In [1]:

```
pip install pygad
```

```
Collecting pygad
  Downloading pygad-3.0.1-py3-none-any.whl (67 kB)
                                            0.0/68.0 kB ? eta -:--:--
     -----------------                      30.7/68.0 kB 660.6 kB/s eta
0:00:01
     -----------------                      30.7/68.0 kB 660.6 kB/s eta
0:00:01
     ------------------------------------ 68.0/68.0 kB 409.3 kB/s eta
0:00:00
Collecting cloudpickle (from pygad)
  Downloading cloudpickle-2.2.1-py3-none-any.whl (25 kB)
Requirement already satisfied: matplotlib in c:\users\91955\appdata\loc
al\programs\python\python310\lib\site-packages (from pygad) (3.7.1)
Requirement already satisfied: numpy in c:\users\91955\appdata\local\pr
ograms\python\python310\lib\site-packages (from pygad) (1.24.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\91955\appda
ta\local\programs\python\python310\lib\site-packages (from matplotlib->
pygad) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\91955\appdata\l
```

In [1]:

```
import numpy
import matplotlib.pyplot
import pygad
```

In [2]:

```
cluster1_num_samples = 10
cluster1_x1_start = 0
cluster1_x1_end = 5
cluster1_x2_start = 2
cluster1_x2_end = 6
cluster1_x1 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x1 = cluster1_x1 * (cluster1_x1_end - cluster1_x1_start) + cluster1_x1_star
cluster1_x2 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x2 = cluster1_x2 * (cluster1_x2_end - cluster1_x2_start) + cluster1_x2_star
cluster2_num_samples = 10
cluster2_x1_start = 10
cluster2_x1_end = 15
cluster2_x2_start = 8
cluster2_x2_end = 12
cluster2_x1 = numpy.random.random(size=(cluster2_num_samples))
cluster2_x1 = cluster2_x1 * (cluster2_x1_end - cluster2_x1_start) + cluster2_x1_star
cluster2_x2 = numpy.random.random(size=(cluster2_num_samples))
cluster2_x2 = cluster2_x2 * (cluster2_x2_end - cluster2_x2_start) + cluster2_x2_star
```

In [3]:

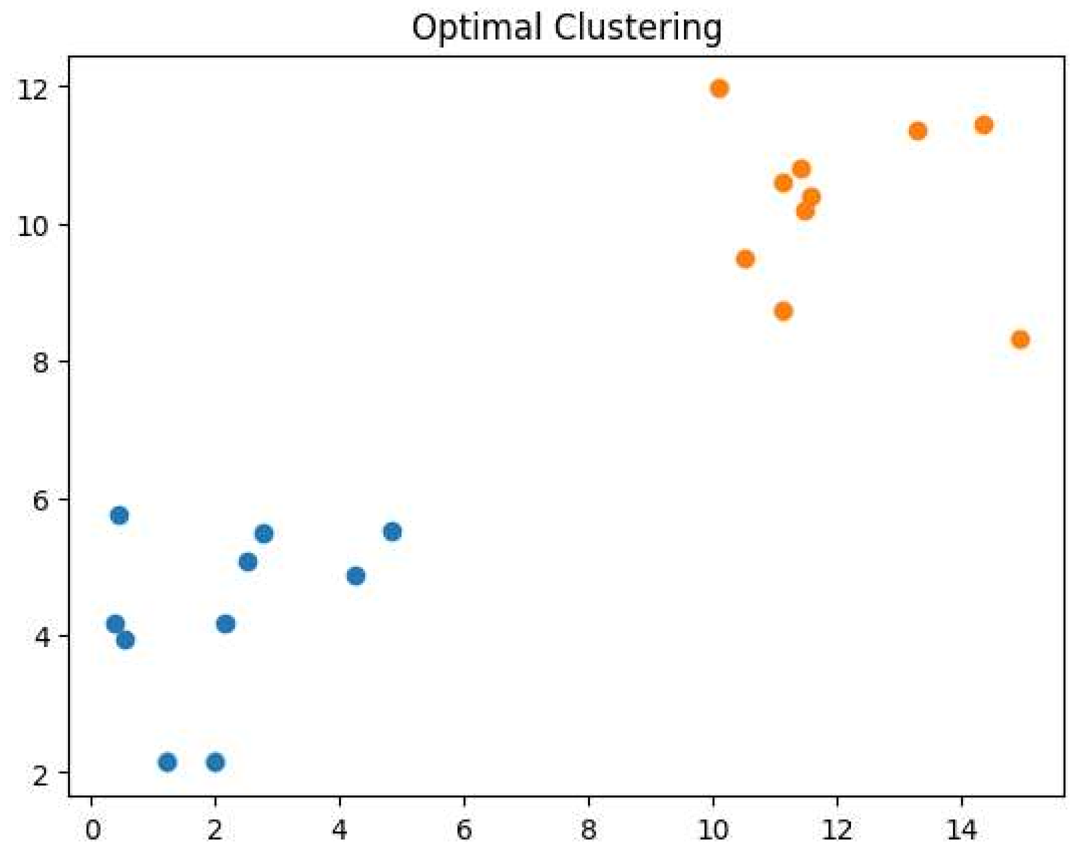
```
 c1 = numpy.array([cluster1_x1, cluster1_x2]).T
c2 = numpy.array([cluster2_x1, cluster2_x2]).T
data = numpy.concatenate((c1, c2), axis=0)
data
```

Out[3]:

```
array([[ 0.38178095,  4.16771023],
       [ 2.16277445,  4.18621198],
       [ 2.50580829,  5.08285849],
       [ 4.26033499,  4.87641047],
       [ 2.00323431,  2.14629212],
       [ 0.44179313,  5.7410927 ],
       [ 1.23006608,  2.14463077],
       [ 2.78039314,  5.49504004],
       [ 4.84866989,  5.52511147],
       [ 0.54428146,  3.94304465],
       [13.30733672, 11.3641438 ],
       [11.49230852, 10.20676623],
       [11.59632511, 10.41402936],
       [11.42145453, 10.8173091 ],
       [11.14342512, 10.62218626],
       [10.5095396 ,  9.50889834],
       [10.10666093, 11.97013301],
       [14.35622771, 11.47121644],
       [14.93217028,  8.33188517],
       [11.12796138,  8.74972797]])
```

In [4]:

```
matplotlib.pyplot.scatter(cluster1_x1, cluster1_x2)
matplotlib.pyplot.scatter(cluster2_x1, cluster2_x2)
matplotlib.pyplot.title("Optimal Clustering")
matplotlib.pyplot.show()
```



In [5]:

```
def euclidean_distance(X, Y):
 return numpy.sqrt(numpy.sum(numpy.power(X - Y, 2), axis=1))
```

In [6]:

```
def cluster_data(solution, solution_idx):
 global num_cluster, data
 feature_vector_length = data.shape[1]
 cluster_centers = []
 all_clusters_dists = []
 clusters = []
 clusters_sum_dist = []
 for clust_idx in range(num_clusters):
    cluster_centers.append(solution[feature_vector_length*clust_idx:feature_vector_l
    cluster_center_dists = euclidean_distance(data, cluster_centers[clust_idx])
    all_clusters_dists.append(numpy.array(cluster_center_dists))
 cluster_centers = numpy.array(cluster_centers)
 all_clusters_dists = numpy.array(all_clusters_dists)
 cluster_indices = numpy.argmin(all_clusters_dists, axis=0)
 for clust_idx in range(num_clusters):
        clusters.append(numpy.where(cluster_indices == clust_idx)[0])
        if len(clusters[clust_idx]) == 0:
            clusters_sum_dist.append(0)
        else:
            clusters_sum_dist.append(numpy.sum(all_clusters_dists[clust_idx, cluster
 clusters_sum_dist = numpy.array(clusters_sum_dist)
 return cluster_centers, all_clusters_dists, cluster_indices, clusters, clusters_sum
```

In [7]:

```
def fitness_func(ga_instance,solution, solution_idx):
 _, _, _, _, clusters_sum_dist = cluster_data(solution, solution_idx)
 fitness = 1.0 / (numpy.sum(clusters_sum_dist) + 0.00000001)
 return fitness
```

In [8]:

```
num_clusters = 2
num_genes = num_clusters * data.shape[1]
ga_instance = pygad.GA(num_generations=100,
 sol_per_pop=10,
 num_parents_mating=5,
 init_range_low=-6,
 init_range_high=20,
 keep_parents=2,
 num_genes=num_genes,
 fitness_func=fitness_func,
 suppress_warnings=True)
ga_instance.run()
```

In [9]:

```
best_solution, best_solution_fitness, best_solution_idx = ga_instance.best_solution(
print("Best solution is {bs}".format(bs=best_solution))
print("Fitness of the best solution is {bsf}".format(bsf=best_solution_fitness))
print("Best solution found after {gen} generations".format(gen=ga_instance.best_solu
```

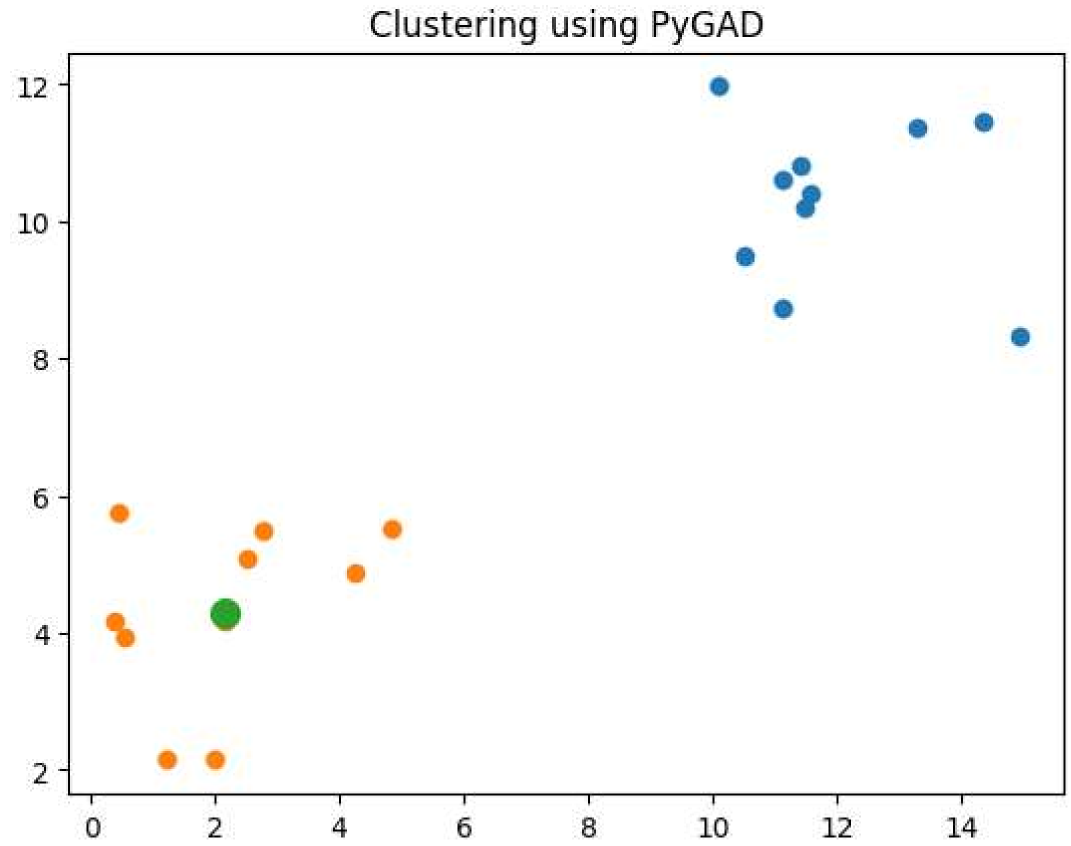
```
Best solution is [11.54917861 10.41378349  2.14742384  4.28340315]
Fitness of the best solution is 0.030340957272971966
Best solution found after 86 generations
```

In [10]:

```
cluster_centers, all_clusters_dists, cluster_indices, clusters, clusters_sum_dist= c
```

In [11]:

```
for cluster_idx in range(num_clusters):
    cluster_x = data[clusters[cluster_idx], 0]
    cluster_y = data[clusters[cluster_idx], 1]
    matplotlib.pyplot.scatter(cluster_x, cluster_y)
matplotlib.pyplot.scatter(cluster_centers[cluster_idx, 0], cluster_centers[cluster_i
matplotlib.pyplot.title("Clustering using PyGAD")
matplotlib.pyplot.show()
```


Clustering using PyGAD