Stable sets and Lovasz number

Loading and initializing

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

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

#auxiliary method to check if the list of vertices v forms a clique in graph
def isClique(v):
    l = len(v)
    for i in range(l):
        for j in range(i+1,l):
            if(G[v[i]][v[j]] == 0):
                return False
    return True

#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
```

Solving SDP(s) and bound on $\alpha(G)$

This solves the original Lovasz SDP:

$$\begin{aligned} \max_{X \in S^n} \quad & \operatorname{Tr}(JX) \\ \text{s.t.} \quad & \operatorname{Tr}(X) = 1 \\ & X_{ij} = 0 \ \forall \{i,j\} \in E \\ & X \succeq 0 \end{aligned} \tag{(P)}$$

[3]:
```
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")
```

```
5.000000081544538
```

The following solves the modified SDP given in the problem set:

$$\begin{aligned} \min_{Z \in S^{n+1}} \quad & Z_{n+1,n+1} \\ \text{s.t.} \quad & Z_{n+1,i} = Z_{ii} = 1, i = 1, \dots, n \\ & Z_{ij} = 0 \text{ if } \{i,j\} \in \bar{E} \\ & Z \succeq 0. \end{aligned} \tag{(D')}$$

[4]:
```
Z = cp.Variable((n+1,n+1), symmetric=True)
constraints = [Z >> 0]
for i in range(n):
    constraints.append(Z[i][n] == Z[i][i])
    constraints.append(Z[i][i] == 1)
    for j in range(i+1,n):
        if(G[i][j] == 0):
            constraints.append(Z[i][j] == 0)
constraints
prob = cp.Problem(cp.Minimize(Z[n][n]), constraints)
print(prob.solve(),"\n")
```

```
4.999970218323207
```

Here we solve the dual of the original SDP (P). The dual is

$$\begin{aligned} \min_{\substack{Y \in S^n \\ t \in \mathbb{R}}} \; & t \\ \text{s.t. } & Y_{ij} = 0 \text{ if } \{i,j\} \in \overline{E} \\ & Y_{ii} = 0 \;\forall 1 \leq i \leq n \\ & tI - Y - J \succeq 0 \end{aligned} \tag{(D)}$$

[5]:
```
Y = cp.Variable((n,n), symmetric=True)
t = cp.Variable()
constraints = [t*np.identity(n) >> Y+J]
for i in range(n):
    constraints.append(Y[i][i] == 0)
    for j in range(i+1,n):
        if(G[i][j]==0):
            constraints.append(Y[i][j] == 0)
constraints
prob = cp.Problem(cp.Minimize(t), constraints)
print(prob.solve(),"\n")
```

```
5.000001684329688
```

Now we (try to) find stable sets of size 5 and 6 by brute force. If no stable set of size 6 is found, we can conclude that there is no stable set of that size, because the method is searching through all possible subsets of vertices of the given size.

[6]:
```
#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)
```

```
[2, 7, 9, 11, 46]
1
```

[7]:
```
#stable sets of length 6
s6 = 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):
                    for p in range(t+1,n):
                        s6 = s6 + isStable([i,j,k,l,t,p])
print(s6)
```

```
0
```

LP with clique inequalities

First we check that this graph has a clique of size 4.

[8]:
```
c4 = 0
for i in range(n):
    for j in range(i+1,n):
        if(G[i,j]==0):
            continue
        for k in range(j+1,n):
            if(G[j,k] == 0 or G[i,k] == 0):
                continue
            for l in range(k+1,n):
                c4 = max(c4,isClique([i,j,k,l]))
print(c4)
```

```
True
```

The following is the LP for η^2_{LP}:

$$\begin{aligned} \max_{x\in\mathbb{R}^n} & \sum_{i=1}^{n} x_i \\ \text{s.t. } & 0 \le x_i \le 1 \ \forall 1 \le i \le n \\ & x_i + x_j \le 1 \text{ if } \{i,j\} \in E \end{aligned}$$

[9]:
```
#C2
x = cp.Variable(n)
constraints = [0 <= x, x <= 1]
for i in range(n):
```

```
    for j in range(i,n):
        if(G[i][j]==1):
            constraints.append(x[i] + x[j] <= 1)
constraints
prob = cp.Problem(cp.Maximize(cp.sum(x)), constraints)
print(prob.solve(solver = cp.ECOS),"\n")
```

24.999999999752085

The following is the LP for η^3_{LP}:

$$
\begin{aligned}
&\max_{x\in\mathbb{R}^n} \sum_{i=1}^{n} x_i \\
&\text{s.t. } 0 \le x_i \le 1 \ \forall 1 \le i \le n \\
&\qquad x_i + x_j \le 1 \text{ if } \{i,j\} \in E && \text{removed} \\
&\qquad x_i + x_j + x_k \le 1 \text{ if } \{i,j\}, \{j,k\}, \{k,i\} \in E
\end{aligned}
$$

[10]:

```
#C3
x = cp.Variable(n)
constraints = [0 <= x, x <= 1]
for i in range(n):
    for j in range(i+1,n):
        if(G[i][j] == 0):
            continue
        for k in range(j+1,n):
            if(isClique([i,j,k])):
                constraints.append(x[i]+x[j]+x[k] <= 1)
constraints
prob = cp.Problem(cp.Maximize(cp.sum(x)), constraints)
print(prob.solve(solver = cp.ECOS),"\n")
```

16.666666666662305

The following is the LP for η^4_{LP}:

$$
\begin{aligned}
&\max_{x\in\mathbb{R}^n} \sum_{i=1}^{n} x_i \\
&\text{s.t. } 0 \le x_i \le 1 \ \forall 1 \le i \le n \\
&\qquad x_i + x_j \le 1 \text{ if } \{i,j\} \in E && \text{removed} \\
&\qquad x_i + x_j + x_k \le 1 \text{ if } \{i,j\}, \{j,k\}, \{k,i\} \in E && \text{removed} \\
&\qquad x_i + x_j + x_k + x_l \le 1 \text{ if } \{i,j\}, \{j,k\}, \{k,l\}, \{l,i\}, \{i,k\}, \{j,l\} \in E
\end{aligned}
$$

[11]:

```
#C4
x = cp.Variable(n)
```

```python
constraints = [0 <= x, x <= 1]
for i in range(n):
    for j in range(i+1,n):
        if(G[i][j] == 0):
            continue
        for k in range(j+1,n):
            if(G[i][k] == 0 or G[k][j] == 0):
                continue
            for l in range(k+1,n):
                if(isClique([i,j,k,l])):
                    constraints.append(x[i]+x[j]+x[k]+x[l] <= 1)
constraints
prob = cp.Problem(cp.Maximize(cp.sum(x)), constraints)
print(prob.solve(solver = cp.ECOS),"\n")
```

```
12.499999999999996
```

[12]:

```python
#stable set of size 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)
```

```
[2, 7, 9, 11, 46]
1
```

[13]:

```python
#stable set of size 6
s6 = 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):
                    for p in range(t+1,n):
                        s6 = s6 + isStable([i,j,k,l,t,p])
print(s6)
```

```
0
```

[14]:
```
#verify if this set of vertices is stable
stb = [2, 7, 9, 11, 46]
subG = np.array([[G[i,j] for i in stb] for j in stb])
print(subG)
```

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