EulerCharacteristicAlgo

May 2, 2022

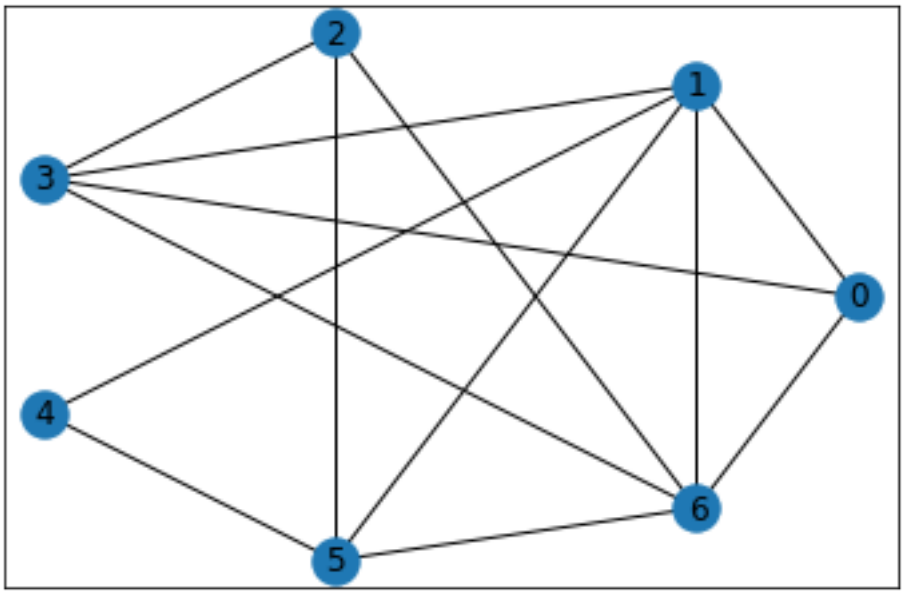
[3]:
```
#import relevant packages
from itertools import combinations
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
```

[6]:
```
#Computes Curvature of nodes in graph G
def curvature_of_nodes(G):
    cl = nx.enumerate_all_cliques(G)
    vertex_dict = dict()
    N = G.number_of_nodes()

    for i in range(N):
        curvature_of_i = 0

        #initialize dictionary keeping track of cliques of vertex
        int_i = dict()
        for num in range(N+1):
            int_i[num] = 0

        #add values to int_i
        cl = nx.enumerate_all_cliques(G)
        for j in nx.cliques_containing_node(G, nodes=i, cliques=cl):

            k = len(j)
            int_i[k] = int_i[k] + 1

        #computing curvature of each node
        for x in range(N+1):
            if x == 0:
                continue
            else:
                curvature_of_i = curvature_of_i + (-1)**(x+1) * int_i[x]/x
        vertex_dict[i] = curvature_of_i

    return vertex_dict
```

```
#sums the entire curvature
def curvature_sum(G):
    values = curvature_of_nodes(G).values()
    return sum(values)

#Computes the Euler Characteristic via discrete Gauss-Bonnet theorem
def Euler_Char(G):
    values = curvature_of_nodes(G).values()
    return round(sum(values))
```

[7]:
```
G = nx.erdos_renyi_graph(7, .7, seed=None, directed=False)
cl = nx.enumerate_all_cliques(G)
#print(nx.cliques_containing_node(G, nodes=1, cliques=cl))


nx.draw_networkx(G, pos=nx.circular_layout(G))

print(curvature_sum(G))
print(Euler_Char(G))
```

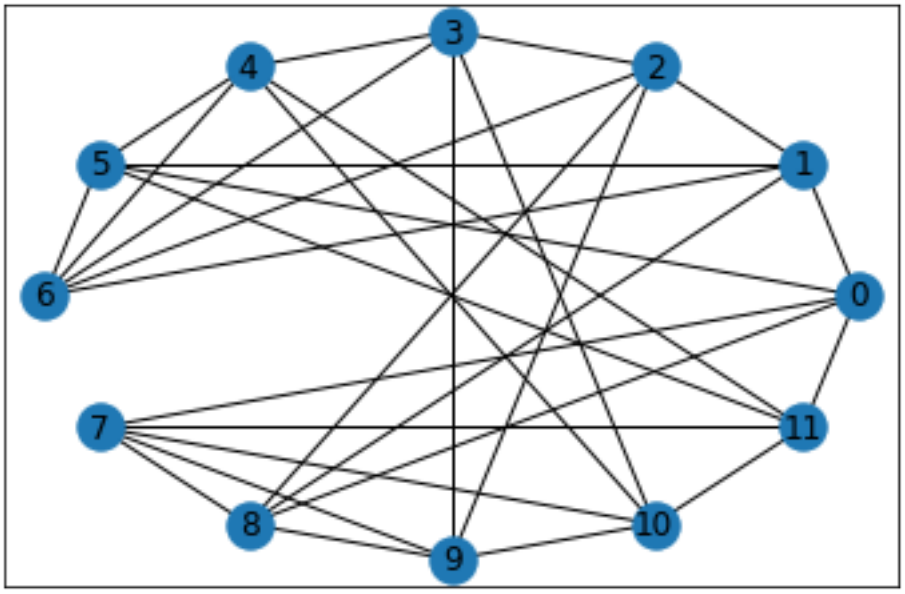
```
1.0
1
```



[8]:
```
G = nx.icosahedral_graph(create_using=None)
nx.draw_networkx(G, pos=nx.circular_layout(G))

print(Euler_Char(G))
```

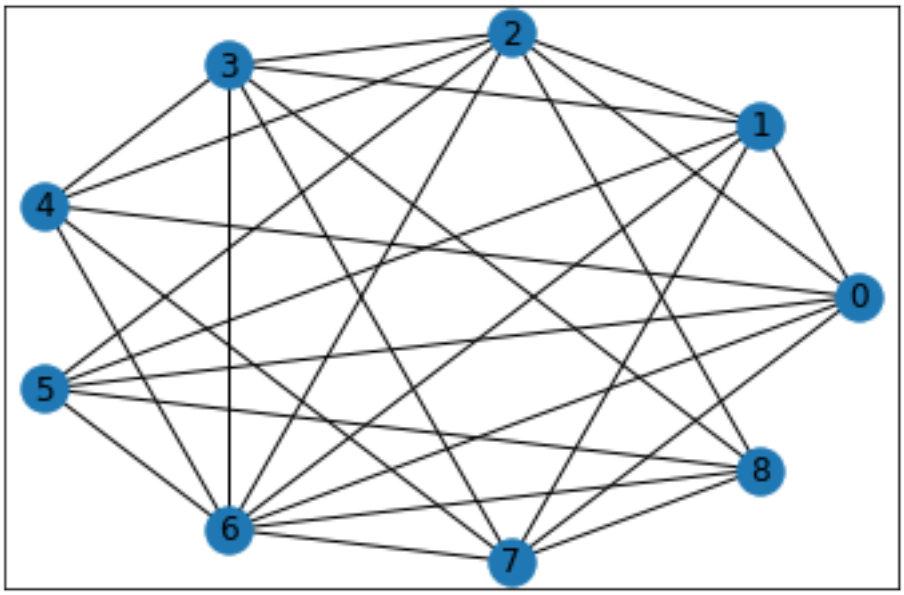
```
print(curvature_sum(G))
```

```
2
2.000000000000001
```



[11]:
```
#Generates Erdos Reyni Graph and computes its Euler Characteristic.
G = nx.erdos_renyi_graph(9, .7, seed=None, directed=False)

print(f"Euler Characteristic of G: {Euler_Char(G)}")
nx.draw_networkx(G, pos=nx.circular_layout(G))
```

```
Euler Characteristic of G: 1
```



[ ]:
```
#Returns possible p for which there is a topological phase transition (Euler
↪entropy becomes singular)
k = 10
S = np.linspace(0, 1, 101)

for j in S:
    X = 0
    for i in range(k):
        G = nx.erdos_renyi_graph(15, j, seed=None, directed=False)
        X = X + Euler_Char(G)
    avg = X/k
    if np.abs(avg) <= .1:
        print(f"Possible phase transition at p = {j} with average :{avg}")
        break
    print(j)
```