sum

August 30, 2021

[89]:
```python
import networkx as nx
import Complexity as cx
import matplotlib.pyplot as plt
import utilities as ut
import numpy as np
import matplotlib as mpl
from math import log
```

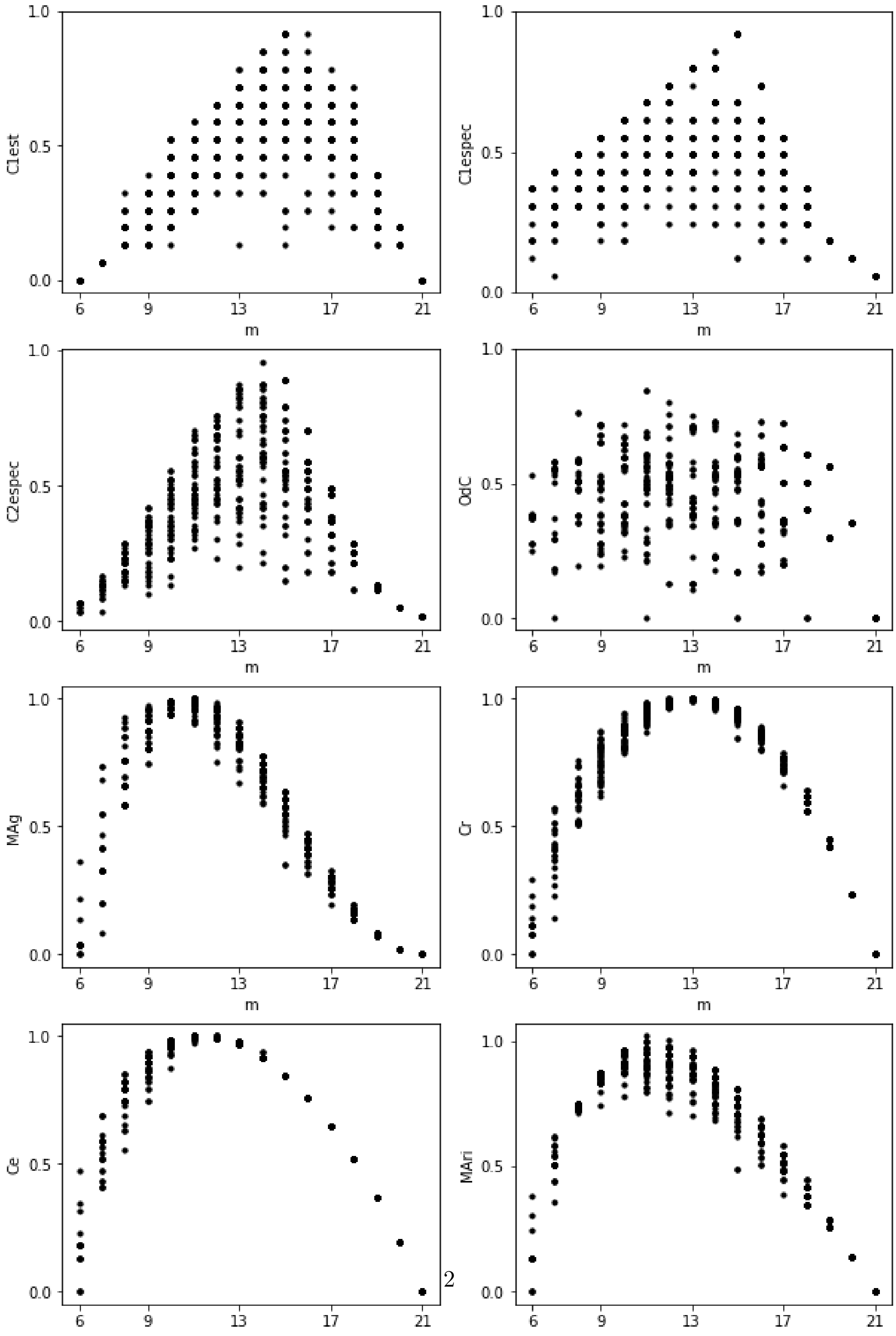
[2]:
```python
methods = ["C1est","C1espec","C2espec","OdC","MAg","Cr","Ce","MAri"]
```

[3]:
```python
#Generates random graphs and data
n=7
graphs,df = ut.random_networks(n=n,use_all_m = True,sample_number = 50)
```

[4]:
```python
#Find the complexities of the graphs
results = []
for item in methods:
    method = getattr(cx,item)
    temp_result = [method(g) for g in graphs]
    results.append(temp_result)
```

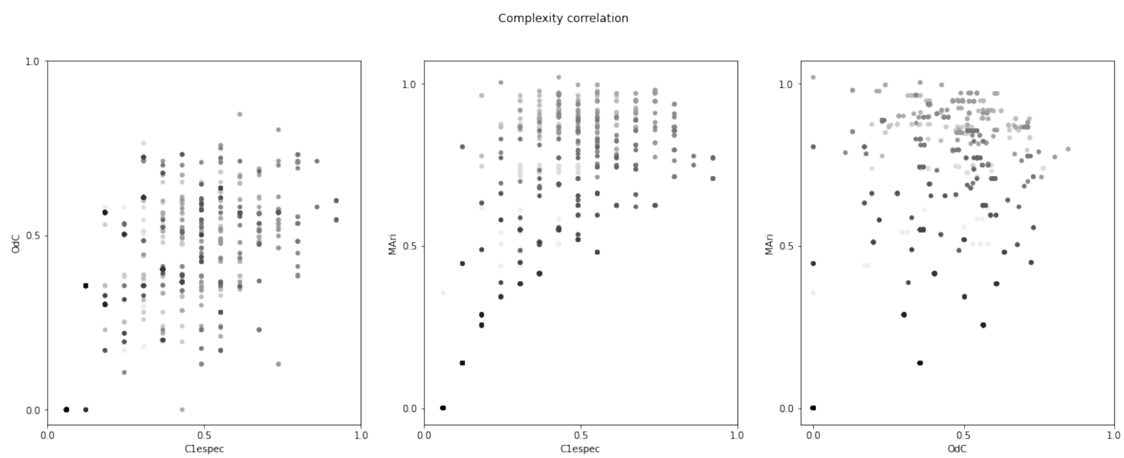
[119]:
```python
n=7
c=0
fig,axes = plt.subplots(4,2,figsize = (10,16))
xticks = np.linspace(n-1,n*(n-1)/2,5)
xticks = [int(item) for item in xticks]
for i in range(4):
    for j in range(2):
        axes[i][j].scatter(df["Number_of_edges"],results[c],s=10,color =␣
 ↪"black")
        axes[i][j].set_yticks([0,0.5,1])
        axes[i][j].set_xticks(xticks)
        axes[i][j].set_xlabel("m")
        axes[i][j].set_ylabel(methods[c])
        c+=1
plt.suptitle("Complexity of graphs with 7 nodes")
plt.show()
```

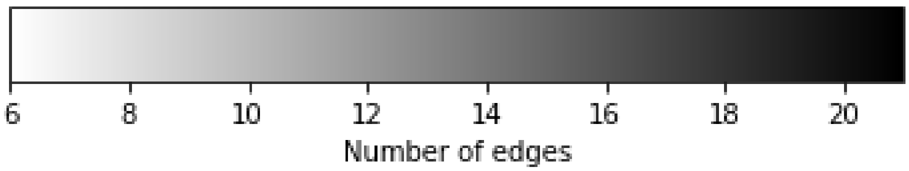
Complexity of graphs with 7 nodes



[84]:
```
fig,axes = plt.subplots(1,3,figsize =(20,7))
axes[0].scatter(results[1],results[3],s=15,c = df["Number_of_edges"],cmap =␣
↪"binary");axes[0].set_xlabel("C1espec");axes[0].set_ylabel("OdC");
axes[1].scatter(results[1],results[7],s=15,c = df["Number_of_edges"],cmap =␣
↪"binary");axes[1].set_xlabel("C1espec");axes[1].set_ylabel("MAri");
axes[2].scatter(results[3],results[7],s=15,c = df["Number_of_edges"],cmap =␣
↪"binary");axes[2].set_xlabel("OdC");axes[2].set_ylabel("MAri");
for i in range(len(axes)):
    axes[i].set_xticks([0,0.5,1])
    axes[i].set_yticks([0,0.5,1])
plt.suptitle("Complexity correlation")
fig, ax = plt.subplots(figsize=(6, 1))
fig.subplots_adjust(bottom=0.5)

cmap = mpl.cm.binary
norm = mpl.colors.Normalize(vmin=(n-1), vmax=(n-1)*n/2)
cb1 = mpl.colorbar.ColorbarBase(ax, cmap=cmap,
                                norm=norm,
                                orientation='horizontal')
cb1.set_label('Number of edges')
fig.show()
```

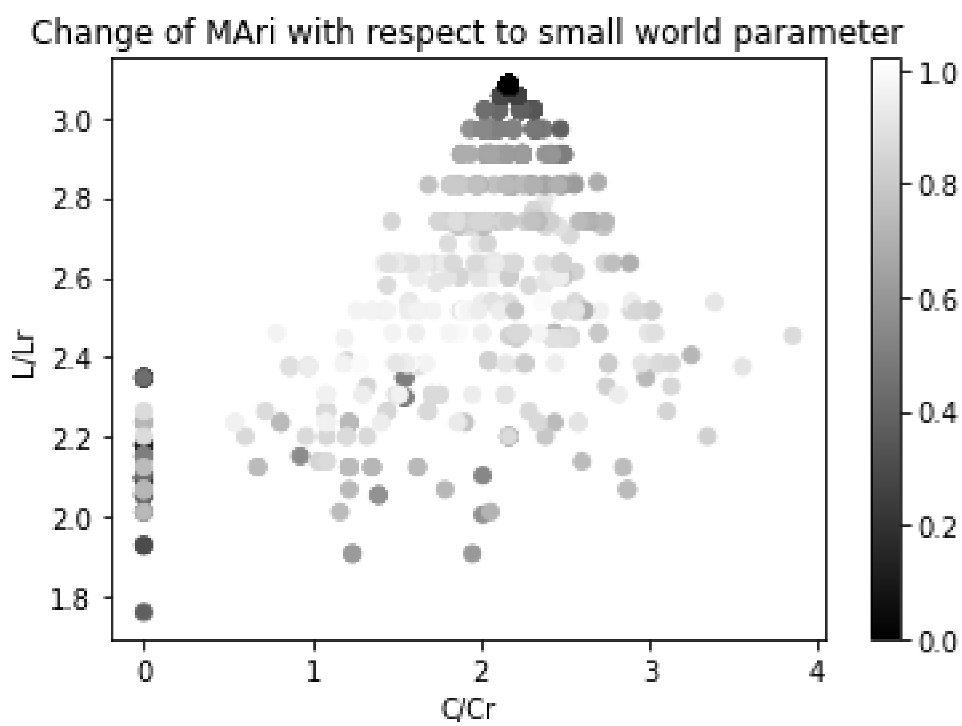
```
<ipython-input-84-11f2d7133ae1>:18: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```





[110]:
```
clusters = [nx.average_clustering(g) for g in graphs]
distances = [nx.average_shortest_path_length(g) for g in graphs]
Cr = [len(g.edges)/(n*(n-1/2)) for g in graphs]
Lr = [log(n)/(2*len(g.edges)/n) for g in graphs]
c_ratio = [item/item1 for item,item1 in zip(clusters,Cr)]
l_ratio = [item/item1 for item,item1 in zip(distances,Lr)]
plt.scatter(c_ratio,l_ratio, c = results[7],cmap = "gray")
plt.colorbar()
plt.title("Change of MAri with respect to small world parameter")
plt.xlabel("C/Cr")
plt.ylabel("L/Lr")
```

[110]: Text(0, 0.5, 'L/Lr')

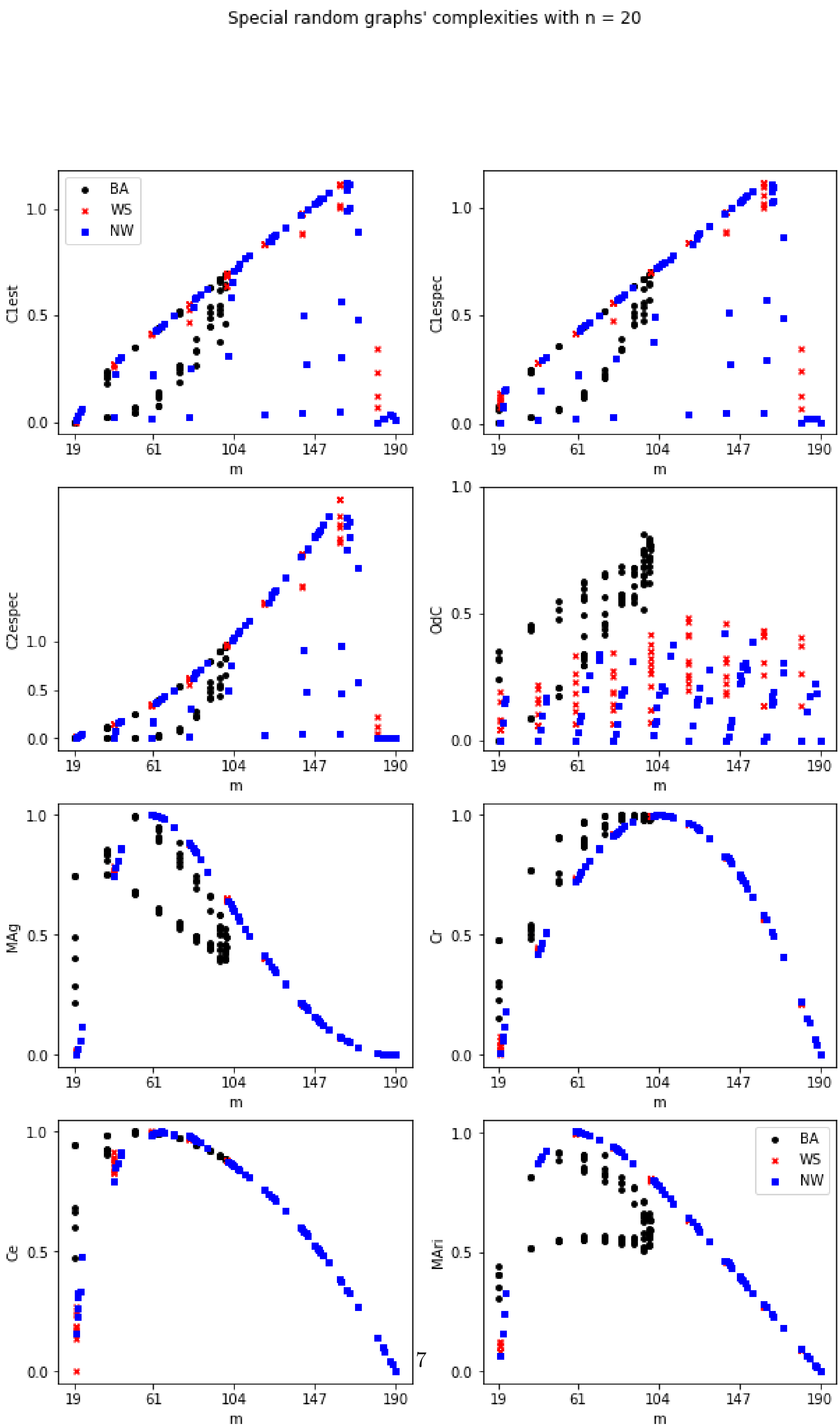


[6]:
```
#Generates special random graphs
n = 20
BA_graphs = ut.BA_random_graphs(n=n,sample_number = 100)
WS_graphs = ut.WS_random_graphs(n=n,sample_number = 100)
NW_graphs = ut.NW_random_graphs(n=n,sample_number = 100)
```

[7]:
```
#Calculates the complexity of special random graphs
BA_result = []
for item in methods:
    method = getattr(cx,item)
    temp_result = [method(g) for g in BA_graphs]
    BA_result.append(temp_result)

WS_result = []
for item in methods:
    method = getattr(cx,item)
    temp_result = [method(g) for g in WS_graphs]
    WS_result.append(temp_result)

NW_result = []
for item in methods:
    method = getattr(cx,item)
    temp_result = [method(g) for g in NW_graphs]
    NW_result.append(temp_result)
```

[122]:
```
# Calculates the complexities of special graphs
n=20
c=0
fig,axes = plt.subplots(4,2,figsize = (10,16))
xticks = np.linspace(n-1,n*(n-1)/2,5)
xticks = [int(item) for item in xticks]
for i in range(4):
    for j in range(2):
        axes[i][j].scatter([len(g.edges) for g in␣
 ↪BA_graphs],BA_result[c],s=15,color = "black",label = "BA")
        axes[i][j].scatter([len(g.edges) for g in␣
 ↪WS_graphs],WS_result[c],marker = "x",s=15,color = "red",label = "WS")
        axes[i][j].scatter([len(g.edges) for g in␣
 ↪NW_graphs],NW_result[c],marker = "s",s=15,color = "blue",label = "NW")
        axes[i][j].set_yticks([0,0.5,1])
        axes[i][j].set_ylabel(methods[c])
        axes[i][j].set_xlabel("m")
        axes[i][j].set_xticks(xticks)
        c+=1
axes[0][0].legend()
plt.legend()
plt.suptitle("Special random graphs' complexities with n = 20")
```

```
plt.show()
```


Special random graphs' complexities with n = 20
BA
WS
NW
C1est
C1espec
C2espec
OdC
MAg
Cr
Ce
MAri
m
19
61
104
147
190
0.0
0.5
1.0

[9]:

```
#Calculates the average complexities
m = np.linspace(n-1,n*(n-1)/2,int(n*(n-1)/2-n+1+1))
m = [int(item) for item in m]
```

[10]:

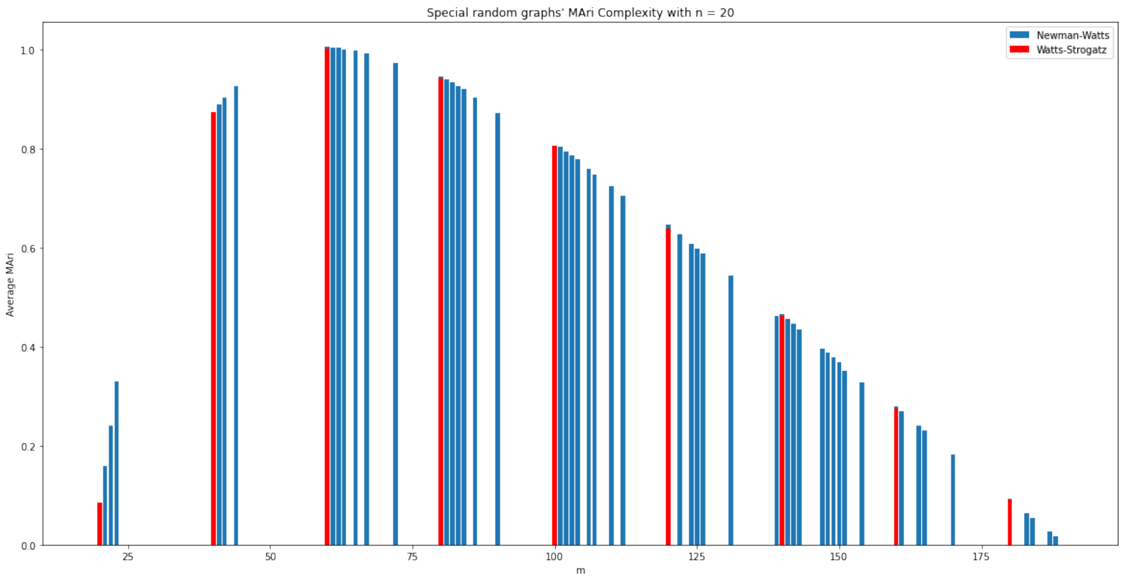
```
BA_avg = [0]*len(m)
WS_avg = [0]*len(m)
NW_avg = [0]*len(m)
c = [0]*len(m)
for g in WS_graphs:
    i = len(g.edges)
    WS_avg[i-n+1] = WS_avg[i-n+1] + cx.MAri(g)
    c[i-n+1] += 1
for i in range(len(c)):
    if c[i] != 0:
        WS_avg[i] = WS_avg[i]/c[i]

c = [0]*len(m)
for g in NW_graphs:
    i = len(g.edges)
    NW_avg[i-n+1] = NW_avg[i-n+1] + cx.MAri(g)
    c[i-n+1] += 1
for i in range(len(c)):
    if c[i] != 0:
        NW_avg[i] = NW_avg[i]/c[i]
```

[57]:

```
plt.figure(figsize=(20,10))
plt.bar(m,NW_avg,label = "Newman-Watts")
plt.bar(m,WS_avg,color = "red", label = "Watts-Strogatz")
plt.legend()
plt.xlabel("m")
plt.ylabel("Average MAri")
plt.title("Special random graphs' MAri Complexity with n = 20")
```

[57]:

```
Text(0.5, 1.0, "Special random graphs' MAri Complexity with n = 20")
```


Special random graphs' MAri Complexity with n = 20
Newman-Watts
Watts-Strogatz
Average MAri
m
0.0
0.2
0.4
0.6
0.8
1.0
25
50
75
100
125
150
175