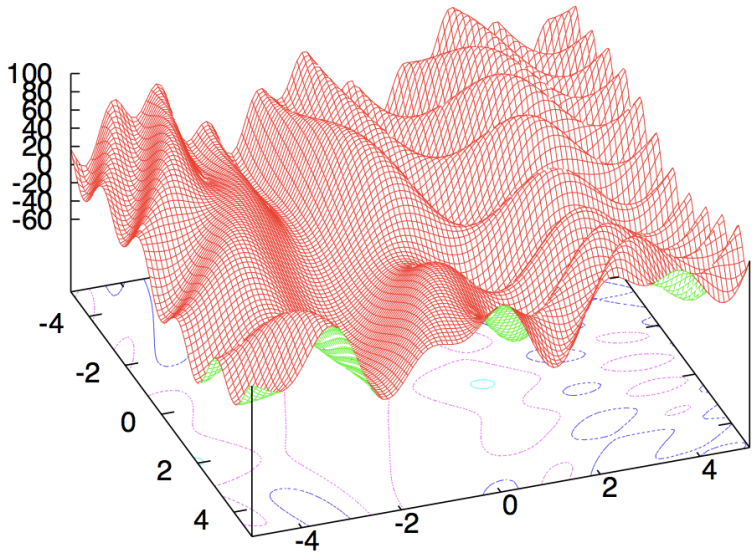
PARTICLE SWARM OPTIMISATION (PSO) OVERVIEW

- Each particle's pBest value only indicates the closest the data has ever come to the target since the algorithm started.
- The gBest value only changes when any particle's pBest value comes closer to the target than gBest. Through each iteration of the algorithm, gBest gradually moves closer and closer to the target until one of the particles reaches the target.
- It's also common to see PSO algorithms using population topologies, or "neighbourhoods", which can be smaller subsets of the local best (lBest) value.
- **NOTE:** the simplest form of PSO algorithm is based purely on the global best (gBest) and personal best (pBest).

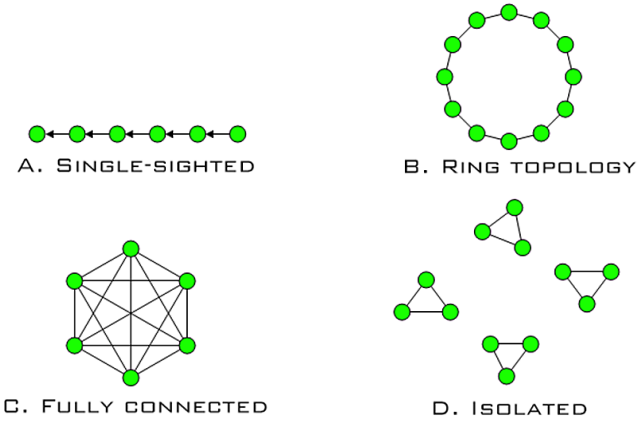
PARTICLE SWARM OPTIMISATION (PSO) LOCAL MINIMA

- Neighbourhoods can involve two or more particles which are predetermined to act together, or subsets of the search space that particles happen into during testing. The use of neighbourhoods often help the algorithm to avoid getting stuck in local minima.

PARTICLE SWARM OPTIMISATION (PSO) LOCAL MINIMA



PARTICLE SWARM OPTIMISATION (PSO) TOPOLOGIES

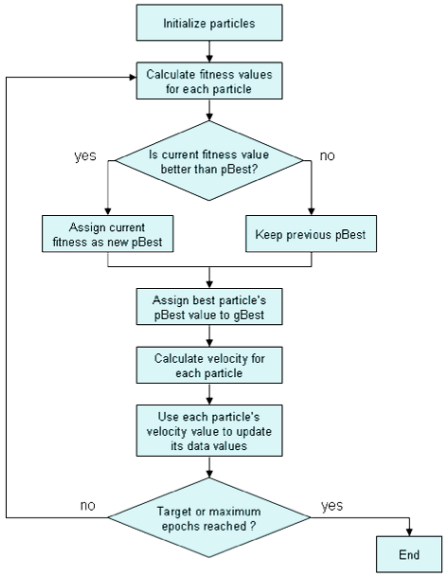


(A) Single-sighted, where individuals only compare themselves to the next best. **(B)** Ring topology, where each individual compares only to those to the left and right. **(C)** Fully connected topology, where everyone is compared together. **(D)** Isolated, where individuals only compare to those within specified groups

PSO IN A NUTSHELL

- An essential aspect of Kennedy and Eberhart's algorithm is the sharing of information between agents. When the swarm is initialised a space is defined. Within it, swarm agents are randomly distributed and a point is established as the target that each agent is meant to find.
- When the search begins, each agent computes a new trajectory based on A) a personal best (pBest): the point it, individually, has found nearest the target and B) a group best (gBest): the nearest pBest in the entire swarm, which is frequently updates and simultaneously shared among all agents. Each agent then randomly selects a point between A and B and moves to it.
- This process repeats very quickly to create an elegant, elliptical movement that appears like the swarming of insects. Distance from an agent to the target is called "fitness." Over time, through individual work and social sharing of "bests," agents seek greater fitness (a value approaching 0), and the swarm coalesces at the target.

PSO ALGORITHM FLOWCHART



PSO ALGORITHM PSEUDO CODE

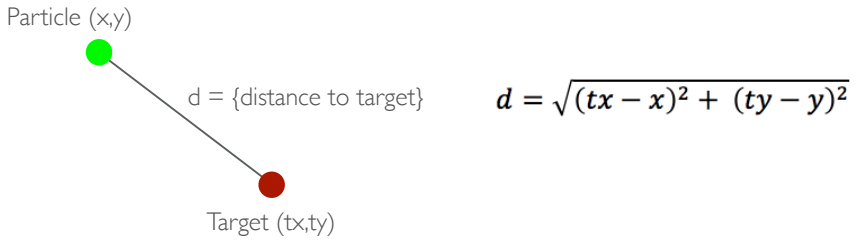
```
For each particle
{
    Initialise particle
}

Do until maximum iterations or minimum error criteria
{
    For each particle
    {
        Calculate Data fitness value
        If the fitness value is better than pBest
        {
            Set pBest = current fitness value
        }
        If pBest is better than gBest
        {
            Set gBest = pBest
        }
    }

    For each particle
    {
        Calculate particle Velocity
        Use gBest and Velocity to update particle Data
    }
}
```

(PSO) CALCULATING FITNESS

- The method for calculating the fitness of each particle will depend on the nature of the problem being solved. Much like the fitness function in a genetic algorithm the fitness function for a PSO will have a big impact on how the algorithm converges.
- Writing a fitness function is tricky and should not be underestimated. For some problems the fitness function will be obvious and for other the fitness function will be much harder to define.
- Remember that complex fitness functions (ones that require many CPU cycles) will slow down the overall speed of the PSO algorithm.
- The next few slides detail some typical fitness function approaches for three different classes of problem.

(PSO) CALCULATING FITNESS LOCATION

- For problems that involve working with a cartesian plane (X,Y) or (X,Y,Z) and some target location on that plane we can use the idea of distance from target location.
- For example, if we take a 2D space and a target location defined by (tx, ty), we can easily calculate the distance from the target for a any particle.



$$d = \sqrt{(tx - x)^2 + (ty - y)^2}$$

(PSO) CALCULATING FITNESS PATTERN

- Distance to target is also a useful concept when it comes to pattern matching problems. For example, if we want to implement a PSO to match strings we would need to calculate how different (distance) each particle string was from the target string.
- One such method for calculating the difference between strings (distance) is the **Levenshtein** distance.
- The Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.
- For example, the Levenshtein distance between **"kitten"** and **"sitting"** is **3**, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:
  - kitten → sitten (substitution of "s" for "k")
  - sitten → sittin (substitution of "i" for "e")
  - sittin → sitting (insertion of "g" at the end).

(PSO) CALCULATING FITNESS ARITHMETIC

- PSO algorithms can also be used to approximate functions and arithmetic expressions. The fitness function for this class of problem is usually straight forward as the function that is being approximated is known.
- For example, suppose we wanted to solve the following equality:

  3 + 11 + 36 = 50 x + y + z = 50

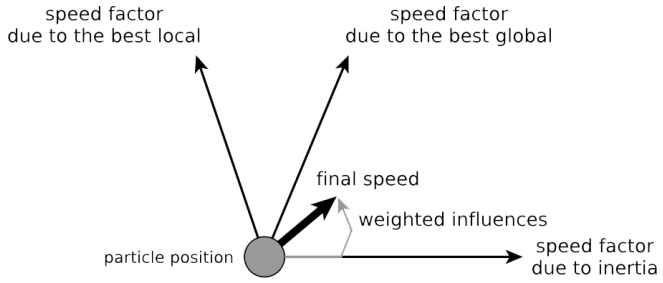
- Each particle would contain three independent variables (x,y,z). The fitness for each particle could be calculated based on the following:

  fitness = (x + y + z) - 50

- This type of fitness function can be easily extended to more complex functions.

(PSO) UPDATING PARTICLE VELOCITY

- There have been many tweaks to the basic velocity update equation since it was first introduced by Kennedy and Eberhart in 1995.
- In its basic form, each particle is updated based on the following:
  - Current velocity **{inertia}**
  - Personal best **{personal influence}**
  - Global best **{social influence}**



(PSO) UPDATING PARTICLE VELOCITY

- The basic form of the velocity update equation is as follows:

$$v_i^{t+1} = v_i^t + c1 \cdot r1(pb_i^t - x_i^t) + c2 \cdot r2(gb^t - x_i^t)$$

- where

$$v_i^{t+1} = new\ velocity$$
$$v_i^t = current\ velocity$$
$$c1, c2 = constants, usually\ 2$$
$$r1, r2 = random\ values\ between\ 0 - 1$$
$$pb_i^t = personal\ best\ at\ time\ t$$
$$x_i^t = current\ x\ position\ at\ time\ t$$
$$gb^t = global\ best\ at\ time\ t$$