Instituto Politécnico Nacional
Escuela Superior de Computo

Algoritmos Bioinspirados
Varga Soriano Flor Arlette
5BM1

Practica 4 PSO

Realizar la minimización de la función x^2+y^2 en el intervalo de valores (-5,5) para x y y.

Versión de PSO: Global
Número de partículas: 20
Iteraciones: 50
w= 0.8
c1= 0.7
c2 = 1

```
import numpy as np
def f(x,y):
    "Objective function"
    return (x)**2 + (y)**2
```

Definimos la función objetivo la cual será x^2+y^2

```
W= 0.8
c1= 0.7
c2 = 1
```

Establecemos los valores de c1, c2 y w

```
n_particles = 20
n_intera = 50
np.random.seed(5)
X = np.random.rand(2, n_particles) * 5
V = np.random.randn(2, n_particles) * 0.1
```

Aquí trazamos la función de f(x,y) en la región de -5,5 . Creamos 20 partículas en ubicaciones aleatorias en la región, junto con velocidades aleatorias

Instituto Politécnico Nacional
Escuela Superior de Computo

Algoritmos Bioinspirados
Varga Soriano Flor Arlette
5BM1

Practica 4 PSO

```
pbest = X
pbest_obj = f(X[0], X[1])
gbest = pbest[:, pbest_obj.argmin()]
gbest_obj = pbest_obj.min()
```

El vector pbest_obj es el mejor valor de la función objetivo encontrado por cada partícula. De manera similar, gbest_obj es el mejor valor escalar de la función objetivo jamás encontrado por el enjambre. Estamos usando las funciones min() y argmin() aquí porque lo configuramos como un problema de minimización

```
def update():
    global V, X, pbest, pbest_obj, gbest, gbest_obj
    # Update params
    r1, r2 = np.random.rand(2)
    V = w * V + c1*r1*(pbest - X) + c2*r2*(gbest.reshape(-1,1)-X)
    X = X + V
    obj = f(X[0], X[1])
    pbest[:, (pbest_obj >= obj)] = X[:, (pbest_obj >= obj)]
    pbest_obj = np.array([pbest_obj, obj]).min(axis=0)
    gbest = pbest[:, pbest_obj.argmin()]
    gbest_obj = pbest_obj.min()
```

Aplicamos las formulas para encontrar la velocidad de cada partícula y así encontrar el pbest y gbest