DSDP5 User Guide — Software for Semidefinite Programming

Steven J. Benson
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL U.S.A.
http://www.mcs.anl.gov/~benson

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA U.S.A
http://www.stanford.edu/~yyye

Technical Report ANL/MCS-TM-277

September 21, 2005

Abstract

DSDP implements the dual-scaling algorithm for semidefinite programming. The source code if this interior-point solver, written entirely in ANSI C, is freely available. The solver can be used as a subroutine library, as a function within the MATLAB environment, or as an executable that reads and writes to files. Initiated in 1997, DSDP has developed into an efficient and robust general purpose solver for semidefinite programming. Although the solver is written with semidefinite programming in mind, it can also be used for linear programming and other constraint cones.

The features of DSDP include:

- a robust algorithm with a convergence proof and polynomially bounded complexity under mild assumptions on the data,
- primal and dual solutions,
- feasible solutions when they exist or approximate certificates of infeasibity,
- initial points that can be feasible or infeasible,
- relatively low memory requirements for an interior-point method,
- sparse and low-rank data structures,
- extensibility that allows applications to customize the solver and improve its performance,
- a subroutine library that enables it to be linked to larger applications,
- scalable performance for large problems on parallel architectures, and
- a well documented interface and examples of its use.

The package has been used in many applications and tested for efficiency, robustness, and ease of use. We welcome and encourage further use under the terms of the license included in the distribution.

1 Notation

The DSDP package implements a dual-scaling algorithm to find solutions (X_j, y_i, S_j) to linear and semidefinite optimization problems of the form

$$(P) \qquad \inf \quad \sum_{j=1}^{p} \langle C_j, X_j \rangle \quad \text{subject to} \quad \sum_{j=1}^{p} \langle A_{i,j}, X_j \rangle = b_i, \quad i = 1, \ldots, m, \qquad X_j \in K_j,$$

$$(D) \qquad \sup \quad \sum_{i=1}^{m} b_i \; y_i \quad \text{subject to} \quad \sum_{i=1}^{m} A_{i,j} y_i + S_j = C_j, \quad j = 1, \ldots, p, \quad S_j \in K_j.$$

In this formulation, b_i and y_i are real scalars.

For semidefinite programming, the data $A_{i,j}$ and C_j are symmetric matrices of dimension n_j ($\mathbb{S}^{n_j}$), and the cone K_j is the set of symmetric positive semidefinite matrices of the same dimension. The inner product $\langle C, X \rangle := C \bullet X := \sum_{k,l} C_{k,l} X_{k,l}$, and the symbol $\succ$ ($\succeq$) means the matrix is positive (semi)definite. In linear programming, A_i and C are vectors of real scalars, K is the nonnegative orthant, and the inner product $\langle C, X \rangle$ is the usual vector inner product.

More generally, users specify $C_j, A_{i,j}$ from an inner-product space V_j that intersects a cone K_j. Using the notation summarized in Table 1, let the symbol $\mathcal{A}$ denote the linear map $\mathcal{A} : V \to \mathbb{R}^m$ defined by $(\mathcal{A}X)_i = \langle A_i, X \rangle$; its adjoint $\mathcal{A}^* : \mathbb{R}^m \to V$ is defined by $\mathcal{A}^* y = \sum_{i=1}^{m} y_i A_i$. Equivalent expressions for (P) and (D) can be written

$$\begin{array}{lllll} (P) & \inf \langle C, X \rangle & \text{subject to} & \mathcal{A}X = b, & X \in K, \\ (D) & \sup \; b^T y & \text{subject to} & \mathcal{A}^* y + S = C, & S \in K. \end{array}$$

Formulation (P) will be referred to as the *primal* problem, and formulation (D) will be referred to as the *dual* problem. Variables that satisfy the linear equations are called feasible, whereas the others are called infeasible. The interior of the cone will be denoted by $\hat{K}$, and the interior feasible sets of (P) and (D) will be denoted by $\mathcal{F}^0(P)$ and $\mathcal{F}^0(D)$, respectively.

Table 1: Basic terms and notation for linear (LP), semidefinite (SDP), and conic programming.

Term	LP	SDP	Conic	Notation
Dimension	n	n	$\sum n_j$	n
Data Space ($\ni C, A_i$)	$\mathbb{R}^n$	$\mathbb{S}^n$	$V_1 \oplus \ldots \oplus V_p$	V
Cone	$x, s \geq 0$	$X, S \succeq 0$	$X, S \in K_1 \oplus \ldots \oplus K_p$	$X, S \in K$
Interior of Cone	$x, s > 0$	$X, S \succ 0$	$X, S \in \hat{K}_1 \oplus \ldots \oplus \hat{K}_p$	$X, S \in \hat{K}$
Inner Product	$c^T x$	$C \bullet X$	$\sum \langle C_j, X_j \rangle$	$\langle C, X \rangle$
Norm	$\|x\|_2$	$\|X\|_F$	$\left(\sum \|X_j\|^2\right)^{1/2}$	$\|X\|$
Product	$[x_1 s_1 \ldots x_n s_n]^T$	XS	$X_1 S_1 \oplus \ldots \oplus X_p S_p$	XS
Identity Element	$[1 \ldots 1]^T$	I	$I_1 \oplus \ldots \oplus I_p$	I
Inverse	$[1/s_1 \ldots 1/s_n]^T$	S^{-1}	$S_1^{-1} \oplus \ldots \oplus S_p^{-1}$	S^{-1}
Dual Barrier	$\sum \ln s_j$	$\ln \det S$	$\sum \ln \det S_j$	$\ln \det S$

2 Dual-Scaling Algorithm

This section summarizes the dual-scaling algorithm for solving (P) and (D). For simplicity, parts of this discussion assume that the cone is a single semidefinite block, but an extension of the algorithm to multiple blocks and other cones is relatively simple. This discussion also assumes that the A_is are linearly independent, there exists $X \in \mathcal{F}^0(P)$, and a starting point $(y, S) \in \mathcal{F}^0(D)$ is known. The next section discusses how DSDP generalizes the algorithm to relax these assumptions.

It is well known that under these assumptions, both (P) and (D) have optimal solutions X^* and (y^*, S^*), which are characterized by the equivalent conditions that the duality gap $\langle X^*, S^* \rangle$ is zero and the product X^*S^* is zero. Moreover, for every $\nu > 0$, there exists a unique primal-dual feasible solution (X_ν, y_ν, S_ν) satisfies the perturbed optimality equation $X_\nu S_\nu = \nu I$. The set of all solutions $\mathcal{C} \equiv \{(X_\nu, y_\nu, S_\nu) : \nu > 0\}$ is known as the central path, and $\mathcal{C}$ serves as the basis for path-following algorithms that solve (P) and (D). These algorithms construct a sequence $\{(X, y, S)\} \subset \mathcal{F}^0(P) \times \mathcal{F}^0(D)$ in a neighborhood of the central path such that the duality gap $\langle X, S \rangle$ goes to zero. A scaled measure of the duality gap that proves useful in the presentation and analysis of path-following algorithms is $\mu(X, S) = \langle X, S \rangle / n$ for all $(X, S) \in K \times K$. Note that for all $(X, S) \in \hat{K} \times \hat{K}$, we have $\mu(X, S) > 0$ unless $XS = 0$. Moreover, $\mu(X_\nu, S_\nu) = \nu$ for all points (X_ν, y_ν, S_ν) on the central path.

The dual-scaling algorithm applies Newton's method to $\mathcal{A}X = b$, $\mathcal{A}^* y + S = C$, and $X = \nu S^{-1}$ to generate

$$\begin{aligned} \mathcal{A}(X + \Delta X) &= b, & (1)\\ \mathcal{A}^*(\Delta y) + \Delta S &= 0, & (2)\\ \nu S^{-1} \Delta S S^{-1} + \Delta X &= \nu S^{-1} - X. & (3) \end{aligned}$$

Equations (1)-(3) will be referred to as the Newton equations; their Schur complement is

$$\nu \begin{pmatrix} \langle A_1, S^{-1} A_1 S^{-1} \rangle & \cdots & \langle A_1, S^{-1} A_m S^{-1} \rangle \\ \vdots & \ddots & \vdots \\ \langle A_m, S^{-1} A_1 S^{-1} \rangle & \cdots & \langle A_m, S^{-1} A_m S^{-1} \rangle \end{pmatrix} \Delta y = b - \nu \mathcal{A} S^{-1}. \quad (4)$$

The left-hand side of this linear system is positive definite when $S \in \hat{K}$. In this manuscript, it will sometimes be referred to as M. DSDP computes $\Delta' y := M^{-1} b$ and $\Delta'' y := M^{-1} \mathcal{A} S^{-1}$. For any ν,

$$\Delta_\nu y := \frac{1}{\nu} \Delta' y - \Delta'' y$$

solves (4). We use the subscript to emphasize that ν can be chosen after computing $\Delta' y$ and $\Delta'' y$ and that the value chosen for the primal step may be different from the value chosen for the dual step.

Using $\Delta_\nu y$ and (3), we get

$$X(\nu) := \nu \left(S^{-1} + S^{-1} (\mathcal{A}^* \Delta_\nu y) S^{-1} \right), \quad (5)$$

which satisfies $\mathcal{A}X(\nu) = b$. Because $X(\nu) \in \hat{K}$ if and only if

$$C - \mathcal{A}^*(y - \Delta_\nu y) \in \hat{K}, \quad (6)$$

DSDP applies a Cholesky factorization on (6) to test the condition. If $X(\nu) \in \hat{K}$, a new upper bound

$$\bar{z} := \langle C, X(\nu) \rangle = b^T y + \langle X(\nu), S \rangle = b^T y + \nu \left(\Delta_\nu y^T \mathcal{A} S^{-1} + n \right) \quad (7)$$

can be obtained without explicitly computing $X(\nu)$. The dual-scaling algorithm does not require $X(\nu)$ to compute the step direction defined by (4), so DSDP does not compute it unless specifically requested. This feature characterizes the algorithm and its performance.

Either (y, S) or X reduces the the dual potential function

$$\psi(y) := \rho \log(\bar{z} - b^T y) - \ln \det S \tag{8}$$

enough at each iteration to achieve linear convergence.

Setup data structures and factor A_i.
Choose y such that $S \leftarrow C - \mathcal{A}^* y \in \hat{K}$.
Choose an upper bound $\bar{z}$ and a barrier parameter ν.
for $k \leftarrow 0, \ldots, k_{max}$ **do**
  Monitor solution and check for convergence.
  Compute M and $\mathcal{A}S^{-1}$.
  Solve $M\Delta' y = b$, $M\Delta'' y = \mathcal{A}S^{-1}$.
  if $C - \mathcal{A}^*(y - \Delta_\nu y) \in \hat{K}$ **then**
    $\bar{z} \leftarrow b^T y + \nu \left(\Delta_\nu y^T \mathcal{A}S^{-1} + n\right)$.
    $\overline{y} \leftarrow y$, $\overline{\Delta y} \leftarrow \Delta_\nu y$, $\overline{\mu} \leftarrow \nu$.
  end if
  Select ν.
  Find α_d to reduce ψ, and set $y \leftarrow y + \alpha_d \Delta_\nu y$, $S \leftarrow C - \mathcal{A}^* y$.
  for $kk = 1, \ldots, kk_{max}$ **do**
    Compute $\mathcal{A}S^{-1}$.
    Solve $M\Delta^c y = \mathcal{A}S^{-1}$.
    Select ν.
    Find α_c to reduce ϕ_ν, and set $y \leftarrow y + \alpha_c \Delta^c_\nu y$, $S \leftarrow C - \mathcal{A}^* y$.
  end for
end for
Optional: Compute X using $\overline{y}$, $\overline{\Delta y}$, $\overline{\mu}$.

3 Feasible Points, Infeasible Points, and Standard Form

The convergence of the algorithm assumes that both (P) and (D) have an interior feasible region and the current solutions are elements of the interior. To satisfy these assumptions, DSDP bounds the variables y such that $l \leq y \leq u$ where $l, u \in \mathbb{R}^m$. By default, $l_i = -10^7$ and $u_i = 10^7$ for each i from 1 through m. Furthermore, DSDP bounds the trace of X by a penalty parameter Γ whose default value is $\Gamma = 10^{10}$. Including these bounds and their associated Lagrange variables $x^l \in \mathbb{R}^m$, $x^u \in \mathbb{R}^m$, and r, DSDP solves following pair of problems:

$$
\begin{array}{lllrcrcl}
(PP) & \text{minimize} & \langle C, X\rangle & + & u^T x^u & - & l^T x^l & \\
 & \text{subject to} & \mathcal{A}X & + & x^u & - & x^l & = \; b, \\
 & & \langle I, X\rangle & & & & & \leq \; \Gamma, \\
 & & X \in K, & & x^u \geq 0, & & x^l \geq 0. &
\end{array}
$$

$$
\begin{array}{lllcl}
(DD) & \text{maximize} & b^T y - \Gamma r & & \\
 & \text{subject to} & C - \mathcal{A}^* y + Ir & = & S \in K, \\
 & & l \leq y \leq u, & & r \geq 0.
\end{array}
$$

The reformulations (PP) and (DD) are bounded and feasible, so the optimal objective values to this pair of problems are equal. Furthermore, (PP) and (DD) can be expressed in the form of (P) and (D).

Unless the user provides a feasible point y, DSDP uses the y values provided by the application (usually all zeros) and increases r until $C - \mathcal{A}^* y + Ir \in \hat{K}$. Large values of r improve robustness, but smaller values often improve performance. In addition to bounding X, the parameter Γ penalizes infeasiblity in (D) and forces r toward zero. The nonnegative variable r increases the dimension m by one and adds an inequality to the original problem. The M matrix treats r separately by storing the corresponding row/column as a separate vector and applying the Sherman-Morrison-Woodbury formula. Unlike other inequalities, DSDP allows r to reach the boundary of the cone. Once $r = 0$, it is fixed and effectively removed from the problem.

The bounds on y add $2m$ inequality constraints to the original problem; and, with a single exception, DSDP treats them the same as the constraints on the original model. The lone difference between these bounds and the other constraints is that DSDP explicitly computes the corresponding Lagrangian variables x^l and x^u at each iteration to quantify the infeasibility in (P). The bounds l and u penalize infeasiblity in (P), force x^l and x^u toward zero, and prevent numerical difficulties created by variables with large magnitude.

The solution to (PP) and (DD) is a solution to (P) and (D) when the optimal objective values of (P) and (D) exist and are equal, and the bounds are sufficiently large. DSDP identifies unboundedness or infeasibility in (P) and (D) through examination of the solutions to (PP) and (DD). Given parameters ϵ_P and ϵ_D,

- if $r \leq \epsilon_r$, $\|\mathcal{A}X - b\|_\infty / \langle I, X\rangle > \epsilon_P$, and $b^T y > 0$, it characterizes (D) as unbounded and (P) as infeasible;
- if $r > \epsilon_r$ and $\|\mathcal{A}X - b\|_\infty / \langle I, X\rangle \leq \epsilon_P$, it characterizes (D) as infeasible and (P) as unbounded.

Normalizing unbounded solutions will provide an approximate certificate of infeasibility. Larger bounds may improve the quality of the certificate of infeasibility and permit additional feasible solutions, but they may also create numerical difficulties in the solver.

4 Iteration Monitor

The progress of the DSDP solver can be monitored by using standard output printed to the screen. The data below shows an example of this output.

```
Iter  PP Objective     DD Objective    PInfeas  DInfeas    Nu     StepLength   Pnrm
-------------------------------------------------------------------------------
0    1.00000000e+02  -1.13743137e+05   2.2e+00  3.8e+02  1.1e+05  0.00  0.00   0.00
1    1.36503342e+06  -6.65779055e+04   5.1e+00  2.2e+02  1.1e+04  1.00  0.33   4.06
2    1.36631922e+05  -6.21604409e+03   5.4e+00  1.9e+01  4.5e+02  1.00  1.00   7.85
3    5.45799174e+03  -3.18292092e+03   1.5e-03  9.1e+00  7.5e+01  1.00  1.00  17.63
4    1.02930559e+03  -5.39166166e+02   1.1e-05  5.3e-01  2.7e+01  1.00  1.00   7.58
5    4.30074471e+02  -3.02460061e+01   3.3e-09  0.0e+00  5.6e+00  1.00  1.00  11.36
...
11   8.99999824e+00   8.99999617e+00   1.1e-16  0.0e+00  1.7e-08  1.00  1.00   7.03
12   8.99999668e+00   8.99999629e+00   2.9e-19  0.0e+00  3.4e-09  1.00  1.00  14.19
```

The program will print a variety of statistics for each problem to the screen.

`Iter`	the iteration number.
`PP Objective`	the upper bound $\bar{z}$ and objective value in (PP).
`DD Objective`	the objective value in (DD).
`PInfeas`	the primal infeasiblity in (P) is $\|x^u - x^l\|_\infty$.
`DInfeas`	the dual infeasibility in (D) is the variable r.
`Nu`	the barrier parameter ν.
`StepLength`	the multiple of the step-directions in (P) and (D).
`Pnrm`	the proximity to the central path: $\|\nabla\psi\|_{M^{-1}}$.

5 Reading SDPA files

DSDP can be used if the user has a problem written in sparse SDPA format. These executables have been put in the directory **DSDPROOT/exec/**. The file name should follow the executable. For example,

```
> dsdp5 truss4.dat-s
```

Other options can also be used with DSDP. These should follow the SDPA filename.

`-gaptol <rtol>`	to stop the problem when the relative duality gap is less than this number.
`-mu0 <mu0>`	to specify the initial barrier parameter ν.
`-r0 <r0>`	to specify the initial value of r in (DD).
`-boundy <1e7>`	to bound the magnitude of each variable y in (DD).
`-save <filename>`	to save the solution into a file with a format similar to SDPA.
`-y0 <filename>`	to specify an initial vector y in (D).
`-maxit <iter>`	to stop the problem after a specified number of iterations.
`-rho <3>`	to set the potential parameter ρ to this multiple of the conic dimension n.
`-dobjmin <dd>`	to add a constraint that sets a lower bound on the objective value at the solution.
`-penalty <1e8>`	to set the penalty parameter Γ for infeasibility in (D).
`-print <1>`	print standard output at each k iteration.
`-bigM <0>`	treat the inequality $r \geq 0$ in (DD) as other inequalities and keep it positive.
`-dloginfo <0>`	to print more detailed output. Higher number produce more output.
`-dlogsummary <1>`	to print detailed timing information about each dominant computations.

6 Applying DSDP to Graph Problems

Within the directory **DSDPROOT/examples/** is a program `maxcut.c` which reads a file containing a graph, generates the semidefinite relaxation of a maximum cut problem, and solves the relaxation. For example,

```
> maxcut graph1
```

The first line of the graph should contain two integers. The first integer states the number of nodes in the graph, and the second integer states the number of edges. Subsequent lines have two or three entries separated by a space. The first two entries specify the two nodes that an edge connects. The optional third entry specifies the weight of the node. If no weight is specified, a weight of 1 will be assigned.

The same options that apply to reading SDPA files also apply here.

A similar program reads a graph from a file, formulates a minimum bisection problem or Lovász Θ problem, and solves it. For example,

```
> theta graph1
```

reads the graph in the file `graph1` and solves this graph problem.

Acknowledgments

We thank Xiong Zhang and Cris Choi for their help in developing this code. Xiong Zhang, in particular, was fundamental to the initial version of DSDP. We also thank Hans Mittelmann for his efforts in testing and benchmarking the different versions of the code. Finally, we thank all of the users who have commented on previous releases and suggested improvements to the software. Their contributions have made DSDP a more reliable, robust, and efficient package.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38.