Particle Swarm Optimization (PSO) Algorithm-Documentation

Supervised By

Dr: Ahmed Elngar

Assistant Professor at Faculty of Computers & Artificial Intelligence - Beni Suef University

Email: elngar_7@yahoo.co.uk

Submitted by

1- Mohamed Abdelrahman Aboelkassem

Computer Science Student at Beni Suef University, E-mail: mohamedabdelrahman345_sd@fcis.bsu.edu.eg

2- Mohamed Gamal Yaseen

Computer Science Student at Beni Suef University, E-mail: mohamedgamal336_sd@fcis.bsu.edu.eg

3- Mohamed Essam Abdelmoneem

Computer Science Student at Beni Suef University, E-mail: mohamedessam346_sd@fcis.bsu.edu.eg

4- Nourhan Mahmoud Hussein

Computer Science Student at Beni Suef University, E-mail: nourhanmahmoud382_sd@fcis.bsu.edu.eg

5- Ola Abdallah Mogawer

Computer Science Student at Beni Suef University, E-mail: olaabdallah319_sd@fcis.bsu.edu.eg

Introduction

Artificial Intelligence (AI) is the intelligence displayed by machines, where the intelligent agent represents a system that is aware of its environment and takes actions that increase its chances of success. Artificial intelligence research is very high-tech, very specialized, and divided into subfields that often fail to communicate with one another. Popular methods of AI today include traditional statistical methods, traditional symbolic artificial intelligence, and computational intelligence (CI). CI is a fairly new field of research. It is a set of nature-inspired computational methodologies and approaches to address complex real-world problems for which traditional approaches are ineffective or inapplicable. CI includes the artificial neural network (ANN), fuzzy logic, and evolutionary computation (EC). Swarm Intelligence (SI) is part of the EC. It investigates the collective behavior of self-organizing, natural or artificial decentralized systems.

Typical SI systems consist of a set of simple agents that interact locally with each other and with their environment. Inspiration often comes from the nature surrounding humans, especially biological systems and the behavior of other living things. Although in the SI system, agents follow very simple rules. There is no specific central control structure that clarifies how individuals should interact. Real customer behaviors at the local level, and to some extent randomly occurring; However, the interactions that occur between these factors give rise to "intelligent" global behavior.

The most well-known examples of SI include ant colonies, bird gathering, animal herding, and fish education. Dorigo proposed an ant colony improvement (ACO) method based on simulating an ant colony. Kennedy and Eberhart proposed a Particle Swarm Improvement (PSO) method based on bird flow simulations. These two are the most popular optimization algorithms affiliated with SI. Additionally, scholars have demonstrated a keen interest in proposing smart new approaches and ideas. Researchers Storn and Price proposed a differential evolution (DE). Karaboga and Basturk proposed an artificial bee colony (ABC), which mimics the foraging behavior of honey bees. Yang proposed a bat algorithm (BA), inspired by the echolocation behavior of microbes.

Reports and numbers prove that the total posts related to PSO are significantly higher than other algorithms, and the number of posts per year related to PSO is the highest among all the seven SI-based algorithms. This indicates that PSO is the most popular SI-based optimization algorithm. Therefore, we focus this review on PSO.

2. Definition

Particle swarm optimization (PSO) is defined as computational procedure to recruit or select the most effective element from a collection of accessible alternatives. Associated optimization drawback either deal with increment or minimization in the true operation for simplest state of affairs while constantly screening the input elements at intervals associated with allowed set of accessible alternatives.

Bird flocks, fish schools, and animal herds constitute representative examples of natural systems where aggregated behaviors are met, producing impressive, collision-free, synchronized moves. In such systems, the behavior of each group member is based on simple inherent responses, although their outcome is rather complex from a macroscopic point of view. For example, the flight of a bird flock can be simulated with relative accuracy by simply maintaining a target distance between each bird and its immediate neighbors. This distance may depend on its size and desirable behavior. For instance, fish retain a greater mutual distance when swimming carefree, while they concentrate in very dense groups in the presence of predators. The groups can also react to external threats by rapidly changing their form, breaking in smaller parts and re-uniting, demonstrating a remarkable ability to respond collectively to external stimuli in order to preserve personal integrity.

3.Basic Idea of PSO

- Each particle is searching for the optimum.
- Each particle is moving and hence has a velocity.
- Each particle remembers the position it was in where it had its best result so far (pbest).
- The particles in the swarm co-operate. They exchange information about what they've discovered. To get the global optimum (gbest)
- So each particle keeps track:
  - its best solution, personal best, pbest.
  - the best value of any particle, global best, gbest

4. Mechanism

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.

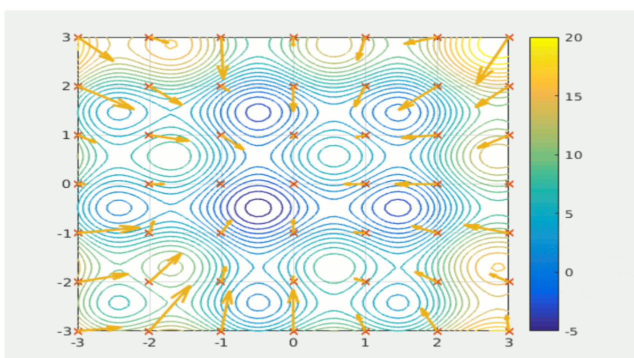
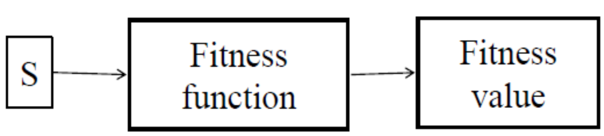


Fig 1: Visualization of the best positions during iterations on the contour plot

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the
particles. The particles fly through the problem space by following the current optimum particles.



PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest.
After finding the two best values, the particle updates its velocity and positions with following equations

$$V_{i+1} = \omega V_i + c_1 r_1 (p_i - x_i) + c_2 r_2 (p_g - x_i) \qquad (1)$$

$$x_{i+1} = x_i + V_{i+1} \qquad (2)$$

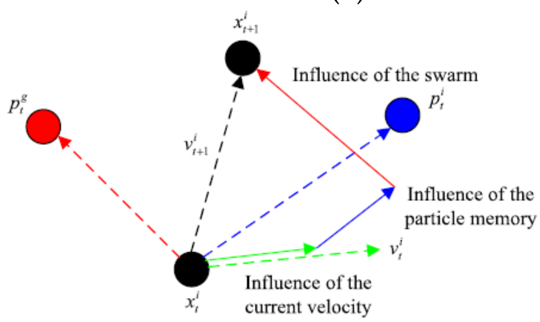


Fig 2: Iteration scheme of the particles

where

- ❑ x : particle's position
- ❑ v : path direction
- ❑ ω: is the inertia weight
- ❑ c_1: acceleration coeffection (cognitive factor)
- ❑ c_2: acceleration coeffection (social factor)
- ❑ p_i: best position of the particle
- ❑ p_g: best position of the swarm
- ❑ r_1, r_2: random variable from 0:1

5. PSO Algorithm

The basic algorithm of PSO Process:

1) Initialize the swarm (population) form the solution space.
2) Evaluate the fitness of each particle according to the objective function.
3) If a particle's current position is better than its previous best position, update it.
4) Determine the best particle according to the swarm.
5) Update particle's velocity: See formula (1).
6) Update particle's position: See formula (2).

Go to step2 and repeat until termination condition

6. Flowchart

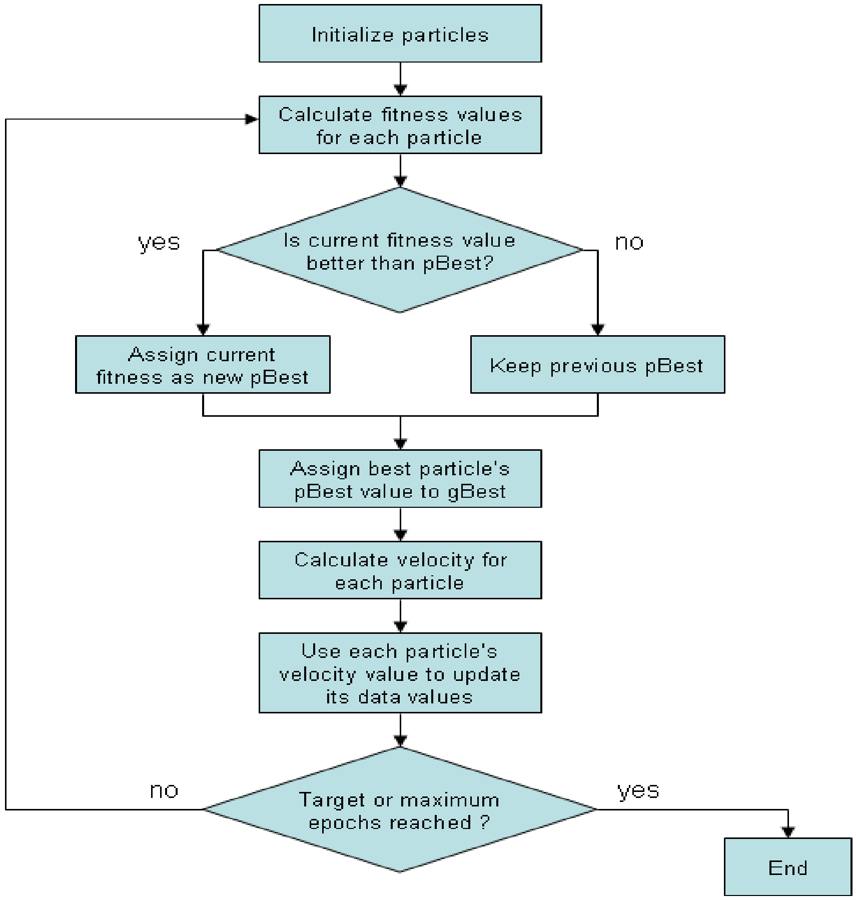


Fig 2: Flowchart of Particle Swarm Optimization Algorithm

7. Pseudo code

```
P=Particle_Initialization();
for i=1 to it-max
    for each particle p in P do
        fp = f(p)
        if fp is better than f(pBest)
            pBest = p;
        end
    end

    gBest=best p in P;
    for each particle p in P do
        v = v + c1*r1*(pBest - p) + c2*r2*(gBest - p)
        p =p + v
    end
end
```

8. Mathematical Formulation and Example

Problem: Find the maximum of the function
$f(x) = -x^2 + 5x + 20$ with $-10 \le x \le 10$ using the PSO algorithm. Use 9 particles with the initial positions
x_1 = -9.6, x_2 = -6, x_3 = -2.6, x_4 = -1.1, x_5 = 0.6 , x_6 = 2.3, x_7 = 2.8, x_8= 8.3, x_9 = 10.

Show the detailed computations for iterations 1, 2 and 3.

Solution:

Step 1: Choose the number of particles

x_1 = -9.6, x_2 = -6, x_3 = -2.6, x_4 = -1.1, x_5 = 0.6 , x_6 = 2.3, x_7 = 2.8, x_8= 8.3, x_9 = 10.

The initial population (i.e. the iteration number $t = 0$) can be represented as x^0, where

$i = 1, 2, 3, 4, 5, 6, 7,8, 9$

x_1^0 = -9.6, x_2^0 = -6, x_3^0 = -2.6, x_4^0 = -1.1, x_5^0 = 0.6, x_6^0 = 2.3, x_7^0 = 2.8, x_8^0 = 8.3, x_9^0 = 10.

Evaluate the objective function values as

$f_1^0 = -120.16$, f_2^0 = - 46 , f_3^0 = 0.24 , f_4^0 = 13.29 , f_5^0 = 22.64 , f_6^0 = 26.21 , f_7^0 = 26.16 , f_8^0= -7.39 , f_9^0= -30

Set the initial velocities of each particle to zero:

$v_1^0 , v_2^0 , v_3^0 , v_4^0 , v_5^0 , v_6^0 \; v_7^0, v_8^0, v_9^0 = 0$

Step 2: Set the iteration number as $t = 0 + 1 = 1$ and go to step 3.

Step 3: Find the personal best for each particle by

$$P_{best,i}^{t+1} = \begin{cases} P_{best,i}^{t} \rightarrow if \rightarrow f(x_i^{t+1}) > P_{best,i}^{t} \\ x_i^{t+1} \rightarrow if \rightarrow f(x_i^{t+1}) \le P_{best,i}^{t} \end{cases}$$

so

$P_{best,1}^1 = -9.6$, $P_{best,2}^1 = -6$, $P_{best,3}^1 = -2.6$, $P_{best,4}^1 = -1.1$, $P_{best,5}^1 = 0.6$, $P_{best,6}^1 = 2.3$, $P_{best,7}^1 = 2.8$, $P_{best,8}^1 = 8.3$, $P_{best,9}^1 = 10$

Step 4: Find the global best by

$$G_{best} = min\{P_{best,i}^{t} \; where \; i = 1,2,3,4,5,6,7,8,9$$

Since, the maximum personal best is:

$P_{best,6}^1 = 2.3$, $G_{best} = 2.3$

Step 5: Considering the random numbers in the range (0,1) As r_1^1=0.213 and r_2^1 =0.876 and find the velocities of the particles by V^{j+1}=v^{j}+cr^j[$P_{best,j}^{r}$-x_j^r]+ $c_2 r_2^1$[$G_{best,j}^{r}$-x_j^r] $where \; i = 1,2,3,4,5,6,7,8,9$

So

v_1^3= 4.4052, v_2^3= 3.0862, v_3^3= 1.8405, v_4^3=1.2909, v_5^3=0.6681, v_6^3= 0.053, v_7^3=-0.1380, v_8^3=-2.1531, v_9^3=-2.7759

Step 6: Find the new values of x_i^1, $i = 1,\ldots,9$ by x_i^{t+1} = x_i^t+ v_i^{t+1}

So

x_1^1=0.8244, x_2^1=1.2708, x_3^1= 1.6924, x_4^1=1.8784, x_5^1= 2.0892, x_6^1= 2.3, x_7^1= 2.362, x_8^1= 3.044,

x_9^1= 3.2548.

Step 7: Find the objective function values of x_i^1= $1,\ldots,9$:

f_1^1= 23.4424, f_2^1= 24.7391, f_3^1= 25.5978, f_4^1= 25.8636, f_5^1= 26.0812, f_6^1= 26.21, f_7^1 = 26.231,

f_8^1= 25.9541, f_9^1= 25.6803

Step 8: Stopping criterion:

If the terminal rule is satisfied, go to step 2, Otherwise stop the iteration and output the results.

Step 2: Set the iteration number as $t = 1 + 1 = 2$, and go to step 3.

Step 3: Find the personal best for each particle.

$P_{best,1}^2$= 0.8244, $P_{best,2}^2 =$ 1.2708, $P_{best,3}^2$ = 1.6924, $P_{best,4}^2$ =1.8784, $P_{best,5}^2$ = 2.0892,

$P_{best,6}^2$= 2.3438, $P_{best,7}^2$ =2.362, $P_{best,8}^2$= 3.044, $P_{best,9}^2$ = 3.2548.

Step 4: Find the global best.

G_{best} = 2.362

Step 5: By considering the random numbers in the range (0, 1) as r_1^3= 0.178 and r_2^3 = 0.507, find the velocities of the particles by v_i^{t+1} =v_i^t +$c_1 r_1^t[P_{best,i}^t$ -x_i^t]+ $c_2 r_2^t$ [G_{best} -x_i^t] $Where\ i = 1,\ldots..9$.

So

v_1^2=11.5099, v_2^2 =8.0412, v_3^2 =4.7651, v_4^2 =5.1982, v_5^2 = 1.6818,

v_6^2 = 0.0438, v_7^2 = -0.04380,v_8^2 =-5.7375, v_9^2 =-7.3755.

Step 6: Find the new values of x_i^2 $where\ i = 1,\ldots..9$ by $x_i^{t+1} = x_i^t + v_i^{t+1}$,

So

x_1^2 = 12.3343, x_2^2= 9.312, x_3^2= 6.4575, x_4^2=5.1982, x_5^2 = 3.7710,

x_6^2= 2.3438, x_7^2 =1.9240, x_8^2 = -2.6935, x_9^2 = -4.1207.

Step 7: Find the objective function values of f_i^2 $Where\ i\ =\ 1,\ldots.,9$;

f_1^2=-70.4644, f_2^2=-20.1532, f_3^2 = 10.5882 , f_4^2=18.9696,

f_5^2= 24.6346, f_6^2 = 26.2256, f_7^2= 25.9182, f_8^2 =-0.7224, f_9^2 =-17.5839.

Step 8: Stopping criterion:

If the terminal rule is fulfilled, go to step 2,Else stop the iteration and output the results.

Step 2: Set the iteration number as $t = 2 + 1 = 3$, and go to step3

Step 3: Find the personal best for each particle.

$P_{best,1}^3$ = 0.8244, $P_{best,2}^3$ = 1.2708, $P_{best,3}^3$ = 1.6924, $P_{best,4}^3$= 1.8784,

$P_{best,5}^3$ = 2.0892, $P_{best,6}^3$ = 2.3438, $P_{best,7}^3$= 2.362, $P_{best,8}^3$= 3.044, $P_{best,9}^3$ = 3.2548.

Step 4: Find the global best.

G_{best} = 2.362.

Step 5: since the random numbers in the $range$ (0,1) as $r_1^3 = 0.178$ and $r_2^3 = 0.507$ find the velocities of the particles by : V^{j+1}=$v^{\ j}$+cr^j[$P_{best,j}^r$-x_j^r]+ $c_2 r_2^1$[$G_{best,j}^r$-x_j^r] $Where\ i = 1, \dots, 9$. So

v_1^3= 4.4052 , v_2^3 = 3.0862, v_3^3 = 1.8405, v_4^3 = 1.2909, v_5^3 = 0.6681, v_6^3 = 0.053, v_7^3 = -0.1380,

v_8^3 = -2.1531, v_9^3 = -2.7759

Step 6: Find the new values of x_1^3 , $i = 1 \dots, 9$ by $x_i^{t+1} = x_i^t + v_i^{t+1}$ So

x_1^3 = 16.7395, x_2^3 = 12.3982, x_3^3 = 8.298, x_4^3 = 6.4892, x_5^3 = 4.4391, x_6^3 = 2.3968, x_7^3 = 1.786,

x_8^3 = -4.8466, x_9^3 = -6.8967.

Step 7: Find the objective function values of f_i^3 $Where\ i = 1, \dots 9$:

f_1^3 = -176.5145, f_2^3= -71.7244, f_3^3= -7.3673, f_4^3= 10.3367, f_5^3= -22.49, f_6^3= 26.2393, f_7^3= 25.7402, f_8^3= -27.7222, f_9^3= -62.0471.

Step 8: Stopping criterion:

If the terminal rule is fulfilled, go to step 2, Else stop the iteration and output the results. Lastly, the values of x_i^2 , $I = 1,2,3,4,5,6,7,8,9$.Did not converge, so we increment the iteration number as $t = 4$ and go to step2 When the positions of all particles converge to similar values, then the method has converged and the corresponding value of x_i^t is the optimum solution. Therefore the iterative process is continued until all particles meet a single value.

9. Types and comparison

There are different types of PSO methods which help to solve different types of optimization problems such as Multi-start (or restart) PSO for when and how to reinitialize particles, binary PSO (BPSO) method for solving discrete-valued problems, Multi-phase PSO (MPPSO) method for partition the main swarm of particles into sub-swarms or subgroups, Multi-objective PSO for solving multiple objective problems.

- **Particle Swarm Optimization (PSO)**

In this algorithm, population parameters were initialized randomly. The velocity of a particle is affected by three components: inertial momentum, social and cognitive components.

$$V_{i+1} = \omega V_i + C_1 r_1 (P_{best_i} - S_i) + C_2 r_2 (g_{best_i} - S_i)$$
$$S_{i+1} = S_i + V_{i+1}$$

- **Modified Particle Swarm Optimization (MPSO)**

In this algorithm, the birds have a memory about the previous best and worst positions so that particles have 2 experiences, a bad experience helps each particle to remember its previous worst position. To calculate the new velocity, the bad experience of each particle is considered with the next equation:

$$V_{i+1} = \omega V_i + C_{1g} r_1 (P_{best_i} - S_i) + C_{1b} r_2 (P_{worst_i} - S_i) + C_2 r_3 (g_{best_i} - S_i)$$

- **Weight Improved Particle Swarm Optimization (WIPSO)**

the WIPSO algorithm finds the answer with less iteration and with higher speed convergence among the proposed methods.
WIPSO algorithms have lower iteration numbers than PSO algorithm.
In the WIPSO algorithm, in order to improve the global search quality of standard PSO, the inertia weight factor and the cognitive and social components (C_1, C_2) have been configured with the next equation:

$$V_{i+1} = W_{new} V_i + C_1 r_1 (P_{best_i} - S_i) + C_2 r_2 (g_{best_i} - S_i)$$

where

$$W_{new} = W_{min} + w r_1$$

- **Multi-Start PSO (MSPSO)**

In the basic PSO, when particles start to converge to the same place, one of the major problems is lack of diversity. Several methods have been developed to constantly introduce randomness, or chaos, into the swarm to avoid this issue of the fundamental PSO. These types of methods are called the Particle Swarm Optimizer Multi-start (or restart) (MSPSO). The multi-start approach is a global search algorithm and has the primary objective of increasing diversity in order to explore greater portions of the search space. It is necessary to note that the swarm can never enter an equilibrium state by continuous injection of random positions, which is why the amount of chaos is reduced in this algorithm.
Initializing position vectors or velocity vectors of particles at random will increase the swarm's diversity. By randomly initializing locations, particles are physically moved to a different random location in the solution space. When position vectors are kept constant and velocity vectors are randomized, particles retain their memory of the best solutions today and before, but are forced to look in various random directions. If the velocity of the randomly initialized particle does not find a better solution, then the particle will be drawn to its best personal location again. When particle locations are reinitialized, the velocities of the particles are usually set to zero and to have a zero momentum at the first iteration after reinitialization. On the other hands, particle velocities can be initialized to small values.
In SOCPSO model the velocity of each particle is updated by

$$v_{ij}^{t+1} = X[\omega v_{ij}^{t} + \phi_{1j}(P_{best_i} - x_{ij}^{t}) + \phi_{2j}(G_{best} - x_{ij}^{t})]$$

- **Multi-phase PSO (MPPSO)**

Multi-phase PSO (MPPSO) method partitions the main swarm of particles into sub-swarms or subgroups, where each sub-swarm performs a different task, exhibits a different behavior and so on. This task or behavior performed by a sub-swarm usually changes over time and information are passed among sub-swarms in this process.

- **Attraction phase:** in this phase, the particles of the corresponding sub-swarm are influenced to move towards the global best position.
- **Repulsion phase**: in this phase, the particles of the corresponding sub-swarm go away from the global best position.

In MPPSO algorithm, the particle velocity updating equation is presented as follows:

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 x_{ij}^{t} + c_1 G_{best}$$

- **Perturbed PSO (PPSO)**

There are several drawbacks to the simple particle swarm optimization (PSO). for example, high speed of convergence also leads to a rapid loss of diversity during the optimization process. Then, the mechanism leads to unnecessary premature convergence. Zhao Xinchao identified a disturbed particle swarm algorithm to resolve this disadvantage, which is based on a new strategy for particle updating and the principle of disturbed global best (p-gbest) within the swarm. The disturbed global best (p-gbest) updating strategy is based on the principle of measure of possibility to model the lack of data on the gbest's true optimality. The particle velocity in PPSO is rewritten by:

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 r_{1j}^{t}[P_{best,i}^{t} - x_{ij}^{t}] + c_2 r_{2j}^{t}[G'_{best} - x_{ij}^{t}]$$

Where

$$G'_{best} = N(G_{best}, \sigma)$$

- **Multi-Objective PSO (MOPSO)**

Multi-objective optimization problems have several objective functions that need to be optimized simultaneously. In multiple-objectives cases, due to lack of common measure and confliction among objective functions, there does not necessarily exist a solution that is best with respect to all objectives. There exist a set of solutions for the multi-objective problem which cannot normally be compared with each other. Such solutions are called non-dominated solutions (or Pareto optimal solutions) only when no improvement is possible in any objective function without sacrificing at least one of the other objective functions.

In multi-objective optimization algorithms, these cases are considered the most difficult.

- **Dynamic Neighborhood PSO (DNPSO)**

The multiple objectives are divided into two classes in this algorithm: f1 and f2, where f1 is defined as the objective of the neighborhood, and f2 is defined as the objective of optimization. The f1 and f2 options are random. Each particle defines a new neighborhood dynamically in each iteration by measuring the distance to all other particles and selecting the nearest neighbors. The distance for the first group of objective functions f1 is defined as the difference between fitness values. The best local value is selected among the neighbors in terms of the fitness value of the second objective function f2 when the neighborhood has been determined. Finally, the global best updating system considers only the solution that dominates the current personal best value. The main drawback is that it is useful only for two objectives.

- **Multi-Objective PSO (MOPSO)**

In the algorithm here. The first approach is that only one objective function at a time tests each particle, and the finding of the best locations is carried out in a similar way to the case of single-objective optimization. The key challenge in such cases is to properly handle the data coming from and objective function so that the particles go towards optimal solutions for Pareto. The most trivial solution for preserving the defined optimal Pareto solutions would be to store non-dominated solutions as the best positions for the particles. The second approach is that, based on the principle of Pareto optimality, each particle is measured by all objective functions, and they produce non-dominated best positions (called leaders). Nevertheless, there can be many non-dominated solutions in the neighborhood of a particle, and the determination of leaders is not easy. On the other hand, only the one that will be used as the best position of a particle is usually selected to participate in the velocity update. Therefore, the selection of the best position is an important task in making particles move to the Pareto optimal front.

$$v_{ij}^{[S]}(t+1) = \omega v_{ij}^{[S]}(t) + c_1 r_1 \left[P_{best\,,i}^{[s]}(t) - x_{ij}^{[S]}(t)\right] + c_2 r_2 [G_{best}^{[q]} - x_{ij}^{[S]}(t)]$$

- **Vector Evaluated PSO (VEPSO)**

S defines the swarm number ($s = 1{,}2{,}3, k$)

i corresponds to the particle number($i = 1{,}2{,}3,\ \ n$)

$v_{ij}^{[S]}$ is the velocity of the -th particle in the -th swarm

$G_{best}^{[q]}$ is the best position found for any particle in the -th swarm

which is evaluated with the -th objective function.

The VEPSO algorithm is called parallel VEPSO because this algorithm also enables the swarms to be implemented in parallel computers that are connected in an Ethernet network.

- **Binary PSO (BPSO)**

The PSO algorithm was developed and most of its updated versions operated in continuous spaces that could not be used to optimize discrete-valued search spaces. In order to function on binary search spaces, they created the PSO, since real-value domains can be converted into binary-value domains. The proposed algorithm is called the binary PSO (BPSO) algorithm, in which the particles represent position in binary space and the position vectors of the particles can take the binary value 0 or 1, i.e. In this case, it maps the n-dimensional binary space (i.e. n-length bit strings) to actual numbers. (where is a fitness function and a real number set respectively). That means a particle's positions must belong to, in order to be calculated by. In BPSO, a particle's velocity is connected to the possibility that the particle's position takes a value of 0 or 1. The update equation for the velocity does not change from that used in the original PSO and the equation.

$$v_{ij}^{t+1} = v_{ij}^{t} + c_1 r_{1j}^{t}\left[P_{best_i} - x_{ij}^{t}\right] + c_2 r_{2j}^{t}\left[G_{best} - x_{ij}^{t}\right]$$

10. Advantages and Disadvantages

It is said that PSO algorithm is the one of the most powerful methods for solving the non-smooth global optimization problems while there are some disadvantages of the PSO algorithm. The advantages and disadvantages of PSO are discussed below:

- Advantages of the PSO algorithm:

The main advantage of using the PSO is its simple concept and ability to be implemented in a few lines of code. Furthermore, PSO also has a short-term memory, which helps the particles to fly over the local best and the global best positions. Alternatives, such genetic algorithms (GA), are more complex and, most of the time, they do not consider the previous iteration or the collective emergent performance. For instance, in GA, if a chromosome is not selected, the information contained by that individual is lost.

- Disadvantages of the PSO algorithm:

Despite its features, a general problem with the PSO, similarly to other optimization algorithms that are not exhaustive methods, such as the brute-force search (Schaeffer et al. 1993), is that of becoming trapped in a local optimum, or sub optimal solution, such that it may work well on one problem but yet fail on another problem.

In general, the main drawbacks of PSO can be summarized as follows.

1) Premature convergence of a swarm: Particles try to converge to a single point. located on a line between the global best and the personal best positions (local best). This point is not guaranteed for a local optimum (Van den Bergh and Engelbrecht 2004). Another reason could be the fast rate of information flow between particles, which leads to the creation of similar particles. This results in a loss in diversity and the possibility of being trapped in local optima is increased (Premalatha and Natarajan 2009).
2) Parameter settings dependence. This leads to the high-performance variances for a stochastic search algorithm (Premalatha and Natarajan 2009). In general. there is not any specific set of parameters for different problems. As an example, and by simple observation of $v_n[t+1] = wv_n[t] + \sum_{i=1}^{2} p_i\, r_i(x_{i_n}[t] - x_n[t])$, increasing the inertia weight w will increase the speed of the particles vi+ 1) and cause more exploration (global search) and less exploitation (local search). As a result, finding the best set of parameters is not a trivial task, and it might be different from one problem to another (Premalatha and Natarajan 2009)

11. Difference between Genetic Algorithm and PSO

- Commonalities

- Both algorithms start with a group of a randomly generated population
- Both have fitness values to evaluate the population.
- Both update the population and search for the optimum with random techniques.
- Both systems do not guarantee success.

- Differences

- PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity.
- In GAs, chromosomes share information with each other, So the whole population moves like a one group towards an optimal area.
- In PSO, only gBest gives out the information to others. It is a one -way information sharing mechanism.
- In GAs, the evolution only looks for the best solution.

- In PSO, all the particles tend to converge to the best solution quickly.

12 Some applications that use PSO.

- Solving Travelling Salesman Problem (TSP)
- Particle Swarm Optimization Methods for Pattern Recognition and Image Processing
- Training Neural Network
- Control applications.
- Sensors and sensor networks.
- Antenna design.
- Applications in computer graphics and visualization.
- Design applications.
- Modeling.
- Signal processing.
- Robotics.
- Prediction and forecasting.

13. Experimental Results and Discussion

This is an experiment work done to remove Backpropagation and in-turn Gradient Descent and use Particle Swarm Optimization technique for Neural Network Training for optimizing the network's weights and biases.

TABLE 1: Datasets description.

Dataset	Dimension	Samples	Classes
MNIST	784	1347	10

After doing the experiment for training Neural Networking for training MNIST dataset using Particle Swarm Optimization with only 500 iterations. The final result of training NN get Bbest at iteration 499 with mean error: 0.0533665225281364, and accuracy: 0.5777777777777777.

The following table show the report of training test dataset of MNIST with main measures.

TABLE 2: Report of Testing Dataset

	Precision	Recall	F1-score	Support
0	0.92	0.83	0.87	41
1	0.00	0.00	0.00	42
2	0.71	0.57	0.63	42
3	0.00	0.00	0.00	52
4	0.97	0.71	0.83	45
5	0.82	0.65	0.73	43
6	0.00	0.00	0.00	49
7	0.77	0.51	0.62	47
8	0.00	0.00	0.00	44
9	0.66	0.51	0.57	45

Micro avg	0.84	0.46	0.60	450
Macro avg	0.58	0.47	0.52	450
Weighted avg	0.58	0.46	0.51	450
Samples avg	0.46	0.46	0.46	450

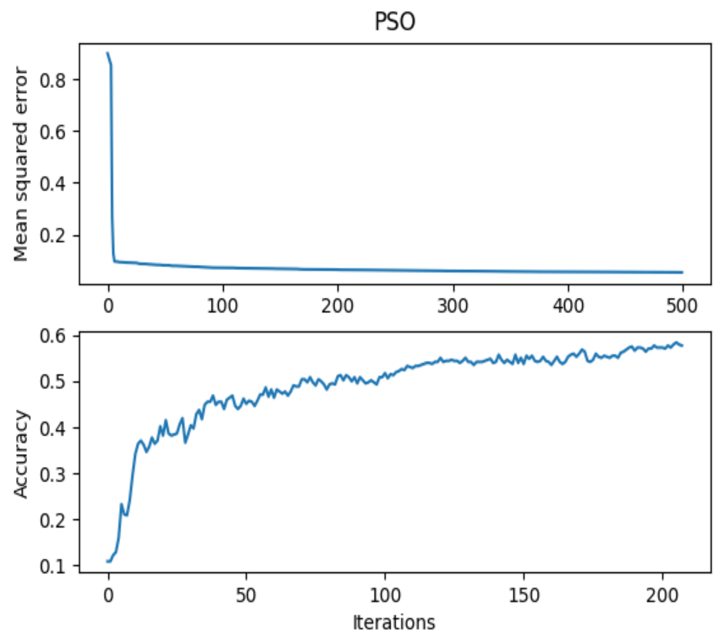


Fig 3: Performance and behavior of the Neural Network over 500 iterations with MNIST Dataset