The Algebraic Degree of Semidefinite Programming and MAX-CUT Problem

Ivan Stamenkovic
Mentor: Serkan Hosten

1. Introduction

We determine the algebraic degree for some classes of semidefinite programming (SDP) problems – in this case – complete graph MAX-CUT problems represented as SDP problems. We explore the relationship between their algebraic degree of SDP and number of vertices for each complete graph.

2. Semidefinite Programming

Let S^n denote set of all symmetric $n \times n$ matrices. Let $X \in S^n$. We say that X is *positive semidefinite* matrix, denoted as $X \succeq 0$, if

$$v^T X v \geq 0, \ \forall v \in \mathbb{R}^n \tag{2.1}$$

which is equivalent to the statements,

(i) all eigenvalues of X must be nonnegative.
(ii) all principal minors of X are nonnegative.

A semidefinite program is an optimization problem. Let S_+^n denote set of all *positive semidefinite* matrices. When written in standard primal form (SDP-P) we have a semidefinite program,

$$\begin{aligned} &\text{minimize (min)} \quad C \bullet X \\ &\text{subject to (}s.t.\text{)} \quad A_i \bullet X = b_i, \ i = 1, \ldots, m, \\ &\qquad X \in S_+^n \end{aligned} \tag{2.2}$$

where "$\bullet$" denotes Frobenius inner product,

$$C \bullet X = trace\left(C \cdot X\right) = \sum_{i=1, j=1}^{n} C_{ij} \cdot X_{ij} \tag{2.3}$$

It is assumed that the matrix C and matrices $A_1, .., A_m$ are symmetric $n \times n$ matrices.

From standard primal form it can be seen that SDP appears similar to linear programming problems (LP). Linear programs are optimization problems in standard primal form,

$$\begin{aligned} &\min \quad c \cdot x \\ &s.t. \;\; a_i \cdot x = b_i, \;\; i = 1,...,m, \\ &\qquad x \in \mathbb{R}^n_{0+} \end{aligned}$$

Also, we have LP in standard dual form,

$$\begin{aligned} &\max \quad \sum_{i=1}^{m} b_i y_i \\ &s.t. \;\; \sum_{i=1}^{m} a_i y_i + s = c \\ &\qquad s \in \mathbb{R}^n_{0+} \end{aligned}$$

While for LP in standard form it is assumed that each component of vector x is nonnegative $\left(x \in \mathbb{R}^n_{0+} \right)$, for SDP in standard form it is assumed that matrix X is *positive semidefinite* i.e. all eigenvalues of X must be nonnegative $\left(X \in S^n_+ \right)$. Thus, it is not a surprise that the set of all LP problems is the subset of the set of all SDP problems - all linear programming problems can be defined as semidefinite programming problems. This is done by (2.4(i-iii)) - (2.4.(i)) set SDP coefficient matrix diagonal (diag(C)) to coefficients of LP objective function (c_i, $i = 1,...,n$) and all non-diagonal entries of C to 0 ($\forall i \neq j,\ C_{ij} = 0,\ (i, j = 1,...,n)$), (2.4.(ii)), and (2.4.(iii)):

$$\begin{aligned} &(i) \;\; diag\left(C\right) = c_i;\ \forall i \neq j,\ C_{ij} = 0,\ (i, j = 1,...,n) \\ &(ii) \;\; diag\left(A_i\right) = \left(a_i\right)_j;\ \forall i \neq j,\ \left(A_i\right)_{jk} = 0,\ (i = 1,...,m; j,k = 1,...,n) \\ &(iii) \;\; diag\left(X\right) = x_i;\ \forall i \neq j,\ X_{ij} = 0,\ (i, j = 1,...,n) \end{aligned} \qquad (2.4)$$

However, for now, there is no plausible reason to define LP problem as SDP problems. Even though for SDP problems efficient numerical algorithms are developed for finding optimal solutions (such as Interior Point Methods (IPM)), the methods for solving LP are more efficient. In addition, LP IPMs are developed that are more efficient than LP Simplex method for large number of decision variables [3].

As in the case of LP, we have SDP in standard dual form (SDP-D),

$$\begin{aligned} &\max \quad b^T y \\ &s.t. \quad A(y) := C - \sum_{i=1}^{m} y_i A_i , \\ &\qquad A(y) \in S_+^n , y \in \mathbb{R}^m \end{aligned} \tag{2.5}$$

Example 1 We define a semidefinite programming problem in dual form

$$\max \quad -42y_1 - (65/2)y_2 - (79/2)y_3 - (21/2)y_4 - (43/2)y_5 - (17/2)y_6 + 309/2$$

$$s.t. \qquad A(y) := \begin{bmatrix} 1 & y_1 & y_2 & y_3 \\ y_1 & 1 & y_4 & y_5 \\ y_2 & y_4 & 1 & y_6 \\ y_3 & y_5 & y_6 & 1 \end{bmatrix} \succeq 0$$

3. The Algebraic Degree of SDP

For this part we follow similar steps as Jie et al. in Section 2 of [1]. First, we use Theorem 3 [1]. We suppose that both primal and dual SDPs are *strictly feasible* i.e. $\exists X \succ 0 : A_i \bullet X = b_i,\ i = 1,...,m$ (primal) and $\exists y \in \mathbb{R}^m : A(y) \succ 0$ (dual). Then, *strong duality* holds and we have a pair of optimal solutions $\left(A(\hat{y}), \hat{X} \right)$, and optimality conditions,

$$A_i \bullet \hat{X} = b_i,\ \forall i = 1,...,m \tag{3.1}$$

$$A(\hat{y}) \cdot \hat{X} = 0 \tag{3.2}$$

$$A(\hat{y}) \succeq 0 \ \ and \ \ \hat{X} \succeq 0 \tag{3.3}$$

Then, based on optimality conditions (3.1, 3.2, 3.3), we form a system of polynomial equations, and try to solve them symbolically by using Singular [4]. In order to illustrate this process we use the SDP problem defined in Example 1.

First, we have six linear polynomial equations based on optimality condition (3.1),

$$\begin{aligned} A_1 \bullet X - b_1 = 0 &\Leftrightarrow -2x_2 + 42 = 0 \\ A_2 \bullet X - b_2 = 0 &\Leftrightarrow -2x_3 + 65/2 = 0 \\ A_3 \bullet X - b_3 = 0 &\Leftrightarrow -2x_4 + 79/2 = 0 \\ A_4 \bullet X - b_4 = 0 &\Leftrightarrow -2x_6 + 21/2 = 0 \\ A_5 \bullet X - b_5 = 0 &\Leftrightarrow -2x_7 + 43/2 = 0 \\ A_6 \bullet X - b_6 = 0 &\Leftrightarrow -2x_9 + 17/2 = 0 \end{aligned} \tag{3.4}$$

Then, we have sixteen nonlinear polynomial equations from optimal condition (3.2) – we equate each of sixteen components of product $A(y) \cdot X$ to 0 (only some are shown below),

$$\begin{aligned} & x_1 + x_2 y_1 + x_3 y_2 + x_4 y_3 = 0 \\ & x_2 + x_5 y_1 + x_6 y_2 + x_7 y_3 = 0 \\ & \ldots \\ & x_4 y_3 + x_7 y_5 + x_9 y_6 + x_{10} = 0 \end{aligned} \tag{3.5}$$

Note that matrix $X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_5 & x_6 & x_7 \\ x_3 & x_6 & x_8 & x_9 \\ x_4 & x_7 & x_9 & x_{10} \end{bmatrix}$

According to Jie et al. [1], the algebraic degree of SDP is "the highest degree of the minimal polynomials of the optimal solution coordinates $\hat{y}_i$ and $\hat{x}_{jk}$." Thus, before determining algebraic degree for Example 1, we need to find optimal solution of Example 1.

Two systems of equations (3.4) and (3.5) form a nonlinear system of twenty two polynomial equations in sixteen variables. In order to determine optimal solution, we solve the system in Singular by using integrated solve() function which give us the list of solutions (Note: This exact method, which uses SDP primal and dual form, is quite inefficient to find optimum solution for even smaller SDP problems). We determine that there are fourteen solutions of which we discard two complex solutions. We plug twelve remaining real solutions into dual SDP objective function to find exact optimum solution,

$$\begin{aligned} & \hat{x}_1 = 83.4550..., \hat{x}_2 = 21, \hat{x}_3 = 16.25, \hat{x}_4 = 19.75, \hat{x}_5 = 21.8798..., \\ & \hat{x}_6 = 5.25, \hat{x}_7 = 10.75, \hat{x}_8 = 3.2453..., \hat{x}_9 = 4.25, \hat{x}_{10} = 6.6871..., \\ & \hat{y}_1 = 1.3573..., \hat{y}_2 = -2.8489..., \hat{y}_3 = -3.3247..., \\ & \hat{y}_4 = -6.3151..., \hat{y}_5 = -1.6026..., \hat{y}_6 = 17.9305..., \end{aligned} \tag{3.6}$$

Next, we use Grobner bases method in Singular [4] to determine univariate polynomial in terms of y_1

$$\begin{aligned} f(y_1) &= 1056340806375930088153140 0y_1^{8} + \\ &+5413684833352588038529783 5y_1^{7} \\ &+.....+480519244588747484701296 77 \end{aligned} \tag{3.7}$$

which can be factored to three polynomials, namely $g(y_1),\ h(y_1),\ p(y_1)$

$$f(y_1) = g(y_1)\cdot h(y_1)\cdot p(y_1)$$

where

$$\begin{aligned} g(y_1) &= y_1\text{-}1 \\ h(y_1) &= 1056340806375930088153140 0y_1^{6} + \\ &5413684833352588038529783 5y_1^{5} +... \\ &-480519244588747484701296 77 \\ p(y_1) &= y_1+1 \end{aligned} \tag{3.8}$$

We also plug in optimal value $\hat{y}_1 = 1.3573...$ into h to see that $\hat{y}_1$ satisfies $h(\hat{y}_1) = 0$. The highest degree of polynomial *h* is six. Thus, we can conclude that $\hat{y}_1$ has algebraic degree six. Same occurs with other $\hat{y}_i$ and $\hat{x}_{jk}$ optimal values. Thus, SDP problem as defined in Example 1 has algebraic degree of six.

In [2], Ranestad and Bothmer devised a general formula for the algebraic degree for SDP. They defined algebraic degree of SDP - $\delta(m,n,r)$ - to be "a measure of the algebraic complexity of the rank *r* solution for a *general* objective function optimized over a *generic m*-dimensional affine space of symmetric $(n\times n)$-matrices." [2]

Here is their general formula (Theorem 1.1 and definition of ψ-function in [2]) in its entirety,

(Theorem 1.1. in [2]) *The algebraic degree,*

$$\delta(m,n,r) = \sum_I \psi_I \psi_{I^c},$$

where the sum runs over all strictly increasing subsequences $I = \{i_1,...,i_{n-r}\}$ *of* $\{1,...,n\}$ *of length n − r and sum* $i_1+...+i_{n-r} = m$, *and* I^c *is the complement* $\{1,...,n\}\setminus I$.

$$\psi_i = 2^{i-1}, \psi_{i,j} = \sum_{k=i}^{j-1}\binom{i+j-2}{k} \text{ when } i<j$$

and

$$\psi_{i_1,\ldots,i_r} = Pf\left(\psi_{i_k,i_l}\right)_{1\le k<l\le r} \text{ if } r \text{ is even}$$
$$\psi_{i_1,\ldots,i_r} = Pf\left(\psi_{i_k,i_l}\right)_{0\le k<l\le r} \text{ if } r \text{ is odd}$$

where $\psi_{i_0,i_k} = \psi_{i_k}$ and *Pf* denote Pfaffian

They also note that this formula for $\delta(m,n,r)$ is valid only within Pataki's Inequalities [1, Proposition 5],

Pataki's Inequalities [1, Proposition 5]. *Let r and n – r be ranks of the optimal matrices* $\hat{Y} = A(\hat{y})$ *and* $\hat{X}$ *. Then,*

$$\binom{n-r+1}{2} \le m \quad \textit{and} \quad \binom{r+1}{2} \le \binom{n+1}{2} - m \tag{3.9}$$

Unfortunately, there exist SDP problems with the rank of optimal solution matrix $A(\hat{y})$ outside Pataki's Inequalities as noted by Nie et al. in [1]. In addition, based on Ranestad and Bothmer definition of algebraic degree [2], both objective function and variables matrix must be *general/generic* - they cannot be special. If that is not the case, the general formula $\delta(m,n,r)$ as defined in [2] cannot be used to determine algebraic degree of SDP.

Note that both $\delta(m,n,r)$ and ψ functions are implemented and source code provided to be used in Singular [4] (see Addendum A – sdp.lib).

4. MAX-CUT

MAX-CUT problem can be stated briefly - given undirected graph $G = (V, E)$ with *n* vertices and *m* edges the goal is to partition *G* into two partitions in such a way that the total sum of weights of edges that crosses partition boundary is as large as possible. It is assumed that the weights of the edges, $w_{ij} \ge 0$.

In 1995, Goemans and Williamson [5] devised a rounding algorithm to approximate exact optimal solution – total weight of the MAX-CUT – to at least to 0.878 times exact optimal solution by using SDP. They first defined MAX-CUT problems as "an optimization problem with quadratic objective function and Boolean variables." [3] For each vertex *V*, they defined Boolean variable x_i ,

$$x_i = \begin{cases} 1 & \text{if vertex } i \text{ is in partition 1} \\ -1 & \text{if vertex } i \text{ is in partition 2} \end{cases} \quad i = 1,\ldots,n$$

As noted in [3],

$$x_i x_j = \begin{cases} 1 & \text{if edge}(i,j) \text{ doesn't cross partition boundary} \\ -1 & \text{if edge } (i,j) \text{ crosses partition boundary} \end{cases} \quad i,j=1,...,n,\ i \neq j$$

The weight of MAX-CUT can be defined as

$$OPT := \max \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j) = \max \frac{1}{4} x^T L x, \tag{4.1}$$

where

$$L = -W + diag(We),$$

where W consists of zero diagonal and off-diagonal weights of the edges $w_{ij} \geq 0,\ i,j=1,...,n$, and e represents all ones vector.
As qtd.in [3], the problem (4.1) can be relaxed to SDP by noting that

$(i)\ x^T L x = trace(L x x^T)$
$(ii)\ X := x x^T \succeq 0$
(iii) every diagonal element of X is 1 since $x_i^2 = 1$
$(iv)\ X$ has rank one

We drop *(iv)* to get SDP relaxation,

$$\begin{aligned} & OPT \leq \overline{OPT} := \max \frac{1}{4} trace(LX) \\ & s.t.\ diag(X) = e, \\ & \quad X \succeq 0 \end{aligned} \tag{4.2}$$[1]

Note that Example 1 is MAX-CUT problem represented as SDP (see procedure example1() in sdp.lib, Addendum A).

5. Computations and Conclusions

For now, we are not interested to use SDP defined in (4.2) to solve MAX-CUT problem by using Goemans and Williamsons rounding algorithm [5]. We use (4.2) to try to determine algebraic degree of SDP relaxation of MAX-CUT problem. We consider complete graphs only i.e. we restrict weights $w_{ij} \geq 1,\ i,j=1,...,n$. We consider graphs

[1] *diag(X)=e* makes *X* matrix special (non-generic)

with $n > 2$ vertices. For each run of procedure cgmc(*n*) defined in sdp.lib (see Addendum A) we also randomize weights from 1 to 100. This make sure that objective function is *general*. We run procedure cgmc(*n*) 100 times for *n*=3, 4, and 50 times for *n*=5. The results are shown in the next table:

n	Algebraic Degree	$\delta(m,n,r)$
3	1: 100/100 (100 %)	not defined ($\delta(3,3,r)$)
4	6: 100/100 (100 %)	$\delta(6,4,2)=30$
5	27: 48/50 (96%) 26: 2/50 (4 %)	$\delta(10,5,2)=207$
6	102(?)	$\delta(15,6,2)=1400$

As seen from the table, we determine that algebraic degrees of SDP of complete graph MAX-CUT problem are 1, 6 and 27 respectively for almost all the runs (if all weights of the graph are 1, algebraic degree of SDP remains at 1 regardless of *n*). Calculated algebraic degrees for *n*=3, 4 and 5 doesn't fit Ranestad's and Bothmer's general formula. For allowed Pataki's Inequalities range for rank of the optimal solution we get algebraic degree of SDP that differs from 1, 6 and 27 (and 26). We determine that differences are due to the SDP relaxation constraint *diag(X)=e* in (4.2) which is non-generic. If that is the case, we cannot use $\delta(m,n,r)$ to determine algebraic degree. As Example 1 illustrates, the rank of the optimal solution is two (see rank_ex1() in sdp.lib – Addendum A) which is within Pataki's Inequalities. Also, the objective function is general. However, matrix $A(y)$ is special since $diag(A(y))=e$.

For $n > 5$, we encountered computation difficulties. Under VM with defined 4GB RAM (3.4GB max used by 32-bit Linux) and Ubuntu Linux, Singular [4] would not finish the Grobner bases method for days – the Singular process would just be terminated without showing any error or with memory full error. However, in one SDP instance, for *n*=6, after several days, we get that algebraic degree of that SDP problem is 102. This instance is defined in sdp.lib as procedure n6_1() – please note that this SDP problem is created with random weights from 1 to 5, and that there is no (1/4) defined in objective function (weights can be multiplied by 4 if there is need to call instead commented objective function).

After querying Sloane (Encyclopedia of Integer Sequences) for sequence 1, 6, 27 and 102, we get two results – A057222 (number of $4 \times n$ binary matrices with 1 unit column up to row and column permutations), and A166888. We do not consider A166888 since this sequence is not always increasing. We have sequence A057222,

n	3	4	5	6	7	8	…
sequence	1	6	27	102	333	969	…

In order to make conjecture that *in general* algebraic degrees of SDP of complete graph MAX-CUT problem for $n>2$ match with A057222 sequence, we need:

(i) to explore cases for $n=5$ where algebraic degree is equal to 26.
(ii) to find at least another case n=6 for which algebraic degree is equal to 102.

References

[1] J. Nie, K. Ranestad and B. Sturmfels. The algebraic degree of semidefinite programming. http://arxiv.org/abs/math/0611562v3

[2] H.C. von Bothmer and K. Ranestad: A general formula for the algebraic degree in semidefinite programming. http://arxiv.org/abs/math/0701877v1

[3] E. de Klerk. *Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications*. Kluwer, 2002.

[4] G.-M. Greuel, G. Pfister, and H. Schonemann. Singular 3-1-0: A computer algebra system for polynomial computations. http://www.singular.uni-kl.de (2009).

[5] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM,* 42(6): 1115–1145, 1995.

Addendum A - The source code of Singular library sdp.lib used in the project:

```
//IS, last modified 2.1.10
////////////////////////////////////////////////////////////////////////////
version="$Id: sdp.lib,v 1.2 2010/1/2 9:09:20 Exp $";
category="Semidefinite programming";
info="
LIBRARY:  sdp.lib     Procedures for Semidefinite programming and beyond
AUTHOR:                  Ivan Stamenkovic, ivanhoe@sfsu.edu

NOTE:

PROCEDURES:
              [parameters in square brackets are optional]
";

LIB "general.lib";
LIB "standard.lib";
LIB "crypto.lib";
LIB "poly.lib";
LIB "matrix.lib";
LIB "solve.lib";
LIB "linalg.lib";

////////////////////////////////////////////////
proc sdpmax(int m, int n, poly o, matrix Ay)
"USAGE:  sdpmax(m,n,o,Ay);
RETURN:
EXAMPLE:example sdpmax; shows an example
"
{
  int i,j,k;
  poly o1=o;
  matrix x[n][n] = symmat(n);
  matrix s[n][n];
  ideal I;
  matrix v[m][1];
  for (i=1; i<=m; i++)
  {
     v[i,1] = o/y(i);
     if (i==1) { I=-trace(transpose(Ay/y(i))*x)-v[i,1]; } else
     { I=I, -trace(transpose(Ay/y(i))*x)-v[i,1]; }
  }
  s = Ay*x;
  for (i=1; i<=n; i++)
  {
   for (k=1; k<=n; k++)
   {
    I=I,s[i,k];
   }
  }
  print(x);
  I;
  solve(I);
  option(redSB);
  ideal J = groebner(I);
J;
factorize(J[1]);
//  G=simplify(G,1);
// vdim(J);

//  maxdeg(factorize(J[1])[1][2]);
//  list l=list(o1);
//  listvar(r1);
//  export(v);
//  v;
//  def AR2=solve(I,6,0,"nodisplay");
//  setring AR2;
//  listvar();
```

```
//  int nsm;
//  for (i=1; i<=size(SOL); i++)
//  {
//   nsm=0;
//   for (j=1; j<=m; j++)
//   {
//     nsm=nsm+v[j,1]*SOL[i][j];
//    nsm=nsm+o/y(j)*SOL[i][j];
//   }
//   nsm;
//  }

}
example
{ "EXAMPLE:"; echo = 2;
  int m = 3;
  int n = 4;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(m..1)),lp;
  poly ofn = y(1)+y(2)+y(3);
  matrix Ay[n][n] = y(3)+1,y(1)+y(2),y(2),y(2)+y(3),
                    y(1)+y(2),1-y(1),y(2)-y(3),y(2),
                   y(2),y(2)-y(3),y(2)+1,y(1)+y(3),
                   y(2)+y(3),y(2),y(1)+y(3),1-y(3);
  sdpmax(m,n,ofn,Ay);
}

////////////////////////////////////////////
proc sdpmin(int m, int n, matrix c, vector b, list #)
"USAGE:  sdpmin(m,n,c,b,A1..Am);
RETURN: the (decimal) number corresponding to the hexadecimal number s
EXAMPLE:example decimal; shows an example
"
{
  if (m!=size(#)) { return("Error!"); }
//  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  int i,k;
  matrix x[n][n] = symmat(n);
  matrix s[n][n] = c;
  ideal I;
  for (i=1; i<=m; i++)
  {
    if (i==1) { I = trace(transpose(#[i])*x)-b[i]; } else
     { I = I,trace(transpose(#[i])*x)-b[i]; }
        s = s - y(i)*#[i];
  }
  s = s*x;
  for (i=1; i<=n; i++)
  {
   for (k=i; k<=n; k++)
   {
         I=I,s[i,k];
       }
      }
  I;
  option(redSB);
  ideal G = groebner(I);
  G;
//  solve(G);
  factorize(G[1]);
//  maxdeg(factorize(G[1])[1][2]);

}
example
{ "EXAMPLE:"; echo = 2;
  int m = 2;
  int n = 3;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  matrix a1[n][n] = 1,0,1,0,3,7,1,7,5;
  matrix a2[n][n] = 0,2,8,2,6,0,8,0,4;
  matrix c[n][n] = 1,2,3,2,9,0,3,0,7;
```

```
  vector b = [11,19];
  sdpmin(m,n,c,b,a1,a2);
}


////////////////////////////////////////////////
proc n6_1()
"USAGE:  n6_1();
RETURN: nothing
EXAMPLE:example n6_1; shows an example
"
{
  int n = 6;
  int m = n*(n-1)/2;
  int i,j,k;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  export(r1);
  matrix W[n][n]=0,5,5,2,3,4,
                 5,0,5,5,3,3,
                 5,5,0,5,1,3,
                 2,5,5,0,4,5,
                 3,3,1,4,0,4,
                 4,3,3,5,4,0;
  matrix L[n][n];
  matrix e[n][1];
  for (i=1; i<=n; i++) { e[i,1]=1; };
print(W);
 L = -W + diag(W*e);
//L;

 matrix Ay[n][n];
 k=0;
 for (i=1; i<=n; i++)
  {
   Ay[i,i]=1;
   for (j=i+1; j<=n; j++)
   {
          if (i!=j)
          {
           k=k+1;
           Ay[i,j]=y(k); Ay[j,i]=y(k);
          }; //end if
         } // end for
       } // end for
// poly ofn = (1/4)*trace(L*Ay);
 poly ofn = trace(L*Ay);
 m;n;ofn;
 print(Ay);
 sdpmax(m,n,ofn,Ay);
}
example
{ "EXAMPLE:"; echo = 2;
   n6_1();
}

////////////////////////////////////////////////
proc t4()
"USAGE:  t4();
RETURN: nothing
EXAMPLE:example t4; shows an example
"
{
  int m = 3;
  int n = 4;
  ring r1 = (real,6),(x(1..(n*(n+1)/2)), y(m..1)),lp;
//  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  poly ofn = y(1)+y(2)+y(3);
  y(1)=0.33774199;y(2)=0.57245486;y(3)=0.32547533;
  matrix Ay[n][n] = 0.32547533+1,0.33774199+0.57245486,0.57245486,0.57245486+0.32547533,
                    0.33774199+0.57245486,1-0.33774199,0.57245486-0.32547533,0.57245486,
```

```
                  0.57245486,0.57245486-0.32547533,0.57245486+1,0.32547533+0.33774199,
                  0.57245486+0.32547533,0.57245486,0.32547533+0.33774199,1-0.32547533;
//  matrix Ay[n][n] = y(3)+1,y(1)+y(2),y(2),y(2)+y(3),
//                  y(1)+y(2),1-y(1),y(2)-y(3),y(2),
//                  y(2),y(2)-y(3),y(2)+1,y(1)+y(3),
//                  y(2)+y(3),y(2),y(1)+y(3),1-y(3);
  print(Ay);
  mat_rk(Ay);
//  sdpmax(m,n,ofn,Ay);
}
example
{ "EXAMPLE:"; echo = 2;
   t4();
}

////////////////////////////////////////////////
proc rank_ex1()
"USAGE:  rank_ex1();
RETURN: the rank of A(y^) optimal solution of example 1
EXAMPLE:example rank_ex1; shows an example
"
{
  int m = 6;
  int n = 4;
  ring r1 = (real,6),(x(1..(n*(n+1)/2)), y(m..1)),lp;
//  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  poly ofn = -42*y(1)-65/2*y(2)-79/2*y(3)-21/2*y(4)-43/2*y(5)-17/2*y(6)+309/2;
  y(1)=1.3573077;y(2)=-2.84894345;y(3)=-3.3247203;y(4)=-6.31517748;y(5)=-
1.60266376;y(6)=17.93051;
  matrix Ay[n][n] = 1,1.3573077,-2.84894345,-3.3247203,
                  1.3573077,1,-6.31517748,-1.60266376,
                  -2.84894345,-6.31517748,1,17.93051,
                  -3.3247203,-1.60266376,17.93051,1;
  print(Ay);
  mat_rk(Ay);
//  sdpmax(m,n,ofn,Ay);
}
example
{ "EXAMPLE:"; echo = 2;
   rank_ex1();
}



////////////////////////////////////////////////
// Complete Graph
////////////////////////////////////////////////
proc cgmc(int n)
"USAGE:  cgmc(int n);
RETURN: nothing
EXAMPLE:example cgmc; shows an example
"
{
  int m = n*(n-1)/2;
  int i,j,k;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  export(r1);
  matrix W[n][n];
  matrix L[n][n];
  matrix e[n][1];
  for (i=1; i<=n; i++) { e[i,1]=1; };

  for (i=1; i<=n; i++)
  {
   for (j=1; j<=n; j++)
   {
          if (i!=j)
          {
//if (random(0,10)==1)
//{
```

```
// k=random(1,10);
//} else { k=0;}
         k = random(1,100);
//       k = 1;
         W[i,j]=k; W[j,i]=k;
         }; //end if
        } // end for
       } // end for

print(W);
 L = -W + diag(W*e);
// L;

 matrix Ay[n][n];
 k=0;
 for (i=1; i<=n; i++)
  {
   Ay[i,i]=1;
   for (j=i+1; j<=n; j++)
   {
          if (i!=j)
          {
           k=k+1;
           Ay[i,j]=y(k); Ay[j,i]=y(k);
          }; //end if
        } // end for
       } // end for
 poly ofn = (1/4)*trace(L*Ay);
// mat_rk(L);
// poly ofn = trace(L*Ay);
 m;n;ofn;
print(Ay);
 sdpmax(m,n,ofn,Ay);

}
example
{ "EXAMPLE:"; echo = 2;
   cgmc(4);
}



////////////////////////////////////////////////
proc t2()
"USAGE:  t2();
RETURN: nothing
EXAMPLE:example t2; shows an example
"
{
  int m = 3;
  int n = 4;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(m..1)),lp;
//  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  poly ofn = y(1)+y(2)+y(3);
  matrix Ay[n][n] = y(3)+1,y(1)+y(2),y(2),y(2)+y(3),
                    y(1)+y(2),1-y(1),y(2)-y(3),y(2),
                    y(2),y(2)-y(3),y(2)+1,y(1)+y(3),
                    y(2)+y(3),y(2),y(1)+y(3),1-y(3);
  sdpmax(m,n,ofn,Ay);
}
example
{ "EXAMPLE:"; echo = 2;
   t2();
}


////////////////////////////////////////////////
proc test1()
"USAGE:  test1();
RETURN: nothing
```

```
EXAMPLE:example test1; shows an example
"
{
  int m = 2;
  int n = 3;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(1..m)),lp;
  matrix a1[n][n] = 1,0,1,0,3,7,1,7,5;
  matrix a2[n][n] = 0,2,8,2,6,0,8,0,4;
  matrix c[n][n] = 1,2,3,2,9,0,3,0,7;
  vector b = [11,19];
  sdpmin(m,n,c,b,a1,a2);
}
example
{ "EXAMPLE:"; echo = 2;
   test1();
}


////////////////////////////////////////////////
proc example1()
"USAGE:  example1();
RETURN: nothing
EXAMPLE:example example1; shows an example
"
{
  int m = 6;
  int n = 4;
  ring r1 = 0,(x(1..(n*(n+1)/2)), y(m..1)),lp;
  poly ofn = -42*y(1)-65/2*y(2)-79/2*y(3)-21/2*y(4)-43/2*y(5)-17/2*y(6)+309/2;
  matrix Ay[n][n] = 1,y(1),y(2),y(3),
                    y(1),1,y(4),y(5),
                    y(2),y(4),1,y(6),
                    y(3),y(5),y(6),1;
  sdpmax(m,n,ofn,Ay);
}
example
{ "EXAMPLE:"; echo = 2;
   example1();
}


////////////////////////////////////////////////
proc fi(list #)
"USAGE:  fi(i1,i2,..,ir);
RETURN:
EXAMPLE:example fi; shows an example
"
{
  int n=size(#);
  if (n==1)
  {
    return (bigint(2)^(#[1]-1));
  }
  int i,j,k;
  bigint l;
  if (n==2)
  {
   l=0;
   j=#[2]-1;
   k=#[1]+#[2]-2;
   for (i=#[1]; i<=j; i++)
   {
    l=l+binomial(k,i);
   }
   return(l);
  }
  // if odd...
  if (n mod 2 != 0)
  {
   matrix Pf[n+1][n+1]; // initialized to 0s
```

```
   for (i=1; i<=n; i++)
   {
    Pf[1,i+1]=fi(#[i]);Pf[i+1,1]=-fi(#[i]);
   }
   for (i=2; i<=n; i++)
   {
    for (j=i; j<=n; j++)
    {
     Pf[i,j+1]=fi(#[i-1],#[j]);Pf[j+1,i]=-fi(#[i-1],#[j]);
    }
   }

  } else //..even
  {
   matrix Pf[n][n];
   for (i=1; i<=n; i++)
   {
    for (j=i; j<n; j++)
    {
     Pf[i,j+1]=fi(#[i],#[j+1]);Pf[j+1,i]=-fi(#[i],#[j+1]);
    }
   }
  }
  return(bigint(intRoot(bigint(det(Pf)))));
}
example
{ "EXAMPLE:"; echo = 2;
  fi(1,3,4,6);
  fi(2,4,5);
  fi(23);
  fi(3,7,8,11,23);
}

////////////////////////////////////////////////
proc delta(int m, int n, int r)
"USAGE:  delta(m,n,r);
RETURN: algebraic degree of SDP
EXAMPLE:example delta; shows an example
"
{
  int len = n-r;
  int i,j,k;
  bigint fs=1;
  list l;
  if (m<=binomial(len+1,2) || binomial(r+1,2)>=(binomial(n+1,2)-m)) { return("error in
parameters."); }
  if (len==1)
  {
   if (m<=n) { fs=fi(m); }
   for (i=1; i<=n; i++)
   {
    if (i!=m) { l=l+list(i); }
   }
   return(fs*fi(l));
  }
  bigint fsum=0;
//  if (len==2)
//  {
//  for (i=1; i<=n; i++)
//  {
//   for (j=i+1; j<=n; j++)
//   {
//    if (i+j==m)
//    {
//     l=list(i,j);
//     fs=fi(l);
//     print(""+string(l)+"*");
//     l=list();
//     for (k=1; k<=n; k++)
//     {
```

```
//      if (i!=k && j!=k) { l=l+list(k); }
//     }
//     print(string(l));
//     fs=fs*fi(l);
//     fsum=fsum+fs;
//     break;
//    }
//   }
//  }
//  } //endif
  string s;
  intvec iv;
  int c;
  bigint bsum;
  s=s+"for (iv[1]=1; iv[1]<=n; iv[1]=iv[1]+1) { ";
  for (i=2; i<=len; i++)
  {
   s=s+"for (iv["+string(i)+"]=iv["+string(i-1)+"]+1; iv["+string(i)+"]<=n;
iv["+string(i);
   s=s+"]=iv["+string(i)+"]+1) {";
  }
  s=s+"bsum=sum(iv); if(bsum>m) {break;} if (bsum==m) {
l=list(iv[1..len]);fs=fi(l);l=list();";
  s=s+"for (j=1; j<=n; j++) { c=1;for (k=1; k<=len; k++) { if (iv[k]==j) { c=0; break; }
} ";
  s=s+"if (c==1) { l=l+list(j); } } fs=fs*fi(l); fsum=fsum+fs;break; }";
  for (i=1; i<=len; i++)
  {
   s=s+"} ";
  }
  execute(s);
  return(fsum);
}
example
{ "EXAMPLE:"; echo = 2;
   delta(8,5,3);
}



////////////////////////////////////////////////
proc dr()
"USAGE:  dr();
RETURN: nothing
EXAMPLE:example dr; shows an example
"
{
  ring r1 = 0,x,lp;
  export(r1);
}
example
{ "EXAMPLE:"; echo = 2;
   dr();
}
```