REPORT ON PARTICLE SWARM OPTIMIZATION AND ITS VARIANTS

INTRODUCTION:

The particle swarm optimization (PSO) algorithm is a stochastic optimization technique based on the swarm, which was proposed by Eberhart and Kennedy (1995). PSO algorithm simulates animals' social behavior, including insects, herds, birds, and fishes. These swarms conform a cooperative way to find food, and each member in the swarms keeps changing the search pattern according to the learning experiences of its own and other members.

The main design idea of the PSO algorithm is closely related to two research: One is an evolutionary algorithm, just like an evolutionary algorithm; PSO also uses a swarm mode which makes it to simultaneously search large regions in the solution space of the optimized objective function. The other is artificial life; namely, it studies artificial systems with life characteristics. In studying the behavior of social animals with the artificial life theory, for how to construct the swarm artificial life systems with cooperative behavior by computer,

Millonas proposed five basic principles (van den Bergh 2001):

(1) Proximity: the swarm should be able to carry out simple space and time computations.

(2) Quality: the swarm should be able to sense the environmental quality change and respond to it.

(3) Diverse response: the swarm should not limit its way to get the resources in a narrow scope.

(4) Stability: the swarm should not change its behavior mode with every environmental change.

(5) Adaptability: the swarm should change its behavior mode when this change is worthy

BACKGROUND:

In order to illustrate production background and development of the PSO algorithm, here we first introduce the early simple model, namely Boid (Bird-oid) model (Reynolds 1987). This model is designed to simulate the behavior of birds, and it is also a direct source of the PSO algorithm.

The simplest model can be depicted as follows. Each individual of the birds is represented by a point in the Cartesian coordinate system, randomly assigned with initial velocity and position. Then run the program in accordance with "the nearest proximity velocity match rule" so that one individual has the same speed as its nearest neighbor. With the iteration going on in the same way, all the points will have the same velocity quickly. As this model is too simple and far away from the real cases, a random variable is added to the speed item. That is to say, at each iteration, aside from meeting "the nearest proximity velocity match," each speed will be added with a random variable, which makes the total simulation approach the real case.

Heppner designed a "cornfield model" to simulate the foraging behavior of a flock of birds (Clerc and Kennedy 2002). Assume that there was a "cornfield model" on the plane, i.e., food's location and birds randomly dispersed on the plane at the beginning.

MOTIVATION:

• Originally a model of social information sharing

• Abstract vs. concrete spaces – cannot occupy same locations in concrete space – can in abstract space (two individuals can have the same idea)

• Global optimum (& perhaps many suboptima)

• Combines:

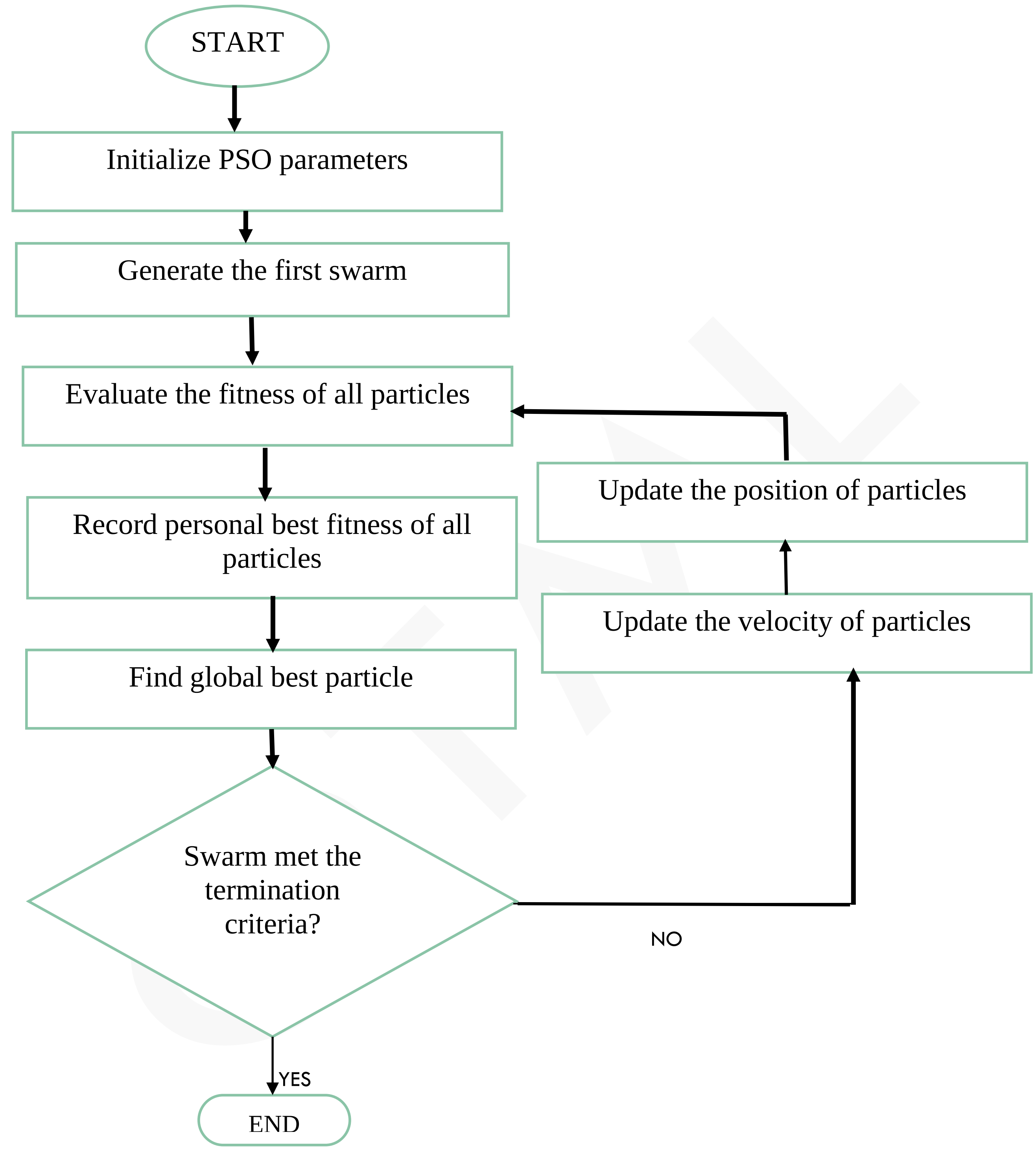
– private knowledge (best solution each has found)

– public knowledge (best solution the entire group has found)

IDEA:

Moving points in the search space, which refines their knowledge by interaction.

FLOWCHART OF ALGORITHM:



Algorithm – Parameters

f : Objective function

Xi: Position of the particle or agent.

Vi: Velocity of the particle or agent.

A: Population of agents.

W: Inertia weight.

C1: cognitive constant.

R1, R2: random numbers.

C2: social constant.

ALGORITHM STEPS:

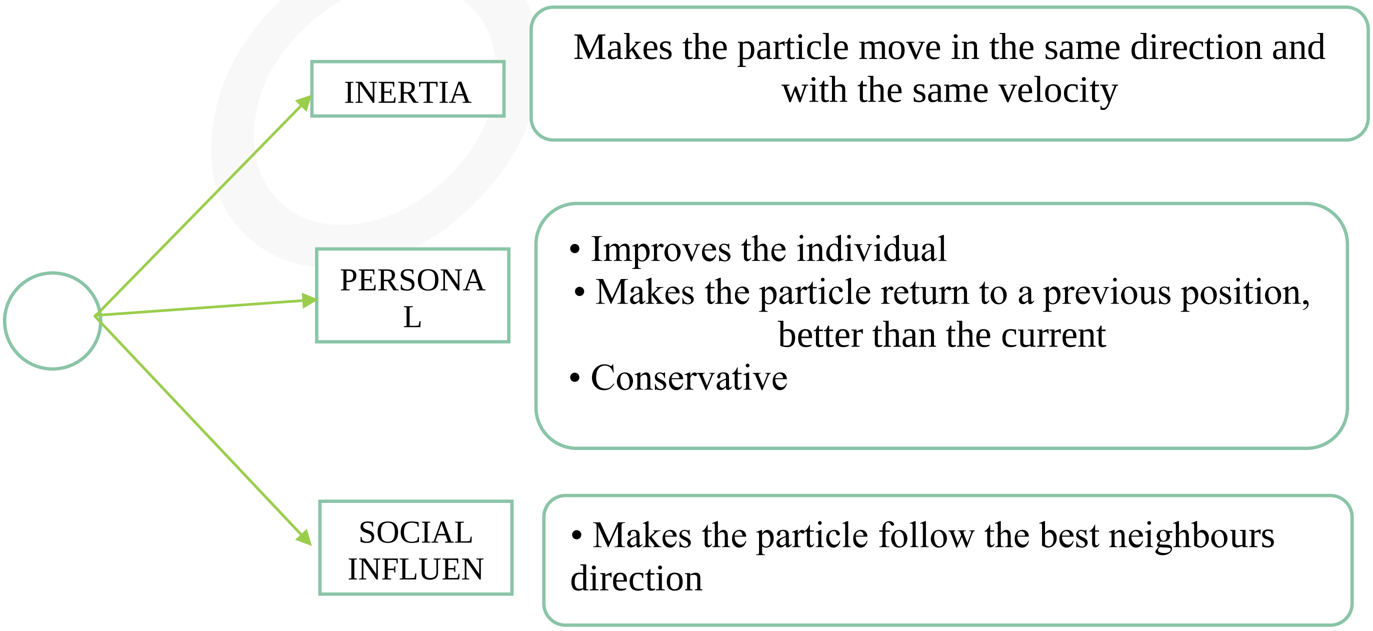
1. Create a ‘population’ of agents (particles) uniformly distributed over X
2. Evaluate each particle’s position according to the objective function
3. If a particle’s current position is better than its previous best position, update it.
4. Determine the best particle (according to the particle’s previous best positions).
5. **Update particles’ velocities**:

$$V_i^{t+1} = \underbrace{V_i^t}_{\text{Inertia}} + \underbrace{C_1R_1(pbest_i^t - X_i)}_{\text{personal influence}} + \underbrace{C_2R_2(gbest_i^t - X_i)}_{\text{social influence}}$$

6. **Move particles to their new positions**:

$$P_i^{t+1} = P_i^t + V_i^{t+1}$$

7. **Go to step 2 until the stopping criteria are satisfied.**



Acceleration coefficients

- When c1=c2=0, then all particles continue flying at their current speed until they hit the search space's boundary. Therefore, the velocity update equation is calculated as:

$$V_i^{t+1} = V_i^t$$

- When c1>0 and c2=0 , all particles are independent. The velocity update equation will be:

$$V_i^{t+1} = V_i^t + C_1R_1(pbest_i^t - X_i)$$

- When c1>0 and c2=0, all particles are attracted to a single point in the entire swarm, and the update velocity will become

$$V_i^{t+1} = V_i^t + C_2R_2(gbest_i^t - X_i)$$

- When c1=c2, all particles are attracted towards the average of pbest and gbest.

Intensification: explores the previous solutions and finds the best solution for a given region

Diversification: searches new solutions and finds the regions with potentially the best solutions

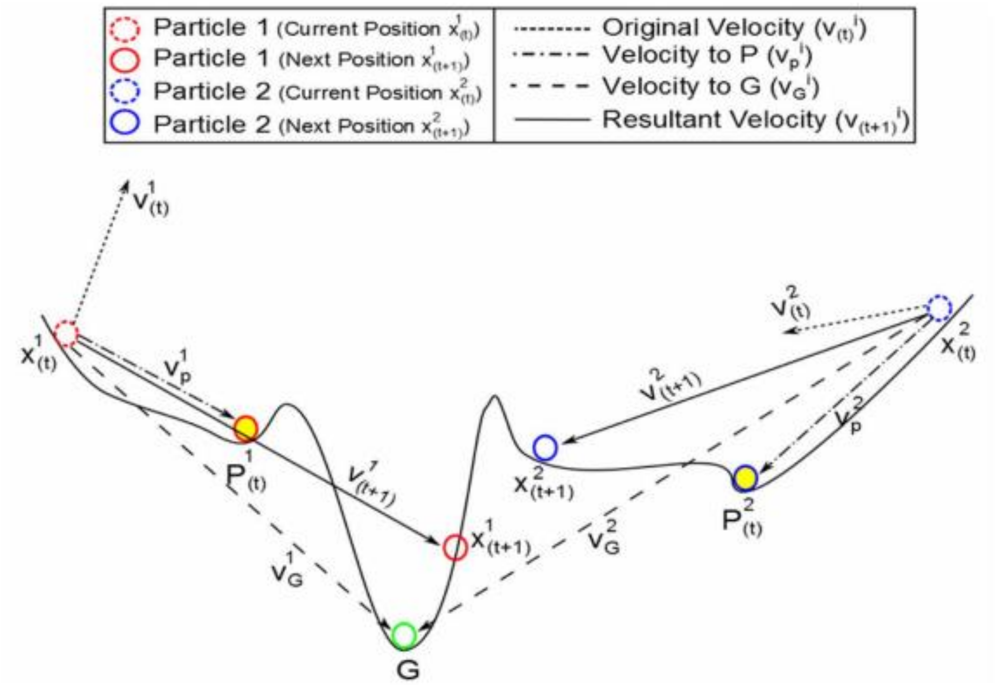
IN PSO:

$$V_i^{t+1} = \underbrace{V_i^t}_{\textbf{Diversification}} + \underbrace{C_1R_1(pbest_i^t - X_i) + C_2R_2(gbest_i^t - X_i)}_{\textbf{Intensification}}$$

Figure 1:

shows an example of the movement of two particles in the search space. As shown, the search space is one-dimensional, and the first, x_t^1, and second, x_t^2 particles are represented by the dotted red and blue circles, respectively. Moreover, the two particles have two different previous best positions, p_t^1 and p_t^2, and one global solution (G). As shown in the figure, there are three different directions or directed velocity, namely; (1) the original direction (v_1 andv_2), (2) the direction to the previous best positions (v_p^1 and v_p^2), and (3) the direction to the best position (v_G^1 and v_G^2). The velocity of the particles are calculated as in Equation (2), and it will be denoted by

(v_{t+1}^1 and v_{t+1}^2). As shown in the figure, the positions of the two particles in the next iteration, t +1, become closer to the global solution.

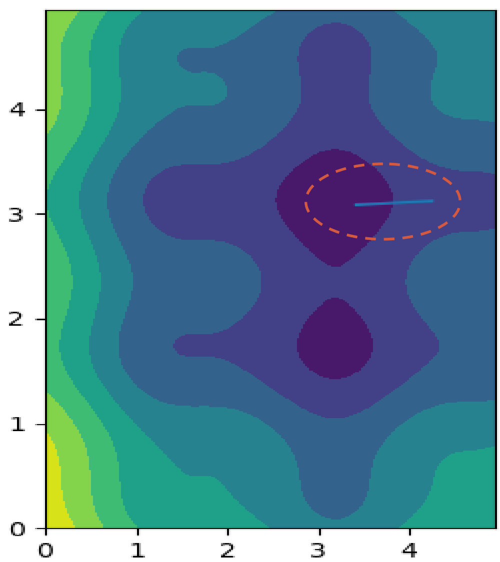


IMPLEMENTATION OF PSO:

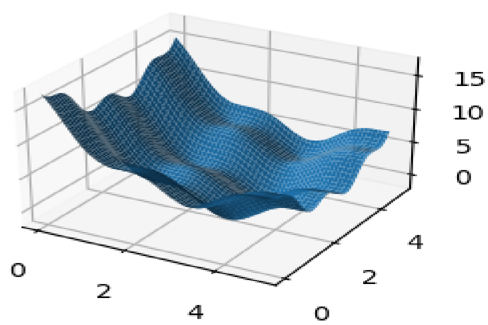
EXAMPLE 1: Minimise the following function

$$(x_1-3.14)**2 + (x_2-2.72)**2 + \sin(3*x_1+1.41) + \sin(4*x_2-1.73)$$

$$0<=x_1,\ x_2<=5$$



best positions during iterations



3D PLOT OF FUNCTION

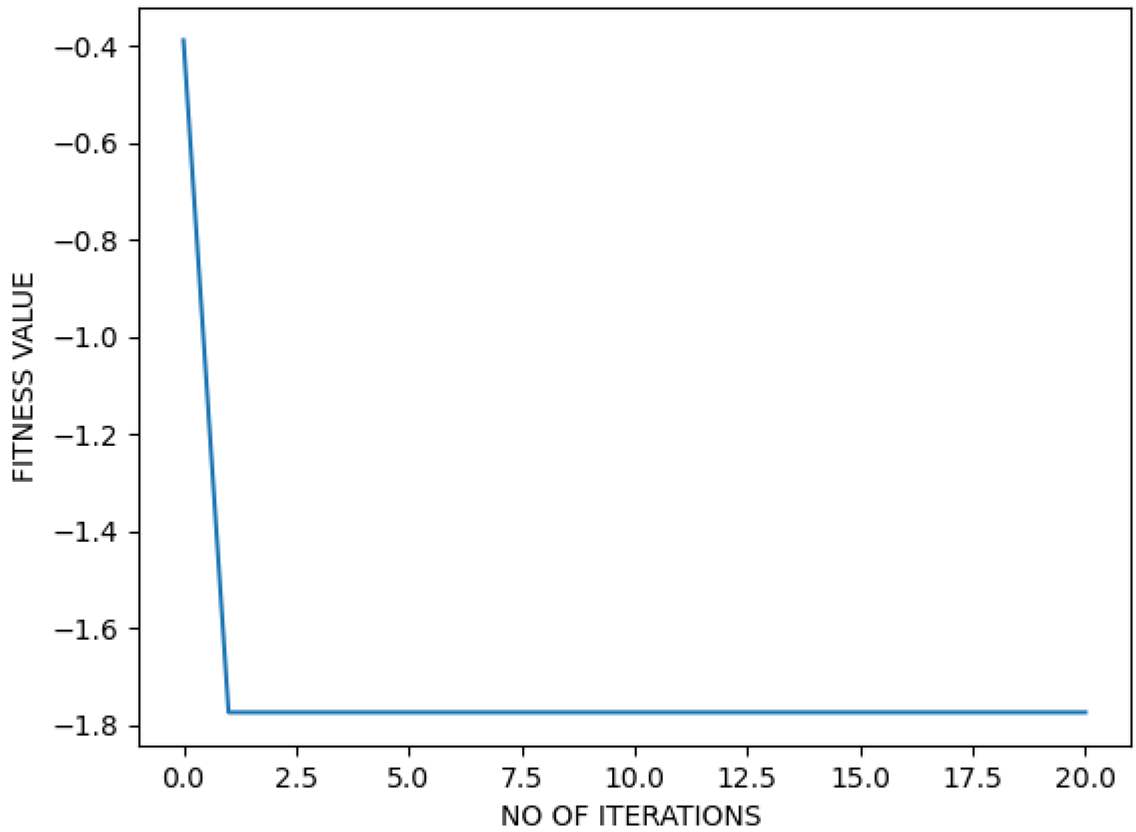


FIG: Convergence curve of the PSO particles

As we can see from the plot above, this function looks like a curved egg carton. It is not a **convex function**, and therefore it is hard to find its minimum because a **local minimum** found is not necessarily the **global minimum**.

FROM OUR ALGORITHM, WE FOUND THE BEST POSITION AND VALUE AS FOLLOWS:

Best value: -1.7730 Best position: [3.265, 3.127]

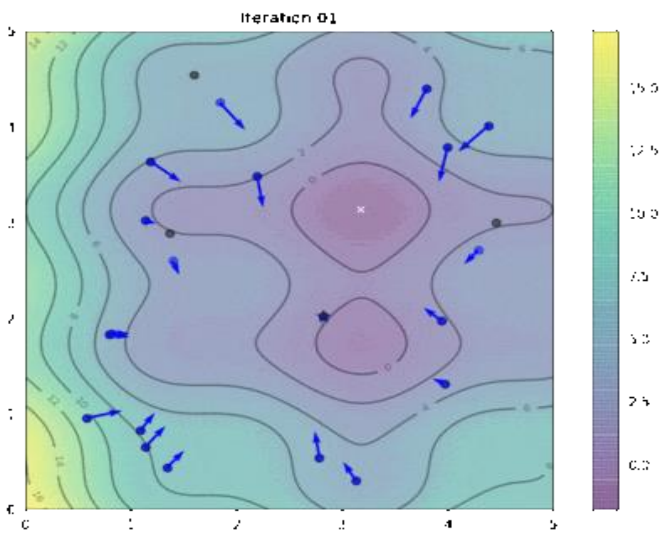


FIG: Animation of particle movements

EXAMPLE 2: MINIMISE THE FOLLOWING FUNCTION (Rastrigin Function in 2d):

$$F(x_1, x_2) = x_1^2 + x_2^2 - 10\cos(6.28x_1) - 10\cos(6.28x_2) + 20$$

$$-5 <= x_1, x_2 <= 5$$

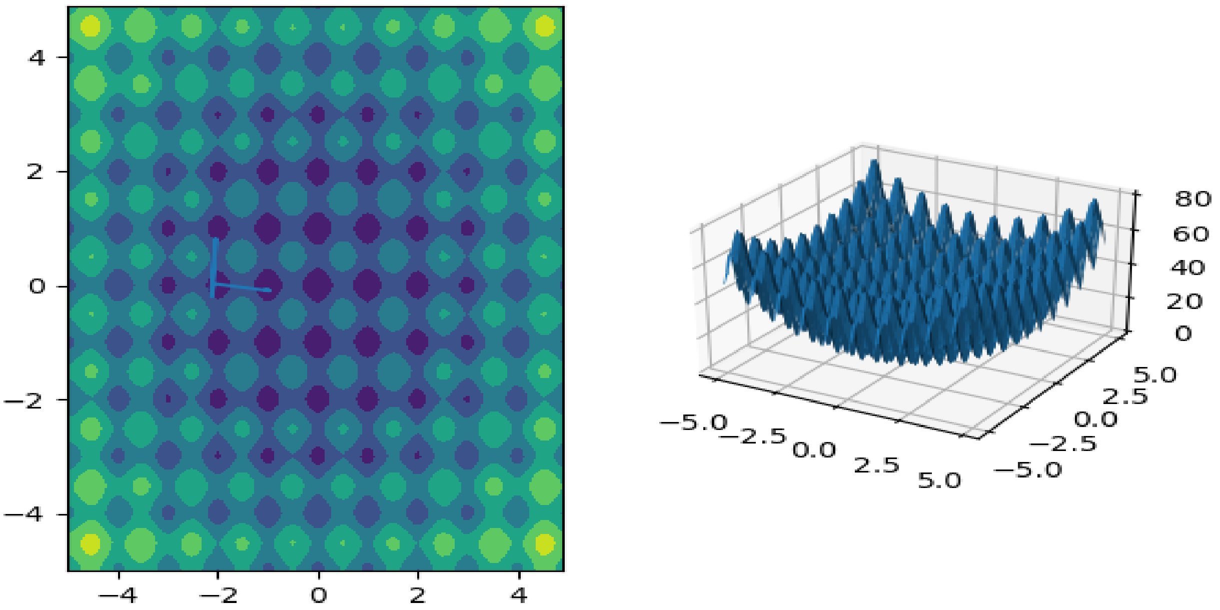


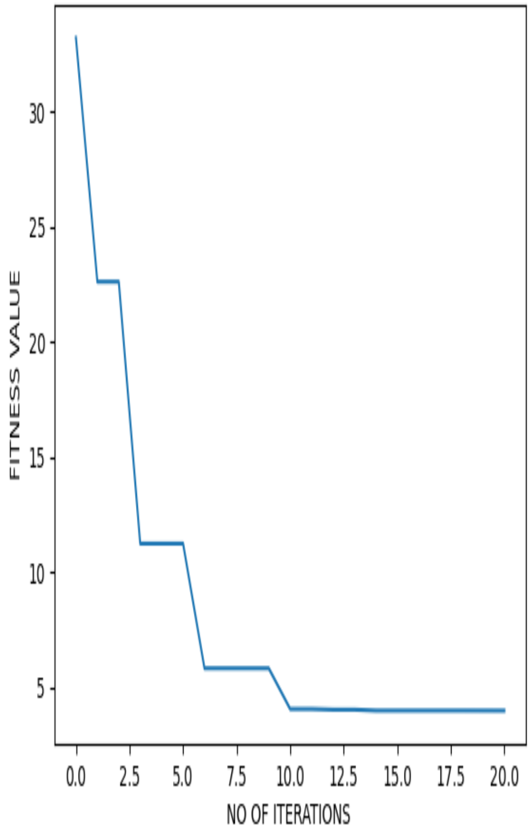
FIG: The contour and surface plot of the Rastrigin function

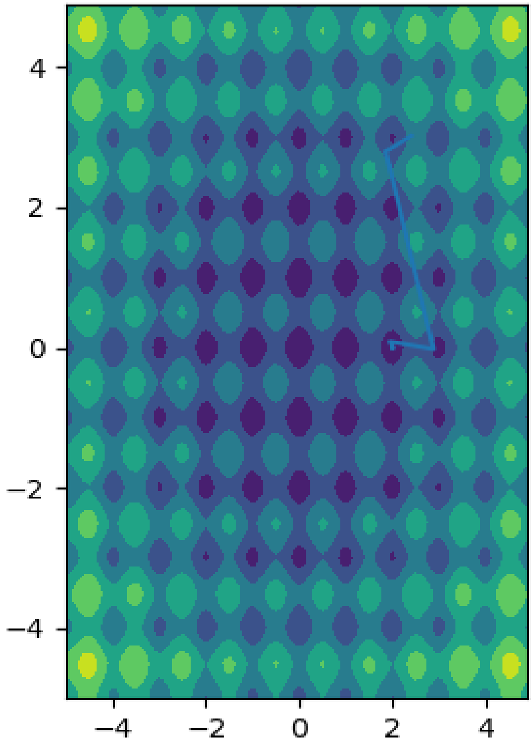
As shown, the function is not convex, and it has many **local optimal solutions**. The optimal solution is located at the **origin,** and the **optimal value is zero.**

In this example, the PSO algorithm has been run four times. In each run, the particles are randomly initialized, as shown in Table. The particles' positions are then iteratively moved using the PSO algorithm. In this example, the maximum number of iterations was 20.

Particles	First Run		Second Run		Third Run		Fourth Run	
	x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
P_1	2.600	-0.782	3.236	4.101	-4.493	3.048	2.029	-3.591
P_2	2.406	3.018	-4.897	-3.291	-4.886	-3.485	4.339	-0.273
P_3	4.500	-4.461	-3.228	-4.770	2.651	-0.169	3.772	-2.131
P_4	-4.235	2.223	-1.049	2.201	0.868	2.148	2.533	-0.269
P_5	-1.470	-3.844	-4.934	3.894	-4.057	-2.564	-0.220	-0.013
Best position	1.991	0	-1.015	2.054	-0.004	-2.026	1.01	-1.00e-03
Best solution	3.9839		5.8335		4.2340		1.0400	

FIGURE A: 1st RUN

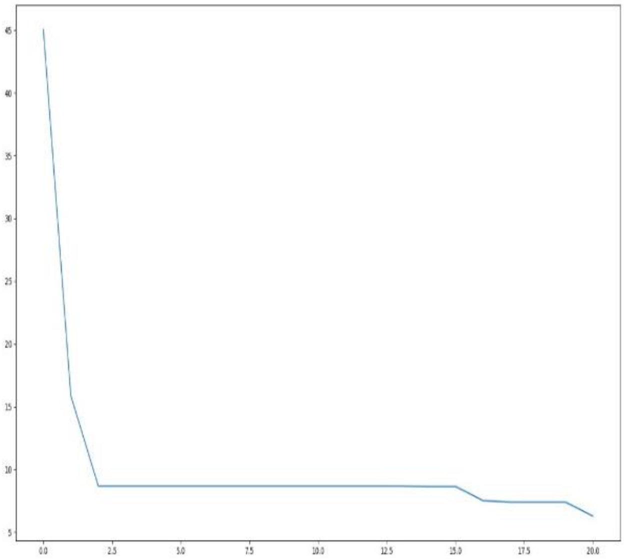


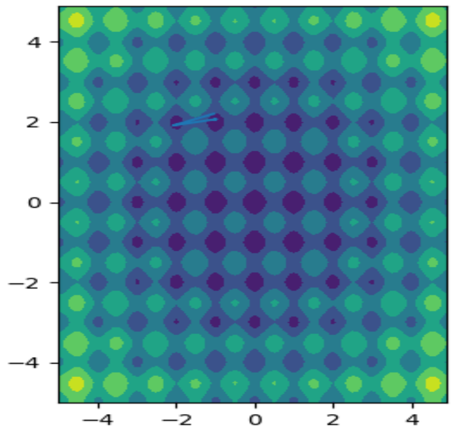


Convergence curve of the PSO particles

Visualization of the convergence of the PSO particles

FIGURE B: 2ND RUN

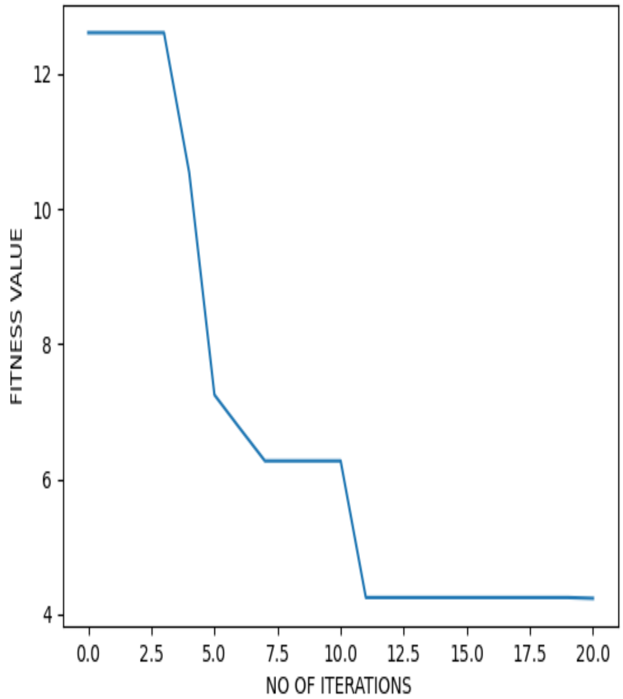




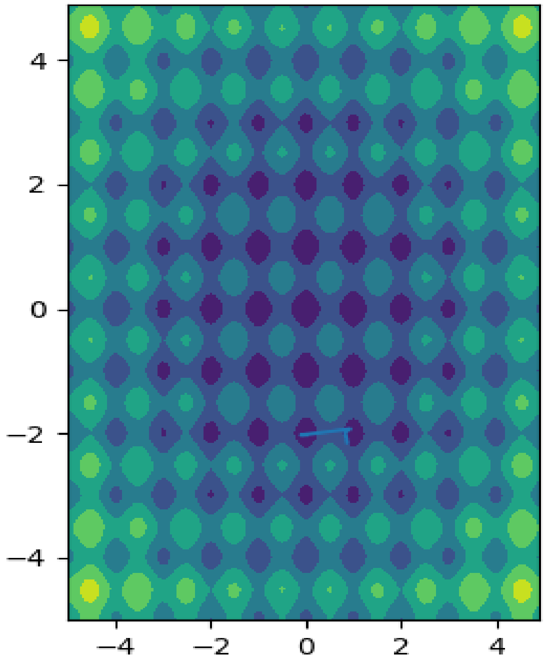
Convergence curve of the PSO particles

Visualization of the convergence of the PSO particles

FIGURE C: 3RD RUN

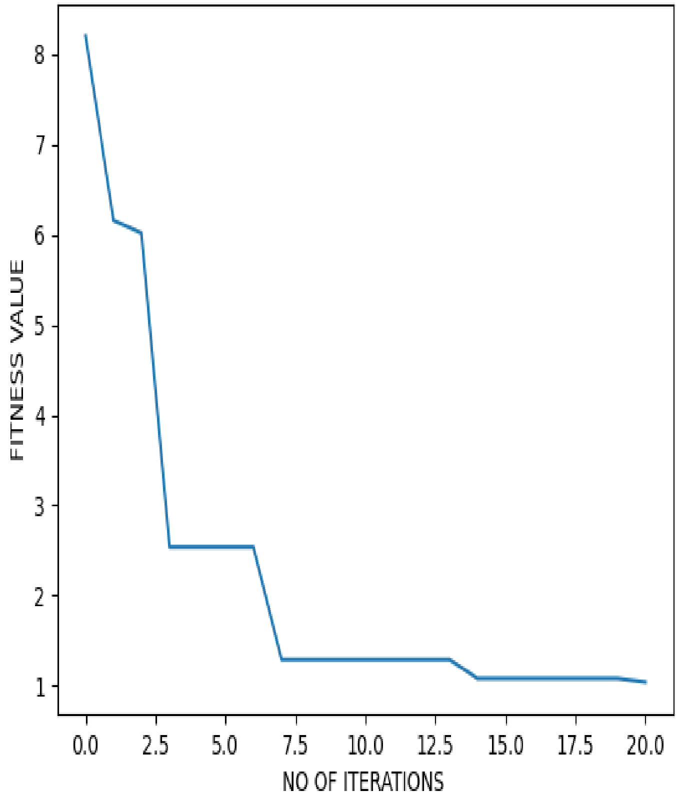


Convergence curve of the PSO particles

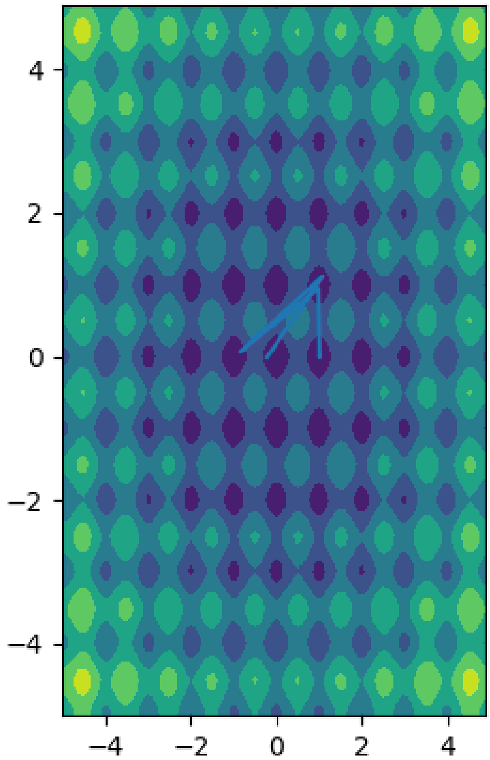


Visualization of the convergence of the PSO particles

FIGURE D: 4TH RUN



Convergence curve of the PSO particles



Visualization of the convergence of the PSO particles

CONCLUSION ON PSO:

The above convergence of the particles in each run. As shown, the four runs converged to different optimal solutions, and all of them did not reach the optimal solution. Therefore, in each run, the PSO algorithm achieved different optimal values.

To conclude, the PSO algorithm can be trapped into a local optimal solution; hence, it cannot find better solutions because its exploration capability is very limited, this problem is common in many optimization algorithms, and it is called Stagnation. However, many other solutions for this problem were used, such as combining PSO with other optimization algorithms, which make a balance between the exploration and exploitation phases. Approximately all the recent optimization algorithms solved this problem, but they do not guarantee to reach the same solution in each run due to the stochastic nature of the optimization algorithms.

ADVANTAGES AND DISADVANTAGES:

It is said that the PSO algorithm is one of the most powerful methods for solving 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:

1) PSO algorithm is a derivative-free algorithm.

2) It is easy to implement so that it can be applied both in scientific research and engineering problems.

3) It has a limited number of parameters, and the impact of parameters to the solutions is small compared to other optimization techniques.

4) The calculation in the PSO algorithm is very simple.

5) some techniques ensure convergence, and the optimum value of the problem calculates easily within a short time.

6) PSO is less dependent on a set of initial points than other optimization techniques.

Disadvantages of the PSO algorithm:

1) PSO algorithm suffers from partial optimism, which degrades the regulation of its speed and direction.

2) Problems with non-coordinate system (for instance, in the energy field) exit

VARIANTS IN PSO:

1. **Shuffled Frog Leaping Algorithm**

Introduction

The shuffled frog leaping algorithm (SFLA) was introduced by Eusuff and Lansey in 2003 for the optimization of water distribution network design. It is a hybrid of PSO and shuffled complex evolution (SCE). SCE is based on the idea of allowing sub-populations to evolve independently and periodically allowing interactions between sub-populations. SFLA is a nature-inspired population-based meta-heuristic that imitates the behavior of a frog population searching for food.

Basic Concepts

Memetic Evolution

A meme is an idea that spreads from one person to another within a culture. A field of study called memetics arose to explore the concepts and transmission of memes in terms of evolutionary model. Internet memes are an example of this memetic theory.

Genetic algorithm is a popular meta-heuristic that uses genes as unit of evolution to solve optimization problems. SFLA uses memetic evolution as its evolutionary model instead. Memetic and genetic evolution are similar in some ways, i.e., possible solutions are created, selected according to some measure of fitness, combined with other solutions. But memetic evolution differs in many ways: genes are typically transmitted between generations. Usually only fitter parents are taken to the next generation and their children repopulate the generation. But information from one meme can be incorporated in other memes immediately rather than waitng for a full generation. Thus the potential advantage of memetic evolution is information is passed between all individuals in the population rather only parent-children in genetic evolution.

Leaping Frogs

For this algorithm, individual frogs are seen as host for memes. Each meme consists of memetypes. These memetypes represent an idea similar to gene representing a trait in genetic algorithm. Each frog has its location and its fitness value as its memetypes. A lower fitness value means higher fitness for minimization problems. The population consists of sub- populations, called memeplexes. A local search is done for each sub-population. Then the sub-populations are shuffled.

Algorithm for Minimization Problems

In minimization problems the fitness corresponds to the value of the function optimized. A low function value indicates high fitness. SFLA begins by randomly creating N frogs, (i.e., N solutions). The N frogs are divided into m sub-populations, also called memeplexes. Local search is performed in each subpopulation. The local search consists of i_{max} iterations. Each iteration, only the frog's position with worst fitness (highest function value) is updated as follows:

$$\bar{x}_w \leftarrow \bar{x}_w + r(\bar{x}_b - \bar{x}_w)$$

where $\bar{x}_w$ is the position of frog with worst fitness, r ∈ [0, 1] is a uniformly distributed random number, and $\bar{x}_b$ is the frog in sub-population with best fitness (least fitness value). However, if fitness does not improve then $\bar{x}_w$ is updated as follows:

$$\bar{x}_w \leftarrow \bar{x}_w + r(\bar{x}_g - \bar{x}_w)$$

where r ∈ [0, 1] is a new random number and $\bar{x}_g$ is the position of globally best frog of all m sub-populations. If $\bar{x}_w$ is still not improved, the $\bar{x}_w$ is updated with a random position.

After local search is done once for each sub-population, the frogs are then shuffled among the sub-populations. Then local search is performed for each sub-population as defined above. This is done for k_{max} iterations.

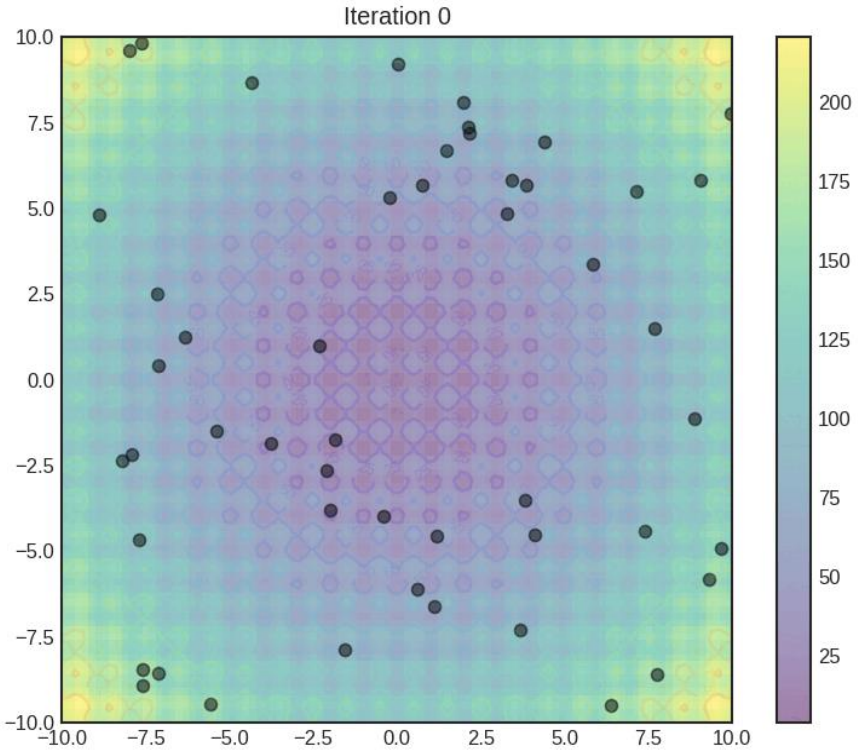


FIG: Convergence of population on rastrign function using SFLA

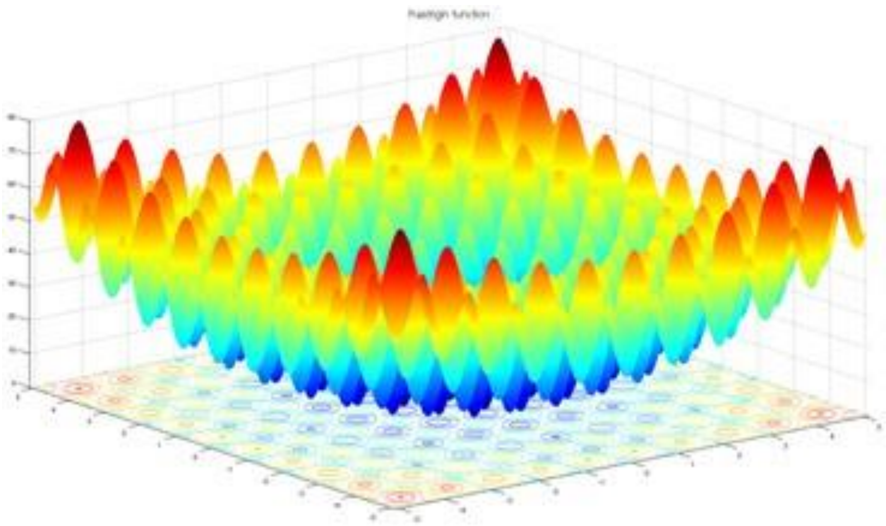
Implementation in Python

```python
from sympy import *
import numpy as np
import random

def sub(f, x):
    var = []
    for i in range(len(x)):
        var.append(('x'+str(i+1), x[i]))
    return f.subs(var).evalf()

class frog:
    def __init__(self, dim, domain, f):
        self.pos = np.zeros(dim)
        for i in range(dim):
            self.pos[i] = domain[i][0] + (domain[i][1]-domain[i][0])*random.random()
        self.fitness = sub(f, self.pos)

    def __gt__(self, other):
        return self.fitness>other.fitness

def local_search(memeplex, frog_g, dim, domain, f, i_max = 10):
    # memeplex is the sub-population, frog_g is the global best frog, i_max is number of iterations
    for i in range(i_max):
        frog_w = max(memeplex) # frog with worst fitness value in memeplex
        frog_b = min(memeplex) # frog with best fitness value in memeplex

        frog_w_new = frog_w.pos + (random.random() * (frog_b.pos - frog_w.pos))
        frog_w_new_fitness = sub(f, frog_w_new)

        if frog_w_new_fitness > frog_w.fitness:
            frog_w_new = frog_w.pos + (random.random() * (frog_g.pos - frog_w.pos))
            frog_w_new_fitness = sub(f, frog_w_new)

        if frog_w_new_fitness > frog_w.fitness:
            frog_w = frog(dim, domain, f)
        else:
            frog_w.pos = frog_w_new
            frog_w.fitness = frog_w_new_fitness

def SFLA(dim, domain, f, k_max, num_frogs=200, num_memeplexes = 20, i_max = 10):
    # num_memelexes is the number of sub_populations, k_max is the number of iterations
    frogs = np.array([frog(dim, domain, f) for i in range(num_frogs)])

    for i in range(k_max):
        np.random.shuffle(frogs)
        memeplexes = np.array_split(frogs, num_memeplexes)

        frog_g = min(frogs)

        for sub_population in memeplexes:
            func_eval += local_search(sub_population, frog_g, dim, domain, f, i_max)

    best = min(frogs)
    return best

dim = 2
var = ''
for i in range(dim):
    var += 'x'+str(i+1)+' '
x = symbols(var)
print(x)
func = '20 + (x1)**2 - 10 * cos(2 * pi * (x1)) + (x2)**2 - 10 * cos(2 * pi * (x2))'
# func = 'x1**2 + (x2 - 2)**2 + 3'
f = sympify(func)
print(f)
domain=[[-10, 10], [-10, 10]]


best=SFLA(dim, domain, f, k_max = 10, num_frogs = 50, num_memeplexes=5)
print(best)
```

Performance for Rastrigin Function

Let N be the number of frogs (individuals), m be the number of sub-population, i be the number of iterations in local search,
k be the number of iterations in SFLA algorithm.
The Rastrigin function is defined on 2 - Dimensions as follows:

$$f(x_1, x_2) = 20 + (x_1^2 - 10cos(2\pi x_1)) + (x_2^2 - 10cos(2\pi x_2))$$



The rastrigin function has multiple local minima, hence gradient-based methods cannot be used to globally optimize the function. SFLA algorithms works very well for this function. For the domain as $x_1, x_2 \in [-10, 10]$ the global minimum is $f(x) = 0$ at $x_1 = 0, x_2 = 0$.
$N = 50$, $m = 5$, $k = 20$, $i = 10$ yields the minimum as $5.0209 \cdot 10^{-10}$ at $x_1 = 1.0492 \cdot 10^{-6}$, $x_2 = 1.1957 \cdot 10^{-6}$ after 1397 function evaluations. This result is very close to actual value and is also reproducable.

Disadvantages

Metaheuristics such as SFLA do not guarantee an optimal solution is ever found.

APPLICATIONS OF PSO:

- integer programming
- minimax problems
  - in optimal control

  – engineering design

  – discrete optimization

  – Chebyshev approximation

  – game theory

• multi-objective optimization

• hydrologic problems

• musical improvisation!

References:

1. Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm by Muzaffar M. Eusuff and Kevin E. Lansey - JOURNAL OF WATER RESOURCES PLANNING AND MANAGEMENT © ASCE / MAY/JUNE 2003
2. Evolutionary Optimization Algorithms by Dan Simon - Book published by WILEY
3. Particle Swarm Optimization: a tutorial - University of Salford Institutional Repository
4. https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

REPORT BY:

CS21B2045 T. LAKSHMI SRINIVAS