Evolutionary Algorithms documentation

1. Project Idea:

The project aims to implement and compare four Swarm Intelligence algorithms to find the global minimum of three benchmark optimization functions. The selected algorithms are Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO).The optimization functions chosen for evaluation are Rastrigin Function, Cross-in-Tray, and Drop-Wave.

2. Main Functionalities:

- Implementation of PSO and GWO algorithms.
- Evaluation of the algorithms' performance in finding the global minimum of benchmark optimization functions.
- Comparison of the effectiveness and efficiency of PSO and GWO in optimization tasks.
- Visualization of the optimization process and results.

3. Similar Applications in the Market:

Similar applications include optimization software tools such as MATLAB's Optimization Toolbox, SciPy's optimization module, and commercial optimization software like Gurobi and CPLEX. However, this project focuses specifically on comparing PSO and GWO algorithms for benchmark optimization functions, offering insights into their relative performance in solving optimization problems.

4. **Literature Review:**

- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks (pp. 1942-1948).

- Mirjalili, S., & Mirjalili, S. M. (2014). A comprehensive review of nature-inspired algorithms. Swarm and Evolutionary Computation, 13, 34-46.

- Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. In Proceedings of World Congress on Nature & Biologically Inspired Computing (pp. 210-214).

- Mirjalili, S., & Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm and Evolutionary Computation, 9, 1-14.

5. Details of Algorithm(s)/Approach(es) Used and Results:

- PSO Algorithm: Implemented using Python, the PSO algorithm is applied to the Ackley, Cross-in-Tray, and Drop-Wave functions.

Results will include convergence plots, optimization trajectories, and the discovered global minimum.

- GWO Algorithm: Also implemented in Python, the GWO algorithm is compared with PSO on the same benchmark functions. Performance metrics such as convergence speed and solution accuracy will be analyzed and compared.

6. Development Platform:

The project is developed using Python programming language with libraries such as NumPy and Matplotlib for algorithm implementation and visualization. Jupyter Notebook is utilized for code development and documentation, providing an interactive environment for experimentation and analysis. Version control is managed using Git, with the project repository hosted on GitHub for collaboration and version tracking.

Particle Swarm Optimization (PSO):

-is a computational method inspired by the social behavior of bird flocking or fish schooling. It's a population-based stochastic optimization technique used to solve optimization problems.

Main ideas of PSO:

1. Population Representation: In PSO, a population of potential solutions is represented as a swarm of particles. Each particle represents a potential solution to the optimization problem.

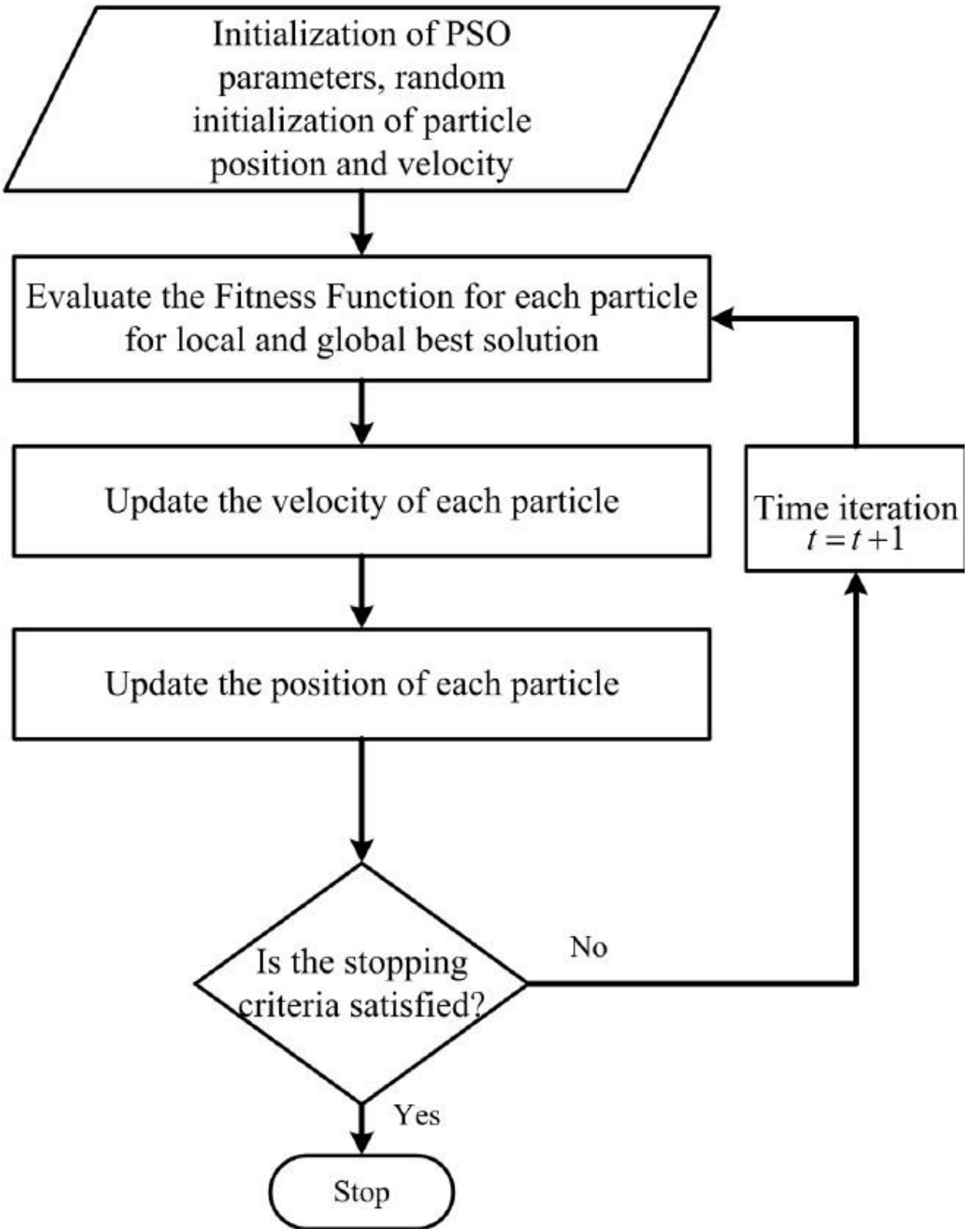
2. Fitness Evaluation: Each particle's fitness or quality is evaluated based on a fitness function, which quantifies how close the particle's solution is to the optimal solution of the problem.

3. Velocity and Position Update: Each particle adjusts its position and velocity in the search space based on its own experience and the experience of its neighboring particles. This adjustment is guided by two main components: the particle's previous best-known position (pBest) and the global best-known position among all particles in the swarm .

4. Exploration and Exploitation: PSO balances exploration (searching widely in the solution space to find promising regions) and exploitation (narrowing down the search around promising regions) through the particles' movements.

5. Iteration: The process iterates for a certain number of iterations or until a termination criterion is met (best solution is found).

Particle Swarm Optimization Algorithm’s flowchart:



Gray Wolf Optimizer(GWO):

Is a relatively recent metaheuristic algorithm inspired by the social behavior and hunting strategies of gray wolves. Developed by Seyedali Mirjalili and Seyed Mohammad Mirjalili in 2014, it's designed to solve optimization problems.

Main ideas of GWO:

1. The algorithm models the social hierarchy and hunting behavior of gray wolves. In a wolf pack, there are typically alpha, beta, delta, and omega wolves, representing the highest to lowest ranks in the pack. The algorithm simulates the hunting process of these wolves, where they collaborate to locate and pursue prey.

2. Population Representation: Similar to other population-based metaheuristic algorithms like Genetic Algorithms and Particle Swarm Optimization, GWO maintains a population of candidate solutions. Each solution is represented as a wolf in the search space.

3. Search Space Exploration: Initially, the positions of the alpha, beta, and delta wolves are determined randomly within the solution space. These wolves represent the best solutions found so far.

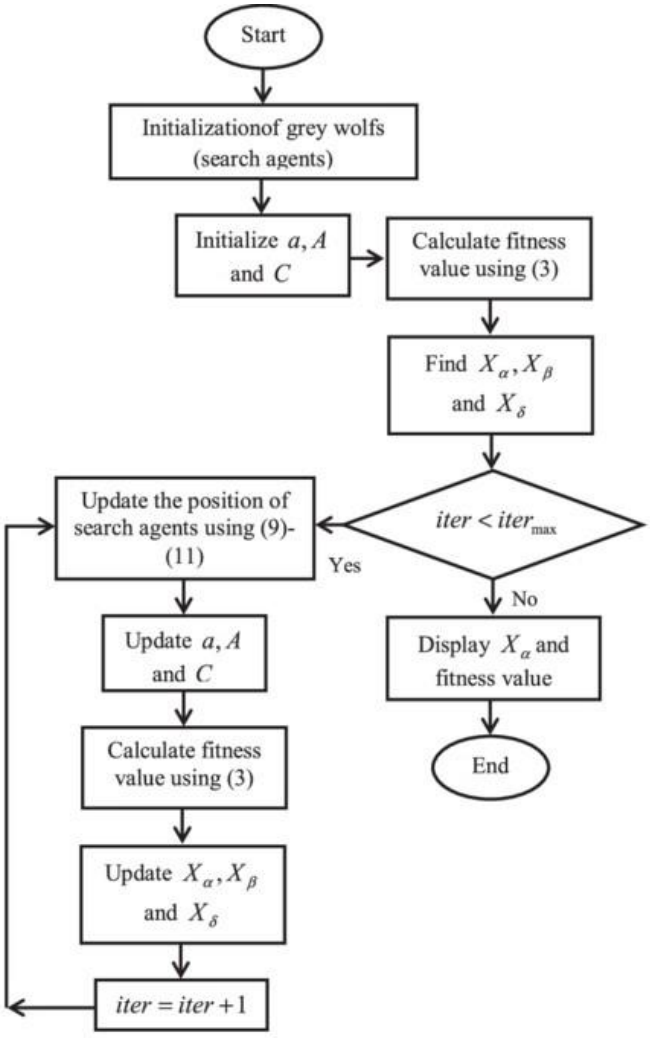
4. Iterative Improvement: During each iteration, the position of each wolf is updated based on its current position and the positions of the alpha, beta, and delta wolves. This update is influenced by three main factors: exploration (seeking new solutions),

exploitation (refining promising solutions), and convergence towards the global optimum.

5. Leadership and Collaboration: The alpha wolf is considered the leader, followed by the beta and delta wolves. Other wolves (omega wolves) in the pack adjust their positions relative to the leaders, mimicking the hunting strategy where subordinate wolves follow the lead of the pack's leaders.

6. Convergence: As the algorithm progresses, the positions of the alpha, beta, and delta wolves converge towards the optimal solution, leading to improved solutions in subsequent iterations.

GWO flowchart:


Start
Initializationof grey wolfs (search agents)
Initialize a, A and C
Calculate fitness value using (3)
Find X_α, X_β and X_δ
$iter < iter_{max}$
Yes
No
Update the position of search agents using (9)-(11)
Update a, A and C
Calculate fitness value using (3)
Update X_α, X_β and X_δ
$iter = iter + 1$
Display X_α and fitness value
End

PSO paper:

Reference:

Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review | Archives of Computational Methods in Engineering (springer.com)

Particle Swarm Optimization: PSO Mechanism
Self-Organization Features SI system has a major feature, namely, self-organization, in which the components of an initially disordered system interact locally to produce a coordination or global order. This process is characterized by spontaneousness; that is, no agent inside or outside of the system dominates the interaction. The self-organization in swarms was interpreted by Bonabeau et al. [25] through three key components as follows: • Robust dynamical nonlinearity (always comprising positive and negative feedback) convenient structures are promotionally being created with the help of positive feedback, while this positive feedback is counterbalanced and the collective pattern is stabilized with the help of negative feedback. • Trade-of between exploration and exploitation A valuable mean creativity artificial approach is provided through a suitable balance that is identifed by SI. • Multiple interactions Information coming from neighbor agents in the swarm are used by individual agents, allowing information to be disseminated throughout the network

4 PSO Pseudocode:

Algorithm 1 PSO pseudocode

Input:
N – Swarm size
D – Problem dimensionality
T – Maximum number of iterations
LB – Lower bound of the search space
UB – Upper bound of the search space

Output:
$\mathbf{g}^t_{best}$ – the best position (solution) found so far

Start
  Initialize the swarm randomly;
  for $i = 1$ to N **do** ▷ Iterate through the swarm
    $\mathbf{v}^0_i \leftarrow$ a random vector within $[LB, UB]^D$; ▷ Initialize particles' velocity using a uniform distribution
    $\mathbf{x}^0_i \leftarrow$ a random vector within $[LB, UB]^D$; ▷ Initialize particles' positions using a uniform distribution
    $\mathbf{p}^0_{best_i} \leftarrow \mathbf{x}^0_i$; ▷ Initialize $\mathbf{p}_{best}$ to its initial position
  end for
  Apply Eq. (2) to find $\mathbf{g}^0_{best}$; ▷ Initialize $\mathbf{g}_{best}$ to position with the minimum fitness value
  $t \leftarrow 1$; ▷ Initialize first iteration number
  while $t \leq T$ **do**
    for $i = 1$ to N **do** ▷ Iterate through the swarm
      $\boldsymbol{r_1}, \boldsymbol{r_2} \leftarrow$ two independent vectors randomly generated from $[0, 1]^D$;
      Apply Eq. (3); ▷ Update particle's velocity
      Apply Eq. (4); ▷ Update particle's position
      if $f(\mathbf{x}^t_i) < f\left(\mathbf{p}^{t-1}_{best_i}\right)$ **then** ▷ If new solution is better than current personal best
        $f\left(\mathbf{p}^t_{best_i}\right) \leftarrow f(\mathbf{x}^t_i)$; ▷ Update the best known position of the particle
      end if
    end for
    Apply Eq. (2) to find $\mathbf{g}^t_{best}$; ▷ Update the swarm's overall best known position
    $t \leftarrow t + 1$;
  end while ▷ Maximum iteration number is reached or termination criterion is satisfied
End

GWO paper:

Reference:

[(PDF) Grey Wolf Optimizer (researchgate.net)]

3. Grey Wolf Optimizer (GWO)

In this section the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

3.1. Inspiration

Grey wolf (Canis lupus) belongs to Canidae family. Grey wolves are considered as apex predators, meaning that they are at the top of the food chain. Grey wolves mostly prefer to live in a pack. The group size is 5-12 on average. Of particular interest is that they have a very strict social dominant hierarchy as shown in Fig. 1.

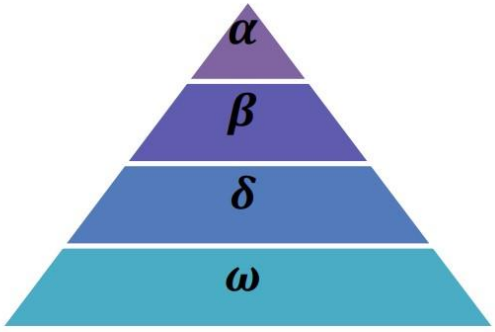


Fig. 1. *Hierarchy of grey wolf (dominance decreases from top down)*

3.2.2. Encircling prey:

As mentioned above, grey wolves encircle prey during the hunt. In order to mathematically model encircling behavior the following equations are proposed:

$$\vec{D} = |\vec{C}.\overrightarrow{X_p}(t) - \vec{X}(t)| \tag{3.1}$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}.\vec{D} \tag{3.2}$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors , $\overrightarrow{X_p}$ is the position vector of the prey, and $\vec{X}$ indicates the position vector of a grey wolf.

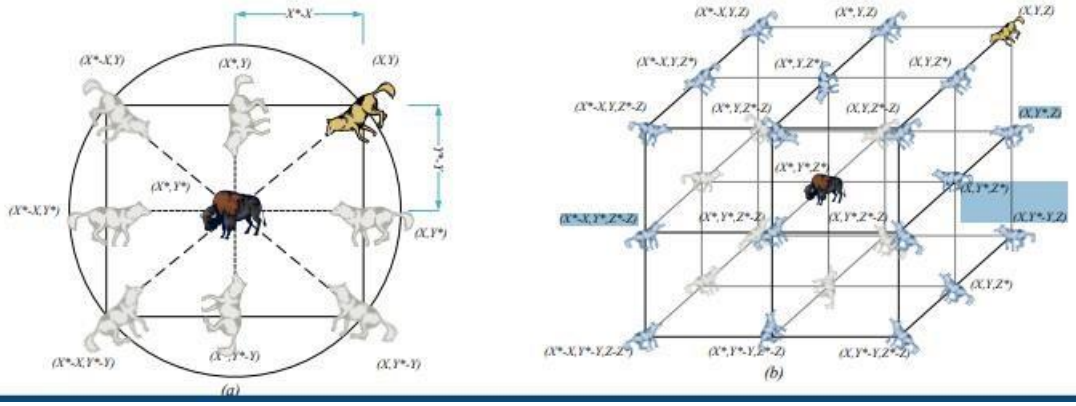
The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a} \tag{3.3}$$

$$\vec{C} = 2.\overrightarrow{r_2} \tag{3.4}$$

where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and r_1, r_2 are random vectors in [0,1].

To see the effects of equations (3.1) and (3.2), a two-dimensional position vector and some of the possible neighbors are illustrated in Fig. 3 (a). As can be seen in this figure, a grey wolf in the position of (X,Y) can update its position according to the position of the prey (X^*,Y^*). Different places around the best agent can be reached with respect to the current position by adjusting the value of $\vec{A}$ and $\vec{C}$ vectors. For instance, $(X^*\text{-}X,Y^*)$ can be reached by setting $\vec{A} = (1,0)$ and $\vec{C} = (1,1)$. The possible updated positions of a grey wolf in 3D space are depicted in Fig. 3 (b). Note that the random vectors r_1 and r_2 allow wolves to reach any position between the points illustrated in Fig. 3. So a grey wolf can update its position inside the space around the prey in any random location by using equations (3.1) and (3.2).



GWO pseudocode:

```
Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=the third best search agent
while (t < Max number of iterations)
    for each search agent
        Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, and X_δ
    t=t+1
end while
return X_α
```

Fig. 6. *Pseudo code of the GWO algorithm*

To see how GWO is theoretically able to solve optimization problems, some points may be noted:

- The proposed social hierarchy assists GWO to save the best solutions obtained so far over the course of iteration
- The proposed encircling mechanism defines a circleshaped neighborhood around the solutions which can be extended to higher dimensions as a hyper-sphere
- The random parameters A and C assist candidate solutions to have hyper-spheres with different random radii
- The proposed hunting method allows candidate solutions to locate the probable position of the prey
- Exploration and exploitation are guaranteed by the adaptive values of a and A • The adaptive values of parameters a and A allow GWO to smoothly transition between exploration and exploitation

- With decreasing A, half of the iterations are devoted to exploration (|A|≥1) and the other half are dedicated to exploitation (|A|<1)
- The GWO has only two main parameters to be adjusted (a and C).

Problems comparing:

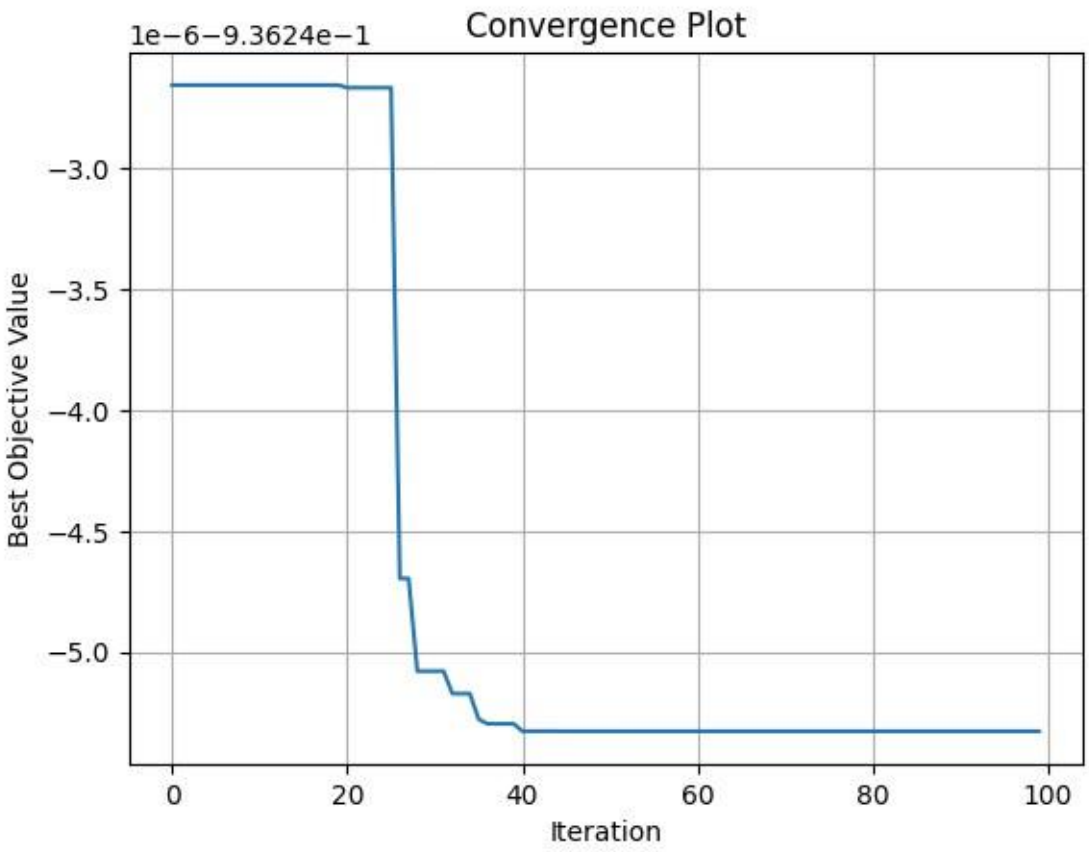
1(DropWave function :

Algorithm:	PSO	GWO	Original funcglobal min

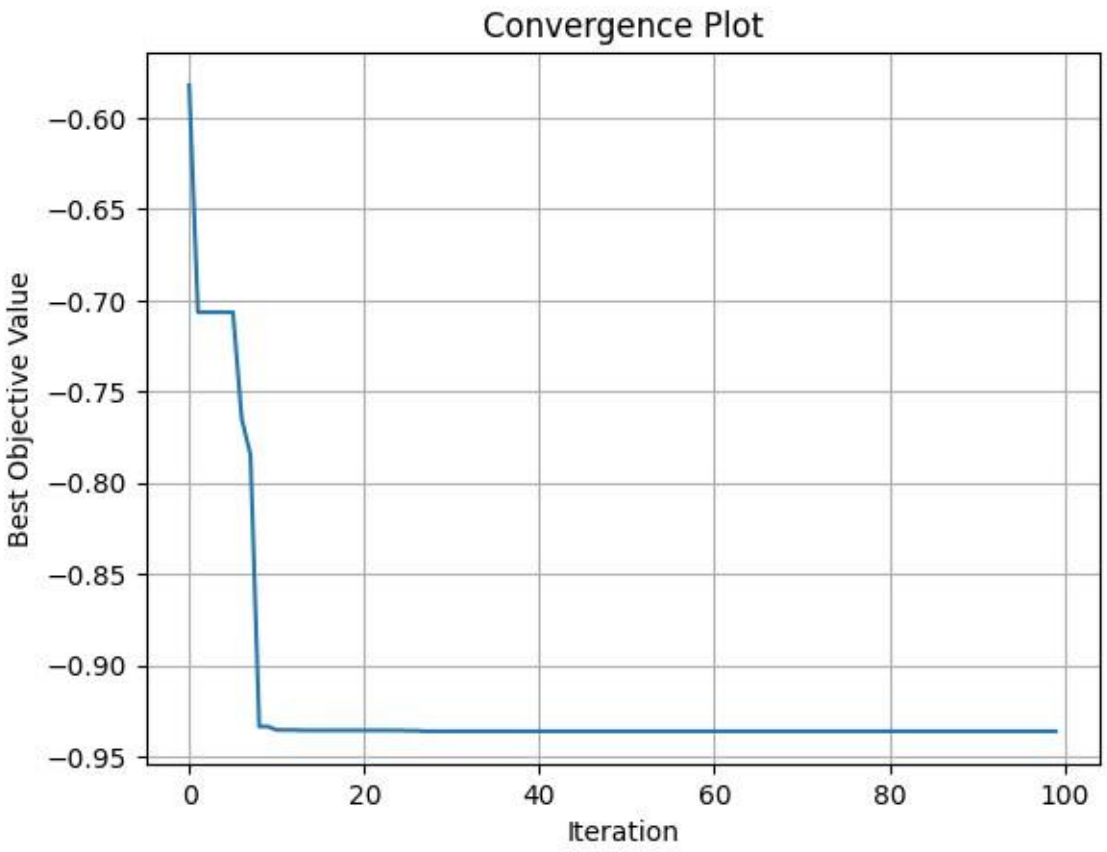
Avg of best solutions through running 30 times	-0.9829950054185971	-0.942618141135305	f(x*) = -1

- By comparing the two values ,we that using PSO at this function is better than GWO.

PSO convergence plot:



GWO convergence plot:



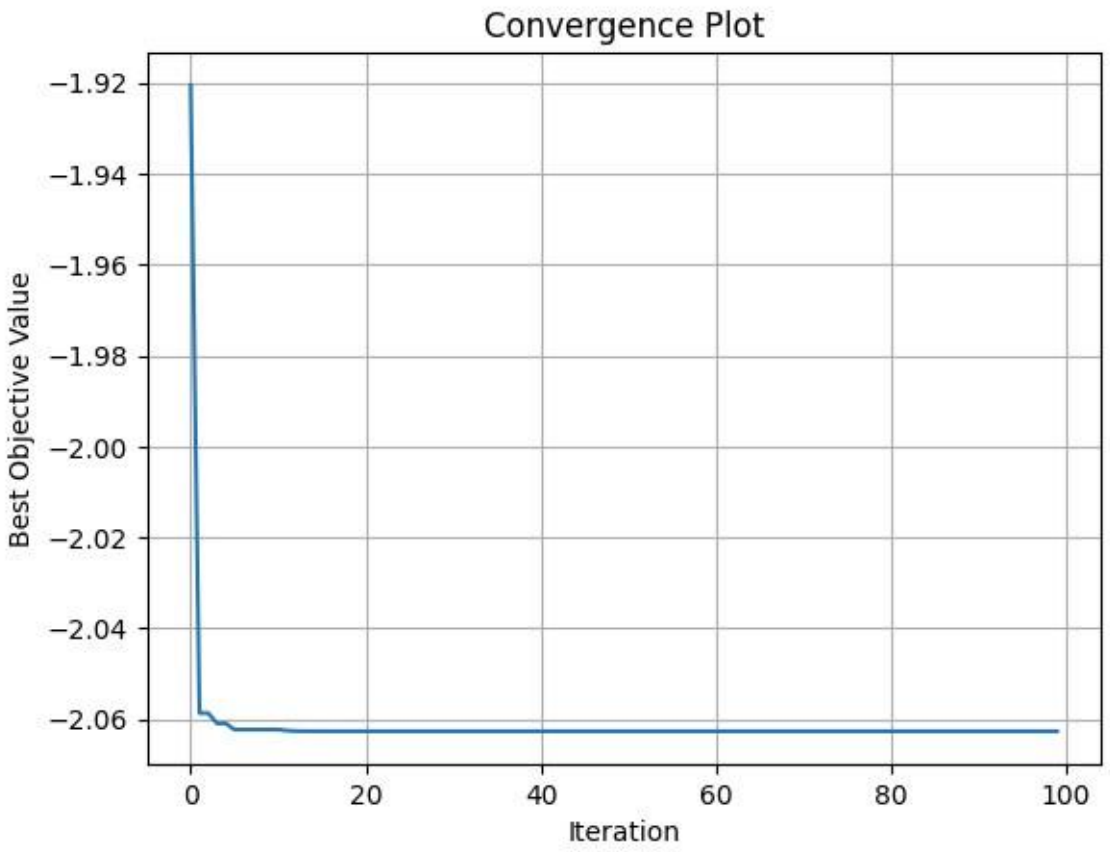
N.B : these results are from running every algorithm 30 times and calculating rhe average of these results.

2)CROSS-IN-TRAY function:

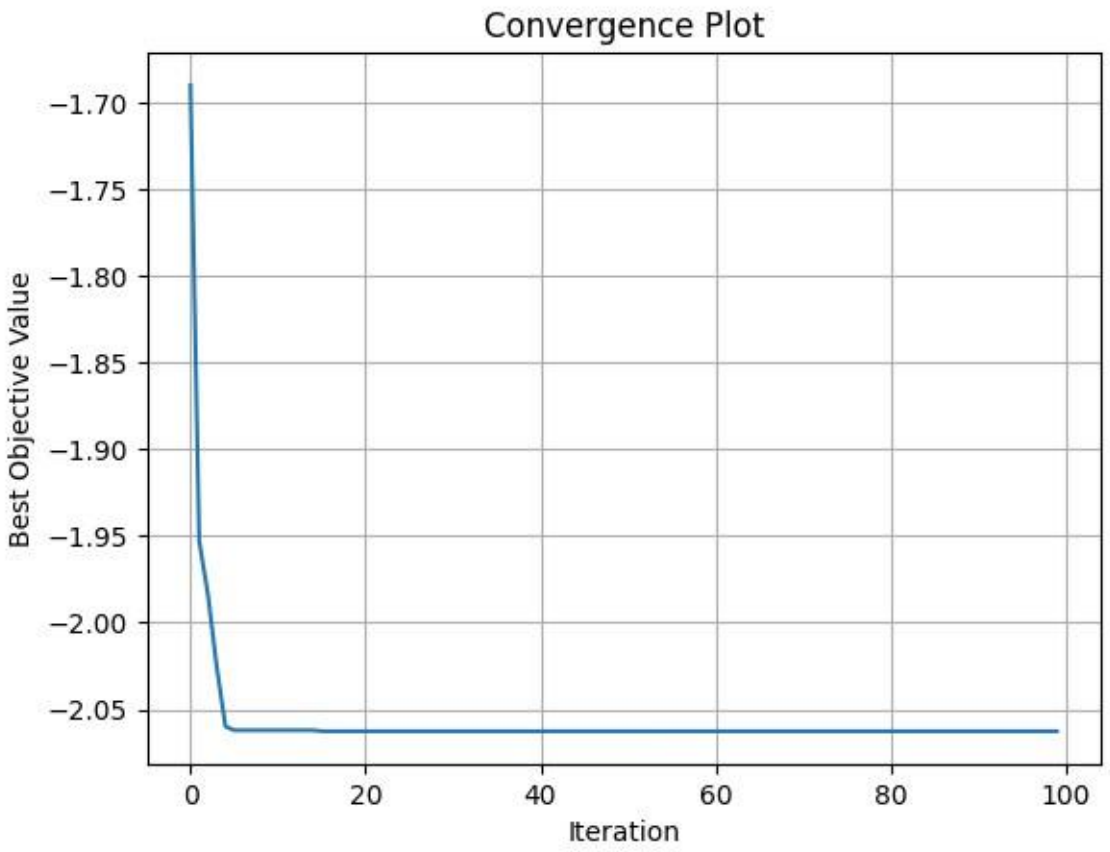
- By comparing the two values ,we that using GWO at this function is better than PSO.

Algorithm:	PSO	GWO	Original funcglobal min
Avg of best solutions through running 30 times	-2.06261187082274	-2.0626109953842198	f(x*) = -2.06261

PSO convergence plot:



GWO convergence plot:

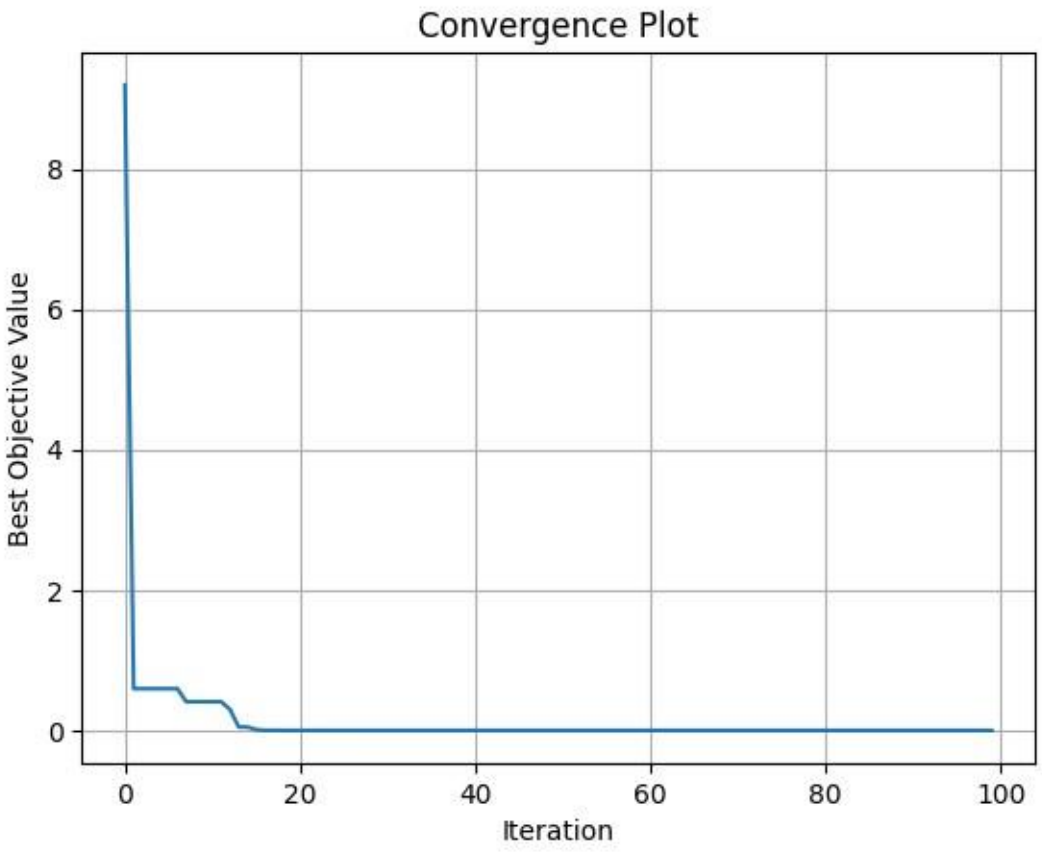


3) RASTRIGIN function :

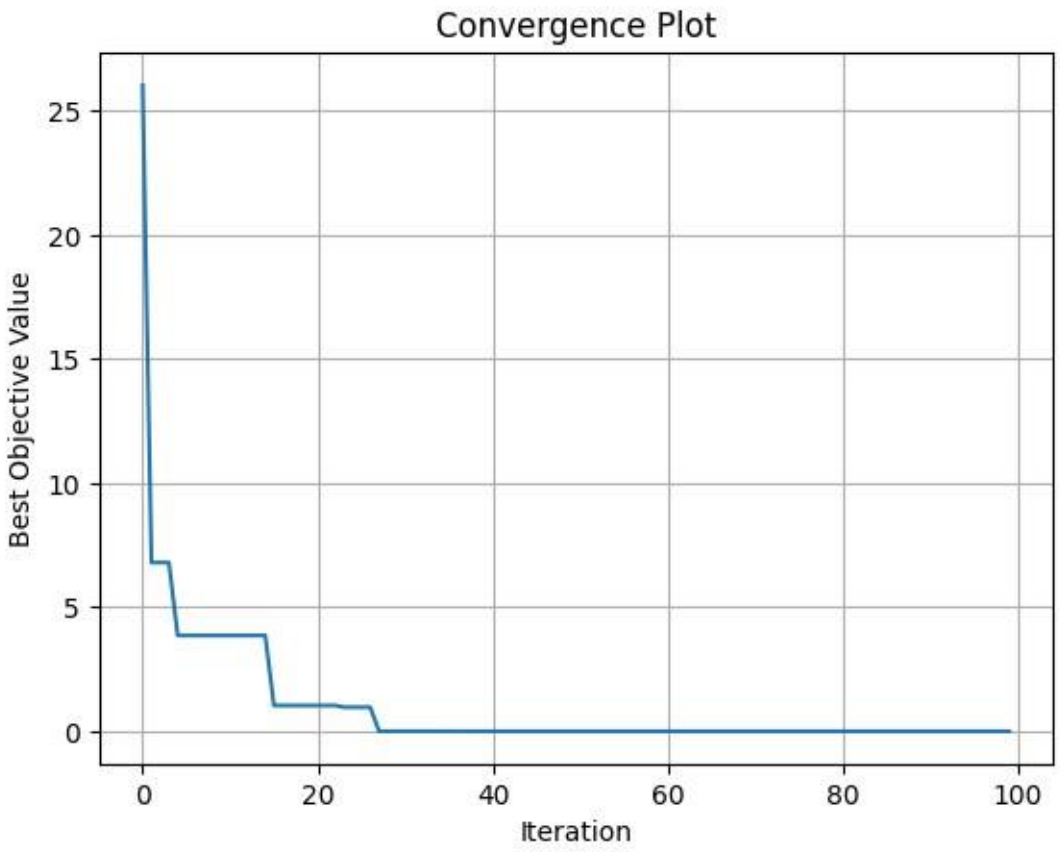
Algorithm:	PSO	GWO	Original funcglobal min
Avg of best solutions through running 30 times	0.06633060380621932	0.27883407067080174	f(x*) = 0

- By comparing the two values ,we that using PSO at this function is better than GWO.

PSO convergence plot:



GWO convergence plot:



Different initializations:

GWO with Rastrigin function:

Number of runs: 80	Number of runs:50
Number of wolves:30	Number of wolves:20
Number of Dimensions:2	Number of Dimensions:2
Number of iterations:150	Number of iterations:150
Lower Bound:-2	Lower Bound:-30

Upper Bound:2	Upper Bound:30
Average objective value:0.0124391102324 62802	Average objective value:0.079635147507 88299

PSO with Rastrigin function:

Dimension:2	Dimension:2
Lower Bound:-30	Lower Bound:-2
Upper Bound:30	Upper Bound:2
Number of particles:50	Number of particles:50
Max iterations:150	Max iterations:50
Number of runs:50	Number of runs:80
Average best score:0.0	Average best score:3.8110795520740482e-06

GWO with Cross in tray function:

Number of runs:50	Number of runs:80
Number of wolves:20	Number of wolves:20

Number of Dimensions:2	Number of Dimensions:2
Number of iterations:150	Number of iterations:150
Lower Bound:-30	Lower Bound:-2
Upper Bound:30	Upper Bound: 2
Average objective value: -2.062611714764542	Average objective value:-2.06261169582630 28

PSO with Cross in tray function:

Dimension:2	Dimension:2
Lower Bound:-30	Lower Bound:-2
Upper Bound:30	Upper Bound:2
Number of particles:50	Number of particles:50
Max iterations:100	Max iterations:150
Number of runs:50	Number of runs:80
Average best score:-2.06261187088227392	Average best score:-2.0626118708227392

GWO with Drop wave function:

Number of runs:80	Number of runs:50

Number of wolves:20	Number of wolves:30
Number of Dimensions:2	Number of Dimensions:2
Number of iterations:150	Number of iterations:100
Lower Bound:-2	Lower Bound:-30
Upper Bound:2	Upper Bound: 30
Average objective value: -0.9689194862285525	Average objective value:-0.9591965819151699

PSO with Drop wave function:

Number of runs:50	Number of runs:80
Number of particles:50	Number of particles:30
Number of Dimensions:2	Number of Dimensions:2
Number of iterations:150	Number of iterations:100
Average best score: -0.9961747196684766	Average best score: -0.980546172191756

Summary : by using PSO and GWO for :

Drop-Wave	Cross-In-Tray	Rastrigin
Original G.min:-1	original G.min : -2.06261	original G.min:0
PSO:-0.98	PSO: -2.062611	PSO: 0.0663
GWO:-0.94	GWO:-2.062612	GWO:0.278