```python
import numpy as np
import matplotlib.pyplot as plt

class PSO(object):
    def __init__(self, population_size, max_steps):
        self.w = 0.6
        self.c1 = self.c2 = 2
        self.population_size = population_size
        self.dim = 2
        self.max_steps = max_steps
        self.x_bound = [-10, 10]
        self.x1 = np.random.uniform(self.x_bound[0], self.x_bound[1],
                                   (self.population_size, self.dim))
        self.x2 = np.random.uniform(self.x_bound[0], self.x_bound[1],
                                   (self.population_size, self.dim))
        self.v1 = np.random.rand(self.population_size, self.dim)
        self.v2 = np.random.rand(self.population_size, self.dim)

        fitness = self.calculate_fitness(self.x1, self.x2)
        self.p1 = self.x1
        self.pg1 = self.x1[np.argmin(fitness)]

        self.p2 = self.x2
        self.pg2 = self.x2[np.argmin(fitness)]

        self.individual_best_fitness = fitness
        self.global_best_fitness = np.min(fitness)

    def calculate_fitness(self, x1, x2):
        return np.sum(np.square(x1)+np.square(x2), axis=1)

    def evolve(self):
        fig = plt.figure()
        for step in range(self.max_steps):
            r1 = np.random.rand(self.population_size, self.dim)
            r2 = np.random.rand(self.population_size, self.dim)

            self.v1 = self.w * self.v1 + self.c1 * r1 * (self.p1 - self.x1) +
            self.c2 * r2 * (self.pg1 - self.x1)
            self.v2 = self.w * self.v2 + self.c1 * r1 * (self.p1 - self.x2) +
            self.c2 * r2 * (self.pg1 - self.x2)

            self.x1 = self.v1 + self.x1
            self.x2 = self.v2 + self.x2

            plt.clf()
            plt.scatter(self.x1[:, 0], self.x1[:, 1], s=30, color='k')
            plt.scatter(self.x2[:, 0], self.x2[:, 1], s=30, color='r')
            plt.xlim(self.x_bound[0], self.x_bound[1])
            plt.ylim(self.x_bound[0], self.x_bound[1])
            plt.pause(0.01)
            fitness = self.calculate_fitness(self.x1, self.x2)

            update_id = np.greater(self.individual_best_fitness, fitness)
            self.p1[update_id] = self.x1[update_id]
            self.p2[update_id] = self.x2[update_id]
            self.individual_best_fitness[update_id] = fitness[update_id]

            if np.min(fitness) < self.global_best_fitness:
                self.pg1 = self.x1[np.argmin(fitness)]
                self.pg2 = self.x2[np.argmin(fitness)]
                self.global_best_fitness = np.min(fitness)
            print('best fitness: %.5f, mean fitness: %.5f' %
            (self.global_best_fitness, np.mean(fitness)))


pso = PSO(100, 100)
pso.evolve()
plt.show()
```