[1]:
```python
import numpy as np
import cvxpy as cp
import scipy
mat = scipy.io.loadmat('AdjacencyMatrix.mat')
G = mat['A']
```

[2]:
```python
n = len(G)
one = [1 for i in range(n)]
J = np.outer(one, one)
I = np.identity(n)

#auxiliary method to check if the list of vertices v forms a stable set in graph
def isStable(v):
    l = len(v)
    for i in range(l):
        for j in range(i+1,l):
            if(G[v[i]][v[j]] == 1):
                return False
    return True
```

Calculation of $\vartheta(G)$

[3]:
```python
X = cp.Variable((n,n), symmetric = True)
constraints = [X >> 0, cp.trace(X) == 1]
for i in range(n):
    for j in range(i,n):
        if(G[i][j] == 1):
            constraints.append(X[i][j] == 0)
constraints
prob = cp.Problem(cp.Maximize(cp.trace(J @ X)), constraints)
print(prob.solve(),"\n")
```

```
5.333333264771721
```

Calculation of $\alpha(G)$

[4]:
```python
#stable sets of length 5
s5 = 0
for i in range(n):
    for j in range(i+1,n):
        if(G[i,j]==1):
            continue
        for k in range(j+1,n):
            if(G[j,k] == 1 or G[i,k] == 1):
                continue
```

```
            for l in range(k+1,n):
                if(not isStable([i,j,k,l])):
                    continue
                for t in range(l+1,n):
                    if(isStable([i,j,k,l,t])):
                        print([i,j,k,l,t])
                        s5 = s5 + 1
print(s5)
```

```
0
```

[5]:
```
#stable sets of length 4
s4 = 0
for i in range(n):
    for j in range(i+1,n):
        if(G[i,j]==1):
            continue
        for k in range(j+1,n):
            if(G[j,k] == 1 or G[i,k] == 1):
                continue
            for l in range(k+1,n):
                if(isStable([i,j,k,l])):
                    #print([i,j,k,l])
                    s4 = s4 + 1
print(s4)
```

```
240
```

[6]:
```
stb = [0, 15, 51, 60]
print(np.array([[G[i][j] for i in stb] for j in stb]))
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

Calculation of $\vartheta'(G)$

Here we solve the following problem:

$$\vartheta'(G) = \begin{cases} \min\limits_{\substack{P \in S^n \\ k \in \mathbb{R}}} \; k \\ \quad \text{s.t. } k(I+A) - J - P \geq 0 \\ \qquad\quad P \succeq 0 \end{cases} \tag{(Q)}$$

[7]:
```
#solving the given problem for \vartheta'(G)
P = cp.Variable((n,n), symmetric = True)
k = cp.Variable(1)
```

```
constraints = [P >> 0]
for i in range(n):
    for j in range(i,n):
        constraints.append(k*(I[i][j]+G[i][j]) >= J[i][j] + P[i][j])
#constraints
prob = cp.Problem(cp.Minimize(k), constraints)
print(prob.solve(),"\n")
```

```
3.999999962758152
```

We showed that optval(Q) is equal to

$$\begin{aligned} \min_{\substack{X \in S^n \\ k \in \mathbb{R}}} \ & k \\ \text{s.t. } & kA \geq X \\ & kI + X \succeq J \end{aligned} \tag{(R)}$$

For sanity check, we also solve this problem and check.

[8]:
```
#solving the given problem for \vartheta'(G)
M = cp.Variable((n,n), symmetric = True)
l = cp.Variable(1)
constraints = [l*I+M >> J]
for i in range(n):
    for j in range(i,n):
        constraints.append(l*G[i][j] >= M[i][j])
#constraints
prob = cp.Problem(cp.Minimize(l), constraints)
print(prob.solve(),"\n")
```

```
3.9999999331124707
```