CONVEX AND CONIC OPTIMIZATION
Homework 5

NILAVA METYA
nilava.metya@rutgers.edu
nm8188@princeton.edu

April 18, 2024

Problem 1

The Lovász sandwich theorem states that for any graph $G(V, E)$, with $|V| = n$, we have

$$\alpha(G) \underset{(1)}{\leq} \vartheta(G) \underset{(2)}{\leq} \chi(\bar{G}).$$

1. We proved inequality (1) in class. Prove inequality (2).

   Hint: You may want to first show that the optimal value of the following SDP also gives $\vartheta(G)$:

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

2. Given an example of a graph G where neither inequality (1) nor inequality (2) is tight.

Solution

1. Let $\mathcal{E}_{ij} = \mathcal{E}_{ji}^\top = e_i e_j^\top$.

$$\begin{aligned} -\min_{X \in S^n} \quad & \mathrm{Tr}(-JX) \\ \text{s.t.} \quad & \mathrm{Tr}(X) = 1 \\ & \mathrm{Tr}((\mathcal{E}_{ij} + \mathcal{E}_{ji})X) = 0 \ \forall \{i,j\} \in E \\ & X \succeq 0 \end{aligned} \tag{(P)}$$

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

   Negative of $\max t$ is same as $\min(-t)$. So let's replace t with $-t$ and change the problem (D) to that of minimizing t (by varying $Y \in S^n, t \in \mathbb{R}$) with all but the last constraint unchanged, and the last constraint changes to $tI \succeq \sum_{\{i,j\}\in E} Y_{ij}E_{ij} + J = Y + J$. That is, our dual is simply

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

Note that both (P) and (D) are strictly feasible. Indeed, $X^* = I_{n\times n}$ is feasible to (P) with $X^* \succ 0$, and $Y^* = 0, t^* = (1+\text{max eigenvalue of } J)$ is feasible to (D) with $t^*I - Y^* - J = t^*I - J \succ 0$.

Note that if (Y,t) is feasible then it must satisfy $0 \le e_1^\top(tI - Y - J)e_1 = t - 0 - 1 \implies t \ge 1$. In particular, division by (feasible) t is possible and preserves inequalities. Next note that $tI - Y - J \succeq 0 \iff t\left(I - \frac{1}{t}Y\right) - J \ge 0 \iff \left(I - \frac{1}{t}Y\right) - \frac{1}{t}J \succeq 0$. Noting that $J = \mathbf{1}\mathbf{1}^\top$ and $t > 0$, and using Schur complements, this is equivalent to $Z := \left[\begin{array}{c|c} I - \frac{1}{t}Y & \mathbf{1} \\ \hline \mathbf{1}^\top & t \end{array}\right] \succeq 0$. Here $\mathbf{1} = \begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} \in \mathbb{R}^n$.

Note that whenever (Y,t) is feasible to (D), then $Z = \begin{bmatrix} I - \frac{1}{t}Y & \mathbf{1} \\ \mathbf{1}^\top & t\end{bmatrix} \succeq 0$ satisfies $Z_{n+1,i} = Z_{ii} = 1 \ \forall 1 \le i \le n$ and $\{ij\} \in \bar{E} \implies Z_{ij} = \frac{-1}{t}Y_{ij} = 0$, whence Z is feasible to (D') and has same objective value $t = Z_{n+1,n+1}$.
Conversely if Z is feasible to (D') then taking $Y = Z_{n+1,n+1}\left(I - Z_{1:n,1:n}\right), t = Z_{n+1,n+1}$ is feasible to (D). Indeed, $\{ij\} \in \bar{E} \implies Y_{ij} = t(0 - Z_{ij}) = 0$ and $Y_{ii} = t(1 - Z_{ii}) = t(1-1) = 0$; further, $Z \succeq 0 \implies t = Z_{n+1,n+1} \ge 0, Z_{1:n,1:n} \succeq \frac{1}{t}\mathbf{1}\mathbf{1}^\top = \frac{1}{t}J \implies tI - Y \succeq J$. This proves feasiblity of (Y,t) to (D), and has the same objective value t as (D').

By the above discussion we conclude that (D) and (D') are equivalent and have the same optimal value $\vartheta(G)$. Thus (D') and (P) have the same optimal value $\vartheta(G)$.

Now we prove inequality (2). Say $G = G(V,E)$ admits a $k-$coloring $C : V \to [k] = \{1,\cdots,k\}$ where $k = \chi(G)$. We take $V = [n]$. In words, vertex i has color $C(i)$, and if $\{i,j\} \in E$ then $C(i) \ne C(j)$. For each color $c \in [k]$ let $\eta(c) = \sum_{i=1}^n \delta^c_{C(i)}$ be the number of vertices that are colored c. Assign a vector $\boldsymbol{v}_i := \dfrac{\boldsymbol{e}_{C(i)}}{\eta(C(i))} \in \mathbb{R}^k$ to each vertex i. Here $\boldsymbol{e}_c \in \mathbb{R}^k$ is the standard basis vector with 1 at position c and 0 elsewhere. Define $M = A^\top A \in S^{n+1}$ where $A = \begin{bmatrix}\boldsymbol{e}_{C(1)} & \cdots & \boldsymbol{e}_{C(n)} & \sum_i \boldsymbol{v}_i\end{bmatrix}$. By construction, $M \succeq 0$. Note the following:

- The c^{th} component of $\sum_j \boldsymbol{v}_j$: $\boldsymbol{e}_c^\top \sum_j \boldsymbol{v}_j = \sum_{j:C(j)=c} \boldsymbol{e}_c^\top \boldsymbol{v}_j = \eta(c)\boldsymbol{e}_c^\top \dfrac{\boldsymbol{e}_c}{\eta(c)} = 1$. Hence,
  - $M_{n+1,i} = \boldsymbol{e}_{C(i)}^\top \sum_j \boldsymbol{v}_j = 1 \ \forall 1 \le i \le n,$
  - $M_{n+1,n+1} = \sum_i \boldsymbol{v}_i^\top \sum_j \boldsymbol{v}_j = \sum_i \dfrac{1}{\eta(C(i))} = k$ where the last equality is because $\frac{1}{\eta(c)}$ occurs exactly $\eta(c)$ times in the given sum for each color $c \in [k]$, so each color contributes 1 to the sum.
- If $1 \le i,j \le n$ then $M_{ij} = \boldsymbol{e}_{C(i)}^\top \boldsymbol{e}_{C(j)} = \delta^{C(j)}_{C(i)}$. Thus
  - if $\{i,j\} \in \bar{E}$ then $C(i) \ne C(j)$ for the coloring to be consistent, which forces $M_{ij} = 0$,

  - and $M_{ii} = 1 \; \forall 1 \leq i \leq n$.

The red points above prove that M is feasible to (D'). The blue point above shows that this M has objective value $k = \chi(G)$. Thus the optimal value of (D'), namely $\vartheta(G)$, is *at most* $M_{n+1,n+1} = \chi(G)$ because it's a minimization problem.

2. The counterexample we present here is the cycle on 5 vertices C_5. First we notice that $\overline{C_5} \cong C_5$ ($\cong$ means graph isomorphism). Indeed if the original C_5 was numbered with $\{0,1,2,3,4\}$ with i and $(i+1) \pmod 5$ being connected, the isomorphism/relabelling $0 \mapsto 0, 2 \mapsto 1, 4 \mapsto 2, 1 \mapsto 3, 3 \mapsto 4$ makes it exactly C_5.

   We will show that $\vartheta(C_5)$ is not an integer by bounding the optimal value between two non-integers.

   Lower bound: For this case we consider the original (maximization) problem (P). Consider the instance $X = \begin{bmatrix} 0.2 & 0 & 0.12 & 0.12 & 0 \\ 0 & 0.2 & 0 & 0.12 & 0.12 \\ 0.12 & 0 & 0.2 & 0 & 0.12 \\ 0.12 & 0.12 & 0 & 0.2 & 0 \\ 0 & 0.12 & 0.12 & 0 & 0.2 \end{bmatrix}$. It satisfies the constraints that its trace is 1 and the entries corresponding to edges is 0. The principal minors are $0.2, 0.04, 0.00512, .00007936, .0000011264$, thus proving that $X \succ 0$. The objective value for this X is 2.2. This establishes $\vartheta(C_5) \geq 2.2$.

   Upper bound: We'll use (D'). Consider the instance $Z = \begin{bmatrix} 1 & 0 & 0.56 & 0.56 & 0 & 1 \\ 0 & 1 & 0 & 0.56 & 0.56 & 1 \\ 0.56 & 0 & 1 & 0 & 0.56 & 1 \\ 0.56 & 0.56 & 0 & 1 & 0 & 1 \\ 0 & 0.56 & 0.56 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 2.5 \end{bmatrix}$. Its principal minors are $1, 1, 0.6864, 0.15754496, 0.0338711552, 0.004793088$ which are all positive, and thus $Z \succ 0$. Z clearly satisfies that the last column (except bottom-right entry) is all 1's, diagonal is all 1's, and the entries corresponding to the edges of $\overline{C_5} \cong C_5$ are all 0. This has objective value 2.5, thus proving that $\vartheta(C_5) \leq 2.5$.

   We have, therefore shown that $\alpha(G) < 2.2 \leq \vartheta(G) \leq 2.5 < \chi(\overline{G})$ for $G = C_5$, because $\alpha(G), \chi(\overline{G}) \in \mathbb{Z}$.

   The above matrices were found by solving the corresponding problems (P) and (D') on `CVXPY` and rounding the optimal solutions to get a feasible solution, thus giving bounds on $\vartheta(G)$.

Problem 2

For a graph $G(V, E)$, with $|V| = n$, we saw in class that an SDP-based upperbound for the stability number $\alpha(G)$ of the graph is given by $\vartheta(G)$ (as defined in Problem 1). We also saw that alternative upperbounds on the stability number can be obtained through the following family of LP relaxations:

$$\begin{aligned} \eta_{LP}^{k} := \max & \sum_{i=1}^{n} x_i \\ \text{s.t. } & 0 \le x_i \le 1, i = 1, \dots, n \\ & C_2 \dots, C_k, \end{aligned}$$

where C_k contains all clique inequalities of order k, i.e. the constraints

$$x_{i_1} + \dots + x_{i_k} \le 1$$

for all $\{i_1, \dots, i_k\} \in V$ defining a clique of size k.

1. Show that for any graph G, we have $\vartheta(G) \le \eta_{LP}^{k} \forall k \ge 2$
2. The file Graph . mat contains the adjacency matrix of a graph G with 50 nodes (depicted above). Compute $\vartheta(G), \eta_{LP}^{2}, \eta_{LP}^{3}, \eta_{LP}^{4}$ and $\alpha(G)$ for this graph. You can directly load the data file in MATLAB. In Python, you can use the following code to do this.

   ```
   import scipy
   mat = scipy.io.loadmat('Graph.mat')
   G = mat['G']
   ```
3. Present a stable set of maximum size. Prove or disprove the claim that this graph has a unique maximum stable set.

Solution

1. We start by proving what's given as a hint.

   Lemma 1
   The optimum value of the following program equals $\vartheta(G)$:

$$\begin{aligned} \max_{Y \in S^{(n+1)}} \; & \sum_{i=1}^{n} Y_{ii} \\ \text{s.t. } & Y \succeq 0, \\ & Y_{n+1,n+1} = 1, \\ & Y_{n+1,i} = Y_{ii} \; \forall 1 \le i \le n \\ & Y_{ij} = 0, \text{ if } \{i, j\} \in E. \end{aligned} \qquad ((1))$$

   Proof. From problem 1, we recall that the optimum value of the following (slightly changed for standard

SDP form) program equals $\vartheta(G)$ which we named as (D').

$$\begin{aligned}
\min_{Z \in S^{n+1}} \quad & \operatorname{Tr}[\mathcal{E}_{n+1,n+1} Z] & & \\
\text{s.t.} \quad & \operatorname{Tr}[(\mathcal{E}_{n+1,i} + \mathcal{E}_{i,n+1})Z] = 2 \ \forall 1 \le i \le n & & Y_{i,n+1} \\
& \operatorname{Tr}[\mathcal{E}_{ii} Z] = 1 \ \forall 1 \le i \le n & & Y_{ii} \\
& \operatorname{Tr}[(\mathcal{E}_{ij} + \mathcal{E}_{ji})Z] = 0 \text{ if } \{i,j\} \in \bar{E} & & Y_{ij} \\
& Z \succeq 0 & &
\end{aligned} \tag{(D')}$$

where $\mathcal{E}_{ij} = e_i e_j^\top \in \mathbb{R}^{(n+1)\times(n+1)}$, and the blue symbols on the right denote the respective dual variables. These dual variables can be thought of as coming from a symmetric matrix Y whose (i,j) entry is useless, so equals to 0, if $\{ij\} \in E$ or $i = j = n+1$. Thus the dual of this problem is given by

$$\begin{aligned}
\max_{Y \in S^{n+1}} \quad & \sum_{i=1}^{n} (2Y_{i,n+1} + Y_{ii}) \\
\text{s.t.} \quad & E_{n+1,n+1} \succeq \sum_{i=1}^{n} Y_{i,n+1}(\mathcal{E}_{n+1,i} + \mathcal{E}_{i,n+1}) + \sum_{i=1}^{n} Y_{ii}\mathcal{E}_{i,i} + \sum_{\{ij\}\in\overline{E}} Y_{ij}(\mathcal{E}_{ij} + \mathcal{E}_{ji})
\end{aligned}$$

which is same as the program

$$\begin{aligned}
\max_{Y \in S^{n+1}} \quad & \sum_{i=1}^{n} (2Y_{i,n+1} + Y_{ii}) \\
\text{s.t.} \quad & E_{n+1,n+1} \succeq Y \\
& Y_{ij} = 0 \text{ if } \{i,j\} \in E \\
& Y_{n+1,n+1} = 0
\end{aligned} \tag{(P')}$$

By strong duality, the optimal value of (P') is $\vartheta(G)$ because of strict feasibility:

- In case of (P'), $Y = \begin{bmatrix} -1 & & & \\ & \ddots & & \\ & & -1 & \\ & & & 0 \end{bmatrix}$ gives $E_{n+1,n+1} - Y = I_{n+1} \succ 0$ and this Y satisfies the two other constraints, namely, the bottom-right entry is 0 and all entries corresponding to edges E are 0.
- The original problem (D') is strictly feasible because it is just a cosmetic modification of (D) in the previous problem which was shown to be strictly feasible.

Taking $M := E_{n+1,n+1} - Y$ in (P') gives the following cosmetic modification:

$$\begin{aligned}
\max_{M \in S^{n+1}} \quad & -\sum_{i=1}^{n} (2M_{i,n+1} + M_{ii}) \\
\text{s.t.} \quad & M \succeq 0 \\
& M_{ij} = 0 \text{ if } \{i,j\} \in E \\
& M_{n+1,n+1} = 1
\end{aligned} \tag{(P'')}$$

Claim 2
If M is feasible to (P'') and satisfies $M_{tt} + M_{t,n+1} \neq 0$ for some $1 \le t \le n$, then the objective can be strictly improved (increased) in the feasible region.

Proof. Let M, t be as in the hypothesis of the claim. The $2 \times 2-$minor formed by rows $t, n+1$ and columns $t, n+1$ is $M_{tt} - M_{t,n+1}^2$, which is non-negative by Sylvester. Consider the following cases on each M_{tt}:

- $M_{tt} = 0$: then $M_{t,n+1}^2 \le M_{tt} = 0 \implies M_{t,n+1} = 0$. So this case is impossible.
- $M_{tt} \ne 0$: So $e_t^\top M e_t = M_{tt} > 0$. Multiply M on left and right by $C = diag\left(1, \cdots, 1, \frac{-M_{t,n+1}}{M_{tt}}, 1, \cdots, 1\right)$ where the nontrivial fraction term is on the t^{th} position. Then $CMC \succeq 0$ (because $M' := CMC = CQ^\top QC = (CQ)^\top (CQ) \succeq 0$). This multiplies the t^{th} row and t^{th} column of M by $\frac{-M_{t,n+1}}{M_{tt}}$. Note that this new matrix $M' \succeq 0$ is feasible because the $(n+1, n+1)$ entry remains untouched and $(u,v)^{\text{th}}$ entry remains 0 if $\{u,v\} \in E$. But now $M'_{tt} = \frac{M_{t,n+1}^2}{M_{tt}}$ and $M'_{t,n+1} = \frac{-M_{t,n+1}^2}{M_{tt}}$ thus increasing the objective by $-\left(\frac{M_{t,n+1}^2}{M_{tt}} - \frac{2M_{t,n+1}^2}{M_{tt}}\right) + (M_{tt} + 2M_{t,n+1}) = \frac{M_{t,n+1}^2}{M_{tt}} + M_{tt} + 2M_{t,n+1} \overset{\text{AM}\ge\text{GM}}{\ge} 2\,|M_{t,n+1}| + 2M_{t,n+1} \ge 0$. Equality holds iff $M_{tt}^2 = M_{t,n+1}^2$ (for the first one) and $M_{t,n+1} \le 0$ (for the second one), which happens iff $M_{t,n+1} = -M_{tt}$. This condition is false, so the equality is strict. In other words, the objective can be strictly improved.

$\blacksquare$

Thus adding the constraint $M_{ii} + M_{i,n+1} = 0 \;\forall 1 \le i \le n$ does not change the optimal value. But this makes the objective equal to $\sum_{i=1}^n M_{ii}$. But note that $M = \begin{bmatrix} T & v \\ v^\top & 1 \end{bmatrix} \succeq 0 \iff M'' = \begin{bmatrix} T & -v \\ -v^\top & 1 \end{bmatrix} \succeq 0$ by the Schur complement of 1 (here $T \in S^n, v \in \mathbb{R}^n$). And $M_{ii} + M_{i,n+1} = 0 \iff M''_{ii} = M''_{i,n+1}$. Considering these, the program (P'') is just

$$\begin{aligned} \max_{M \in S^{n+1}} \; & \sum_{i=1}^n M_{ii} \\ \text{s.t.} \; & M \succeq 0 \\ & M_{ij} = 0 \text{ if } \{i,j\} \in E \\ & M_{n+1,n+1} = 1 \\ & M_{n+1,i} = M_{ii} \;\forall 1 \le i \le n. \end{aligned}$$

which is exactly the same as (1). $\blacksquare$

Claim 3
The spectrahedron that occurs as the feasible set of problem (1) is compact.

Proof. The spectrahedron is described by non-strict algebraic inequalities, hence closed. It is enough to show bounded. Say M is feasible. $M \succeq 0 \implies M_{ii} - M_{ii}^2 \ge 0$ (by Sylvester's theorem on the minnor coming from rows $i, n+1$ and columns $i, n+1$) whence $M_{ii} \in [0,1]$ for each $1 \le i \le n$. Now for any off diagonal entry i,j we have, by Sylvester's theorem on minor on rows i,j and columns i,j, $M_{ii}M_{jj} \ge M_{ij}^2 \implies M_{ij} \in [-1,1]$. So each entry of the M is bounded, proving that M is bounded. $\blacksquare$

Now suppose $Y = \begin{bmatrix} T & v \\ v^\top & 1 \end{bmatrix} \succeq 0$ is an optimal solution to this SDP (1) (which exists due to compactness), with optimal value $\vartheta(G)$. Equivalently $T - vv^\top \succeq 0$ and $v \in \mathbb{R}^n$ are the diagonal elements of T. Consider a point $x = v \in \mathbb{R}^n$, so $x_i = Y_{ii} = v_i \ge 0 \;\forall 1 \le i \le n$. $Y \succeq 0 \implies x_i - x_i^2 = Y_{ii} - Y_{i,n+1}^2 \ge 0 \forall i$ (by taking the minor of rows $i, n+1$ and columns $i, n+1$, and using Sylvester), whence $0 \le x_i \le 1 \forall i$. Furthermore if $(i_1, \cdots, i_k)$ form a clique of size k, then

$$\begin{aligned}
0 &\le \left(\sum_{j=1}^{k} e_{i_j}\right)^{\top} (T - vv^{\top}) \left(\sum_{j=1}^{k} e_{i_j}\right) \\
&= \sum_{j=1}^{k} (T - vv^{\top})_{i_j, i_j} + \sum_{1 \le j \ne s \le n} (T - vv^{\top})_{i_j, i_s} \\
&= \sum_{j=1}^{k} (T_{i_j, i_j} - x_{i_j}^2) + \sum_{1 \le j \ne s \le n} (T_{i_j, i_s} - x_{i_j} x_{i_s}) \\
&= \sum_{j=1}^{k} (x_{i_j} - x_{i_j}^2) - \sum_{1 \le j \ne s \le n} x_{i_j} x_{i_s} && \because \{i_j, i_s\} \in E \text{ if } j \ne s \\
&= \sum_{j=1}^{k} x_{i_j} - \left(\sum_{j=1}^{k} x_{i_j}\right)^2 = \left(\sum_{j=1}^{k} x_{i_j}\right)\left(1 - \sum_{j=1}^{k} x_{i_j}\right)
\end{aligned}$$

thus proving that $\sum_{j=1}^{k} x_{i_j} \in [0, 1]$. This means that x satisfies all clique inequalities, that is, it is feasible to the LP corresponding to η_{LP}^k. Since they are maximization problems, the optimal values η_{LP}^k should be **at least** as much as the objective of the above feasible solution namely $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} Y_{ii} = \vartheta(G)$ for each k.

2. Complete code at the end.

SDP and independence number:

I fed problems (P), (D') and (D) to CVXPY and got the following values for $\vartheta(G)$:

(P) : 5.000000081544538
(D') : 4.999970218323207
(D) : 5.000001684329688

Keeping numerical error in mind, this suggests that $\alpha(G) \le 5$. Indeed, we find by brute-force that there is no stable set of size 6 but $\{2, 7, 9, 11, 46\}$ is stable (and unique for size 5). So $\alpha(G) = 5$.

LP with clique inequalities:

We consider the variables $x \in \mathbb{R}^n$ satisfying $1 \le x_i \le 0\ \forall i$. If G has a clique of size k, then the constraints $C_2, \cdots, C_k$, can be replaced by only C_k because of the constraints $x_i \ge 0$ (the subgraph induced by a subset of vertices in a clique is itself a clique). Since we consider only $\eta_{LP}^2, \eta_{LP}^3, \eta_{LP}^4$, we check if the given graph has a $4-$clique. It indeed has cliques of size 4. Removing the lower clique inequality constraints from the LP saves $\sim$ 1sec for η_{LP}^3 and $\sim$ 15sec for η_{LP}^4.

The following code (in Python) checks if there is a $4-$clique. The `isClique()` helper method checks if a set of vertices form a clique.

```
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)
```

The CVXPY solver, for the LP's with cliques, gives

$$\begin{aligned}\eta^2_{LP} &\simeq 24.999999999\\ \eta^3_{LP} &\simeq 16.666666666\\ \eta^4_{LP} &\simeq 12.499999999.\end{aligned}$$

3. We presented a stable set of size 5 in the previous part: $\{2, 7, 9, 11, 46\}$. Note that vertex indexing starts from 0, not 1, here. Indeed the submatrix of the adjacency matrix for these rows and columns is found by the following code.

```
stb = [2, 7, 9, 11, 46]
subG = np.array([[G[i,j] for i in stb] for j in stb])
print(subG)
```

The output is the 5×5 matrix with all entries 0 suggesting that all of them are pairwise non-adjacent. Since we brute-forced over all possible $5-$subsets and checked whether it's stable, and the output was only the above set, this is the unique maximum stable set (checked by brute force by going through all possible $5-$subsets).

Problem 3

1. Consider two graphs G_A and G_B (with possibly a different number of nodes) and denote their adjacency matrices by A and B respectively. Express the adjacency matrix of their strong graph product $G_A \otimes G_B$ in terms of A and B.
2. Compute the Shannon capacity of the graph given in Problem 2.2.

Solution

1. We assume the graphs have no self loops. Say the adjacency matrix of $G_A \otimes G_B$ is H. Say $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{m\times n}$ so $H \in \mathbb{R}^{mn\times mn}$. In general, it is easy to see that if M is the adjacency matrix of a (simple) graph G then the adjacency matrix of the reflexive closure of G (that is, simply adding self loops on each vertex) is $H + I_{mn}$ (I is the identity matrix). We go to the reflexive closure because it's easy to describe the strong product of two graphs that have self loops on each of its vertices. Let the reflexive closures of G_A, G_B be F_A, F_B respectively. It is easy to describe the strong product of F_A, F_B. (i_1, i_2) is connected to (j_1, j_2) iff $\{i_1, j_1\} \in E(F_A)$ and $\{i_2, j_2\} \in E(F_B)$ [1]. From definition, it is clear that if we remove all the self loops from $F_A \otimes F_B$ we get back $G_A \otimes G_B$. In other words, the reflexive closure of $G_A \otimes G_B$ is $F_A \otimes F_B$. So $H + I$ is the adjacency matrix of $F_A \otimes F_B$. But

$$\begin{aligned}
&(H+I)_{(i_1,i_2),(j_1,j_2)} = 1\\
\iff &\{i_1, j_1\} \in E(F_A) \text{ and } \{i_2, j_2\} \in E(F_B)\\
\iff &(I+A)_{i_1,j_1}(I+B)_{i_2,j_2} = 1\\
\iff &((I+A)\otimes(I+B))_{(i_1,i_2),(j_1,j_2)} = 1
\end{aligned}$$

   where the last step follows by the fact on Kronecker products that $(X \otimes Y)_{(i_1,i_2),(j_1,j_2)} = X_{i_1,j_1} Y_{i_2,j_2}$.

   Thus $H + I_{mn} = (A + I_n) \otimes (B + I_m) = \underbrace{I \otimes I}_{I_{mn}} + A \otimes I_m + I_n \otimes B + A \otimes B \implies H = \boxed{A \otimes I_m + I_n \otimes B + A \otimes B}$.

2. We *assume* that the Lovasz SDP bound for $\alpha(G)$ is tight, so that $\vartheta(G) = 5$. From lectures we know that $5 = \vartheta(G) \geq \Theta(G) = \sup_k \alpha(G^k)^{\frac{1}{k}} \geq \alpha(G) = 5$ whence $\Theta(G) = 5$.

[1]we secretly combined the (edge $\vee$ equal) condition to one edge condition by declaring that each i is connected to itself.

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:

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

```
[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:

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

```
[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

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

[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}:

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

[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}:

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

[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}:

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

[11]:
```
#C4
x = cp.Variable(n)
```

```
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]:
```
#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]:
```
#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]]
```