QÜESTIIÓ, Vol 19, 1, 2, 3 pp. 233-257, 1995

IMPLEMENTACIÓ D'UN ALGORISME PRIMAL-DUAL DE PUNT INTERIOR AMB FITES SUPERIORS A LES VARIABLES

J. CASTRO*

Universitat Politècnica de Catalunya

Aquest document presenta una implementació d'un algorisme primal-dual de punt interior, que permet considerar variables afitades superiorment. A diferència d'altres codis que obtenen la direcció del mètode de Newton a cada iteració usant tècniques de matrius simètriques indefinides, la implementació feta descompon el procés fins arribar a un sistema simètric i definit positiu. Això permet que es pugui fer una factorització simbòlica prèvia per tal de conèixer la topologia de la descomposició de Cholesky que s'efectuarà a cada iteració, agilitzant així el procés de càlcul. El codi desenvolupat es compara amb dos paquets de programació lineal capdavanters: Minos 5.3 (una implementació eficient de l'algorisme del símplex) i LoQo (codi de punt interior basat també en un algorisme primal-dual). Els problemes resolts en la comparació pertanyen al conjunt de problemes de la Netlib (una bateria estàndard de problemes de programació lineal).

An implementation of a primal-dual interior point algorithm with upper-bounded variables.

Key words: Algorisme Primal-Dual, Mètodes de Barrera Logarísmica, Mètodes de Punt Interior, Programació Lineal, Resultats Computacionals.

*J. Castro. Dept. d'Estadística i Investigació Operativa, Universitat Politècnica de Catalunya. c. Pau Gargallo 5, 08028 Barcelona, Spain.

- Article rebut el desembre de 1994.
- Acceptat el juny de 1995.

1. INTRODUCCIÓ

Des de l'aparició el 1984 d'un algorisme basat en transformacions projectives a Karmarkar(1984) per resoldre problemes de programació lineal (anomenat també algorisme de Karmarkar), es va iniciar una forta activitat de recerca dins del camp dels anomenats mètodes de punt interior. Aquests, a diferència del fins llavors majoritàriament emprat algorisme del símplex, arriben al punt òptim a través de l'interior de la zona factible delimitada pel conjunt de constriccions, en comptes de bellugar-se per la frontera com ho feien els iterats obtinguts amb el símplex. L'algorisme de les transformacions projectives, però, va ser precedit per l'algorisme de l'el.lipsoid (Khachiyan(1979)), també de complexitat polinòmica, encara que mai va ser un competidor del símplex des del punt de vista pràctic. Des d'aquell article inicial d'en Karmarkar nous mètodes van aparèixer, entre ells els anomenats d'escalat afí, on la transformació projectiva de l'algorisme de Karmarkar era substituïda per una transformació afí (Barnes(1986), Monma i Morton(1987), Vanderbei *et al*(1986)). Posteriorment van aparèixer els anomenats mètodes primal-dual (Monteiro i Adler(1989)), basats en la substitució de les constriccions de no-negativitat de les variables per una barrera logarísmica (Wright(1991)). Els mètodes de punt interior han guanyat un ampli reconeixement com a procediment d'optimització per a problemes lineals en els darrers anys, i s'han mostrat força més eficients que l'algorisme del símplex.

El codi presentat en aquest document implementa un mètode primal-dual de punt interior. Malgrat ser, des d'un punt de vista formal, més complexos que els mètodes d'escalat afí, els mètodes primal-dual s'han mostrat computacionalment més eficients que aquells. La implementació feta tracta les fites superiors a les variables de forma explícita, sense afegir constriccions addicionals. Això fa que no incrementem la dimensió del sistema simètric i definit positiu a solucionar a cada iteració (donat que aquesta dimensió és exactament el nombre de constriccions del nostre problema). A més, el fet de que la topologia d'aquest sistema sigui sempre la mateixa a cada iteració, permet usar rutines de factorització simbòlica per tal de determinar a priori el nombre i la situació dels elements nous no-zero creats en fer la descomposició de Cholesky. Aquesta factorització simbòlica caldrà fer-la només una vegada al principi de l'execució, i serà aprofitada posteriorment a cada iteració. En aquest punt el codi difereix clarament respecte altres implementacions capdavanteres d'algorismes primal-dual (com ara Vanderbei(1992), Vanderbei(1994)), que solucionen un sistema simètric i indefinit mitjançant una factorització de Bunch-Parlett (Duff *et al*(1986), Vanderbei i Carpenter(1993)), que, en teoria, ha de ser lleugerament menys eficient que una factorització de Cholesky coneixent el patró d'esparsitat.

El present document s'estructurarà de la següent forma. En primer lloc es detallaran les bases de l'algorisme primal-dual amb fites superiors a algunes variables. A continuació es comentaran certs aspectes referents a detalls de la implementació

feta. Finalment el codi desenvolupat es compararà amb dos codis capdavanters d'optimització lineal, usant una bateria estàndard de 80 problemes de programació lineal.

2. L'ALGORISME PRIMAL-DUAL CONSIDERANT FITES SUPERIORS A LES VARIABLES

En aquesta secció es farà un breu esment del mètode primal-dual usat, una especialització del mètode general per a problemes amb fites superiors en algunes variables. No es pretén, però, aprofondir en els aspectes de convergència ni justificar el mètode. Per més detalls sobre els mètodes primal-dual hom pot consultar Arbel(1993) i Monteiro i Adler(1989).

2.1. Formulació dels problemes primal i dual

Considerem el següent problema de minimització amb fites superiors a algunes variables

$$(1)\qquad \begin{aligned} &\min\ c^t x \\ &\text{subj. a } Ax = b \\ &\qquad 0 \le x_u \le \bar{x}_u \\ &\qquad 0 \le x_l \end{aligned}$$

on $x_u \in \mathbb{R}^{n_u}$, $x_l \in \mathbb{R}^{n_l}$, $x = (x_u^t, x_l^t)^t$, $x \in \mathbb{R}^n$ (llavors $n = n_u + n_l$), $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ i $A \in \mathbb{R}^{m \times n}$. Considerant un particionament escaient de c i A, i afegint folgues $f \in \mathbb{R}^{n_u}$ per als límits superiors, el problema anterior pot ser escrit com:

$$(2)\qquad \begin{aligned} &\min\ c_u^t x_u + c_l^t x_l \\ &\text{subj. a } A_u x_u + A_l x_l = b \\ &\qquad x_u + f = \bar{x}_u \\ &\qquad x \ge 0 \\ &\qquad f \ge 0 \end{aligned}$$

on $c_u \in \mathbb{R}^{n_u}$, $c_l \in \mathbb{R}^{n_l}$, $A_u \in \mathbb{R}^{m \times n_u}$ i $A_l \in \mathbb{R}^{m \times n_l}$.

Considerant les variables duals $y \in \mathbb{R}^m$, $u \in \mathbb{R}^{n_u}$, i les folgues $z = (z_u^t, z_l^t)^t$ ($z_u \in \mathbb{R}^{n_u}$, $z_l \in \mathbb{R}^{n_l}$, per tant $z \in \mathbb{R}^n$) i $w \in \mathbb{R}^{n_u}$, el problema dual associat a (2) pot ser escrit com:

$$
\begin{aligned}
&\max\ b^t y + \bar{x}_u^t u \\
&\text{subj. a } A_u^t y + u + z_u = c_u \\
&\qquad\qquad A_l^t y + z_l = c_l \\
&\qquad\qquad u + w = 0 \\
&y \in \mathbb{R}^m \quad u \in \mathbb{R}^{n_u} \\
&z_l \geq 0 \quad z_u \geq 0 \quad w \geq 0
\end{aligned} \tag{3}
$$

De la tercera constricció es té que $u = -w$ i, aleshores, eliminant u el problema dual pot ser formulat com:

$$
\begin{aligned}
&\max\ b^t y - \bar{x}_u^t w \\
&\text{subj. a } A_u^t y + z_u - w = c_u \\
&\qquad\qquad A_l^t y + z_l = c_l \\
&y \in \mathbb{R}^m \\
&z_l \geq 0 \quad z_u \geq 0 \quad w \geq 0
\end{aligned} \tag{4}
$$

2.2. Obtenció dels Lagrangians a través una barrera logarísmica

Als problemes primal i dual prèviament definits les restriccions de no-negativitat de les variables poden ser substituïdes per una penalització de barrera logarísmica (tal i com descriu Wright(1991)). Llavors el problema primal (2) pot ser expressat com:

$$
\begin{aligned}
&\min\ c_u^t x_u + c_l^t x_l - \mu \textstyle\sum_{i=1}^{n} \ln x_i - \mu \sum_{i=1}^{n_u} \ln(\bar{x}_{u_i} - x_{u_i}) \\
&\text{subj. a } A_u x_u + A_l x_l = b \\
&\qquad \mu \geq 0
\end{aligned} \tag{5}
$$

mentre que el dual (4) pot formular-se tal i com segueix:

$$
\begin{aligned}
&\max\ b^t y - \bar{x}_u^t w + \mu \textstyle\sum_{i=1}^{n} \ln z_i + \mu \sum_{i=1}^{n_u} \ln w_i \\
&\text{subj. a } A_u^t y + z_u - w = c_u \\
&\qquad\qquad A_l^t y + z_l = c_l \\
&y \in \mathbb{R}^m \quad \mu \geq 0
\end{aligned} \tag{6}
$$

A continuació, associant els multiplicadors de Lagrange $y \in \mathbb{R}^m$ a les constriccions d'igualtat de (5) construim el Lagrangià L_p del problema primal:

$$(7)\quad L_p(x,y,\mu) = c_u^t x_u + c_l^t x_l - \mu \sum_{i=1}^{n} \ln x_i - \mu \sum_{i=1}^{n_u} \ln(\bar{x}_{u_i} - x_{u_i}) - y^t(A_u x_u + A_l x_l - b)$$

Anàlogament, associant els multiplicadors $x_u \in \mathbb{R}^{n_u}$ i $x_l \in \mathbb{R}^{n_l}$ a les constriccions d'igualtat de (6) obtenim L_d, el Lagrangià del dual:

$$\begin{aligned}(8)\quad L_d(x,y,z,w,\mu) &= b^t y - \bar{x}_u^t w + \mu \sum_{i=1}^{n} \ln z_i + \\ &+ \mu \sum_{i=1}^{n_u} \ln w_i - x_u^t(A_u^t y + z_u - w - c_u) - x_l^t(A_l^t y + z_l - c_l)\end{aligned}$$

2.3. Condicions d'optimalitat de Kuhn-Tucker de primer ordre

Si denotem per e_l el vector l-dimensional de 1's i considerem X_u, X_l, X, Z_u, Z_l, Z, W, F matrius diagonals, on:

$$\begin{aligned} e_l &= (1_1,\ldots,1_l)^t \\ X_u &= \operatorname{diag}(x_{u_1},\ldots,x_{u_{n_u}}) \\ X_l &= \operatorname{diag}(x_{l_1},\ldots,x_{l_{n_l}}) \\ X &= \begin{pmatrix} X_u & 0 \\ 0 & X_l \end{pmatrix} \\ (9)\qquad Z_u &= \operatorname{diag}(z_{u_1},\ldots,z_{u_{n_u}}) \\ Z_l &= \operatorname{diag}(z_{l_1},\ldots,z_{l_{n_l}}) \\ Z &= \begin{pmatrix} Z_u & 0 \\ 0 & Z_l \end{pmatrix} \\ W &= \operatorname{diag}(w_1,\ldots,w_{n_u}) \\ F &= \operatorname{diag}(\bar{x}_{u_1} - x_{u_1},\ldots,\bar{x}_{u_{n_u}} - x_{u_{n_u}}) \end{aligned}$$

llavors les condicions d'optimalitat necessàries de primer ordre de L_p poden ser escrites com:

$$(10)\quad \frac{\partial L_p}{\partial x_u} = c_u - \mu X_u^{-1} e_{n_u} + \mu F^{-1} e_{n_u} - A_u^t y := 0$$

$$(11)\quad \frac{\partial L_p}{\partial x_l} = c_l - \mu X_l^{-1} e_{n_l} - A_l^t y := 0$$

$$(12)\quad \frac{\partial L_p}{\partial y} = -(A_u x_u + A_l x_l - b) := 0$$

Per la seva banda, les condicions d'optimalitat de primer ordre del Lagrangià dual (8) són:

$$\frac{\partial L_d}{\partial x_u} = c_u - A_u^t y - z_u + w := 0 \tag{13}$$

$$\frac{\partial L_d}{\partial x_l} = c_l - A_l^t y - z_l := 0 \tag{14}$$

$$\frac{\partial L_d}{\partial y} = b - A_u x_u - A_l x_l := 0 \tag{15}$$

$$\frac{\partial L_d}{\partial z} = \mu Z^{-1} e_n - X e_n := 0 \tag{16}$$

$$\frac{\partial L_d}{\partial w} = \mu W^{-1} e_{n_u} - F e_{n_u} := 0 \tag{17}$$

Les condicions (12) i (15) són la mateixa, i imposen la *factibilitat primal* a l'òptim. Les condicions (13) i (14) imposen la *factibilitat dual* de la solució. Les condicions (16) i (17) (usant ()) poden ser reescrites com:

$$\left.\begin{aligned} Z_u X_u e_{n_u} &= \mu e_{n_u} \\ Z_l X_l e_{n_l} &= \mu e_{n_l} \end{aligned}\right\} \tag{18}$$

$$F W e_{n_u} = \mu e_{n_u} \tag{19}$$

i les anomenarem condicions de *complementarietat* (això degut a que, quan $\mu \to 0$ tenim que (18) i (19) són exactament les condicions del teorema de la folga complementària, les quals han de ser garantides per la solució del problema primal i dual). Finalment, es pot veure fàcilment que, verificant-se la factibilitat primal, la factibilitat dual i la complementarietat, automàticament queden satisfetes les dues condicions restants (10) i (11):

• pel que fa a (10) veiem que:

$$\begin{aligned} 0 &\stackrel{?}{=} c_u - \mu X_u^{-1} e_{n_u} + \mu F^{-1} e_{n_u} - A_u^t y \\ &= A_u^t y + z_u - w - \mu X_u^{-1} e_{n_u} + \mu F^{-1} e_{n_u} - A_u^t y \quad \text{[usant (13)]} \\ &= Z_u e_{n_u} - W e_{n_u} - \mu X_u^{-1} e_{n_u} + \mu F^{-1} e_{n_u} \quad \text{[usant (9)]} \\ &= (Z_u e_{n_u} - \mu X_u^{-1} e_{n_u}) + (\mu F^{-1} e_{n_u} - W e_{n_u}) \\ &= 0 \quad \text{[usant (18) i (19)]} \end{aligned}$$

• per la seva banda, per (11) observem que:

$$\begin{aligned} 0 &\stackrel{?}{=} c_l - \mu X_l^{-1} e_{n_l} - A_l^t \\ &= A_l^t y + z_l - \mu X_l^{-1} e_{n_l} - A_l^t y \quad \text{[usant (14)]} \\ &= Z_l e_{n_l} - \mu X_l^{-1} e_{n_l} \quad \text{[usant (9)]} \\ &= 0 \quad \text{[usant (8)]} \end{aligned}$$

Per tant, les condicions finals de Kuhn-Tucker de primer ordre que ha de verificar un punt per ser considerat òptim del nostre problema seran les sis següents:

$$X_u Z_u e_{n_u} = \mu e_{n_u} \tag{20}$$

$$X_l Z_l e_{n_l} = \mu e_{n_l} \tag{21}$$

$$A_u x_u + A_l x_l = b \tag{22}$$

$$A_l^t y + z_l = c_l \tag{23}$$

$$FWe_{n_u} = \mu e_{n_u} \tag{24}$$

$$A_u^t y + z_u - w = c_u \tag{25}$$

Clarament, quan $n_u = 0$ ($n = n_l$, és a dir, no hi ha cap variable amb fita superior) només cal considerar les equacions (21, 22 i 23), les quals són, efectivament, les condicions d'optimalitat del mètode primal-dual estàndard sense fites a les variables (com es pot veure a Arbel(1993)).

2.4. Solució del sistema no-lineal

El sistema no-lineal d'equacions (20–25) resultant es resoldrà usant el mètode de Newton, on el sistema lineal d'equacions $J^i d^i = -f^i$ —essent J^i el jacobià del sistema, d^i la direcció de Newton i f^i l'avaluació del sistema al punt actual— a ser resolt a cada iteració i és:

(26)

$$\begin{array}{|cccccc|}
\hline
Z_n & & & X_u & & \\
 & Z_l & & & X_l & \\
A_u & A_l & & & & \\
 & & A_l^t & & \mathbb{I}_{n_l} & \\
-W & & & & & F \\
 & & A_u^t & \mathbb{I}_{n_u} & & -\mathbb{I}_{n_u} \\
\hline
\end{array}
\begin{array}{|c|}
\hline
dx_u \\
dx_l \\
dy \\
dz_u \\
dz_l \\
dw \\
\hline
\end{array}
=
\begin{array}{|c|}
\hline
\mu e_{n_u} - X_u Z_u e_{n_u} \\
\mu e_{n_l} - X_l Z_l e_{n_l} \\
b - Ax \\
c_l - A_l^t y - z_l \\
\mu e_{n_u} - WFe_{n_u} \\
c_u - A_u^t y - z_u + w \\
\hline
\end{array}$$

Definint els nous vectors (que representen els termes independents del sistema lineal anterior):

$$b_{1_l} = \mu e_{n_l} - X_l Z_l e_{n_l} \tag{27}$$

$$b_{1_u} = \mu e_{n_u} - X_u Z_u e_{n_u} \tag{28}$$

$$b_2 = \mu e_{n_u} - WFe_{n_u} \tag{29}$$

$$b_3 = b - Ax \tag{30}$$

$$b_{4_l} = c_l - A_l^t y - z_l \tag{31}$$

$$b_{4_u} = c_u - A_u^t y - z_u + w \tag{32}$$

s'arriba a que la solució de (26) ve donada per:

$$(ASA^t)dy = b_3 + ASr \tag{33}$$

$$dx = S(A^t dy - r) \tag{34}$$

$$dw = F^{-1}(b_2 + W dx_u) \tag{35}$$

$$dz_u = b_{4_u} + dw - A_u^t dy \tag{36}$$

$$dz_l = b_{4_l} - A_l^t dy \tag{37}$$

essent

$$\begin{array}{rcl} r & = & (r_u^t, r_l^t)^t \quad r \in \mathbb{R}^n \quad r_u \in \mathbb{R}^{n_u} \quad r_l \in \mathbb{R}^{n_l} \\ r_u & = & F^{-1}b_2 + b_{4_u} - X_u^{-1}b_{1_u} \quad r_l = b_{4_l} - X_l^{-1}b_{1_l} \end{array} \tag{38}$$

i

$$\begin{array}{rcl} S & = & \begin{pmatrix} S_u & 0 \\ 0 & S_l \end{pmatrix} \quad S \in \mathbb{R}^{n \times n}, \quad S_u \in \mathbb{R}^{n_u \times n_u}, \quad S_l \in \mathbb{R}^{n_l \times n_l u} \\ S_u & = & FX_u(Z_u F + X_u W)^{-1} \qquad S_l = Z_l^{-1} X_l \end{array} \tag{39}$$

on S_u i S_l poden ser calculades directament, donat que no són més que productes i sumes de matrius diagonals. A l'apèndix 1 es pot trobar el procés de reducció de (26) a la seqüència (33–37).

El codi presentat soluciona a cada iteració el sistema (26) a través de (33–37), a diferència d'altres —com Vanderbei(1992)— on (26) es soluciona usant tècniques per matrius simètriques indefinides basades en la factorització de Bunch-Parlett (Duff *et al*(1986), Vanderbei i Carpenter(1993), Vanderbei(1994)).

També cal fer notar que en la solució del nostre sistema cal invertir les matrius diagonals X, Z, W, F definides a (9). Naturalment això només és possible si cap element diagonal és igual a 0. Per tant, per poder garantir l'obtenció de (33–37) cal que les variables primals i duals no arribin mai a les seves fites (inferiors o superiors), és a dir, hem d'obligar a tenir sempre punts estrictament interiors respecte les seves fites.

2.5. Actualització del nou punt i del paràmetre μ de penalització

Un cop s'han calculat les direccions (33–37), cal actualitzar les noves variables primals i duals per a la següent iteració. És a dir, calcularem:

$$
(40) \qquad \begin{aligned} x^{(i+1)} &= x^{(i)} + \alpha_p dx \\ y^{(i+1)} &= y^{(i)} + \alpha_d dy \\ z^{(i+1)} &= z^{(i)} + \alpha_d dz \\ w^{(i+1)} &= w^{(i)} + \alpha_d dw \end{aligned}
$$

on α_p i α_d són tals que preserven el fet de que cada iterat sigui un punt interior (és a dir, α_p manté que $0<x_l^{(i+1)}$ i $0<x_u^{(i+1)}<\bar{x}_u$, mentre que α_d garanteix que $0<z$ i $0<w$).

Només queda fer l'actualització del paràmetre μ de la barrera logarísmica abans de tornar a calcular les direccions (33–37) pel nou iterat calculat prèviament. Com és habitual als mètodes primal-dual, el paràmetre μ s'actualitza en funció de la distància que hi ha al punt actual entre la funció primal i la funció dual (que anomenarem *gap dual*). Al nostre cas concret, tenim que:

$$
\begin{aligned}
\text{[usant (2) i (4)]} \qquad gap\ dual &= c^t x - (b^t y - \bar{x}_u^t w) \\
\text{[usant (2)]} \qquad &= c^t x - x^t A^t y + \bar{x}_u^t w \\
&= (c^t - y^t A)x + \bar{x}_u^t w \\
\text{[usant (4)]} \qquad &= (z_u^t - w^t \quad z_l^t) \begin{pmatrix} x_u \\ x_l \end{pmatrix} + \bar{x}_u^t w \\
&= z_u^t x_u + z_l^t x_l - x_u^t w + \bar{x}_u^t w \\
\text{[usant (2)]} \qquad &= z^t x + f^t w \\
\text{[usant (20), (21) i (24)]} \qquad &= \mu n + \mu n_u \\
&= (n + n_u)\mu \\
&\Downarrow \\
\mu &= \frac{gap\ dual}{(n + n_u)}
\end{aligned}
$$

Per tal de que a la següent iteració es millori el gap dual (garantint la convergència de l'algorisme), es prendrà com nou paràmetre $\mu^{(i+1)}$ una fracció de l'anterior. És a dir:

$$
(41) \qquad \mu^{(i+1)} = \sigma \frac{gap\ dual}{(n + n_u)} \quad \text{on} \quad 0 < \sigma < 1
$$

3. IMPLEMENTACIÓ DE L'ALGORISME

En aquesta secció es detallaran alguns aspectes de la implementació de l'algorisme realitzada. Els cinc punts concrets a tractar seran la forma de solucionar el sistema (33) a cada iteració, com escollir el punt inicial $(x^{(0)}, y^{(0)}, z^{(0)}, w^{(0)})$, com calcular

les passes α_p i α_d de (40), com fer l'actualització del paràmetre μ de la barrera logarísmica i quines són les condicions d'aturada del procés iteratiu de minimització.

3.1. Obtenció de *dy* a cada iteració

El pas més costós, des del punt de vista computacional, de tot l'algorisme és l'obtenció de *dy* a través del sistema (33). Donat que es garanteix que en tot moment el punt actual és interior respecte les fites simples de les variables, tenim que la matriu ASA^t serà simètrica i definida positiva, sempre i quan A sigui de rang complet (és a dir, sempre i quan rang(A)= m). D'igual forma, per a la majoria de problemes la matriu A acostuma a ser força esparsa, i sovint aquesta propietat es manté en construir ASA^t. Per tant sembla raonable aplicar tècniques de resolució de sistemes especialment dissenyades per a matrius simètriques definides positives i esparses (és a dir, s'emprarà una factorització de Cholesky per a matrius esparses i de gran dimensió). Concretament, la implementació realitzada utilitza el paquet SPARSPAK (George i Liu(1981)) dissenyat per aquest tipus de sistemes.

En fer la factorització de Cholesky de ASA^t cal tenir molt present l'ordenació de les seves files (o columnes), donat que una mala ordenació pot donar lloc a que es creïn un gran nombre d'elements no-zero, tot degradant l'esparsitat original de la matriu. Per tant, abans de fer la factorització, cal reordenar les files de ASA^t de forma que garantim la creació del menor nombre possible d'elements. Hi ha dos possibles mètodes a aplicar per tal de trobar aquesta ordenació millor: l'ordenació del *mínim grau* ("mínimum-degree ordering"), i la del *mínim ompliment* ("mínimum-local-fill-in ordering") (Duff *et al*(1986)). Al codi desenvolupat s'ha usat la rutina GENQMD del paquet SPARSPAK, la qual implementa l'algorisme del mínim grau.

Cal també observar que la topologia de la matriu ASA^t no varia durant tota l'execució del programa. Tan sols hi ha una variació en els coeficients de la matriu diagonal S definida a (39). Aquest fet és clau, donat que implica que només cal fer una única reordenació de files mitjançant l'algorisme del mínim grau al principi, en comptes d'una vegada a cada iteració. D'igual forma també es pot explotar el fet de tenir una topologia constant fent una factorització simbòlica prèvia del sistema. Això ens proporciona el patró d'esparsitat del sistema un cop ja factoritzat, i agilitza clarament el posterior procés de càlcul.

Per tal de calcular de forma eficient ASA^t (només la part sub i diagonal, però, ja que és una matriu simètrica) el codi usa una variació de la idea proposada a Monma i Morton(1987). Considerem que la part subdiagonal de ASA^t s'emmagatzema per columnes al vector `lnz(maxlnz)`, on `maxlnz` ès el nombre d'elements no nuls de la part subdiagonal de ASA^t un cop realitzada la factorització simbòlica, mentre que els elements diagonals es troben al vector `diag(m)`. El nombre d'elements que ori-

ginalment es tenen a ASA^t (abans de factoritzar) correspon al nombre de parells de files de A ($A_{i_1,.}$,$A_{i_2,.}$) tal que alguna variable x_j apareix amb un coeficient no nul en ambdues constriccions i_1 i i_2 (és a dir $\exists\ j : a_{i_1,j} \neq 0$ i $a_{i_2,j} \neq 0$). Considerem que el nombre de parells de files que donen lloc a un element no nul de la part sub i diagonal de ASA^t és `np`. Llavors tindrem un vector `ilnz(np)` que ens dirà per a cada parell de files amb elements coincidents quina posició ocupa dins els vectors `lnz(.)` o `diag(.)`, tenint en compte la informació de la factorització simbòlica prèviament feta. Per tal de que `ilnz(.)` mantingui una única numeració, els vectors `lnz(.)` i `diag(.)` s'emmagatzemen a memòria de forma consecutiva, dins un vector que podem anomenar `asa(maxlnz+m)`, i llavors aquelles posicions i tals que $1 \leq$ `ilnz(i)` $\leq$ `maxlnz` correspondran a la part subdiagonal, mentre que aquelles on `maxlnz` $<$ `ilnz(i)` $\leq$ `maxlnz+m` estaran associades a la part diagonal. També cal un vector `ifillin(nfill)`, on `nfill` és el nombre d'elements no-zero creats a la part subdiagonal en factoritzar (és a dir, `nfill` $=$ `maxlnz` $-$ (`np` $-$ `m`)). Aquest vector ens dóna les posicions dins `lnz(.)` dels nous elements creats, les quals han de ser inicialitzades a zero a cada iteració. A més, disposarem de tres vectors més anomenats `ka(np+1)`, `la(.)` i `va(.)`. El vector `ka(.)` per cada parell de files (i_1,i_2) de A que donen lloc a un element no-zero té un punter als vectors `la(.)` i `va(.)`. El primer d'aquests, `la(.)`, ens diu quines són les columnes j amb elements no-zero a les dues files i_1 i i_2 que estem considerant, mentre que `va(.)` ens dóna directament el producte $a_{i_1,j} \cdot a_{i_2,j}$. El nombre total de columnes amb elements no-zero d'un parell de files es troba fent `ka(i+1)` $-$ `ka(i)`, essent i l'entrada associada a aquesta parella de files dins el vector `ka(.)`. Per exemple, si considerem la matriu de constriccions A següent:

$$A = \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} & 3 & 2 & 1 & & \\ 3 & & 5 & & 1 & \\ 2 & 3 & -1 & & & 1 \end{pmatrix} \end{array}$$

on `m` $= 3$, `np` $= 6$, `maxlnz` $= 3$ i `nfill` $= 0$, tenim que els vectors anteriorment definits són:

(i)	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ka(i)	=	1	4	5	7	10	12	16								
la(i)	=	2	3	4	3	2	3	1	3	5	1	3	1	2	3	6
va(i)	=	9	4	1	10	9	−2	9	25	1	6	−5	4	9	1	1
ilnz(i)	=	4	1	2	5	3	6									

Amb aquesta estructura de dades, considerant que la matriu diagonal S es troba al vector `s(n)` de dimensió nombre de columnes de A, la creació de la part sub i

diagonal de ASA^t es redueix simplement a fer:

```
per i  = 1 fins nfill fer
         asa(ifillin(i)) = 0.0
fi-per
per i  = 1 fins np fer
         acum = 0.0
         per j  = ka(i) fins ka(i + 1) − 1 fer
                  acum = acum + va(j)· s(la(j))
         fi-per
asa(ilnz(i)) = acum
fi-per
```

Tot i que la matriu ASA^t es suposa definida positiva, podria ser que, si A no és de rang complet, s'arribés a tenir pivots diagonals amb valor nul en fer la descomposició de Cholesky. Aquest fet podria ser evitat si s'usés pivotació parcial en escollir el pivot, però això ens modificaria l'esparsitat de la descomposició desaprofitant la factorització simbòlica feta prèviament. El que realment s'ha fet, seguint una recomanació de Monma i Morton(1987), ha estat, en comptes de solucionar el sistema $(ASA^t)x = b$, solucionar el sistema escalat $(A(S/\gamma)A^t)x = b/\gamma$, on $\gamma = \max\{S_{ii}, i = 1,\ldots,m\}$. Si en solucionar el nou sistema escalat trobem un pivot nul, automàticament li assignem un valor petit (p.e. 10^{-12}), i continuem la factorització. Això és equivalent a introduir una petita perturbació dins la matriu A de forma que eliminem el fet de tenir alguna fila combinació lineal d'altres. Els resultats obtinguts amb aquesta regla han estat força satisfactoris. Tanmateix, a més d'aquesta simple estratègia, s'han provat d'altres de més acurades, com ara la triangularització de Gill-Murray (Gill, Murray i Wright(1981)), sense haver obtingut un millor resultat.

3.2. Elecció del punt inicial

Un dels punts clau dins de l'algorisme és el càlcul del punt inicial d'iteració. Aquest punt, que ha de ser estrictament interior, s'inicialitza atenent a dos criteris principals. El primer és el de no fixar variables a un valor que estigui a prop de les seves fites, donat que llavors tindríem un punt "poc interior". El segon és que, donat que el punt òptim ha de satisfer les condicions (20–25), interessa que el punt inicial satisfaci ja d'entrada el màxim nombre de condicions d'optimalitat possible. Dit això, el criteri seguit a l'hora de trobar el punt inicial és el següent:

a) Variables x: $x_l = 100$ i $x_u = \min\left\{100, \frac{\bar{x}_u}{2}\right\}$.

b) Variables y: $y = 0$ (y són variables lliures i poden tenir qualsevol valor).

c) Variables z i w: donat que hem fixat $y = 0$, llavors les condicions de factibilitat dual (23) i (25) queden directament com

$$\begin{aligned} z_l &= c_l \\ z_u - w &= c_u \end{aligned}$$

La primera condició es podrà satisfer si $\forall i \; c_{l_i} > 0$, ja que sinó tindríem una component de z_l no-interior. A més, no només interessa que totes les components de c_l siguin positives, sinó que també cal que estiguin lluny de zero per evitar tenir un punt "poc interior". Per tal de garantir això, z_l s'inicialitza com $z_{l_i} = \max\{c_{l_i}, 100\}$. La segona condició sempre la satisfarem inicialitzant w i z_u com segueix:

$$w_i, z_{u_i} = \begin{cases} w_i = 100, z_{u_i} = c_{u_i} + 100 & \text{si } c_{u_i} \geq 0 \\ z_{u_i} = 100, w_i = 100 - c_{u_i} & \text{si } c_{u_i} < 0 \end{cases}$$

3.3. Càlcul de les passes α_p i α_d

Les passes α_p i α_d usades a (40) es calculen de forma que en tot moment es mantinguin els valors de les variables entre fites. A més, sempre que sigui possible fer-ho sense violar les fites de les variables, s'intenta que prenguin el valor de 1, ja que així estaríem resolent el sistema lineal (26) usant la direcció exacta del mètode de Newton. Per tal de garantir ambdós objectius α_p i α_d són calculades com:

$$\alpha_p = \min\left\{\rho \cdot \min\{\frac{-x_i}{dx_i} \;\forall\; i : dx_i < 0\}, \rho \cdot \min\{\frac{\overline{x}_u - x_{u_i}}{dx_{u_i}} \;\forall\; i : dx_{u_i} > 0\}, 1\right\}$$

(42)

$$\alpha_d = \min\left\{\rho \cdot \min\{\frac{-z_i}{dz_i} \;\forall\; i : dz_i < 0\}, \rho \cdot \min\{\frac{-w_i}{dw_i} \;\forall\; i : dw_i < 0\}, 1\right\}$$

on ρ és un valor proper a 1, usat per treure fora de la seva fita la variable que ens proporciona la passa màxima. A la implementació feta ρ=0.99.

3.4. Actualització del paràmetre μ

L'expressió de l'actualització del paràmetre μ a la nova iteració i+1 venia donada per l'equació (41) i era $\mu^{(i+1)} = \sigma \cdot gap\ dual/(n + n_u)$, on σ havia de ser un valor tal que $0 < \sigma < 1$. En general un valor de σ=0.1 acostuma a ser prou satisfactori. Tanmateix

s'ha usat una actualització dinàmica de μ, tal i com es descriu a Vanderbei(1994). Sigui $\alpha_{pd} = \min\{\alpha_p, \alpha_d\}$, α_p i α_d definits a (42), llavors el paràmetre σ es troba a cada iteració com:

$$\sigma = \left(\frac{1 - \alpha_{pd}}{10\alpha_{pd} + 1} \right)^2 \tag{43}$$

Això fa que, quan α_{pd} és proper a 1 (és a dir, no ha estat necessari escurçar o bé α_p o bé α_d a causa de les fites) tenim que σ tendeix a 0, amb el qual disminuirem $\mu^{(i+1)}$. Per l'altra banda, quan α_{pd} s'apropa a 0 (alguna variable està a prop de la seva fita) augmentem $\mu^{(i+1)}$. Donat que el paràmetre μ controla la barrera logarísmica, actualitzant σ d'aquesta forma estem fent que, quan les variables estan lluny de les fites, disminuim la barrera logarísmica, mentre que quan alguna està a prop incrementem el seu pes dins el Lagrangià.

3.5. Condicions d'aturada

Les condicions d'aturada implementades són les habituals dels mètodes primal-dual: un punt es considera òptim quan supera un test de factibilitat primal, un de factibilitat dual i un sobre el *gap dual* entre les funcions objectiu primal i dual. Concretament les tres condicions a complir són:

$$\begin{aligned}
\frac{||Ax - b||}{1 + ||b||_2} &< \varepsilon_p \qquad \text{Factibilitat Primal} \\
\frac{\left\| \begin{pmatrix} A_u^t y + z_u - w \\ A_l^t y + z_l \end{pmatrix} - c \right\|_2}{1 + ||c||_2} &< \varepsilon_d \qquad \text{Factibilitat Dual} \\
\frac{|c^t x - (b^t y - \vec{x}_u^t w)|}{1 + c^t x} &< \varepsilon_g \qquad \text{Gap Dual}
\end{aligned} \tag{44}$$

A la implementació realitzada els valors usats per ε_p, ε_d i ε_g són $\varepsilon_p = \varepsilon_d = 10^{-6}$ i $\varepsilon_g = 10^{-8}$.

3.6. Resultats computacionals

Presentem en aquesta secció els resultats computacionals obtinguts amb la implementació feta de l'algorisme presentat en apartats anteriors. S'han usat un total de 80 problemes de programació lineal pertanyents a la bateria de tests de la Netlib (Gay(1985)). Tots aquests tests han estat agafats a través d'un ftp anònim al compte

Taula I

Característiques dels problemes test

Nom	Constr.	Vars.	No-zeros A
25fv47	822	1571	11127
80bau3b	2263	9799	29063
adlittle	57	97	465
afiro	28	32	88
agg	489	163	2541
agg2	517	302	4515
agg3	517	302	4531
bandm	306	472	2659
beaconfd	174	262	3476
blend	75	83	521
bnl1	644	1175	6129
bnl2	2325	3489	16124
boeing1	351	384	3865
boeing2	167	143	1339
bore3d	234	315	1525
brandy	221	249	2150
czprob	930	3523	14173
d2q06c	2172	5167	35674
d6cube	416	6184	43888
degen2	445	534	4449
degen3	1504	1818	26230
e226	224	282	2767
etamacro	401	688	2489
fffff800	525	854	6235
finnis	498	614	2714
fit1d	25	1026	14430
fit1p	628	1677	10894
fit2d	26	10500	138018
fit2p	3001	13525	60784
ganges	1310	1681	7021
gfrd-pnc	617	1092	3467
greenbea	2393	5405	31499
grow15	301	645	5665
grow22	441	946	8318
grow7	141	301	2633
israel	175	142	2358
kb2	44	41	291
lotfi	154	308	1086
maros	847	1443	10006
maros-r7	3137	9408	151120

Nom	Constr.	Vars.	No-zeros A
nesm	663	2923	13988
pilot	1442	3652	43220
pilot87	2031	4883	73804
pilotnov	976	2172	13129
recipe	92	180	752
sc105	106	103	281
sc205	206	203	552
sc50a	51	48	131
sc50b	51	48	119
scagr25	472	500	2029
scagr7	130	140	553
scfxm1	331	457	2612
scfxm2	661	914	5229
scfxm3	991	1371	7846
scorpion	389	358	1708
scrs8	491	1169	4029
scsd1	78	760	3148
scsd6	148	1350	5666
scsd8	398	2750	11334
sctap1	301	480	2052
sctap2	1091	1880	8124
sctap3	1481	2480	10734
seba	516	1028	4874
share1b	118	225	1182
share2b	97	79	730
shell	537	1775	4900
ship04l	403	2118	8450
ship04s	403	1458	5810
ship08l	779	4283	17085
ship08s	779	2387	9501
ship12l	1152	5427	21597
ship12s	1152	2763	10941
sierra	1228	2036	9252
standata	360	1075	3038
standgub	362	1184	3147
standmps	468	1075	3686
stocfor1	118	111	474
stocfor2	2158	2031	9492
woodlp	245	2594	70216
woodw	1099	8405	37478

netlib.att.com, i es troben dins del directory */netlib/lp/data*. Només s'han usat problemes sense variables lliures a la seva formulació, donat que la implementació feta no contempla aquesta possibilitat. La Taula I mostra les principals característiques dels problemes test usats. La taula presenta per a cada problema el seu nom (columna

"Nom"), el nombre de constriccions del problema (columna "Constr."), el nombre de variables (columna "Vars."), i el nombre d'elements no-zero de la matriu de constriccions (columna "No-zeros A").

Taula II

Resultats obtinguts amb els problemes test

	IP			LoQo			Minos 5.3		
Nom	Valor òptim	Iter.	CPU	Valor òptim	Iter.	CPU	Valor òptim	Iter.	CPU
25fv47	5501.8	45	23.9	5501.8	29	19.8	5501.8	6477	75.8
80bau3b	987224.2	71	61.3	987224.2	44	52.4	987229.0	11158	239.2
adlittle	225495.0	20	0.1	225495.0	14	0.1	225495.0	108	0.3
afiro	464.8	13	0.0	464.8	13	0.0	464.8	14	0.1
agg	35991767.3	43	6.8	35991767.3	26	1.9	35991767.3	158	1.6
agg2	20239252.3	37	9.5	20239252.3	22	4.4	20239252.4	223	2.2
agg3	10312115.9	39	9.8	10312115.9	22	4.4	10312115.9	255	2.2
bandm	158.6	34	1.5	158.6	20	1.6	158.6	474	2.7
beaconfd	33592.5	18	1.3	33592.5	14	1.2	33592.5	95	1.1
blend	30.8	20	0.1	30.8	14	0.2	30.8	115	0.4
bnl1	1977.6	51	7.5	1977.6	35	7.1	1977.6	1166	9.5
bnl2	1811.2	61	124.7	1811.2	40	122.4	1811.2	5441	125.0
boeing1	335.2	44	4.0	335.2	28	3.1	335.2	579	3.0
boeing2	315.0	32	0.8	315.0	28	1.1	315.0	159	0.7
bore3d	1373.1	29	0.9	1373.1	17	0.9	1373.1	118	1.1
brandy	1518.5	29	1.2	1518.5	22	1.5	1518.5	310	1.7
czprob	2185196.7	56	12.8	2185196.7	38	10.9	2185196.7	1506	17.8
d2q06c	122784.2	58	263.6	122784.2	36	247.0	122784.2	43934	1246.4
d6cube	315.5	51	80.8	315.5	23	75.2	315.5	96780	1049.1
degen2	1435.2	26	6.7	1435.2	14	4.7	1435.2	974	6.5
degen3	987.3	38	165.6	987.3	17	73.4	987.3	7109	158.4
e226	18.8	39	1.7	18.8	22	1.4	18.8	455	1.9
etamacro	755.7	49	5.6	755.7	30	8.9	755.7	761	4.1
fffff800	555679.6	78	17.8	555679.6	36	10.6	555679.6	272	3.3
finnis	172791.1	42	2.7	172791.1	26	2.2	172791.0	547	3.4
fit1d	9146.4	56	4.8	9146.4	21	5.3	9146.4	2626	6.4
fit1p	9146.4	26	381.6	9146.4	26	4.8	9146.4	894	13.6
fit2d	68464.3	48	44.7	68464.3	24	195.0	68464.3	32235	291.2
fit2p	*(a)*			68464.3	24	41.1	68464.3	14240	1128.3
ganges	109585.7	41	15.2	109585.7	23	10.7	109585.8	663	9.5
gfrd_pnc	6902236.0	55	1.9	6902236.0	19	1.6	6902236.0	686	5.7
greenbea	*(b)*			72555247.5	47	86.9	72462405.9	24537	645.5
grow15	106870941.1	24	2.0	106870941.3	23	3.0	106870941.3	426	4.3
grow22	160834336.2	25	3.3	160834336.5	28	5.2	160834336.5	676	8.0
grow7	47787811.8	23	0.8	47787811.8	20	1.2	47787811.8	175	1.4
israel	896644.8	71	12.3	896644.8	28	1.2	896644.8	250	1.2
kb2	1749.9	28	0.1	1749.9	21	0.2	1749.9	50	0.2
lotfi	25.3	29	0.5	25.3	25	0.8	25.3	205	0.8
maros	58063.7	65	22.2	58063.7	28	14.0	58063.7	1910	20.0
maros_r7	*(a)*			1497185.2	22	3603.3	*(c)*		

(Continua)

Taula II (cont.)

Resultats obtinguts amb els problemes test

Nom	IP			LoQo			Minos 5.3		
	Valor òptim	Iter.	CPU	Valor òptim	Iter.	CPU	Valor òptim	Iter.	CPU
nesm	14076036.6	65	19.4	14076036.5	38	22.2	14076073.1	3061	26.4
pilot	557.5	72	520.5	557.5	44	463.3	557.4	15996	738.5
pilot87	301.7	79	2312.7	301.6	46	1923.7	301.7	24284	3041.2
pilotnov	4497.3	40	40.1	4497.3	29	39.7	4497.3	2525	34.9
recipe	266.6	18	0.1	266.6	13	0.3	266.6	27	0.4
sc105	52.2	15	0.1	52.2	14	0.1	52.2	44	0.3
sc205	52.2	17	0.2	52.2	17	0.4	52.2	77	0.6
sc50a	64.6	15	0.0	64.6	14	0.1	64.6	28	0.2
sc50b	70.0	12	0.0	70.0	13	0.1	70.0	15	0.2
scagr25	14753433.1	47	1.4	14753433.1	18	1.0	14753433.1	319	2.4
scagr7	2331389.8	23	0.2	2331389.8	15	0.2	2331389.8	106	0.5
scfxm1	18416.8	38	1.7	18416.8	26	1.9	18416.8	361	2.1
scfxm2	36660.3	46	4.8	36660.3	28	4.2	36660.3	753	6.5
scfxm3	54901.3	46	8.4	54901.3	28	6.4	54901.3	1097	13.4
scorpion	1878.1	23	0.7	1878.1	16	0.7	1878.1	134	1.3
scrs8	904.3	37	2.5	904.3	23	2.9	904.3	682	5.3
scsd1	8.7	14	0.3	8.7	15	0.9	8.7	359	1.8
scsd6	50.5	17	0.8	50.5	17	1.6	50.5	1106	5.2
scsd8	905.0	17	1.8	905.0	17	3.4	905.0	3026	25.6
sctap1	1412.3	30	0.8	1412.3	18	0.9	1412.2	257	1.6
sctap2	1724.8	33	7.7	1724.8	16	4.9	1724.8	661	9.0
sctap3	1424.0	36	11.9	1424.0	16	6.2	1424.0	953	20.2
seba	15711.6	40	68.4	15711.6	19	2.1	15711.6	366	3.5
share1b	76589.3	89	1.0	76589.3	26	0.7	76589.3	250	0.8
share2b	415.7	20	0.2	415.7	14	0.2	415.7	111	0.4
shell	1208825347.3	32	1.6	1208825346.9	25	3.0	1208825346.0	321	4.5
ship04l	1793324.5	21	2.4	1793324.6	19	2.9	1793324.5	290	4.3
ship04s	1798714.7	24	1.6	1798714.7	19	2.1	1798714.7	157	2.9
ship08l	1909055.2	27	6.9	1909055.2	20	6.5	1909055.2	473	10.0
ship08s	1920098.2	24	3.3	1920098.2	20	3.7	1920098.2	256	5.3
ship12l	1470187.9	25	10.0	1470187.9	24	9.7	1470187.9	961	18.6
ship12s	1489236.1	25	4.8	1489236.1	22	4.8	1489236.1	437	9.0
sierra	15394362.2	25	6.6	15394362.2	21	5.9	15394362.2	864	12.2
standata	1257.7	29	1.4	1257.7	23	1.9	1257.7	131	2.3
standgub	1257.7	29	1.4	1257.7	23	2.0	1257.7	105	2.2
standmps	1406.0	42	3.7	1406.0	32	3.4	1406.0	383	3.0
stocfor1	41132.0	22	0.2	41132.0	17	0.2	41132.0	92	0.4
stocfor2	39024.4	45	19.6	39024.4	31	11.7	39024.4	3092	69.8
wood1p	1.4	38	57.1	1.4	28	53.8	1.4	870	21.1
woodw	1.3	51	49.1	1.3	27	41.1	1.3	3572	76.5
PROMIG		37	58.1		23	46.0		3747	97.5

(a) No hi ha suficient memòria per executar aquest problema.

(b) Problemes de convergència.

(c) Error numèric. No es poden satisfer les constriccions de forma acurada.

La implementació feta (a la qual ens referirem com IP) ha esta codificada en ANSI−C i s'ha comparat amb dos codis capdavanters de programació lineal. El primer d'ells és el paquet Minos 5.3 (Murtagh i Saunders(1983)), el qual està basat en el mètode del símplex per a programació lineal. El segon és el paquet LoQo (Vanderbei(1992)), una implementació eficient d'un mètode primal-dual per a programació lineal i quadràtica. Totes les execucions s'han realitzat sobre una SunSparc 10/41 sota UNIX, amb un únic processador RISC de 40Mhz i aproximadament 20Mflops, i amb 64 Mbytes de memòria (32Mbytes reals i 32Mbytes mapejats a disc). La Taula II mostra els resultats obtinguts amb cada paquet. Per a cada un dels tres codis (IP, LoQo i Minos 5.3) es presenta el valor òptim assolit (columna "Valor òptim"), el nombre d'iteracions realitzat (columna "Iter.") i el temps de CPU en segons requerit (columna "CPU").

Es pot observar a la Taula II com els tres codis en tot moment van assolir els mateixos valors òptims (hi ha petites diferències, però són menyspreables). També es pot veure que IP i LoQo realitzen moltes menys iteracions que Minos, degut a que estan basats en un mètode de punt interior en comptes de en el símplex. I comparant només aquests dos, en general IP realitza més iteracions que LoQo. El codi desenvolupat (IP) no va poder executar dos problemes ("fit2p" i "maros-r7") degut a problemes de memòria. LoQo no va tenir aquests problemes perquè no resol el sistema (26) a través de (33–37), sinó directament aplicant tècniques per a matrius simètriques i indefinides. Aquest fet explica també el comportament tan diferent que tenen ambdós codis en alguns exemples (per exemple, als problemes "fit1p" i "fit2d"). IP va tenir problemes de convergència al problema "greenbea", mentre que Minos no va poder satisfer les constriccions de forma acurada al test "maros-r7". La darrera fila de la Taula II mostra el valor promig per a cada codi pel que fa al nombre d'iteracions i temps de CPU. Pot observar-se com els dos codis de punt interior es mostren més eficients que Minos. També es veu que LoQo té un millor rendiment que IP. Tanmateix IP realitza moltes més iteracions que LoQo, el qual indica que efectivament IP té un cost per iteració menor que LoQo, com s'havia previst inicialment per la forma especial de solucionar el sistema (26).

El dit anteriorment sobre el rendiment de cada codi pot observar-se clarament a les Fig. 1 i 2. La Fig. 1 mostra el temps de CPU (eix y) pels tres codis segons la mida del problema (eix x). Com a mida del problema s'ha agafat el nombre de constriccions de cada un (tot i que es podria haver seguit un altre criteri com el nombre de variables, d'elements no-zero de la matriu o una expressió funció d'aquests valors). Només es presenten els problemes amb una mida relativament gran (per ex., més de 500 constriccions), per simplificar la gràfica. El temps de CPU s'ha escalat logarísmicament. Pot observar-se que, en general, Minos es troba per sobre dels dos codis de punt interior, mentre que LoQo dóna un millor rendiment. IP es troba en una situació intermèdia. Per la seva banda la Fig. 2 ens mostra, per a IP i LoQo, el temps per iteració (a l'eix y com abans) segons la mida del problema (eix x). Només

es mostren, com a la figura anterior, els problemes amb més de 500 constriccions. De nou també el temps per iteració s'ha escalat logarísmicament. Pot observar-se com IP té tendència a donar millors temps per iteració. Malgrat això, el fet de que requereixi força més iteracions que LoQo per convergir a l'òptim fa que no tingui un comportament global tan eficient com aquell.

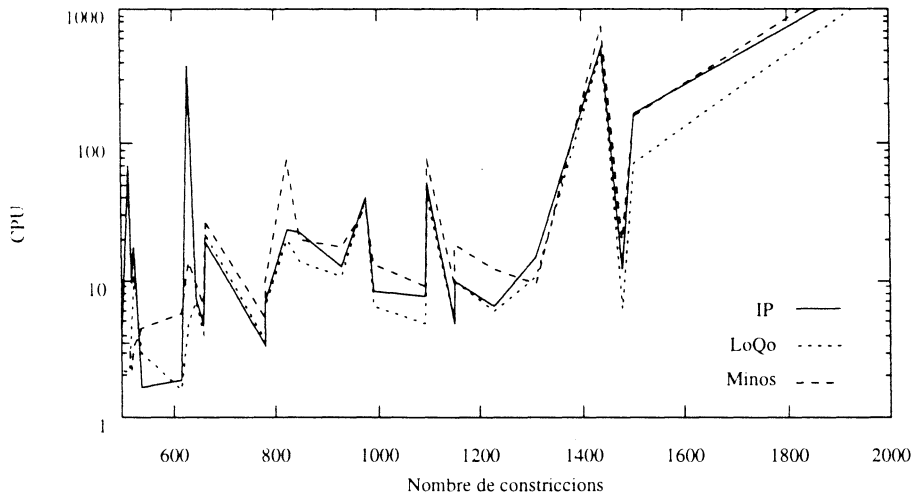

Figura 1. Temps de CPU per IP, LoQo i Minos segons la mida del problema.

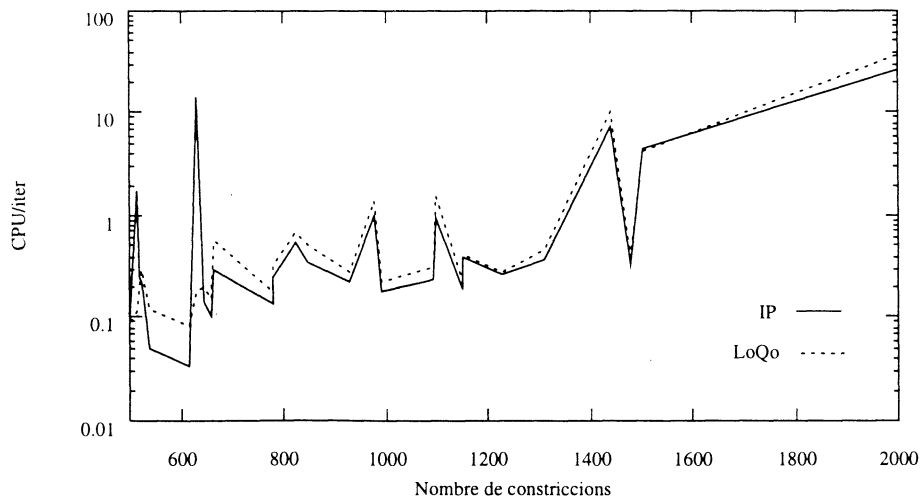

Figura 2. Temps per iteració per IP i LoQo segons la mida del problema.

5. CONCLUSIONS

Aquest document ha presentat un algorisme primal-dual de punt interior especialitzat per tractar les fites superiors a les variables de forma explícita. Una implementació de l'algorisme ha estat comparada amb dos paquets capdavanters de programació lineal. La implementació feta s'ha mostrat més eficient que Minos5.3, una implementació eficient del mètode del símplex, però menys que LoQo, basat també en un mètode primal-dual. Tanmateix, tot i que el codi desenvolupat té un major cost que LoQo, també realitza moltes més iteracions. S'ha pogut comprovar que el seu cost per iteració és menor, degut a la forma especial de calcular la direcció de Newton resolent un sistema simètric i definit positiu (LoQo resol un sistema simètric i indefinit). Això fa que es pugui explotar el fet de tenir la mateixa topologia durant tota l'execució i poder realitzar una factorització simbòlica prèvia. Queda obert el problema de millorar la convergència del codi desenvolupat pel que fa al nombre d'iteracions realitzades, amb el qual els temps de càlcul s'atansarien — i potser millorarien— els del codi LoQo.

AGRAÏMENTS

Voldria agrair la tasca dels revisors anònims, tant per la lectura tan acurada que han fet de l'article, la qual m'ha fet adonar de petits detalls que havia obviat, com per tots el seus comentaris i suggeriments.

APÈNDIX 1. SOLUCIÓ DEL SISTEMA LINEAL D'EQUACIONS PLANTEJAT A (26)

El sistema lineal d'equacions a resoldre a cada iteració a l'algorisme presentat és:

$$Z_u dx_u + X_u dz_u = b_{1_u} \tag{45}$$

$$Z_l dx_l + X_l dz_l = b_{1_l} \tag{46}$$

$$-W dx_u + F dw = b_2 \tag{47}$$

$$A_u dx_u + A_l dx_l = b_3 \tag{48}$$

$$A_u^t dy + dz_u - dw = b_{4_u} \tag{49}$$

$$A_l^t dy + dz_l = b_{4_l} \tag{50}$$

Aïllant dz_u de (49), dz_l de (50), dw de (47), dx_u de (45) i dx_l de (46) obtenim:

$$dz_u = b_{4_u} - A_u^t dy + dw \tag{51}$$

$$dz_l = b_{4_l} - A_l^t dy \tag{52}$$

$$dw = F^{-1}(b_2 + W dx_u) \tag{53}$$

$$dx_u = Z_u^{-1}(b_{1_u} - X_u dz_u) \tag{54}$$

$$dx_l = Z_l^{-1}(b_{1_l} - X_l dz_l) \tag{55}$$

Substituïnt (51) i (53) a (54) tenim que:

$$\begin{aligned} dx_u &= Z_u^{-1}(b_{1_u} - X_u[b_{4_u} - A_u^t dy + dw]) \\ &= Z_u^{-1}(b_{1_u} - X_u b_{4_u} - X_u dw + X_u A_u^t dy) \\ &= Z_u^{-1}(b_{1_u} - X_u b_{4_u} - X_u[F^{-1}(b_2 + W dx_u)] + X_u A_u^t dy) \\ &= Z_u^{-1}(b_{1_u} - X_u b_{4_u} - X_u F^{-1} b_2 - X_u F^{-1} W dx_u + X_u A_u^t dy) \\ &= Z_u^{-1} b_{1_u} - Z_u^{-1} X_u b_{4_u} - Z_u^{-1} X_u F^{-1} b_2 - Z_u^{-1} X_u F^{-1} W dx_u + Z_u^{-1} X_u A_u^t dy \end{aligned}$$

Agrupant termes s'arriba a

$$(\mathbb{1} + Z_u^{-1} X_u F^{-1} W) dx_u = Z_u^{-1} b_{1_u} - Z_u^{-1} X_u b_{4_u} - Z_u^{-1} X_u F^{-1} b_2 + Z_u^{-1} X_u A_u^t dy$$

Definint T_u com

$$T_u = \mathbb{1} + Z_u^{-1} X_u F^{-1} W$$

la seva inversa ve donada directament com

$$T_u^{-1} = F Z_u (F Z_u + X_u W)^{-1} \tag{56}$$

Llavors l'expressió anterior de dx_u pot ser escrita com

$$dx_u = T_u^{-1}(Z_u^{-1} b_{1_u} - Z_u^{-1} X_u b_{4_u} - Z_u^{-1} X_u F^{-1} b_2 + Z_u^{-1} X_u A_u^t dy) \tag{57}$$

Anàlogament, substituïnt (52) a (55) s'obté directament que

$$\begin{aligned} dx_l &= Z_l^{-1}(b_{1_l} - X_l[b_{4_l} - A_l^t dy]) \\ dx_l &= Z_l^{-1} b_{1_l} - Z_l^{-1} X_l b_{4_l} + Z_l^{-1} X_l A_l^t dy \end{aligned} \tag{58}$$

Ara, usant (58) i (59), l'equació (48) queda com segueix:

$$\begin{aligned} A_u[T_u^{-1}(Z_u^{-1} b_{1_u} - Z_u^{-1} X_u b_{4_u} - Z_u^{-1} X_u F^{-1} b_2 + Z_u^{-1} X_u A_u^t dy)] + \\ + A_l[Z_l^{-1} b_{1_l} - Z_l^{-1} X_l b_{4_l} + Z_l^{-1} X_l A_l^t dy] = b_3 \\ A_u T_u^{-1} Z_u^{-1}(b_{1_u} - X_u b_{4_u} - X_u F^{-1} b_2) + A_u T_u^{-1} Z_u^{-1} X_u A_u^t dy + \\ + A_l Z_l^{-1}(b_{1_l} - X_l b_{4_l}) + A_l Z_l^{-1} X_l A_l^t dy = b_3 \end{aligned} \tag{59}$$

Definint S_u i S_l com

$$S_u = T_u \; {}^1Z_u \; {}^1X_u = FX_u(FZ_u + X_uW) \; {}^1 \tag{60}$$

$$S_l = Z_l \; {}^1X_l \tag{61}$$

i agrupant termes, l'expressió (60) queda

$$(A_uS_uA_u^t + A_lS_lA_l^t)dy = b_3 + A_uT_u \; {}^1Z_u \; {}^1(X_uF \; {}^1b_2 + X_ub_{4_u} - b_{1_u}) + A_lZ_l \; {}^1(X_lb_{4_l} - b_{1_l})$$

$$(A_uS_uA_u^t + A_lS_lA_l^t)dy = b_3 + A_uS_u(F \; {}^1b_2 + b_{4_u} - X_u \; {}^1b_{1_u}) + A_lS_l(b_{4_l} - X_l \; {}^1b_{1_l})$$

L'expressió final del càlcul dy és:

$$(ASA^t)dy = b_3 + ASr \tag{62}$$

on r i S són respectivament:

$$r = (r_u^t, r_l^t)^t \qquad r_u = F \; {}^1b_2 + b_{4_u} - X_u \; {}^1b_{1_u} \qquad r_l = b_{4_l} - X_l \; {}^1b_{1_l} \tag{63}$$

$$S = \begin{pmatrix} S_u & 0 \\ 0 & S_l \end{pmatrix} \tag{64}$$

Un cop hem calculat dy, substituint el seu valor a (58) i (59) dóna:

$$\begin{aligned} dx_u &= T_u \; {}^1(Z_u \; {}^1b_{1_u} - Z_u \; {}^1X_ub_{4_u} - Z_u \; {}^1X_uF \; {}^1b_2 + Z_u \; {}^1X_uA_u^tdy) \\ &= T_u \; {}^1Z_u \; {}^1X_u(X_u \; {}^1b_{1_u} - b_{4_u} - F \; {}^1b_2 + A_u^tdy) \\ dx_u &= S_u(A_u^tdy - r_u) \end{aligned} \tag{65}$$

$$\begin{aligned} dx_l &= Z_l \; {}^1b_{1_l} - Z_l \; {}^1X_lb_{4_l} + Z_l \; {}^1X_lA_l^tdy \\ &= Z_l \; {}^1X_l(X_l \; {}^1b_{1_l} - b_{4_l} + A_l^tdy) \\ dx_l &= S_l(A_l^tdy - r_l) \end{aligned} \tag{66}$$

Agrupant (66) i (67) tenim que l'expressió final del calcul de dx és:

$$dx = S(A^tdy - r) \tag{67}$$

Un cop dy i dx són coneguts, dz i dw es calculen directament a través de les expressions inicials (51), (52) i (53).

BIBLIOGRAFIA

[1] **Arbel, A.** (1993). *Exploring Interior-Point Linear Programming. Algorithms and Software.* The MIT Press, Cambridge, Massachusetts.

[2] **Barnes, E.R.** (1986). "A variation on Karmarkar's algorithm for solving linear programming problems". *Mathematical Programming*, **36**, 174–182.

[3] **Duff, I.S., A.M. Erisman** i **J.K. Reid.** (1986). *Direct Methods for Sparse Matrices.* Oxford University Press, New York.

[4] **Gay, D.M.** (1985) "Electronic mail distribution of linear programming test problems". *Mathematical Programming Society COAL Newsletter*, **13**, 10–12.

[5] **George, J.A.** i **J.W.H. Liu.** (1981). *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, USA.

[6] **Gill, P.E., W. Murray** i **M.H. Wright.** (1981). *Practical Optimization.* Academic Press, London, UK.

[7] **Karmarkar, N.K.** (1984). "A new polynomial time algorithm for linear programming". *Combinatorica*, **4**, 373–395.

[8] **Khachiyan, G.** (1979). "A polynomial algorithm in linear programming". *Doklady Akademii Nauk SSSR*, **244(S)**, 1093–1096, traduït en *Soviet Mathematics Doklady*, **20(1)**, 191–194.

[9] **Monma, C.L** i **A.J Morton** (1987). "Computational experience with a dual affine variant of Karmarkar's method for linear programming". *Operations Research Letters*, **6**, 261–267.

[10] **Monteiro, R.D.C** i **I. Adler** (1989). "Interior path following primal-dual algorithms. Part I: linear programming". *Mathematical Programming*, **44**, 27–41.

[11] **Murtagh, B.A.** i **M.A. Saunders** (1983). "MINOS 5.0. User's guide". Dept. of Operations Research, Stanford University, CA, USA.

[12] **Vanderbei, R.J.** (1992). "LOQO User's Manual". Princeton University, Princeton, NJ, USA.

[13] **Vanderbei, R.J.** (1994). "An interior point code for quadratic programming". Princeton University, Princeton, NJ, USA.

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

[15] **Vanderbei, R.J., M.S. Meketon** i **B.A. Freedman** (1986). "A modification of Karmarkar's linear programming algorithm". *Algorithmica*, **1**, 395–407.

[16] **Wright, M.H.** (1991). "Interior methods for constrained optimization". *Acta Numerica*, 341–407.

ENGLISH SUMMARY:

AN IMPLEMENTATION OF A PRIMAL-DUAL INTERIOR POINT ALGORITHM WITH UPPER-BOUNDED VARIABLES

J. Castro

This paper presents an implementation of a primal-dual interior point algorithm considering upper-bounded variables without adding additional constraints. The algorithm is described and formulated, following the traditional way of the primal-dual methods, which attempt to solve the Khun-Tucker first order optimality conditions of the primal and dual Lagrangian functions including a logarithmic barrier function. Imposing the first order optimality conditions means solving a nonlinear system of equations. This is done through Newton's method of solving sets of nonlinear equations. In the algorithm developed, Newton's direction is computed by solving a symmetric and positive definite system with matrix ASA^t, A being the constraints matrix and S a diagonal matrix where $S_{ii}>0$. In this point the code differs from other implementations which find out Newton's direction by solving a symmetric indefinite system. In this last case the sparsity pattern of the Cholesky decomposition is not known a priori, since partial pivoting must be applied. However, in the approach of the algorithm presented, since the system to be solved is symmetric and positive definite, it can be assumed that it is sufficiently well-conditioned (thus avoiding partial pivoting), and a previous symbolic factorization can be made in order to know the sparsity pattern of the Cholesky decomposition. In theory, this fact should improve the performance of the process of computing Newton's direction.

Once a brief outline of the algorithm has been made, the paper focuses on some implementation details. The data structure employed to store the information for managing and creating the sub and diagonal part of system matrix ASA^t is instrumental to ensure a good performance. Another key point is the choice of the initial iteration point of Newton's method. When using Newton's method, the step size usually employed is one. In this case, due to the variables bounds, sometimes this step will have to be shorter. The way of computing this step is also detailed. The logarithmic barrier function considered in the formulation of the primal and dual Lagrangian functions has a parameter (denoted as μ) which must be updated at each iteration, and an account is given of how to update it. Finally, a brief exposition of the optimality conditions required to consider a point as optimal is made.

The implementation developed is compared with two state of the art linear programming codes: Minos 5.3 (an implementation of the simplex method) and LoQo

(a primal-dual interior point code for linear and quadratic programming). To test the codes 80 linear programming problems taken from the *Netlib* standard collection have been employed. The results obtained show that the performance of the code developed is between those of LoQo and Minos (LoQo is the most efficient of the three). However, the code developed is faster in performing one single iteration than LoQo (as it was theoretically expected), but it has a slower convergence since it requires much more iterations. It remains as an open problem to improve its convergence to reach (even improve) the efficiency of the LoQo package.