Primal-dual interior point method for multicommodity network flows with side constraints and comparison with alternative methods †

J. Castro and N. Nabona
Statistics and Operations Research Dept., Univ. Politècnica de Catalunya
Pau Gargallo 5, 08071 Barcelona, Spain
phone: 34-3-4017335 fax: 34-3-4017040 email: jcastrop@eio.upc.es

Abstract

This document presents a primal-dual interior point algorithm for the solution of large multicommodity network flow problems with or without side constraints. The method exploits the structure of the problem and uses a preconditioned conjugate gradient solver. The algorithm has been implemented for the case of pure multicommodity problems (without side constraints), and some computational results are presented, comparing the performance of the code developed with alternative ones.

Keywords

Computational Benchmarks, Interior Point Methods, Linear Programming, Multicommodity Network Flows, Primal-Dual Algorithm, Side Constraints

1 INTRODUCTION

Multicommodity network flows (Kennington and Helgasson (1980)) are used as a modelling tool in many applications in routing, telecommunication networks, allocation and transportation problems and in electrical power systems. It is thus important to have efficient tools to optimize this kind of problems. Interior point methods have recently gained wide recognition as an optimization procedure for applications with general linear constraints and both their general primal-dual and dual-affine-scaling formulations have

† Appeared in *System Modelling and Optimization*, J. Doležal and J. Fidler (eds), Chapman & Hall, 1995, p. 451–458.

been extended to the linear multicommodity network flow problem (Choi and Goldfarb (1990), Kamath et al. (1993)).

In the work described here the primal-dual interior point algorithm has been specialized for solving multicommodity network flow problems, considering additional side constraints. The method is heavily based on the use of a preconditioned conjugate gradient algorithm for solving repeatedly a part of the systems of the type $ASA^t dy = \bar{b}$, ASA^t being symmetric and positive definite. Two specialized multicommodity network flow codes (Kennington (1979), Castro and Nabona (1995)), both based on the primal partitioning algorithm (Kennington and Helgasson (1980)), have been run on the same test problems solved with the interior point code in order to compare the performance of the interior point solution with that of the specialized network codes. The performance of the multicommodity primal-dual interior point code developed is also compared with that of a general primal-dual interior point code (Vanderbei (1993)) so that it is possible to appreciate the computational advantages of using multicommodity specialization within the interior point scheme. Randomly generated multicommodity test problems of sizes ranging from 100 to 10000 arcs and numbers of commodities ranging from 1 to 200 were solved and their results are reported.

2 OUTLINE OF THE PRIMAL-DUAL INTERIOR POINT FOR UPPER-BOUNDED LINEAR PROGRAMMING

Let us consider the following minimization problem with upper bounds in some variables

$$\begin{aligned} \min \quad & c^t x && (1)\\ \text{subj. to} \quad & Ax = b && (2)\\ & \underline{0} \le x_u \le \overline{x}_u && (3)\\ & \underline{0} \le x_l && (4) \end{aligned}$$

where $x_u \in \mathbb{R}^{n_u}$, $x_l \in \mathbb{R}^{n_l}$, $x = (x_u^t, x_l^t)^t$, $x \in \mathbb{R}^n$ (thus $n = n_u + n_l$), $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Considering an appropriate partitioning of c and A, equations (1) and (2) can be rewritten as:

$$c_u^t x_u + c_l^t x_l \quad (5)$$

$$A_u x_u + A_l x_l = b \quad (6)$$

where $c_u \in \mathbb{R}^{n_u}$, $c_l \in \mathbb{R}^{n_l}$, $A_u \in \mathbb{R}^{m \times n_u}$ and $A_l \in \mathbb{R}^{m \times n_l}$. The dual of the minimization problem stated can be cast as:

$$\begin{aligned} \max \quad & b^t y - \overline{x}_u^t w \\ \text{subj. to} \quad & A_u^t y + z_u - w = c_u \\ & A_l^t y + z_l \qquad = c_l \end{aligned}$$

$$z = (z_u^t, z_l^t)^t \geq 0 \quad w \geq 0$$

where $z_u \in \mathbb{R}^{n_u}$, $z_l \in \mathbb{R}^{n_l}$ (thus $z \in \mathbb{R}^n$) and $w \in \mathbb{R}^{n_u}$.

Considering a logarithmic barrier function (μ being its penalty term) for the nonnegativity constraints of the variables, and adding slacks $f \in \mathbb{R}^{n_u}$ for the upper bounds ($x_u + f = \overline{x}_u$), the Kuhn-Tucker optimality conditions for both the dual and the primal can be written as:

$$b_{1_l} \equiv \mu e_{n_l} - X_l Z_l e_{n_l} = 0 \tag{7}$$

$$b_{1_u} \equiv \mu e_{n_u} - X_u Z_u e_{n_u} = 0 \tag{8}$$

$$b_2 \equiv \mu e_{n_u} - FW e_{n_u} = 0 \tag{9}$$

$$b_3 \equiv b - (A_u x_u + A_l x_l) = 0 \tag{10}$$

$$b_{4_l} \equiv c_l - (A_l^t y + z_l) = 0 \tag{11}$$

$$b_{4_u} \equiv c_u - (A_u^t y + z_u - w) = 0 \tag{12}$$

e_l being the l-dimensional vector of 1's, and where matrices X_u, X_l, Z_l, Z_u, F and W are diagonal and defined as $M \in \mathbb{R}^{l \times l} = \text{diag}(m_1, \ldots, m_l)$. It is clear than when $n_u = 0$ ($n = n_l$) only equations (7, 10 and 11) hold, thus having the optimality conditions of the standard primal-dual algorithm with no upper bounds.

When using Newton's method to find a point satisfying (7–12), linear systems of the type $J_i d_i = -f_i$ must be solved at each iteration i. These solutions amount to finding dy and then computing dx, dw, dz_u, dz_l, in:

$$(ASA^t)dy = b_3 + ASr \tag{13}$$

$$dx = S(A^t dy - r) \tag{14}$$

$$dw = F^{-1}(b_2 + W dx_u) \tag{15}$$

$$dz_u = b_{4_u} + dw - A_u^t dy \tag{16}$$

$$dz_l = b_{4_l} - A_l^t dy \tag{17}$$

where

$$\begin{gathered} r = (r_u^t, r_l^t)^t \quad r \in \mathbb{R}^n \quad r_u \in \mathbb{R}^{n_u} \quad r_l \in \mathbb{R}^{n_l} \\ r_u = F^{-1} b_2 + b_{4_u} - X_u^{-1} b_{1_u} \quad r_l = b_{4_l} - X_l^{-1} b_{1_l} \end{gathered} \tag{18}$$

and

$$\begin{gathered} S = \begin{pmatrix} S_u & \mathbf{0} \\ \mathbf{0} & S_l \end{pmatrix} \quad S \in \mathbb{R}^{n \times n}, \quad S_u \in \mathbb{R}^{n_u \times n_u}, \quad S_l \in \mathbb{R}^{n_l \times n_l} \\ S_u = F X_u (Z_u F + X_u W)^{-1} \qquad S_l = Z_l^{-1} X_l \end{gathered} \tag{19}$$

(where S_u and S_l can be directly computed, since they are made of products and sums of diagonal matrices). A justification of this process can be found in Castro (1995a).

It is quite clear that the main computational burden in solving system (7–12) is the repeated solution of the linear system (13).

3 FORMULATION OF THE LINEAR MULTICOMMODITY NETWORK FLOWS WITH SIDE CONSTRAINTS

The multicommodity network flow problem corresponds to the minimization problem (1–4). Let us consider a network with m^* nodes, n^* arcs (where the last one is a rooted arc added to avoid the singularity of the network matrix $A^* \in \mathbb{R}^{m^* \times n^*}$) and k commodities. Adding slacks to the mutual capacity constraints ($s_{mc} \in \mathbb{R}^{n^*}$) and the side constraints ($s_{sc} \in \mathbb{R}^t,\ t \geq 0$), and denoting by $x_i = (x_{i_a}^t x_{i_r})^t \in \mathbb{R}^{n^*}$ i=1,...,k the flows for each commodity ($x_{i_r} \in \mathbb{R}$ being the rooted arc and $x_{i_a} \in \mathbb{R}^{n^*-1}$ the remaining ones, for commodity i) with capacities $\overline{x}_i \in \mathbb{R}^{n^*-1}$ (thus considering the rooted arcs as uncapacitated ones), by $b_{mc} \in \mathbb{R}^{n^*}$ the mutual capacities (for the rooted arcs an arbitrary mutual capacity can be considered), by $b_i \in \mathbb{R}^{m^*}$ i=1,...,k the node supplies/demands of each commodity i, by $T_i \in \mathbb{R}^{t \times n^*}$ the matrices defining the side constraints structure, and by $\overline{b}_{sc}, \underline{b}_{sc} \in \mathbb{R}^t$ the upper and lower bounds of the side constraints, then the multicommodity network problem can be stated as:

$$\min \quad \sum_{i=1}^{k} c_i^t x_i \tag{20}$$

subj. to

$$\begin{array}{|c|c|c|c|c|c|} \hline A^* & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & A^* & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline \mathbf{0} & \mathbf{0} & \dots & A^* & \mathbf{0} & \mathbf{0} \\ \hline \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} & \mathbb{1} & \mathbf{0} \\ \hline T_1 & T_2 & \dots & T_k & \mathbf{0} & \mathbb{1} \\ \hline \end{array} \begin{array}{|c|} \hline x_1 \\ \hline x_2 \\ \hline \vdots \\ \hline x_k \\ \hline s_{mc} \\ \hline s_{sc} \\ \hline \end{array} = \begin{array}{|c|} \hline b_1 \\ \hline b_2 \\ \hline \vdots \\ \hline b_k \\ \hline b_{mc} \\ \hline \overline{b}_{sc} \\ \hline \end{array} \tag{21}$$

$$\underline{0} \leq x_{i_a} \leq \overline{x}_i \quad 0 \leq x_{i_r} \quad i = 1, \dots, k \tag{22}$$

$$\underline{0} \leq s_{sc} \leq \overline{b}_{sc} - \underline{b}_{sc} \tag{23}$$

$$\underline{0} \leq s_{mc} \leq b_{cm} \tag{24}$$

In this case the total number of variables and constraints is given by $n = (k+1)n^*+t$ and $m = km^*+n^*+t$, and the partitioning of the variables $x = (x_u^t, x_l^t)^t$ is $x_u^t = (x_{1_a}^t, \dots, x_{k_a}^t, s_{mc}^t, s_{sc}^t)$ and $x_l^t = (x_{1_r}, \dots, x_{k_r})$.

In the multicommodity problem, matrix S defined in (19) can be partitioned as:

$$S = \begin{array}{|c|c|c|c|}\hline S_1 & & & \\ \hline & \ddots & & \\ \hline & & S_k & \\ \hline & & & S_{mc} \\ \hline & & & S_{sc} \\ \hline \end{array} \tag{25}$$

Applying equations (13–17) to the multicommodity problem, one can take advantage of the special structure of matrix A, especially when solving $(ASA^t)dy = b_3 + ASr$. In this case, and considering (21) and (25), the structure of matrix ASA^t is as follows:

$$ASA^t = \begin{array}{|c|c|c|c|c|c|}\hline A^*S_1A^{*^t} & \mathbf{0} & \dots & \mathbf{0} & A^*S_1 & A^*S_1T_1^t \\ \hline \mathbf{0} & A^*S_2A^{*^t} & \dots & \mathbf{0} & A^*S_2 & A^*S_2T_2^t \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline \mathbf{0} & \mathbf{0} & \dots & A^*S_kA^{*^t} & A^*S_k & A^*S_kT_k^t \\ \hline S_1A^{*^t} & S_2A^{*^t} & \dots & S_kA^{*^t} & \sum_{i=1}^k S_i + S_{mc} & \sum_{i=1}^k S_iT_i^t \\ \hline T_1S_1A^{*^t} & T_2S_2A^{*^t} & \dots & T_kS_kA^{*^t} & \sum_{i=1}^k T_iS_i & S_{sc} + \sum_{i=1}^k T_iS_iT_i^t \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|}\hline B & C_1 & C_2 \\ \hline C_1^t & D_1 & D_2 \\ \hline C_2^t & D_3 & D_4 \\ \hline \end{array} = \begin{array}{|c|c|}\hline B & C \\ \hline C^t & D \\ \hline \end{array} \tag{26}$$

4 NUMERICAL SOLUTION SCHEME

Because of the structure of $A^*S_iA^{*^t}$ and that of A^*S_i, when a solution for system (13) is attempted directly using the Cholesky decomposition, submatrix D_1 become completely dense. Since the dimension of D_1 is n^* this would mean having to store and process $n^*(n^*+1)/2$ values. For large networks this amount of memory can become prohibitive. This is stated in Choi and Goldfarb (1990), but no procedure is given there to circumvent this difficulty. The algorithm developed considers the solution of the linear system (13) ($ASA^tdy = \bar{b}$) taking into account the partition indicated in (26); thus the system to be solved can be written as:

$$\begin{array}{|c|c|}\hline B & C \\ \hline C^t & D \\ \hline \end{array} \quad \begin{array}{|c|}\hline dy_1 \\ \hline dy_2 \\ \hline \end{array} = \begin{array}{|c|}\hline \bar{b}_1 \\ \hline \bar{b}_2 \\ \hline \end{array}$$

whose solution is directly obtained by block multiplication:

$$\begin{aligned}(D - C^t B^{-1} C) dy_2 &= (\bar{b}_2 - C^t B^{-1} \bar{b}_1) \\ B dy_1 &= (\bar{b}_1 - C dy_2)\end{aligned} \tag{27}$$

B is made of k diagonal blocks $A^* S_i A^{*^t}$. Each block has the same very sparse topological structure of nonzero elements. If $\mathcal{A}$ is the set of arcs of the network, and $\mathcal{I}_v$ the set of incident arcs to node v, $A^* S_i A^{*^t}$ can be computed as follows:

$$A^* S_i A^{*^t} = \underset{\substack{v=1,\ldots,m^* \\ w=1,\ldots,m^*}}{(a_{vw})} = \begin{cases} \sum\limits_{\forall a} - S_{i_{(a)}} & \text{if } a \equiv (v,w) \in \mathcal{A} \text{ and } (w,v) \notin \mathcal{A} \\ \sum\limits_{\forall a,b} \big(- S_{i_{(a)}} - S_{i_{(b)}} \big) & \text{if } a \equiv (v,w) \in \mathcal{A} \text{ and } b \equiv (w,v) \in \mathcal{A} \\ \sum\limits_{\forall a \in \mathcal{I}_v} S_{i_{(a)}} & \text{if } (v = w) \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

and any solution having B as the system matrix can be decomposed in k systems of equations (thus the process could be parallelized). The minimum order degree algorithm was used to reorder the nodes of the network to avoid fill-in when making the Cholesky decomposition of the k blocks of B. Neither the calculation of $B^{-1}\bar{b}_1$ nor the solution of $B dy_1 = (\bar{b}_1 - C dy_2)$ involves too much work. However, when computing dy_2, matrix $\widehat{D} = D - C^t B^{-1} C$ should be formed, which would mean solving $n^* + t$ systems of equations to obtain $B^{-1}C$ and, afterwards, a Cholesky decomposition of $\widehat{D}$.

A better choice is to use a preconditioned conjugate gradient (PCG) algorithm to obtain dy_2, and then to compute dy_1 directly. In the PCG algorithm the only operation directly made with the system matrix is a product of it by a vector v. But we have: $(D - C^t B^{-1} C)v = Dv - C^t B^{-1} w$, with $w = Cv$. Thus one can take advantage of the sparsity of D and C, and the fact that the computation of $B^{-1}w$ is numerically efficient.

In order to speed up the PCG algorithm, a positive definite matrix M must be determined such that $M^{-1}\widehat{D}$ becomes less ill-conditioned than $\widehat{D}$, and that the computation of $Mz = r$ does not involve too much work. For pure multicommodity network problems (without side constraints) the system matrix is $\widehat{D}_1 = D_1 - C_1^t B^{-1} C_1$. In this case, and considering a splitting of matrix $\widehat{D}_1$ such that $\widehat{D}_1 = P - Q$, where $P = D_1$ and $Q = C_1^t B^{-1} C_1$ (both positive definite), it can be proved (Castro (1995b)) that:

$$\widehat{D}_1^{-1} = \big(\sum_{i=0}^{\infty} (P^{-1}Q)^i \big) P^{-1} \tag{29}$$

The preconditioner M^{-1} to be used will be an approximation of $\widehat{D}_1^{-1}$, and can be obtained by truncating (29) at some term ϕ:

$$M^{-1} = (\mathbb{1} + (P^{-1}Q) + (P^{-1}Q)^2 + \ldots + (P^{-1}Q)^{\phi-1})P^{-1} \tag{30}$$

The higher ϕ is, the better the preconditioning, and the fewer iterations of the PCG will be required. However, it must be noted that the product of Q by a vector r implies the solution of $B^{-1}(C_1 r)$, and this should be performed at each iteration of the PCG algorithm, increasing the execution time considerably. Thus ϕ must be chosen in order to balance both objectives: to decrease the PCG iterations and to improve the time per PCG iteration. Various tests have shown than, in general, the best results are obtained with $\phi = 1$. In this case $M^{-1} = P^{-1} = D_1^{-1}$, which means that $z = M^{-1}r$ can be obtained in $O(n)$ operations (since D_1 is a diagonal matrix).

5 COMPUTATIONAL RESULTS

The multicommodity primal-dual interior point algorithm outlined in the above sections was implemented for the case of problems without side constraints, using the preconditioning previously stated. The code was written in ANSI-C, and to test its performance four types of problems, obtained from different network generators, were used: Rmfgen, Grid-on-torus, Gridgraph and Gridgen (Dimacs (1991)). These generators do not consider the case of multicommodity flows, and the output networks had to be converted to a multicommodity one. The conversion algorithm is described in Castro (1995b). Five particular instances were created with each of these generators. The first two are problems with few commodities and medium-sized networks, whereas the last three correspond to small-sized networks with many commodities. Each problem will be denoted by $\mathrm{L}_j^{i)}$, i=1,...,4, j=1,...,5, i denoting the generator employed (1 for Rmfgen, 2 for Grid-on-torus, 3 for Gridgraph and 4 for Gridgen). Table 1 presents the characteristics of each problem, showing for each test problem the number of commodities, nodes and arcs of the network, and the total number of constraints and variables of the linear program to be solved (columns *Rows A* and *Columns A*).

The interior point multicommodity code developed (denoted by IPM) was compared with MINOS 5.3 (Murtagh and Saunders (1983)), a general-purpose package, PPRN (Castro and Nabona (1995)) and MCNF85 (Kennington (1979)), two specialized multicommodity network flow codes, and LoQo (Vanderbei (1993)), a state-of-the-art primal-dual interior point code. Table 2 shows the CPU seconds required by each code. The fastest execution for each test is marked with an asterisk (*). All runs were carried out on a SunSparc 10/41 (one CPU), with a 40MHz clock, ≈100Mips and ≈20Mflops CPU,

Table 1 Linear test problems

Test	*Commodities*	*Nodes*	*Arcs*	*Rows A*	*Columns A*
$L_1^{1)}$	8	2048	9472	25856	85248
$L_2^{1)}$	16	2048	9472	42240	161024
$L_3^{1)}$	50	128	496	6896	25296
$L_4^{1)}$	150	128	496	19696	74896
$L_5^{1)}$	200	128	496	26096	99696
$L_1^{2)}$	8	1500	9000	21000	81000
$L_2^{2)}$	16	1500	9000	33000	153000
$L_3^{2)}$	50	100	600	5600	30600
$L_4^{2)}$	150	100	600	15600	90600
$L_5^{2)}$	200	100	600	20600	120600
$L_1^{3)}$	8	2502	5000	25016	45000
$L_2^{3)}$	16	2502	5000	45032	85000
$L_3^{3)}$	50	227	450	11800	22950
$L_4^{3)}$	150	227	450	34500	67950
$L_5^{3)}$	200	227	450	45850	90450
$L_1^{4)}$	8	976	7808	15616	70272
$L_2^{4)}$	16	976	7808	23424	132736
$L_3^{4)}$	50	101	606	5656	30906
$L_4^{4)}$	150	101	606	15756	91506
$L_5^{4)}$	200	101	606	20806	121806

and 64Mbytes of main memory. From Table 2 it can be concluded that the performance of IPM increases with the size of the problem, this code thus being a good choice for large multicommodity network flow problems, especially for the case of small-sized networks with many commodities.

6 REFERENCES

Castro, J. (1995a) An implementation of a primal-dual interior point algorithm with upper bounded variables. *Qüestiió*, **19**, to appear. (written in Catalan)

Castro, J. (1995b) *Efficient methods for the solution of multicommodity network flow problems.* Ph.D. dissertation, Statistics and Operations Reseach Dept., Universitat Politècnica de Catalunya, Barcelona, Spain. (written in Catalan)

Table 2 CPU seconds of each code for the linear test problems

Test	IPM	MINOS	PPRN	MCNF85	LoQo
$L_1^{1)}$	7095.4	15147.7	737.9*	1778.2	(*d*)
$L_2^{1)}$	16737.9	(*a*)	6838.7	5651.3*	(*d*)
$L_3^{1)}$	178.6*	2639.4	275.1	398.6	3402.8
$L_4^{1)}$	1839.9*	(*b*)	8069.0	11319.0	(*d*)
$L_5^{1)}$	1710.0*	(*c*)	15415.3	26479.9	(*d*)
$L_1^{2)}$	12296.3	(*b*)	4962.2	4833.0*	(*d*)
$L_2^{2)}$	(*d*)	(*c*)	37470.5	34383.0*	(*d*)
$L_3^{2)}$	287.0	1402.8	169.2*	466.9	5211.1
$L_4^{2)}$	4352.4*	105082.5	7605.5	15836.2	(*d*)
$L_5^{2)}$	11974.4*	(*c*)	22218.4	81903.5	(*d*)
$L_1^{3)}$	2818.3	(*b*)	1409.2*	2134.1	(*d*)
$L_2^{3)}$	16485.5	(*c*)	14139.8*	14709.9	(*d*)
$L_3^{3)}$	236.3*	3172.8	364.9	533.9	3180.4
$L_4^{3)}$	962.9 *	(*a*)	4480.5	6142.8	(*d*)
$L_5^{3)}$	2083.1*	(*a*)	11736.8	19458.0	(*d*)
$L_1^{4)}$	12216.2	205836.0	3424.7*	(*e*)	(*d*)
$L_2^{4)}$	(*d*)	(*c*)	40974.1*	(*e*)	(*d*)
$L_3^{4)}$	114.1	918.2	39.4*	(*e*)	(*d*)
$L_4^{4)}$	584.8	15236.6	415.8*	(*e*)	(*d*)
$L_5^{4)}$	915.2*	(*c*)	1273.4	(*e*)	(*d*)

(*a*) Too many constraints. (*b*) Error during execution. (*c*) Execution too long.
(*d*) Not enough memory. (*e*) Feasibility error.

Castro, J. and Nabona, N. (1995) An implementation of linear and nonlinar multicommodity network flows. Accepted for publication in the *European Journal of Operational Research*.

Choi, I.C. and Goldfarb, D. (1990) Solving multicommodity network flow problems by an interior point method. *SIAM Proceedings in Applied Mathematics*, **46**, 58–69.

DIMACS. (1991) The first DIMACS international algorithm implementation challenge: The bench-mark experiments. Technical Report, DIMACS, New Brunswick, NJ.

Kamath, A.P, Karmarkar, N.K. and Ramakrishnan, K.G. (1993) Computational and Complexity Results for an Interior Point Algorithm on Multicommodity Flow Problems. TR-21/93, Dipartimento di Informatica, Università di Pisa, Italy.

Kennington, J.L. and Helgasson, R.V. (1980) *Algorithms for Network Programming*. John Wiley & Sons, Inc., New York, NY.

Kennington, J.L. (1979) A primal partitioning code for solving multicommodity flow problems (version 1). Technical Report 79008. Dept. of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, USA.

Murtagh, B.A. and Saunders, M.A. (1983) MINOS 5.0. User's guide. Dept. of Operations Research, Stanford University, CA, USA.

Vanderbei, R.J. and Carpenter, T.J. (1993) Symmetric indefinite systems for interior point methods. *Mathematical Programming*, **58**, 1-32.