Working Paper

HOPDM
Modular Solver for LP Problems
User's Guide to version 2.12

Jacek Gondzio and Marek Makowski

WP-95-50
June 1995

IIASA International Institute for Applied Systems Analysis ▫ A-2361 Laxenburg ▫ Austria
Telephone: +43 2236 807 ▫ Fax: +43 2236 71313 ▫ E-Mail: info@iiasa.ac.at

HOPDM
Modular Solver for LP Problems
User's Guide to version 2.12

Jacek Gondzio and Marek Makowski

WP-95-50
June 1995

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Foreword

Several of the practical problems investigated at IIASA can be formulated as linear programming problems. The pressure to use more-and-more refined models cause a need for linear programming software for larger-and-larger LP problems. In the present Working Paper the most recent version of the software package HOPDM is described. The basis of this is an interior point method and the package is amplified with a nice and appropriate set of preprocessing analyses. For IIASA it is very important to have this package available, because for the large practical LP problem arising at IIASA,

HOPDM outperforms the available commercial packages. These large practical LP problems regard land–use problems and energy problems. Performance results with respect to the latter type of problems are used as illustration of the present paper.

Abstract

The paper provides a description of HOPDM, a library of routines for solving large scale linear programming problems and its implementation at IIASA. HOPDM stands for Higher Order Primal Dual Method. The algorithm implemented in HOPDM is a new variant of a primal–dual logarithmic barrier method that uses multiple correctors of centrality. The newest version of the library — HOPDM 2.12 is a robust and efficient LP code that compares favorably with the up to date commercial solvers.

The paper contains an outline of the algorithm implemented in HOPDM and information about results of tests done with large LP problems developed at IIASA. Finally, the paper provides with details of the implementation of HOPDM and its use at IIASA, as well as with information about availability of the portable version of the HOPDM library.

Keywords: Large scale linear programming, interior point method, primal-dual algorithm, multiple centrality correctors.

Contents

HOPDM
Modular Solver for LP Problems
User's Guide to version 2.12

Jacek Gondzio[*] and Marek Makowski

1 Introduction

HOPDM (which stands for <u>H</u>igher <u>O</u>rder <u>P</u>rimal <u>D</u>ual <u>M</u>ethod) described in this paper is a robust and efficient LP code that compares favourably with the up to date commercial solvers, especially for large scale LP problems. The algorithm implemented in HOPDM is a new variant [Gon94b] of a primal–dual logarithmic barrier method that uses multiple correctors of centrality. The newest version of HOPDM (version 2.12) has been recently installed at IIASA but it is also available as a library of routines for solving large scale linear programming problems.

The remaining part of this paper is organized as follows: Section 2 presents the HOPDM algorithm. The following Section 3 gives a few results of testing the code on large problems developed at IIASA. The second part of the paper contains the User's Guide. Section 4.1 summarizes general issues related to using HOPDM. Section 4.2 provides details of the implementation of HOPDM at IIASA, whereas Section 4.3 provides with information on how to get and compile the source code. Available support and recommended way to deal with the problems related to the use of HOPDM is summarized in Section 5. Finally, Appendix A contains a sample of a specification file, Appendix B provides sources for getting more information about HOPDM and testing examples, and Appendix C informs about availability of a current version of this User Guide via the IIASA World Wide Web.

2 HOPDM algorithm

The algorithm implemented in a version 2.12 of HOPDM is a new variant of a primal–dual method proposed in [Gon94a]. Below we present its scheme.

2.1 Basic primal–dual algorithm

The first theoretical results for the primal–dual algorithm come from [Meg89, KMY89]. Descriptions of its efficient implementations can be found in [LMS94, Meh92] and in the survey [GoT95].

[*]Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 102 Bd Carl Vogt, CH-1211 Geneva 4, Switzerland; on leave from the Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland, e-mail: gondzio@ibspan.waw.pl

Let us consider a primal linear programming problem

$$\begin{array}{rl} \text{minimize} & c^T x, \\ \text{subject to} & Ax = b, \\ & x + s = u, \\ & x, s \geq 0, \end{array} \tag{1}$$

where $c, x, s, u \in \mathcal{R}^n, b \in \mathcal{R}^m, A \in \mathcal{R}^{m \times n}$ and its dual

$$\begin{array}{rl} \text{maximize} & b^T y - u^T w, \\ \text{subject to} & A^T y + z - w = c, \\ & z, w \geq 0, \end{array} \tag{2}$$

where $y \in \mathcal{R}^m$ and $z, w \in \mathcal{R}^n$.

Next, let us replace the nonnegativity of constraints in the primal formulation with logarithmic barrier penalty terms in the objective function. It gives the following logarithmic barrier function

$$L(x, s, \mu) = c^T x - \mu \sum_{j=1}^{n} \ln x_j - \mu \sum_{j=1}^{n} \ln s_j. \tag{3}$$

The first order optimality conditions for (3) are

$$\begin{array}{rcl} Ax & = & b, \\ x + s & = & u, \\ A^T y + z - w & = & c, \\ XZe & = & \mu e, \\ SWe & = & \mu e, \end{array} \tag{4}$$

where X, S, Z and W are diagonal matrices with the elements x_j, s_j, z_j and w_j, respectively, e is an n-vector of all ones, $z = \mu X^{-1} e$, and μ is a barrier parameter.

A single iteration of the basic primal–dual algorithm makes one step of Newton's method applied to the first order optimality conditions (4) with a given μ and then μ is updated (usually decreased). The algorithm terminates when infeasibility and complementarity gap are reduced below a predetermined tolerance.

Having an $x, s, z, w \in \mathcal{R}^n_+, y \in \mathcal{R}^m$, Newton's direction is obtained by solving the following system of linear equations

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & W & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \\ \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_u \\ \xi_c \\ \mu e - XZe \\ \mu e - SWe \end{bmatrix}, \tag{5}$$

where

$$\begin{array}{lrcl} & \xi_b & = & b - Ax, \\ & \xi_u & = & u - x - s, \\ \text{and} & \xi_c & = & c - A^T y - z + w, \end{array}$$

denote the violations of the primal and dual constraints, respectively.

The set of linear equations (5) reduces to the *augmented system*

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}, \tag{6}$$

or further to the *normal equations system*

$$(A\Theta A^T)\Delta y = A\Theta r + h, \tag{7}$$

where

$$\begin{aligned} \Theta &= (X^{-1}Z + S^{-1}W)^{-1}, \\ r &= \xi_c - X^{-1}(\mu e - XZe) + S^{-1}(\mu e - SWe) - S^{-1}W\xi_u, \\ h &= \xi_b. \end{aligned} \tag{8}$$

Computing $(\Delta x, \Delta y)$ from (6) or Δy from (7) is usually divided into two phases: *factorization* of the matrix to some easily invertible form followed by *solution* that exploits this factorization.

Once direction $(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w)$ has been computed, the maximum stepsizes α_P and α_D that maintain nonnegativity of variables in the primal and dual spaces are found. Next, a new iterate is computed using a step reduction factor $\alpha_0 = 0.99995$

$$\begin{aligned} x^{k+1} &= x^k + \alpha_0\alpha_P\Delta x, \\ s^{k+1} &= s^k + \alpha_0\alpha_P\Delta s, \\ y^{k+1} &= y^k + \alpha_0\alpha_D\Delta y, \\ z^{k+1} &= z^k + \alpha_0\alpha_D\Delta z, \\ w^{k+1} &= w^k + \alpha_0\alpha_D\Delta w. \end{aligned} \tag{9}$$

After making the step, the barrier parameter μ is updated and the process is repeated.

2.2 Predictor–corrector technique

Factorization of the matrix (6) or (7) is usually at least an order of magnitude more expensive than the *solution*. The factorizations of systems (5) often take 60% to 90% of the total CPU time needed to solve a problem. It is thus natural to look for a possibility of reducing their number to the necessary minimum, even at the expense of some increase of a cost of a single iteration.

The predictor–corrector technique proposed by Mehrotra [Meh92] decomposes a direction vector $(\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)$ (denoted with Δ) into two parts

$$\Delta = \Delta_a + \Delta_c, \tag{10}$$

where Δ_a and Δ_c denote affine–scaling and centering components, respectively. The term Δ_a is obtained by solving (5) with $\mu = 0$ and Δ_c is the solution of equation similar to (5) with the right hand side vector

$$(0, 0, 0, \mu e - XZe, \mu e - SWe)^T,$$

where $\mu > 0$ is some centering parameter. The term Δ_a is responsible for "optimization" while Δ_c keeps the current iterate away from the boundary.

Let us observe that the affine scaling (predictor) direction Δ_a solves the linear system (5) for the right hand side equal to the current violation of the first order optimality conditions for (1)–(2), i.e. with $\mu = 0$. This direction is usually "too optimistic" — if a full step of length one could be made in it, the LP problem would be solved in one step. Predictor–corrector makes a hypothetical step in this direction. The maximum stepsizes in the primal, α_{Pa} and in the dual, α_{Da} spaces preserving nonnegativity of (x, s) and (z, w), respectively are determined and the predicted complementarity gap

$$g_a = (x + \alpha_{Pa}\Delta x)^T(z + \alpha_{Da}\Delta z) + (s + \alpha_{Pa}\Delta s)^T(w + \alpha_{Da}\Delta w)$$

is computed. It is then used to determine the barrier parameter

$$\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}, \tag{11}$$

where $g = x^T z + s^T w$ denotes current complementarity gap.

For such value of μ, the corrector direction Δ_c is computed

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & W & 0 & S \end{bmatrix} \begin{bmatrix} \Delta_c x \\ \Delta_c y \\ \Delta_c s \\ \Delta_c z \\ \Delta_c w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mu e - \Delta X_a \Delta Z_a e \\ \mu e - \Delta S_a \Delta W_a e \end{bmatrix}, \tag{12}$$

and, finally, the direction Δ of (10) is determined.

2.3 Multiple centrality correctors

Although the theory requires that subsequent iterates are in the neighborhood of the central path, in the computational practice, they may stay quite far away from it without negative consequences for the ability of making large steps (and the fast convergence). What really hurts a primal–dual algorithm is a large discrepancy between complementarity products $x_j z_j$ and $s_j w_j$.

The step Δ of (5) aims at drawing all complementarity products to the same value μ. Usually, there is little hope to reach such a goal. The approach proposed by Gondzio in [Gon94a] combines the choice of *targets* (cf. [JRTV93]) that are supposed to be easier to reach with the use of multiple correctors.

Below, we present this approach in more detail.

Assume (x, s) and (y, z, w) are primal and dual solutions at a given iteration of the primal–dual algorithm (x, s, z and w) are strictly positive. Next, assume that a *predictor* direction Δ_p at this point is determined and the maximum stepsizes in primal, α_P and dual, α_D spaces are computed which preserve nonnegativity of primal and dual variables, respectively.

We look for a *corrector* direction Δ_m such that larger stepsizes in primal and dual spaces are allowed for a composite direction

$$\Delta = \Delta_p + \Delta_m. \tag{13}$$

To enlarge these stepsizes from α_P and α_D to $\tilde{\alpha}_P = \min(\alpha_P + \delta_\alpha, 1)$ and $\tilde{\alpha}_D = \min(\alpha_D + \delta_\alpha, 1)$, respectively, a corrector term Δ_m has to compensate for the negative components in the primal and dual variables

$$\begin{aligned} (\tilde{x}, \tilde{s}) &= (x, s) + \tilde{\alpha}_P(\Delta_p x, \Delta_p s), \\ (\tilde{y}, \tilde{z}, \tilde{w}) &= (y, z, w) + \tilde{\alpha}_D(\Delta_p y, \Delta_p z, \Delta_p w). \end{aligned} \tag{14}$$

We suppose to reach the goal by adding the corrector term Δ_m that drives from this exterior trial point to the next iterate $(\hat{x}, \hat{s}, \hat{y}, \hat{z}, \hat{w})$ lying in the vicinity of the central path. However, we are aware that there is little chance to attain the analytic center in one step, i.e., to reach the point

$$v = (\mu e, \mu e) \in \mathcal{R}^{2n}, \tag{15}$$

in the space of complementarity products. Hence we compute the complementarity products of the trial point

$$\tilde{v} = (\tilde{X}\tilde{z}, \tilde{S}\tilde{w}) \in \mathcal{R}^{2n}, \tag{16}$$

and concentrate effort only on correcting their outliers. We thus project these complementarity products componentwise on a hypercube $H = [\beta_{min}\mu, \beta_{max}\mu]^{2n}$. This gives the following target

$$v_t = \pi(\tilde{v}|H) \in \mathcal{R}^{2n}, \tag{17}$$

and the following definition of the direction corrector term

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & W & 0 & S \end{bmatrix} \begin{bmatrix} \Delta_m x \\ \Delta_m y \\ \Delta_m s \\ \Delta_m z \\ \Delta_m w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_t - \tilde{v} \end{bmatrix}. \tag{18}$$

Note that the right hand side in the above system of equations has nonzero elements only in a subset of positions of $v_t - \tilde{v}$ that refer to the complementarity products which do not belong to $(\beta_{min}\mu, \beta_{max}\mu)$.

Once corrector term Δ_m is computed, the new stepsizes $\hat{\alpha}_P$ and $\hat{\alpha}_D$ are determined for the composite direction

$$\Delta = \Delta_p + \Delta_m, \tag{19}$$

and the primal–dual algorithm can move to the next iterate.

Quite often, the effort to factorize $A\Theta A^T$ matrix is tens or even hundreds of times larger than the one required in the following solves.

Whenever this is the case, the correcting process is repeated a desireable number of times. An advantage of the approach is that computing every single corrector term needs exactly the same effort (it is dominated by the solution of the system of equations (18)).

The questions arise about the choice of "optimal" number of corrections for a given problem and the criteria to stop corrections if it does not create improvement. These questions are answered in [Gon94a].

The maximum number of centrality corrections, K allowed when solving a given problem depends on the ratio of the factorization and solve efforts $r_{f/s} = E_f/E_s$. These efforts, in turn, obviously depend on the method used to solve KKT systems. HOPDM applies sparse Cholesky decomposition [Gon92, Gon93] to handle normal equations systems (7), hence the following estimates for the efforts are used

$$E_f = \sum_{i=1}^{m} l_i^2 \qquad \text{and} \qquad E_s = 2 \times \sum_{i=1}^{m} l_i + 12 \times n, \tag{20}$$

where l_i is a number of non-zero elements in i-th row of $A\Theta A^T$. Note that the ratio $r_{f/s} = E_f/E_s$ can be determined after preprocessing the KKT system and before the optimization starts. Naturally, the greater the value of the $r_{f/s}$ coefficient, the more corrections are worth trying.

The HOPDM algorithm with multiple centrality corrections [Gon94a] is a nontrivial extension of the predictor–corrector technique of [Meh92]. Undoubtedly, one of the reasons of the superiority for the multiple centrality corrections over the classical (second order) predictor–corrector algorithm is a clever choice of *targets* (cf. [JRTV93]) that, in our approach, do not have to be *analytic centers*, i.e., the perfectly centered points on the central trajectory. Instead, we choose weighted analytic centers that stay in a wide neighborhood of the central path and can be reached much easier.

3 Results of tests

To give the reader some idea about the efficiency of our code, we have compared it with the newest available commercial implementation of the simplex algorithm, Cplex, version 3.0 [Bix94]. Both codes were run on the same 50MHz Sun Sparc 10 workstation with 64MB of memory. Let us mention that the three large linear problems developed by the Environmentally Compatible Energy Strategies Project at IIASA [NGI+93] that are used in this comparison are public domain (see Appendix B.3).

Table 1. Problem statistics.

Problem	Original size			Cplex presolve			HOPDM presolve		
	m	n	nonz	m	n	nonz	m	n	nonz
mod2	35664	31728	220116	29293	29175	126163	27144	27275	121474
world	35510	32734	220748	29196	30897	126582	27029	28986	121905
world5	52238	36141	316737	35960	32659	147813	34432	31568	143846

Table 1 gives the statistics of linear programs used in this comparison. It contains original sizes of the problems and their dimensions after the execution of default presolve analyses (a presolve routine of HOPDM is described in detail in [Gon94b]).

Table 2. Solution statistics.

Problem	Cplex 3.0		HOPDM 2.12		$\frac{t_{Cplex}}{t_{HOPDM}}$
	iters	time	iters	time	
mod2	119258	26818.63	48	1437.63	18.6
world	134099	32801.67	51	1796.32	18.2
world5	59956	15717.30	40	2148.33	7.3

Table 2 gives statistics on the solution of these problems. It reports iterations and CPU time in seconds to reach optimality (or, as is the case of world5 example, to state infeasibility). The solution time contains all processing time except reading MPS input and writing solution report. Additionally, Table 2 gives in its last column the ratio of the solution times of two codes compared. From the analysis of Table 2 results, it is obvious that on these three nontrivial LPs, HOPDM 2.12 outperforms Cplex 3.0 simplex code.

However, we want to warn the reader from drawing conclusions based on this very limited comparison. Results of Table 2 should rather be considered as an indication that there exist numerous classes of real–life linear optimization problems (usually of considerable size) that can be solved significantly faster with the "new" interior point algorithm than with the "old" simplex method.

Results from a comparision of an older version of HOPDM with another widely used LP code MINOS using a larger set of problems provided by the Food and Agriculture Project of IIASA are presented in [GoM95][1].

Finally we want to point out that HOPDM has been successfully tested on all problems (including the set of infeasible problems) contained in the Netlib collection that is commonly used for testing LP solvers. The results of those tests can be found in [Gon94a, Gon94b].

4 User's guide

HOPDM is available is two versions:

- Customized version installed at IIASA (further on referred to as H_IIASA). This version has additional features, namely it provides dynamic allocation of memory and allows for formulation of the problem in both MPS and LP-DIT formats.
- Fully portable version written in Fortran 77, which is available in the source form (further on referred to as H_SRC). This version can be compiled on any computer with a Fortran 77 compiler.

Therefore the User's Guide is divided into three subsections. Section 4.1 provides information useful for all users of HOPDM. Problems specific for the H_IIASA and H_SRC versions are dealt with in Sections 4.2 and 4.3, respectively.

4.1 General information

In order to solve an LP problem a user can select some of parameters that control the optimization algorithm and he/she has to provide the problem formulation. The selection of controlling parameters is conventionally called problem specification and it is discussed in Section 4.1.1. Requirements for the formulation of LP problem are presented in Section 4.1.2.

HOPDM can output results of the optimization in several ways, which can be controlled by a user. The output generated by HOPDM is summarized in Section 4.1.3.

There are no particular size limits of problems to be solved, except total amount of operational memory available on a given computer. A 64MB workstation, for example, can handle problems of about 50,000 constraints and 100,000 variables. However, in order to make efficient use of the computer resources one has to either select a proper version of H_IIASA or to make appropriate redefinitions of the corresponding `PARAMETER` statements in H_SRC (cf. Sections 4.2 and 4.3 for details).

It is difficult to calculate even approximately the memory requirement for a problem characterized only by the numbers of rows and columns. The specified numbers are usually increased during the conversion of the problem to the standard form (1) and during splitting dense columns. Subsequently, the presolve analysis may decrease these dimensions substantially. The major factors determining the memory requirement are however the density of the final matrix A and `NZL` (the number[2] of non-zero elements of the Cholesky factor of the matrix $A\Theta A^T$). In order to decrease the value of `NZL` matrix A is permuted according to the ordering resulting from the minimum degree heuristic (see [Liu89]).

[1] A draft version of this paper is available as WP-93-002.

[2] This number is reported in the log-file of HOPDM.

4.1.1 Problem specification

HOPDM has only a small number of user controllable parameters. All parameters have their own default values. For typical applications of HOPDM there is no need for changing the default settings. There are two exceptions to this rule:

1. If a user wants to use a complete dual solution (shadow prices), then the presolve level should be lowered at least to 4.

2. If the problem is read from the MPS-formatted file, a user has to specify (possibly overestimated) numbers of rows, columns and non-zero elements of the LP problem.

The specifications, if needed, should be prepared in a free format ASCII file, which meets the following requirements:

- The first statement must read `begin`.
- The last statement must read `end`.
- Statements between the first and the last ones may appear in any order.
- Each line contains only one statement.
- Each statement is composed of a key word (sometimes two) that counts at most 12 characters, and an argument, if required.
- Only first three characters of a (first) word are processed.
- Argument, if required, should start in column 13.

The following key words are recognized (values in `< >` denote the corresponding default settings, the word `none` means that the key word has no argument):

minimize < **none** > : minimize the objective function (default),

maximize < **none** > : maximize the objective function,

mps file < **mps** > : the name of the MPS input file,

log file < **log** > : the name of the log file,

sol file < **sol** > : the name of the MPS-like output (solution) file,

opt tol < **1.0D-8** > : (relative) optimality tolerance,

presolve < **6** > : level of presolve analysis desired,

rows < **0** > : upper limit for the number of rows in LP,

columns < **0** > : upper limit for the number of columns in LP,

elements < **0** > : upper limit for the number of nonzeros in LP,

objective < **none** > : the name of the objective row to be chosen in the MPS file,

rhs < **none** > : the name of the RHS section to be chosen in the MPS file,

ranges < **none** > : the name of the RANGES section to be chosen in the MPS file,

bounds < **none** > : the name of the BOUNDS section to be chosen in the MPS file.

The specified dimensions (number of rows, columns and elements) must account for additional columns, rows and elements generated by HOPDM during the conversion of a given LP problem to the standard form (1) and during splitting dense columns. Note that the default zero values for numbers of rows, columns and elements will be replaced by actual dimensions read from the LP-DIT format input file and computed during preprocessing of the problem. Therefore the dimensions should not be specified, if the LP problem is provided in the LP-DIT format.

The keywords are case sensitive and should be written in either capital or small case letters. For example,

```
rows        1000
```

means the same as

```
ROWS        1000
```

However, the keyword `Rows` will not be understood.

Two parameters that can be supplied in the `specs` file influence the optimization in an important way. The first is the optimality tolerance. HOPDM solves problems by default to 8-digit exact optimum, i.e., the optimum satisfies

$$\frac{|c^T x - (b^T y - u^T w)|}{1 + |b^T y - u^T w|} \leq 10^{-8}. \tag{21}$$

In case of numerical difficulties reported during the solution process for some extremely diffcult problems, this tolerance should be relaxed e.g. to 1.0D-6.

The second important parameter is the level of presolve analysis desired. Its value `6` means default presolve as described in detail in [Gon94b]. A larger value `7` would impose accepting any implied bounds on LP variables, including very large values for bounds. While this may result in further reduction of the problem size, it can sometimes lead to creating formulation that is more difficult for an IPM solver due to the presence of unrealistically large variable bounds. Lower values of `presolve` parameter disable specific options:

`5` turns off splitting dense columns,
`4` additionally turns off the heuristic to make A sparser,
`3` additionally turns off column aggregation technique,
`2` additionally turns off the elimination of dominated columns,
`1` additionally turns off the elimination of redundant rows,
`0` turns off any presolve analysis.

The presolve level should be lowered to at least 4, if a complete dual solution (shadow prices) is needed.

An example of the specification file is reproduced in Appendix A.

4.1.2 Formulation of LP problems

The way in which an LP problem is formulated can have a dramatic impact not only on the solution time. Different formulations of the same real problem may result in completely different solution processes, although, clearly, leading to the same optimal[3] solution. Some

[3] More exactly to a solution that has the same optimal value. Quite often a real-life problem has many (very different) optimal solutions which has the same (or nearly the same) value of the goal function (see e.g. [Mak80]). Application of a regularization technique can help to select out of many optimal solution the one that has some additional properties (see e.g. [Mak94a] for more details).

formulations (for example the one leaving dense columns in matrix A) may require much more computer resources, some others (like the one containing unrealistically large variable bounds) may create unexpected numerical difficulties. This topic is of critical importance for applications of any LP model to real-life problems. However, it is far beyond the scope of this paper. Therefore we can only strongly recommend a reader who deals with large scale real-life problems to consult existing literature (e.g. [GdHR+93, Gon94b, JdJRT93, HuJ90, Mak94c, Wil90]) on problems related to the formulation of LP and the interpretation of results.

An LP problem to be solved by H_SRC has to be provided in the MPS format, which is the de facto industry standard for formulation of LP problem in the form of ASCII text file. There is a number of slightly different formats which are called MPS format. HOPDM accepts a commonly used version of it (cf. [Mur81]). Users of H_IIASA should provide the problem in either the MPS format or in the LP-DIT format (cf. [Mak94b] and Section 4.2 for details).

4.1.3 Output generated by HOPDM

HOPDM generates a number of files with two types of information: final solution of the problem and *log* information about optimization process. Default names of files are used unless a corresponding name is defined in the specification file.

The following files are created by HOPDM (default names are given in <>):

full log and diagnostics <log>: contains many useful statistics on the LP problem itself and on the solution process. In particular, it contains the log information on subsequent iterations of the primal-dual algorithm and, eventually, error messages, if an unusual termination occurs.

short log and diagnostics <*terminal screen*>: a shorter version of diagnostics is output to the `stderr` file, which is usually a terminal screen. This information can be redirected to a file by an appropriate modification of a command, for example: `hopdm >& my_log &`.

summary file <fort.99>: contains a short summary of the problem and status of the solution, it is useful for handling a series of optimization problems.

solution <sol>: contains a solution in one of the commonly used forms for solutions of LP problems output as an ASCII file. The optimal solution may contain only a subset of rows, in case rows elimination technique was applied while solving the problem.

A `log` file reports useful data on the problem solved. It gives for example brief information on reductions obtained during presolve analysis, information on the results of preprocessing for sparsity in Cholesky decomposition (the number of sub-diagonal elements in triangular factor, the number of flops required to compute this decomposition, etc) and the results of scaling matrix A. Additionally, it monitors the progress of optimization. A single iteration report contains the following lines (an example contains the report of iteration 17 when solving problem ETAMACRO):

```
PCCHCK: ||A*x-b||=1.609D-09  ||x+s-u||=1.438D-08  ||At*y+z-w-c||=1.165D-07
NUMFCT: Max. pivot in row    325,  Dii=  1.4159D+10
NUMFCT: Min. pivot in row     84,  Dii=  6.0514D-10
IRSOLV: Iter= 1 null space error= 9.83D-13
```

```
IRSOLV: Iter= 1 null space error= 1.22D-12
IRSOLV: Iter= 1 null space error= 6.65D-14
PCPDM:  It=  17  O= 1  GP= 1.05D-01  BRR= 5.31D-06 AP=8.06D-01 AD=9.26D-01
```

The first line in this report gives current violation of primal constraints, variable bounds and dual constraints, respectively, i.e., infinity norms of vectors ξ_b, ξ_u and ξ_c used in (5).

The following two lines inform about the largest and the smallest pivot elements found during the Cholesky decomposition of normal equations (7). Large discrepancy of these values may sometimes indicate future numerical difficulties.

The following three lines inform about the precision in the computations of subsequent terms of Newton's direction. In the above example, direction is composed from: the affine-scaling term, Mehrotra's corrector term and one generalized centering corrector term (three solves of (5), (12) and (18), respectively).

The last line in this partial report is a summary of the primal-dual iteration. It contains: the iteration number, the number of generalized centrality correctors used (O=1), current duality gap (GP), the value of barrier parameter μ (BRR) and the stepsizes in the primal and dual spaces (AP and AD), respectively.

A shorter report directed to the screen contains only the first and the last line from the example presented above.

Users of the H_IIASA version can optionally supress generation of a solution in the ASCII file and/or provide solution in LP-DIT format (see Section 4.2 for details).

4.2 Using HOPDM at IIASA

The H_IIASA version of HOPDM available on all computers running Solaris provides full functionality of the H_SRC version and some additional features, mainly the dynamic allocation of memory and an optional definition of the problem in the efficient binary format LP-DIT (see [Mak94b] for the description and documentation).

For efficiency reasons there are two versions of H_IIASA, called `hopdm` and `hopdm4`, respectively. They are identical with only one difference: `hopdm` handles indices of rows and columns as `INTEGER*2` variables in Fortran and as `short int` variables in C++, therefore `hopdm` can be used for solving problems in which the resulting formulation (after transformation of a given LP problem to the internal HOPDM's standard form) the number of columns and rows is smaller than 32K. When in doubt, a user should try to run `hopdm`, which will exit with an error message, if the problem is too big. Larger problems should be solved with the second version of H_IIASA called `hopdm4`, which uses `INTEGER*4` and `long int` types of variables (in Fortran and C++, respectively) for handling indices, therefore the only size limitation of the problem is due to the size of memory available on the computer on which the `hopdm4` is run. In case of insufficient amount of memory an appropriate message provides information about the part of the data, for which memory cannot be allocated. Since there are no other differences between `hopdm` and `hopdm4` versions of H_IIASA, we will refer to both versions using the name `hopdm`.

It is assumed that a user of `hopdm` is familiar with the information contained in Section 4.1, therefore this section contains only additional information. The `hopdm` has been developed using the functions available as source code (see Section 4.3 for details) with the `hmain2.f` function replaced by `C++` code, which provides additional functionality of `hopdm` that can be summarized as follows:

Definition of LP problem in the LP-DIT format: Generation of an LP problem in the MPS form is error prone and, especially for larger problems, requires substantial amount of computer resources. Therefore an efficient binary format has been implemented. This format is supported by the LP-DIT library [Mak94b], which allows for easier generation of LP problems. Additionally LP-DIT library provides functions for a problem modification, which facilitates multiple criteria model analysis and/or scenario analysis of LP models substanially. The LP-DIT library can be used with any problem generator written in C++ and/or Fortran for generating the resulting formulation of the problem in the LP-DIT format (see [Mak94b] for details). The problems already available in the MPS format can be converted to the LP-DIT by `mps2dit` (a utility distributed with LP-DIT and installed at IIASA). The users of `hopdm4` should use the corresponding version of `mps2dit` which is called `mps2dit4`.

Solution in the LP-DIT format: Typically the analyst needs only a small fraction of a solution of a problem (the larger the problem the smaller this fraction is). However, especially for large scale problems, writing and processing of a solution file can require a lot of computing resources (CPU time and disc space) as well as it may take a nonnegligable amount of the wall-clock-time to write/read an ASCII file. Whenever this is the case, it is rational to use LP-DIT binary format for processing a solution.

Dynamic allocation of memory: Instead of a static allocation of memory provided by the Fortran version of HOPDM, H_IIASA allocates storage according to the needs of a particular problem. If the problem definition is provided in the MPS format, then maximum numbers of rows, columns and non-zero elements must be specified in the specification file. If the problem is provided in LP-DIT format no specifications of those numbers are needed, because the default zero values of those parameters are replaced by the exact dimensions provided in the LP-DIT format.

Use of command line arguments: A user can specify additional options and parameters using the command line arguments. The list of available options can be obtained by the command: `hopdm -h`.

Short log redirection: H_IIASA redirects the short log by default to a file named `log0`. One can redirect this information to a terminal screen by using the `-l-` option of the command line.

4.3 Retrieving and compiling source code

HOPDM code is available in a form of Fortran source files (cf. Appendix B.1). All its routines are written in a standard Fortran 77, which ensures portability. HOPDM library has already been installed on a PC computer, IBM's Power PC workstation and Sun Sparc workstation. We provide the user with an example `makefile` for a Sun Sparc workstation.

HOPDM is a library of routines that communicate with each other through the parameter lists only. These parameters are described in detail in every source file. All routines are excessively documented (description of the routine function, its input and output parameters and comment lines take, in the average, more than 50% of each source file).

The simplest usage of the library is to apply it as a stand–alone LP solver that reads LP data (in a widely accepted MPS format), solves the problem and prints the output (in an MPS-like format). This, in fact, reflects the structure of the main (HMAIN2.F) routine.

For the user's convenience, three PARAMETER declarations (placed at the beginning of MAIN routine) set up all the dimensions of different arrays used in the HOPDM. They should be chosen appropriately for every HOPDM installation depending on the amount of memory available on a given computer. Current version of MAIN routine has two alternative settings of parameters: for 32 MB and for 64 MB workstations, respectively.

A distributed version of the source files uses half length integers `INTEGER*2` to handle row and column indices which impose implicit constraints on the size of problems that can be solved: they cannot exceed 32627 constraints and variables. A user is provided with a `Two2four` script that uses `sed` utility to automatically replace all `INTEGER*2` declarations with `INTEGER*4` removing thus any implicit limits on the size of solvable problems.

The executable program is called `hopdm`. Information about problem specification and definition is provided in Section 4.1. Once the program has been compiled and linked successfully, its functions can be verified by running it on small examples of linear programs AFIRO, ADLITTLE and a larger one PILOTNOV all from the Netlib collection [Gay85].

To facilitate the HOPDM usage, we supply the users with a few LP test problems. However, we would like to draw the reader's attention to a reach collection of LP test problems gathered by David Gay [Gay85] that can be found on Netlib (cf. Appendix B.2). The three examples mentioned above and distributed with HOPDM come from this source.

5 Support and reporting bugs

Although a great deal of effort has been made to provide users of HOPDM with a robust and efficient code, it is still likely that sometimes a user of the versions described in this paper will meet difficulties, especially when solving highly complicated and very large problems. Users, who apply the code to solving practical problems may also have suggestions for improvements of the interface. Therefore the authors would appreciate comments, suggestions and reports of problems with using HOPDM. The later should always be accompanied with the problem formulation (in form of the MPS format or LP-DIT format file) and with the specification file (if that has been used).

Users of the H_IIASA version should not hesitate to contact Marek Makowski in case of questions, problems or comments related to using HOPDM. Other users should contact Jacek Gondzio directly.

Acknowledgment

The development of the HOPDM software has greatly benefited from the few years cooperation of the authors within the framework of the collaborative research organized by the Methodology of Decision Analysis Project with the Polish Academy of Sciences and the Institute of Automatic Control of the Warsaw University of Technology. The research has also been supported by the grant No PB 8 S505 01505 of the Committee for Scientific Research of Poland and by the grant #12-34002.92 of the Swiss National Foundation for Scientific Research.

Both H_IIASA and H_SRC versions include the `MMD` routine by Joseph Liu, which is an implementation of the Multiple Minimum Degree ordering (cf [Liu89]) kindly provided by the author to be used exclusively for research and teaching purposes.

The authors would like to thank all users of HOPDM at IIASA, who provided valuable comments and LP problems that helped us to improve the described implementation. In particular we want to thank Sabine Messner for providing a number of difficult and large examples (some of them are reported in Section 3) and Günther Fischer for providing another set of large examples (reported in [GoM95]).

References

[Bix94] R. Bixby, *Progress in linear programming*, ORSA Journal on Computing **6**, no. 1 (1994) 15–22.

[Gay85] D. Gay, *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society COAL Bulletin no. 13 (1985) 10–12.

[GdHR+93] O. Güler, D. den Hertog, C. Roos, T. Terlaky and T. Tsuchiya, *Degeneracy in interior point methods for linear programming: a survey*, Annals of Operations Research **46** (1993) 107–138.

[GoM95] J. Gondzio and M. Makowski, *Solving a class of LP problems with primal-dual logarithmic barrier method*, European Journal of Operational Research **80**, no. 1 (1995) 184–192.

[Gon92] J. Gondzio, *Splitting dense columns of constraint matrix in interior point methods for large scale linear programming*, Optimization **24** (1992) 285–297.

[Gon93] ______, *Implementing Cholesky factorization for interior point methods of linear programming*, Optimization **27** (1993) 121–140.

[Gon94a] ______, *Multiple centrality corrections in primal dual method for linear programming*, Technical Report 1994.20, Department of Management Studies, University of Geneva, Geneva, Switzerland, 1994. (revised in May, 1995).

[Gon94b] ______, *Presolve analysis of linear programs prior to applying the interior point method*, Technical Report 1994.3, Department of Management Studies, University of Geneva, Geneva, Switzerland, 1994.

[GoT95] J. Gondzio and T. Terlaky, *A computational view of interior point methods for linear programming*, in Advances in Linear and Integer Programming, J. Beasley, ed., Oxford University Press, Oxford, England, 1995. (to appear).

[HuJ90] I. Huntley and D. James, eds., *Mathematical Modelling, A Source Book of Case Studies*, Oxford University Press, Oxford, New York, Tokyo, 1990.

[JdJRT93] B. Jansen, J. de Jong, C. Ross and T. Terlaky, *Sensitivity analysis in Linear Programming: Just be careful !*, Technical Report AMER.93.022, Shell Internationale Research, Maatschappij B.V., The Netherlands, 1993.

[JRTV93] B. Jansen, C. Roos, T. Terlaky and J. Vial, *Primal–dual target following algorithms for linear programming*, Technical Report 93–107, Faculty of Technical Mathematics and Informatics, Technical University Delft, Delft, The Netherlands, 1993. (to appear in the special issue of *Annals of Operation Research*, K. Anstreicher and R. Freund (eds.).).

[KMY89] M. Kojima, S. Mizuno and A. Yoshise, *A primal–dual interior point algorithm for linear programming*, in Progress in Mathematical Programming: Interior Point and Related Methods, N. Megiddo, ed., Springer–Verlag, New York, 1989, pp. 29–48.

[Liu89] J. Liu, *The evolution of the Minimum Degree Ordering algorithm*, SIAM Review **33**, no. 1 (1989) 1–19.

[LMS94] I. Lustig, R. Marsten and D. Shanno, *Interior point methods for linear programming: Computational state of art*, ORSA Journal on Computing **6**, no. 1 (1994) 1–14.

[Mak80] M. Makowski, *Modeling the expansion of the water system in the Upper Noteć region*, in Proceedings of Joint Task Force Meeting on Development Planning for the Noteć (Poland) and Silistra (Bulgaria) Regions, A. Albegov and R. Kulikowski, eds., Collaborative Paper, vol. CP-80-09, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980, pp. 267–295.

[Mak94a] ———, *Design and implementation of model-based decision support systems*, Working Paper WP-94-86, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[Mak94b] ———, *LP-DIT, Data Interchange Tool for Linear Programming Problems, (version 1.20)*, Working Paper WP-94-36, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[Mak94c] ———, *Methodology and a modular tool for multiple criteria analysis of LP models*, Working Paper WP-94-102, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[Meg89] N. Megiddo, *Pathways to the optimal set in linear programming*, in Progress in Mathematical Programming: Interior Point and Related Methods, N. Megiddo, ed., Springer–Verlag, New York, 1989, pp. 131–158.

[Meh92] S. Mehrotra, *On the implementation of a primal–dual interior point method*, SIAM Journal on Optimization **2** (1992) 575–601.

[Mur81] B. Murtagh, *Advanced Linear Programming: Computation and Practice*, McGraw-Hill, New York, 1981.

[NGI+93] N. Nakicenovic, A. Gruebler, A. Inaba, S. Messner, S. Nilsson, Y. Nishimura, H.-H. Rogner, A. Schaefer, L. Schrattenholzer, M. Strubegger, J. Swisher, D. Victor and D. Wilson, *Long-term strategies for mitigating global warming*, Energy – The International Journal, Special Issue **18**, no. 5 (1993) 409–601.

[Wil90] H. P. Williams, *Model Building in Mathematical Programming*, J. Wiley & Sons, Chichester, New York, 1990.

A Sample of specification file

The following specification file can be used for solving the `afiro` example from the Netlib collection [Gay85], if the problem is provided in the MPS format.

```
begin
rows        100
cols        200
elements    1020
MPS FILE    afiro.mps
LOG FILE    afiro.log
SOLUT FILE  afiro.res
opt tol     1.0D-8
presolve    6
minimize
end
```

For the problem provided in the LP-DIT format the declarations of numbers of `rows`, `cols` and `elements` should be removed from the specification file. If shadow prices are needed then the `presolve` level should be smaller than or equal to 4.

B Getting more information

The following subsections provide sources for getting more information about the HOPDM and about LP test examples.

B.1 How to get the source code?

The Fortran 77 sources of HOPDM code (version 2.11 of March 1995) are distributed free of charge for academic use through Netlib. To get HOPDM connect by `ftp` to the netlib ftp site `netlib.att.com` as an *anonymous* user. Once you are done, change directory to `netlib/opt`. HOPDM can there be found as `hopdm.shar.Z` file.

B.2 How to get the Netlib LP tests?

To get the collection of LP test problems gathered by David Gay [Gay85] connect to netlib ftp site (see Appendix B.1) and change directory to `netlib/lp`. In subdirectory `data` you will find 95 feasible LP test problems. In subdirectory `infeas` you will find 29 infeasible LP test problems. Additionally, in subdirectory `data/kennington` you will find 16 large LP problems collected by Kennington.

B.3 How to get other large scale LP tests?

All test problems (except UK.mps) used in the comparison of [Gon94a] are available via `ftp` from Computational Optimization Laboratory of the University of Iowa, USA. To get them, connect by ftp to `col.biz.uiowa.edu` as an *anonymous* user. Once you are done, change directory `pub/testprob/lp/gondzio`. In particular, this directory contains MOD2 and WORLD problems used in a comparison of Section 3.

B.4 How to get papers that describe HOPDM?

Two reports [Gon94b, Gon94a] that describe in detail the novel implementational techniques applied in HOPDM are available through World Wide Web of Argonne National Laboratory, USA. To see the archive look at the URL:
`http://www.mcs.anl.gov/home/otc/InteriorPoint/`.
Then click on "papers in the archive" and search for interesting reports. You will get either PostScript (*.ps) or DeViceIndependent (*.dvi) versions of the above reports.

C Availability of this User's Guide

All Working Papers published by the Methodology of Decision Analysis Project are available from the Publication Department of IIASA. Most of them (including all papers written by the authors of this WP) are available via the WWW of IIASA:
`http://www.iiasa.ac.at`
The Welcome Page of the IIASA WWW provides an easy access to the IIASA Publications, which can be examined in various ways (by author's name, project, date, etc). Postscript files can be obtained free of charge via WWW. Hard copies can be ordered from the Publication Department of IIASA (orders can be placed also via WWW).

Updated versions of this User's Guide will be made available, if the need arises, also in the form of a IIASA WP.