What is SDP? (for a beginner of SDP)

Copyright (C) 2005 SDPA Project

1 Introduction

This note is a short course for SemiDefinite Programming’s (SDP) beginners. SDP has various applications in, for example, control theory, financial engineering, quantum chemistry, combinatorial optimization, etc., and also has a theoretical neatness.

We start from the mathematical definition of the SDP in Section 2. Then we describe its mathematical properties in Section 3. Section 4 is devoted to one algorithm to solve SDP, the Primal-Dual Interior-Point Method. Finally, we introduce a computer software, the SDPA, which is an implementation of Primal-Dual Interior-Point Methods in Section 5.

The purpose of this note is to give an overview of SDPs from its mathematical and computational aspects. For a more complete description refer to other survey papers. We assume that the reader is familiar with basic notions of linear algebra and Linear Programming (LP).

2 Mathematical Definition of the SDP

The SDP (SemiDefinite Programming) minimizes/maximizes a linear objective function over linear equality constraints with a positive semidefinite matrix condition. We define the SDP as follows:

\[
\begin{align*}
\text{SDP : } & \quad \text{minimize} & & \sum_{i=1}^{m} c_i x_i \\
& \quad \text{subject to} & & X = \sum_{i=1}^{m} F_i x_i - F_0, \\
& & & X \succeq O.
\end{align*}
\]

All matrices involved in the formulation are symmetric matrices. We use the notations \( S^n, S^n_+, \) and \( S^n_{++} \) to denote the set of \( n \times n \) symmetric matrices, the set of \( n \times n \) symmetric positive semidefinite matrices, and the set of \( n \times n \) symmetric positive definite matrices, respectively. We use the notation \( X \succeq O \) and \( X \succ O \) to indicate \( X \in S^n_+ \) and \( X \in S^n_{++}, \) respectively. Thus, all the eigenvalues of \( X \in S^n_+ \) are nonnegative, and positive for \( X \in S^n_{++}. \) In addition, the inner product in \( S^n \) is defined as \( U \cdot V = \sum_{i=1}^{n} \sum_{j=1}^{n} U_{ij} V_{ij} \) for \( U, V \in S^n. \)

In the above SDP, the input data are \( F_0 \in S^n \) (coefficient matrix), \( F_1, F_2, \ldots, F_m \in S^n \) (input data matrices), \( c_1, c_2, \ldots, c_m \in \mathbb{R} \) (coefficient values), and \( n \) (size of matrices). We want to find \( x \in \mathbb{R}^m \) and \( X \in S^n \) which minimizes the linear objective function satisfying the linear equality constraint and the positive semidefinite condition.

From the above SDP (the primal SDP), we can define the dual SDP via its Lagrangian dual problem.

\[
\begin{align*}
\text{dual SDP : } & \quad \text{maximize} & & F_0 \cdot Y \\
& \quad \text{subject to} & & F_i \cdot Y = c_i \quad (i = 1, 2, \ldots, m), \\
& & & Y \succeq O.
\end{align*}
\]

In the same way as LP, the dual SDP essentially has the same information as the primal SDP. We call the pair of the primal and the dual SDPs as SDP simply.

\[
\begin{align*}
\mathcal{P} : & \quad \text{minimize} & & \sum_{i=1}^{m} c_i x_i \\
& \quad \text{subject to} & & X = \sum_{i=1}^{m} F_i x_i - F_0, \\
& & & X \succeq O. \\
\mathcal{D} : & \quad \text{maximize} & & F_0 \cdot Y \\
& \quad \text{subject to} & & F_i \cdot Y = c_i \quad (i = 1, 2, \ldots, m), \\
& & & Y \succeq O.
\end{align*}
\]

If \( (x, X, Y) \in (\mathbb{R}^m, S^n, S^n) \) satisfies constraints of both primal and dual SDPs, we call it a feasible solution. In addition, if \( X \succ O \) and \( Y \succ O, \) we call it an interior feasible solution.
Next, we present a simple example of an SDP application. We want to detect whether we can find a vector which makes all eigenvalues of a linear combination of the given symmetric matrices negative. More precisely, we want to find \( x \in \mathbb{R}^m \) such that all eigenvalues of \( F_0 + \sum_{i=1}^m F_i x_i \) become negative (where \( F_i \in \mathbb{S}^n (i = 0, \ldots, m) \)). This problem is a kind of Linear Matrix Inequality (LMI) arising from a stability condition in control theory. To solve the problem, we introduce an auxiliary scalar \( \lambda \) and formulate it as the following SDP.

\[
\text{SDP from LMI : } \begin{cases} 
\text{minimize} & \lambda \\
\text{subject to} & X = \lambda I - (F_0 + \sum_{i=1}^m F_i x_i), \\
& X \succeq O,
\end{cases}
\]

where \( I \) is the identity matrix. If the optimal value of \( \lambda \) is negative, the corresponding optimal solution (vector) enables us to make all the eigenvalues of the linear combination of these matrices negative.

3 Mathematical Properties of SDPs

We start from properties of the set of positive semidefinite matrices, 
\[ \mathbb{S}_+^n = \{ X \in \mathbb{S}^n : X \succeq O \}, \]
which characterizes the SDPs.

The set \( \mathbb{S}_+^n \) satisfies the two conditions to become a cone.

\[
\begin{cases} 
\forall X \in \mathbb{S}_+^n, \forall \lambda \geq 0 & \Rightarrow \lambda X \in \mathbb{S}_+^n, \\
\forall X, \forall Y \in \mathbb{S}_+^n & \Rightarrow X + Y \in \mathbb{S}_+^n.
\end{cases}
\]

We can easily verify these facts by 
\[ X \in \mathbb{S}_+^n \iff v^T X v \geq 0 \text{ for } \forall v \in \mathbb{R}^n. \]

Furthermore, \( \mathbb{S}_+^n \) can be classified as a self-dual homogeneous cone, or symmetric cone, since
\[
\begin{array}{c}
\{ Y \in \mathbb{S}_+^n : X \cdot Y \geq 0 \text{ for } \forall X \in \mathbb{S}_+^n \} = \mathbb{S}_+^n \quad \text{(self-dual)}, \\
\forall X \in \mathbb{S}_+^n \exists \text{ bijective map } f : \mathbb{S}_+^n \to \mathbb{S}_+^n \text{ s.t. } f(X) = I \quad \text{(homogeneous)}.
\end{array}
\]

In the above sense, SDP is a member of the same group as LP and Second Order Cone Programming (SOCP). In fact, we have an LP when we restrict all involved matrices in the SDP into the space of diagonal matrices. Furthermore, SDPs also cover the class of SOCPs. Therefore, many theorems and methods for LPs have been extended for SDPs (with slight modifications).

The duality theorem is the most powerful and useful theorem in Mathematical Programming. The duality theorem is composed of two levels, the weak duality theorem and the strong duality theorem. When we say the duality theorem, we usually mean the strong duality theorem.

**Theorem 3.1 (The weak duality theorem)**: When \( (x, X, Y) \) is a feasible solution, the objective value of the primal problem is greater than or equal to that of the dual problem.

**Theorem 3.2 (The strong duality theorem)**: When there exists an interior feasible solution, both the primal and the dual problems have optimal solutions and their optimal objective values coincide.

To prove the strong duality theorem, we need a knowledge of the Farkas Lemma and the contents exceeds the level of this note. Here, we prove only the weak duality theorem. Since \( (x, X, Y) \) satisfies the primal and dual constraints, we have
\[
\sum_{i=1}^m c_i x_i - F_0 \cdot Y = \sum_{i=1}^m (F_i \cdot Y) x_i - \left( \sum_{i=1}^m F_i x_i - X \right) \cdot Y = X \cdot Y \geq 0.
\]

The last inequality holds because \( X, Y \in \mathbb{S}_+^n \).

A significant importance of SDP is that we can solve SDPs effectively by sophisticated algorithms, for example, Primal-Dual Interior-Point Methods (PD-IPM) introduced in the next section.
Therefore, many methods including the PD-IPM to solve the SDPs can be reduced to the problem of finding solutions which satisfy the KKT condition. In the KKT condition, the 1st, the 2nd and the 3rd conditions ensure the feasibility, while the last condition ensures optimality, which is equivalent to \( X \cdot Y = 0 \) due to the 3rd condition.

## 4 Primal-Dual Interior-Point Methods

Here we introduce one of many sophisticated methods for solving SDPs, the Primal-Dual Interior-Point Methods (PD-IPM).

In the PD-IPM, the central path plays an central role as the name indicates.

The Central Path \( \{ (x(\mu), X(\mu), Y(\mu)) : \mu > 0 \} \), where \( (x(\mu), X(\mu), Y(\mu)) \) satisfies

\[
\begin{align*}
X(\mu) &= \sum_{i=1}^{m} F_i x(\mu)_i - F_0, \\
F_i \cdot Y(\mu) &= c_i \quad (i = 1, \ldots, m), \\
x(\mu) &\succeq O, \quad Y(\mu) \succeq O, \\
X(\mu) \cdot Y(\mu) &= \mu I.
\end{align*}
\]

It is well-known that the central path is a continuous and smooth curve and that there is a unique point \((x(\mu), X(\mu), Y(\mu))\) on the central path when \(\mu > 0\) is fixed. Let \((x^*, X^*, Y^*)\) be an optimal solution which satisfies the KKT condition. Then we can prove a fact which is the base of the PD-IPM,

\[
\lim_{\mu \to +0} (x(\mu), X(\mu), Y(\mu)) = (x^*, X^*, Y^*).
\]

An intuitive reason is that the system which defines the central path converges to the KKT condition when \(\mu \to +0\). Furthermore, \(X(\mu) \cdot Y(\mu)/n = \mu\) holds on the central path where \(n\) is the dimension of \(X\) and \(Y\).

In the PD-IPM framework, we numerically trace the central path decreasing \(\mu\) toward the optimal solution.

**PD-IPM Framework**

**Step 0 (Initialization):** Choose an initial point \((x^0, X^0, Y^0)\) such that \(X^0 \succ O\) and \(Y^0 \succ O\). Let \(k = 0\). Choose parameters \(\beta \) \((0 < \beta < 1)\) and \(\gamma \) \((0 < \gamma < 1)\).

**Step 1 (Terminal Check):** If \((x^k, X^k, Y^k)\) approximately satisfies the KKT condition, we print it as an approximate optimal solution and terminate.

**Step 2 (Search Direction):** We compute the search direction \((dx, dX, dY)\) which is an approximate difference between \((x^k, X^k, Y^k)\) and the point on the central path \((x(\beta \mu^k), X(\beta \mu^k), Y(\beta \mu^k))\) where \(\mu^k = X^k \cdot Y^k/n\) and \(\beta\) plays a role to decrease \(\mu\) for the next point.

**Step 3 (Step Length):** We compute the largest \(\alpha\) such that \(X^k + \alpha dX \succeq O\), \(Y^k + \alpha dY \succeq O\).
Step 4 (Update): \((x^{k+1}, X^{k+1}, Y^{k+1}) \leftarrow (x^k, X^k, Y^k) + \gamma \alpha(dx, dX, dY)\) and \(k \leftarrow k + 1\). Then goto Step 1.

Throughout all iterations, we keep the condition \(X^k \succ O\) and \(Y^k \succ O\) by choosing properly \(\alpha\) in Step 3 and \(\gamma\) in Step 4. This is the origin of ‘Interior-Point Methods.’ The largest \(\alpha_p\) for \(X^k\) in Step 3 is basically determined by the following eigenvalue computation.

\[
\alpha_p = \min \{ -1/\lambda_{\min}(L^{-1}dX(L^T)^{-1}), 1 \} \quad \text{if} \quad \lambda_{\min}((L^{-1})dX(L^T)^{-1}) < 0,
\]

where \(L\) is the Cholesky factorization of \(X^k\) and \(\lambda_{\min}(X)\) is the minimum eigenvalue of \(X\), respectively. We compute \(\alpha_p\) for \(Y^k\) in a similar fashion and we choose \(\alpha = \min\{\alpha_p, \alpha_d\}\) in Step 3.

For the computation of the search direction \((dx, dX, dY)\) in Step 2, we employ the Newton method. Let \((dx, dX, dY)\) be a slight movement from the current point \((x^k, X^k, Y^k)\) toward the central path for \(\beta \mu^k\). That is, we want to solve the following system.

\[
\begin{cases}
X^k + dX = \sum_{i=1}^m F_i(x_i^k + dx_i) - F_0, \\
F_i \cdot (Y^k + dY) = c_i \quad (i = 1, \ldots, m), \\
(X^k + dX)(Y^k + dY) = \beta \mu^k I.
\end{cases}
\]

We omit the positive semidefinite condition since it is considered in Step 3. We can not easily solve the above system and get a solution \((dx, dX, dY)\) due to the nonlinear term \(dXdY\) in the last equality. Since this term is of second order, corresponding to a slight movement, we ignore it and we can reduce the above system to the next linear equation,

\[
B \ dx = r,
\]

where

\[
B_{ij} = (X^k)^{-1} F_i Y^k \bullet F_j,
\]

\[
r_i = F_i \bullet \left( (X^k)^{-1} (\beta \mu^k Y^k)^{-1} - \sum_{j=1}^m F_j x_j^k - F_0 Y^k \right) - (c_i - F_i \bullet Y^k).
\]

We call this linear equation as the Schur Complement Equation (SCE) and its coefficient matrix as the Schur Complement Matrix (SCM). Since the SCM is always positive definite throughout all the iteration of the PD-IPM, we can utilize its Cholesky factorization to solve the SCE. From a solution \(dx\) of the SCE, it is a relative easy task to compute \(dX\) and \(dY\). We should mention that in the above system, the number of equality constraints and the number of variables do not coincide. Therefore, many approaches have been proposed to overcome this difficulty. See the papers regarding AHO, HRVW/KSH/M, NT directions for more details.

From the viewpoint of parallel computation, the most intense computation cost is due to the evaluation of the SCM and its Cholesky factorization. To shorten the total computation time, it is natural to think in replacing these two time consuming subroutines with their parallel implementation.

5 Simple Usage of the SDPA

It is impossible to execute the complex algorithm of the PD-IPM by hands. We had better use a computer software to solve SDPs. Here, we introduce how to use the SDPA (SemiDefinite Programming Algorithm), one of the distinguished computer software. The SDPA is an implementation of the PD-IPM described previously. Comprehensive information of the SDPA can be obtained from the SDPA home-page.

http://grid.r.dendai.ac.jp/sdpa/

For simplicity, we use the command prompt version of the SDPA. The SDPA also includes a C++ callable library. In addition, the SDPA family have a MATLAB interface (SDPA-M) and a parallel version of the SDPA (SDPARA).
Now, we will solve a simple example.

\[ \begin{align*}
\mathcal{P}: \quad & \text{minimize} \quad 48x_1 - 8x_2 + 20x_3 \\
& \text{subject to} \quad X = \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} x_1 + \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} x_2 + \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} x_3 - \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix}, \\
& X \succeq O.
\end{align*} \]

\[ \begin{align*}
\mathcal{D}: \quad & \text{maximize} \quad \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix} \cdot Y \\
& \text{subject to} \quad \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} \cdot Y = 48, \quad \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} \cdot Y = -8, \\
& \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} \cdot Y = 20, \ Y \succeq O.
\end{align*} \]

Hence, we have the following input data.

\[
m = 3, \ n = 2, \ c = \begin{pmatrix} 48 \\ -8 \\ 20 \end{pmatrix}, \ F_0 = \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix}, \ F_1 = \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix}, \ F_2 = \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix}, \ F_3 = \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix}.
\]

We make an input data file ‘example1.dat-s’ from the above data. The file ‘example1.dat-s’ is included in the SDPA download package and ‘dat-s’ is the extension of the SDPA sparse format file.

** START of example1.dat-s **

"Example 1: mDim = 3, nBLOCK = 1, {2}"

3 = mDIM
1 = nBLOCK
2 = bLOCKsTRUCT
{48, -8, 20}
0 1 1 1 -11
0 1 2 2 23
1 1 1 1 10
1 1 1 2 4
2 1 2 2 -8
3 1 1 2 -8
3 1 2 2 -2

** END of example1.dat-s **

The 1st line enclosed by double quotations is a comment. The 2nd line is \( m \). The 3rd and 4th line express the dimension of the matrices \( X \) and \( Y \). The SDPA can handle a more general matrix structure called the block diagonal structure. In this simple example, however, we have only one block, therefore \( nBLOCK = 1 \) and \( bLOCKsTRUCT = n \). The 5th line corresponds to the elements of \( c \). The 6th to 12th lines correspond to the information of \( F_k \) (\( k = 0, \ldots, m \)). Each line is composed as follows.

\( k \ l \ i \ j \) value

\( k \) is the index of \( F_k \), \( l \) is the block number (in the simple example, \( l = 1 \)), \( i \) and \( j \) are indices of \( F_k \), and then \textit{value} is the value of \((i,j)\) elements of \( F_k \). Note that since \( F_k \) is symmetric, only the upper triangular part is stored.

By invoking the SDPA command as follow,

\$ ./sdpa example1.dat-s example1.result
we will receive the output.

SDPA start ... (start at September 01 2004 12:00:01)
data is example1.dat-s : sparse
parameter is ./param.sdpa
out is example1.result

<table>
<thead>
<tr>
<th>mu</th>
<th>thetaP</th>
<th>thetaD</th>
<th>objP</th>
<th>objD</th>
<th>alphaP</th>
<th>alphaD</th>
<th>beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0e+00</td>
<td>1.0e+00</td>
<td>-0.00e+00</td>
<td>+1.20e+03</td>
<td>1.0e+00</td>
<td>9.1e-01</td>
<td>2.00e-01</td>
</tr>
<tr>
<td>1</td>
<td>1.6e+03</td>
<td>5.8e-17</td>
<td>9.4e-02</td>
<td>+8.39e+02</td>
<td>+7.51e+01</td>
<td>2.3e+00</td>
<td>9.6e-01</td>
</tr>
<tr>
<td>2</td>
<td>1.7e+02</td>
<td>1.9e-16</td>
<td>3.6e-03</td>
<td>+1.96e+02</td>
<td>-3.74e+01</td>
<td>1.3e+00</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>3</td>
<td>1.8e+01</td>
<td>1.3e-16</td>
<td>7.5e-18</td>
<td>-6.82e+00</td>
<td>-4.19e+01</td>
<td>9.9e-01</td>
<td>9.9e-01</td>
</tr>
<tr>
<td>4</td>
<td>1.9e+00</td>
<td>1.4e-16</td>
<td>7.5e-18</td>
<td>-3.81e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>1.0e-00</td>
</tr>
<tr>
<td>5</td>
<td>1.9e+00</td>
<td>1.3e-16</td>
<td>7.5e-18</td>
<td>-4.15e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>1.0e-00</td>
</tr>
<tr>
<td>6</td>
<td>1.9e-02</td>
<td>1.3e-16</td>
<td>7.5e-18</td>
<td>-4.19e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>9.0e+01</td>
</tr>
<tr>
<td>7</td>
<td>1.9e-03</td>
<td>1.3e-16</td>
<td>1.5e-15</td>
<td>-4.19e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>1.0e-00</td>
</tr>
<tr>
<td>8</td>
<td>1.9e-04</td>
<td>1.4e-16</td>
<td>9.3e-19</td>
<td>-4.19e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>1.0e-00</td>
</tr>
<tr>
<td>9</td>
<td>1.9e-05</td>
<td>1.3e-16</td>
<td>3.0e-17</td>
<td>-4.19e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>9.0e+01</td>
</tr>
<tr>
<td>10</td>
<td>1.9e-06</td>
<td>1.2e-16</td>
<td>3.3e-15</td>
<td>-4.19e+01</td>
<td>-4.19e+01</td>
<td>1.0e-00</td>
<td>9.0e+01</td>
</tr>
</tbody>
</table>

phase.value = pdOPT
  Iteration = 10
  mu = 1.9180668442024251e-06
relative gap = 9.1554440289307566e-08
  gap = 3.8361336884048502e-06
digits = 7.0383205880484550e+00
objValPrimal = -4.189999616386419e+01
objValDual = -4.1899999999997291e+01
p.feas.error = 1.5099033134902129e-14
d.feas.error = 3.1334934647020418e-12
total time = 0.000
  main loop time = 0.000000
  total time = 0.000000
file read time = 0.000000

The result phase.value = pdOPT means that the SDPA found an approximate optimal solution for a predefined numerical error threshold. objValPrimal and objValDual are the optimal objective values of the primal and the dual SDPs.

In this simple example, we can verify the objective value. The reason is that only one point can be feasible for the dual SDP,

\[ Y = \begin{pmatrix} 59/10 & -11/8 \\ -11/8 & 1 \end{pmatrix}. \]

Therefore, the objective value becomes ‘−41.9’.