```
import numpy as np

# PSO Parameters
NUM_POINTS = 50  # Number of data points
NUM_CLUSTERS = 3  # Number of clusters
NUM_PARTICLES = 30
NUM_ITERATIONS = 100
INERTIA_WEIGHT = 0.5
COGNITIVE_CONSTANT = 1.5
SOCIAL_CONSTANT = 1.5

# Randomly generate data points
data_points = np.random.rand(NUM_POINTS, 2)  # 2D data points

# Objective function to minimize the sum of squared distances
def objective_function(centroids):
    total_distance = 0
    for point in data_points:
        distances = [np.linalg.norm(point - centroid) for centroid in centroids]
        total_distance += min(distances)
    return total_distance

# Initialize particles (each particle represents a set of cluster centroids)
particles = [np.random.rand(NUM_CLUSTERS, 2) for _ in range(NUM_PARTICLES)]
velocities = [np.random.uniform(-1, 1, (NUM_CLUSTERS, 2)) for _ in range(NUM_PARTICLES)]

# Personal and global best positions and scores
personal_best_positions = np.copy(particles)
personal_best_scores = np.array([objective_function(p) for p in particles])

global_best_position = personal_best_positions[np.argmin(personal_best_scores)]
global_best_score = np.min(personal_best_scores)

def particle_swarm_clustering():
    global global_best_position, global_best_score

    for iteration in range(NUM_ITERATIONS):
        for i in range(NUM_PARTICLES):
            velocities[i] = (INERTIA_WEIGHT * velocities[i] +
                             COGNITIVE_CONSTANT * np.random.random() * (personal_best_positions[i] - particles[i]) +
                             SOCIAL_CONSTANT * np.random.random() * (global_best_position - particles[i]))

            particles[i] += velocities[i]

            current_score = objective_function(particles[i])
            if current_score < personal_best_scores[i]:
                personal_best_positions[i] = particles[i]
                personal_best_scores[i] = current_score

                if current_score < global_best_score:
                    global_best_position = particles[i]
                    global_best_score = current_score

    return global_best_position, global_best_score

# Run the PSO clustering
best_centroids, best_score = particle_swarm_clustering()
print(f"Best Cluster Centroids: {best_centroids}, Best Score: {best_score}")
```

```
Best Cluster Centroids: [[0.59754598 0.17170333]
 [0.1825933  0.71918648]
 [0.68880233 0.66659985]], Best Score: 10.678539870863613
```