Usage of Post-Processing Software for Semidefinite Programming with Projection and Rescaling Method

Shin-ichi Kanoh *

1 Introduction

The main purpose of this document is to explain how to use the programs that implement the algorithms proposed in [1]. Our programs are concerned with the following semidefinite programming:

$$\begin{array}{llllll} \text{(P)} & \inf\limits_{x} & \langle c, x\rangle & \text{s.t.} & \mathcal{A}x = b, & x \in \mathcal{K}, \\ \text{(D)} & \sup\limits_{(z,y)} & b^\top y & \text{s.t.} & z = c - \mathcal{A}^* y, & (z, y) \in \mathcal{K} \times \mathbb{R}^m, \end{array}$$

where $\mathbb{E}$ is a real-valued vector space with an inner product $\langle \cdot, \cdot \rangle$, $\mathcal{K} \subseteq \mathbb{E}$ is a direct product of nonnegative orthant $\mathbb{R}^n_+$ and positive semidefinite cones $\mathbb{S}^r_+$, $\mathcal{A} : \mathbb{E} \to \mathbb{R}^m$ is a linear operator, $b \in \mathbb{R}^m$, $c \in \mathbb{E}$ and $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{E}$ is the adjoint operator of $\mathcal{A}$. In the study of [1], an algorithm for finding the approximate optimal solution of (P) and an algorithm for finding the approximate optimal solution of (D) are proposed and are called Algorithm 4 and Algorithm 5, respectively. See [1] for details on these algorithms. This document explains how to use our programs that represent these algorithms. Our programs are written in MATLAB and use the functions included in the SeDuMi package [2]. Therefore, this document assumes that SeDuMi is properly installed, and that you are working in MATLAB.

2 How to use our programs

This section describes how to use our programs. Algorithm 4 and Algorithm 5 are implemented in "PostProcessingAlg.m" and "Alt_PostProcessingAlg.m", respectively. Even though Algorithm 4 is an algorithm for finding an approximate optimal solution to (P), it can find an approximate optimal dual solution. Similarly, Algorithm 5 can find an approximate optimal primal solution. In other words, if we are interested in (P), we might obtain a more accurate optimal solution to (P) by using Algorithm 5 in conjunction with Algorithm 4 rather than using Algorithm 4 alone. Therefore, if there is sufficient time to solve the problem, it is recommended that both algorithms be used to solve the problem.

2.1 Input

The input format of our programs is the SeDuMi format. Thus, our programs require the objects **[A, b, c, K]** representing the problem as input. Note that, unlike SeDuMi, our programs can not handle quadratic cones. In addition to the problem data, our programs require two inputs, **sol** and **para**.

*Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan. email: s2130104@s.tsukuba.ac.jp

A structure **sol** is defined with fields **sol.x**, **sol.y** and **sol.z**. This structure will be used to give our programs an approximate optimal solution (x^0, y^0, z^0) to (P) and (D). A structure **para** is used to change the values of parameters used in our programs. This structure can have the following fields.

- Options regarding termination conditions
  1. **para.Tol**(1e-12): Desired accuracy. Algorithms 4 and 5 terminate when $UB-LB \leq$**para.Tol** is satisfied.
  2. **para.Maxtime**(inf): The maximum execution time. The unit of input value is seconds.
  3. **para.PraTerCon**(30): If the projection and rescaling method called in Algorithms 4 and 5 returns **para.PraTerCon** consecutive incorrect outputs or proves **para.PraTerCon** consecutive times that there is no ε-feasible solution to the input problem, Algorithms 4 and 5 terminate. See [1] and [1] for the definition of incorrect output and ε-feasible solution, respectively.

In addition to the above parameters, **para** has the following fields for passing information about approximate solutions other than **sol** to Algorithms 4 and 5.

- Options to use approximate optimal solutions other than the input solution
  1. **para.LowerBound**: This structure is defined with fields **para.LowerBound.sol** and **para.LowerBound.val**. If an approximate optimal dual solution y such that $c - \mathcal{A}^* y \in \mathcal{K}$ is known before running Algorithms 4 or 5, this option will be used.
     - **para.LowerBound.sol**: A vector y such that $c - \mathcal{A}^* y \in \mathcal{K}$.
     - **para.LowerBound.val**: The value of $b^\top y$.
  2. **para.StockP**: This structure is defined with fields **para.StockP.Sol** and **para.StockP.err2**. If (approximate) primal solutions $x^1, \ldots, x^k$ are known before running Algorithms 4 or 5, this option will be used.
     - **para.StockP.Sol**: A matrix that stores the (approximate) primal solutions as
     $$\textbf{para.StockP.Sol (:, i)} = x^i.$$
     - **para.StockP.err2**: A vector that stores the value of $\max\left\{0, \frac{-\lambda_{\min}(x)}{1+\max\limits_{j=1,\ldots,m}|b_j|}\right\}$ for the corresponding primal solution, as
     $$\textbf{para.StockP.err2 (1, i)} = \max\left\{0, \frac{-\lambda_{\min}(x^i)}{1+\max\limits_{j=1,\ldots,m}|b_j|}\right\}.$$
  3. **para.StockD**: This structure is defined with fields **para.StockD.Sol** and **para.StockD.err4**. If (approximate) dual solutions $y^1, \ldots, y^k$ are known before running Algorithms 4 or 5, this option will be used.
     - **para.StockD.Sol**: A matrix that stores the (approximate) dual solutions as
     $$\textbf{para.StockD.Sol (:, i)} = y^i.$$
     - **para.StockD.err4**: A vector that stores the value of $\max\left\{0, \frac{-\lambda_{\min}(c-\mathcal{A}^* y)}{1+\max\limits_{j=1,\ldots,d}|c_j|}\right\}$ for the corresponding dual solution, as
     $$\textbf{para.StockD.err4 (1, i)} = \max\left\{0, \frac{-\lambda_{\min}(c-\mathcal{A}^* y^i)}{1+\max\limits_{j=1,\ldots,d}|c_j|}\right\}.$$

The above options will be used when both Algorithms 4 and 5 are run consecutively. These two algorithms might obtain several solutions of (P) and (D) before terminating and are designed to preserve them. By selecting from among the stored solutions the one that is most likely to be accurate as the optimal solution, these algorithms make the accuracy of their output robust. Now consider the case where we first use Algorithm 5 and then Algorithm 4 to find the solution to (P) and (D). Suppose that Algorithm 5 obtains some solutions to (P). Then, by giving them to Algorithm 4 using the above options, Algorithm 4 can extract a primal solution from among several primal solutions obtained by Algorithm 5, in addition to the primal solutions found by itself.

Furthermore, these two algorithms work more efficiently when dual feasible solutions are known before execution. Since Algorithm 5 can theoretically find an approximate optimal solution to (D), [1] recommends using Algorithm 5 before Algorithm 4 as long as $\lambda_{\min}(z^0) \geq$-1e-12 holds, where $\lambda_{\min}(z^0)$ is the smallest eigenvalue of z^0.

2.2 Post-processing using Algorithm 4 and Algorithm 5

To perform post-processing using Algorithm 4, call PostProcessingAlg.m as follows.
>>> **[bound, TmpSol, info] = PostProcessingAlg(A,b,c,K,sol,para);**
Similarly, to perform post-processing using Algorithm 5, call Alt_PostProcessingAlg.m as follows.
>>> **[bound, TmpSol, info] = Alt_PostProcessingAlg(A,b,c,K,sol,para);**

The outputs of these algorithms are described in the next section. As in [1], Algorithms 4 and 5 will work more efficiently when x_0 and z_0 are highly accurate approximate optimal interior feasible solutions, respectively. Therefore, when calling Algorithm 4 and Algorithm 5, the minimum eigenvalue of x_0 and z_0 should be positive, respectively. If the input solution is not an interior point of the cone $\mathcal{K}$, our programs generate an approximate interior feasible solution by adding a small perturbation to the input solution before starting the operation. However, as in the numerical results of [1], it is recommended to use our programs after preparing an approximate optimal interior feasible solution x_0 or z_0 with a minimum eigenvalue greater than or equal to -1e-12 unless the problem size is small.

2.3 Output

Our two programs "PostProcessingAlg.m" and "Alt_PostProcessingAlg.m" return three output variables, **bound**, **TmpSol**, and **info**.

- **bound**: The structure **bound** has the same fields as **para.LowerBound**.
  1. **bound.sol**: A vector y that holds as the current dual feasible solution in our program. If our program does not find a dual feasible solution, **bound.sol** is empty.
  2. **bound.val**: The value of $b^\top y$.
- **TmpSol**: This structure has the same fields as **sol**. **TmpSol** consists of the solution to (P) and (D) that seems to be the most accurate among the solutions obtained by our algorithm.
  1. **TmpSol.x**: A primal solution x
  2. **TmpSol.y**: A dual solution y. If **bound.sol** is not empty, **TmpSol.y** = **bound.sol** holds.
  3. **TmpSol.z**: A vector $c - \mathcal{A}^* y$.
- **info**: This structure summarizes the results of our program. It has the following fields:

1. **info.Psol**: A matrix that stores the vectors returned by the projection and rescaling algorithm as the solution to (P) in our program.
2. **info.Dsol**: A matrix that stores the vectors returned by the projection and rescaling algorithm as the solution to (D) in our program.
3. **info.UB**: The value of UB at the end of our program.
4. **info.LB**: The value of LB at the end of our program.
5. **info.LB_eps**: The input value θ when the projection and rescaling algorithm called in Post-ProcessingAlg.m proves for the first time that there is no ε-feasible solution to the input problem.
6. **info.UB_eps**: The input value θ when the projection and rescaling algorithm called in Alt_PostProcessingAlg.m proves for the first time that there is no ε-feasible solution to the input problem.
7. **info.msg**: A summary of which termination conditions our program satisfied.
   - If **info.msg** = **"Complete"**, then our programs terminated because $UB-LB \leq$ **para.Tol** held.
   - If **info.msg** = **"Time Over"**, then our programs terminated because its execution time exceeded **para.Maxtime**.
   - If **info.msg** = "Terminate due to numerical error", then our programs terminated because the projection and rescaling methods returned **para.PraTerCon** consecutive incorrect outputs.
   - If **info.msg** = **"Dual improving ray"**, then the program PostProcessingAlg.m terminated because an improving ray of (D), i.e., a vector $f \in \mathbb{R}^m$ such that $b^\top f > 0$ and $-\mathcal{A}^* f \in \mathcal{K}$, was obtained. The obtained improving ray is stored in **info.ir**.
   - If **info.msg** = **"Primal improving ray"**, then the program Alt_PostProcessingAlg.m terminated because an improving ray of (P), i.e., a vecotr $s \in \mathcal{K}$ such that $\mathcal{A}s = 0$ and $\langle c, s\rangle < 0$, was obtained. The obtained improving ray is stored in **info.ir**.
   - If **info.msg** = **"Primal reducing direction"**, then the program PostProcessingAlg.m terminated because a reducing direction for (P), i.e., a vector $f \in \mathbb{R}^m$ such that $b^\top f = 0$ and $-\mathcal{A}^* f \in \mathcal{K} \setminus \{0\}$, was obtained. The obtained reducing direction is stored in **info.rd**.
   - If **info.msg** = **"Dual reducing direction"**, then the program Alt_PostProcessingAlg.m terminated because a reducing direction for (D), i.e., a nonzero vecotr $s \in \mathcal{K}$ such that $\mathcal{A}s = 0$ and $\langle c, s\rangle = 0$, was obtained. The obtained reducing direction is stored in **info.rd**.
   - If **info.msg** = **"Primal problem might have no interior feasible solution "**, then the program PostProcessingAlg.m terminated because the projection and rescaling methods returned **para.PraTerCon** consecutive certificates that there was no ε-feasible solution to the input problems.
   - If **info.msg** = **"Dual problem might have no interior feasible solution "**, then the program Alt_PostProcessingAlg.m terminated because the projection and rescaling methods returned **para.PraTerCon** consecutive certificates that there was no ε-feasible solution to the input problems.

3 Concluding remarks

If you want to use Algorithm 4 and Algorithm 5 consecutively, use Programs "ProposedAlgorithm.m" or "ProposedAlgorithm_type2.m". To execute Algorithm 5 and Algorithm 4 in that order, call Program "ProposedAlgorithm.m". To execute Algorithm 4 and Algorithm 5 in that order, call Program "ProposedAlgorithm_type2.m". See File "Test.m" for specific usage of "ProposedAlgorithm.m" and "ProposedAlgorithm_type2.m".

References

[1] Shinichi Kanoh and Akiko Yoshise. Post-processing with projection and rescaling algorithms for semidefinite programming, 2024.

[2] Jos F. Sturm. Using sedumi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.