In [1]:

```
pip install pygad
```

```
Collecting pygad
  Downloading pygad-3.0.1-py3-none-any.whl (67 kB)
     -------------------------------------- 68.0/68.0 kB 307.0 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:\reddy\python37\lib\site-packages (from pygad) (3.4.
2)
Requirement already satisfied: numpy in c:\reddy\python37\lib\site-packages (from pygad) (1.21.0)
Requirement already satisfied: cycler>=0.10 in c:\reddy\python37\lib\site-packages (from matplotlib
->pygad) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\reddy\python37\lib\site-packages (from matpl
otlib->pygad) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in c:\reddy\python37\lib\site-packages (from matplotli
b->pygad) (8.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\reddy\python37\lib\site-packages (from matplo
tlib->pygad) (2.4.7)
Requirement already satisfied: python-dateutil>=2.7 in c:\reddy\python37\lib\site-packages (from ma
tplotlib->pygad) (2.8.2)
Requirement already satisfied: six in c:\reddy\python37\lib\site-packages (from cycler>=0.10->matpl
otlib->pygad) (1.16.0)
Installing collected packages: cloudpickle, pygad
Successfully installed cloudpickle-2.2.1 pygad-3.0.1
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

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

In [3]:

```
 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_start
cluster1_x2 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x2 = cluster1_x2 * (cluster1_x2_end - cluster1_x2_start) + cluster1_x2_start
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_start
cluster2_x2 = numpy.random.random(size=(cluster2_num_samples))
cluster2_x2 = cluster2_x2 * (cluster2_x2_end - cluster2_x2_start) + cluster2_x2_start
```

In [4]:

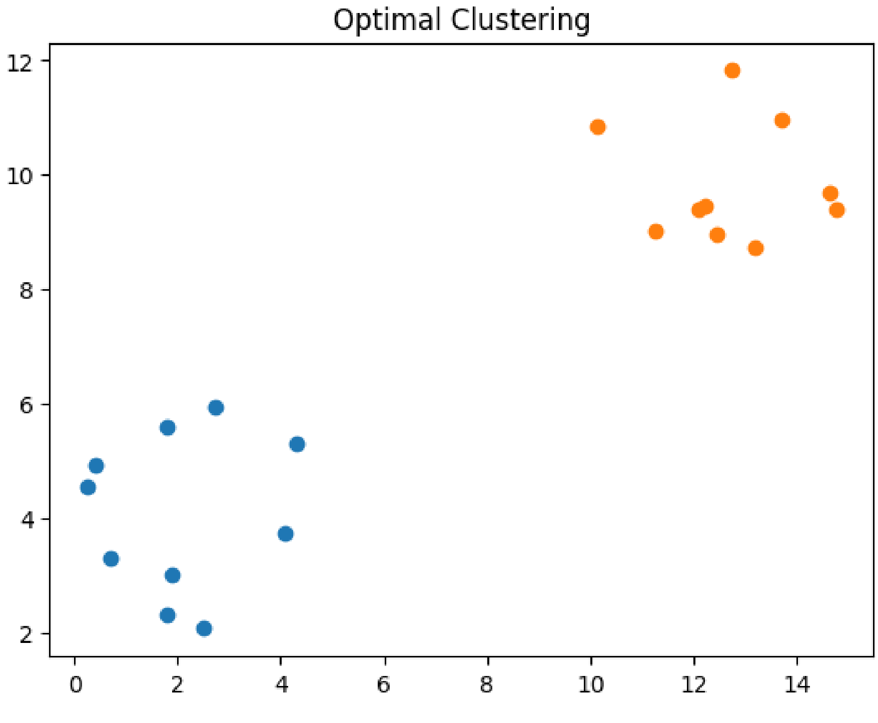
```
c1 = numpy.array([cluster1_x1, cluster1_x2]).T
c2 = numpy.array([cluster2_x1, cluster2_x2]).T
data = numpy.concatenate((c1, c2), axis=0)
data
```

Out[4]:

```
array([[ 0.24838267,  4.5637342 ],
       [ 2.49743886,  2.06865154],
       [ 2.73916016,  5.94601865],
       [ 1.80388526,  5.60154272],
       [ 0.41419872,  4.91703655],
       [ 0.69773324,  3.28809349],
       [ 4.30807113,  5.30291468],
       [ 1.80801035,  2.31913295],
       [ 1.88337443,  3.01952445],
       [ 4.09547538,  3.74723214],
       [12.0900452 ,  9.40640908],
       [12.21189822,  9.45109729],
       [12.74741462, 11.82393866],
       [14.76229139,  9.3854692 ],
       [11.27228902,  9.01521448],
       [13.20506364,  8.7192891 ],
       [10.14785192, 10.84914076],
       [14.64657805,  9.69953057],
       [12.44693217,  8.9635972 ],
       [13.6999755 , 10.97131405]])
```

In [5]:

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



In [6]:

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

In [8]:

```
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_length*(clust
        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, clusters[clust_idx]]))

    clusters_sum_dist = numpy.array(clusters_sum_dist)
    return cluster_centers, all_clusters_dists, cluster_indices, clusters, clusters_sum_dist
```

In [10]:

```
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 [11]:

```
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 [12]:

```
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_solution_generation)
```

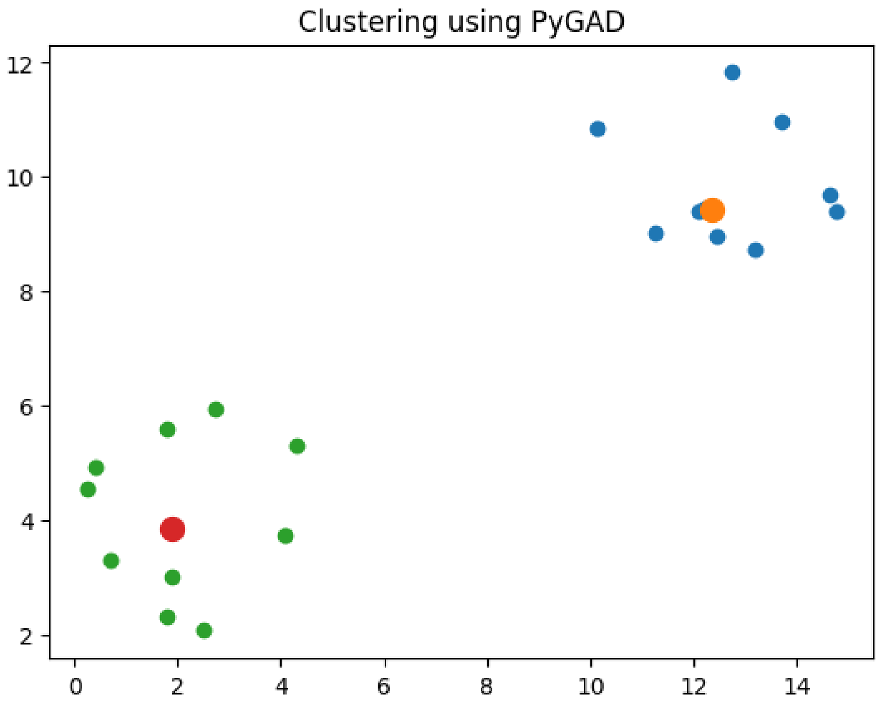
```
Best solution is [12.34028601  9.42154463  1.90483625  3.86779388]
Fitness of the best solution is 0.030128176115088987
Best solution found after 77 generations
```

In [16]:

```
cluster_centers, all_clusters_dists, cluster_indices, clusters, clusters_sum_dis= cluster_data(best_
```

In [17]:

```
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_idx, 1], linew
matplotlib.pyplot.title("Clustering using PyGAD")
matplotlib.pyplot.show()
```



In [ ]: