ORFE 523: Convex and Conic Optimization
Homework 6

Zachary Hervieux-Moore

Thursday 4$^{\text{th}}$ May, 2017

Exercise 1: Show that the following decision problem is NP-hard:

Given a set of quadratic functions $q_i(x) = x^T Q_i x + b_i^T x + c_i$, $i = 1, \ldots, m$, defined with rational data $Q_i \in \mathbb{S}^{n \times n}$, $b_i \in \mathbb{R}^n$, $c_i \in \mathbb{R}$, decide if the set

$$S := \{x \in \mathbb{R}^n : q_i(x) \leq 0, i = 1, \ldots, m\}$$

is unbounded.

Answer: We will do a reduction from testing if a matrix is copositive which we known is NP-hard. A matrix Q is copositive if

$$x^T Q x \geq 0 \; \forall \; x \geq 0$$

Equivalently, a matrix is not copositive if

$$\exists x \geq 0 \text{ s.t. } x^T Q x < 0$$

That is, a matrix is not copositive if the following set is unbounded

$$S = \{x \in \mathbb{R}^n : -x \leq 0, x^T Q x < 0\}$$

This set is unbounded since if $\bar{x} \geq 0$ and $\bar{x}^T Q \bar{x} < 0$, then so is $\alpha \bar{x}$ for all $\alpha \in (0, \infty)$. Thus, S is unbounded if Q is not copositive. If Q is copositve, then $S = \emptyset$ which is bounded. Thus, Q copositve $\iff S$ bounded. Now, we have a strict inequality in S which is not of the form desired. However, we can change S to

$$S = \{x \in \mathbb{R}^n : -x \leq 0, x^T Q x + \epsilon \leq 0\}$$

without changing the outcome. This is because, if Q is not copositive, we can scale $\bar{x}^T Q \bar{x} < 0$ for some $\bar{x} \geq 0$ such that $\bar{x}^T Q \bar{x} \leq -\epsilon$. Thus, we again have that Q copositve $\iff S$ bounded. We have reduced testing copositivity to testing whether or not testing a set of quadratic inequalities is unbounded or not. Or, testing if the set

$$S := \{x \in \mathbb{R}^n : q_i(x) \leq 0, i = 1, \ldots, m\}$$

is unbounded is NP-hard.

Exercise 2:

1) Suppose you had a blackbox that given a 3SAT instance would tell you whether it is satisfiable or not. How can you make polynomial many calls to this blackbox to find a satisfying assignment to any satisfiable instance of 3SAT?

2) Suppose you had a blackbox that given a graph G and an integer k would tell you whether G has a stable set of size larger or equal to k. How can you make polynomially many calls to this blackbox to find a maximum stable set of a given graph?

Answer:

1) First, suppose there are n variables in the 3SAT instance. Suppose this instance is ϕ. First test if ϕ is satisfiable. If it is, then we try to find the assignment, if not, return that it is unsatisfiable. Now, to find the assignment, first append to ϕ the following clause

$$\phi^{(1)} = \phi \wedge (x_1 \vee x_1 \vee x_1)$$

Then put $\phi^{(1)}$ into the blackbox. This forces x_1 to be 1 if it is satisfiable. If it is not satisfiable, then $x_1 = 0$. We keep appending these variables according to what assignment satisfies the previous $\phi^{(i)}$. For example, if $\phi^{(1)}$ is unsatisfiable, then we force $x_1 = 0$ and test x_2 like so,

$$\phi^{(1)} = \phi \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1) \wedge (x_2 \vee x_2 \vee x_2)$$

Continue this chaining of clauses until you get to the n^{th} variable. this works because we are forcing each variable one by one to take a specific value but still confirming that the whole thing is satisfiable. After the n^{th} variable, this will return the satisfying assignment.

2) With our blackbox, we aim to remove nodes one by one. If a node is in a unique stable set of maximal size, then it will necessarily reduce the maximal stable set by 1. Note, that if there are multiple maximal stable sets, then we remove nodes until there is only one maximal stable set with no worry. First, we must find the size of the maximal stable set. This can be done in n calls size we know that $k \in [1, n]$. Thus, we can find k, the maximal stable set size, by inputing the original graph

and k from 1 to n and record the first time the black box returns no. When it returns no, the maximal stable set size is $k-1$.

Now, the algorithm goes as follows: randomly remove one of the nodes in the graph. Input this graph and k found as before. If the blackbox returns yes, the node is not in the stable set and we keep the node removed permanently. If the blackbox returns no, then the node is important and so we add it back to the graph. We also keep track of this node so that we do not draw it again. We repeat this process of removing nodes until we are left with k nodes. This is a maximal stable set.

This works because we ensure that there is some stable set of size k at each step of the algorithm. Thus, when we get down to k nodes, it must be the stable set.

Exercise 3: Recall that the spectral radius of a matrix $A \in \mathbb{R}^{n\times n}$, denoted by $\rho(A)$, is the maximum of the absolute values of its eigenvalues. We call a matrix "stable" if $\rho(A) < 1$. Let us call a pair of real $n \times n$ matrices $\{A_1, A_2\}$ stable if $\rho(\Sigma) < 1$, for any finite product Σ out of A_1 and A_2. (For example, Σ could be $A_2A_1, A_1A_2, A_1A_1A_2A_1$, and so on.)

Find the largest γ such that the pair

$$A_{1,\alpha} = \alpha \begin{bmatrix} -1 & -1 \\ -4 & 0 \end{bmatrix}, A_{2,\alpha} = \alpha \begin{bmatrix} 3 & 3 \\ -2 & 1 \end{bmatrix}$$

is stable for all $\alpha \in [0, \gamma)$. It is enough to find this cutoff value to three digits after the decimal point.

Answer: First, we come up with an upper bound for γ. We notice that that

$$\rho(A_{1,1}A_{2,1}) = 15.3459$$

Therefore, we must have that $\alpha < \frac{1}{\sqrt{15.3459}} = 0.2553$. Now, we write a YALMIP program that lowerbounds α that achieves this. We wish to find a homogeneous polynomial $V(x)$, such that $V(x)$ is a Lyapunov function. That is $V(A_ix) < V(x)$ and $V(x) > 0$. We relax this into a SOS by requiring $0 \leq V(x) - V(A_ix)$ and $0 \leq V(x)$. However, the trivial solution to this is $V(x) = 0$ so we must get rid of this solution by making the norm of x squared the lower bound. That is,

$$\sum_{i=1}^{n} x_i^d \leq V(x) - V(A_ix) \; \forall \; x \neq 0$$

$$\sum_{i=1}^{n} x_i^d \leq V(x) \; \forall \; x \neq 0$$

We have that if $V(x) > 0$ then $\sum_{i=1}^{n} x_i^d \leq V(x)$ for any $x \neq 0$ simply by rescaling. Thus, putting this together into a YALMIP code (appended below) and manually fiddling with α by hand, we see that we can achieve the upper bound. Thus, we conclude that $\gamma = 0.255$ up to three digits.

Code Appendix:

```
clear all;
clc;

sdpvar x1 x2
x = [x1;x2];

alpha = 0.2552;
A = [-1,-1;-4,0];
B = [3,3;-2,1];

d = 4;
[p, cp, mp] = polynomial(x,d,d)

% Lyapunov should be positive
F = [sos(p) - (x1^d + x2^d)];
% Lyapunov V(Ax) < V(x)
F = F + [sos(p - replace(p,[x1,x2],alpha*A*[x1;x2]) - (x1^d + x2^d))];
F = F + [sos(p - replace(p,[x1,x2],alpha*B*[x1;x2]) - (x1^d + x2^d))];
F = F + [cp];

options = sdpsettings('verbose',2,'solver','sdpt3');
[info,z,Q] = solvesos(F,[], options);

sol_Q=Q{1} % get the Gram matrix
sdisplay(z{1}) % get the monomial vector
eig(sol_Q) % check Gram matrix is psd
```

Exercise 4: Consider the following decision problem: Given $A_i \in \mathbb{S}^{n\times n}$, $b_i \in \mathbb{R}$, $i = 1, \ldots, m$, all with rational entries, decide if there exists a matrix $X \in \mathbb{S}^{n\times n}$ such that

$$\begin{aligned} \mathrm{Tr}(A_i X) &= b_i \\ X &\preceq 0 \end{aligned}$$

Determining the complexity of this question is one of the main outstanding open problem in semidefinite programming. At the moment, the problem is not even known to be in NP.

1) One may be tmepted to conclude that the problem is in NP because if the SDP is feasible, then we can just write down a solution and check its validity (testing positive semidefiniteness of a given $n \times n$ matrix can be done in $O(n^3)$). Produce a family of SDP feasibility problems where any feasible solution takes an exponential number of bits to write down with respect to the input size. (You don't need to put your SDP family in the "standard form" given above.)

2) Produce a family of SOCP feasibility problems where any feasible solution takes an exponential number of bits to write down with respect to the input size. *Hint: See if you can make your SDP problems involve only* 2×2 *matrices.*

3) Here is another shockingly simple problem whose complexity is unknown: Given positive integers $a_1, \ldots, a_r, k$, decide if

$$\sqrt{a_1} + \cdots + \sqrt{a_r} \leq k$$

Show that if SDP feasibility is in NP (resp. P), then this problem is in NP (resp. P).

Answer: 1) Consider the SDP where we check feasibility of the block matrices

$$\begin{bmatrix} x_i & x_{i-1} \\ x_{i-1} & 1 \end{bmatrix} \succeq 0, \ \forall\ i \in \{1, \ldots, n\}$$

Then, by Sylvester's criterion, we must have $x_i \geq x_{i-1}^2$ for all i. Thus, by recursively applying this rule, we get that

$$x_n \geq x_1^{2^n}$$

That is, the feasible solution takes an exponential number of bits to write down with respect to the input size n.

2) We can easily turn the SDP above by having the constraints be (again by Sylvester's)

$$\left\| \begin{matrix} (1 - e_i^T x)/2 \\ E_{i-1,i-1} x \end{matrix} \right\|_2 \leq (1 + e_i^T x)/2, \ \forall \ i \in \{1, \ldots, n\}$$
$$x_i \geq 0, \ \forall \ i \in \{1, \ldots, n\}$$

3) Define the SDP as

$$\begin{bmatrix} \frac{x_i}{a_i} & 1 \\ 1 & x_i \end{bmatrix} \succeq 0, \ \forall \ i \in \{2, \ldots, r\}$$
$$\sum_{i=1}^{n} x_i \leq k$$

Note that this is well posed since the a_i's are all positive integers and hence the data is rational. Then we get that $x_i \geq \sqrt{a_i}$. Thus, if we put these constraints, we get

$$\sqrt{a_1} + \cdots + \sqrt{a_r} \leq k$$

Thus, if SDP feasibility is NP (resp. P), then this problem is also in NP (resp. P) as this problem reduces to SDP feasibility.