QÜESTIIÓ, vol. 22, 1, p. 103-116, 1998

IMPLEMENTACIÓN DE UN ALGORITMO PRIMAL-DUAL DE ORDEN SUPERIOR MEDIANTE EL USO DE UN MÉTODO PREDICTOR-CORRECTOR PARA PROGRAMACIÓN LINEAL*

J. CASTRO**

Universitat Rovira i Virgili

Se presenta una implementación de un algoritmo primal-dual de punto interior para la solución de problemas lineales. El algoritmo difiere de otros ya existentes (como el implementado en el sistema LoQo) en el hecho de que soluciona las denominadas «ecuaciones normales en forma primal» (LoQo soluciona el denominado «sistema aumentado») y en que realiza una clara distinción entre variables acotadas superior e inferiormente, y aquellas sólo acotadas inferiormente. La eficiencia de la implementación es comparada con el sistema LoQo. Para la comparación se utilizan 80 problemas lineales de la colección Netlib (Gay (1985)) (una batería estándar de problemas de programación lineal). Este trabajo es el primero de una serie de dos, cuyo objetivo es la resolución eficiente de problemas cuadráticos por técnicas de punto interior.

An implementation of a higher-order primal-dual interior point algorithm using a predictor-corrector method for linear programming.

Palabras clave: Algoritmo primal-dual, métodos de punto interior, método predictor-corrector, programación lineal

*Este trabajo ha sido subvencionado por la Ayuda Iberdrola a la Investigación Científica y al Desarrollo Tecnológico 95-005, y por el proyecto CICYT TAP96-1044-J02-93.

** J. Castro. Estadística i Investigació Operativa. Dept. d'Enginyeria Química. Universitat Rovira i Virgili. Autovia de Salou, s/n. 43006 Tarragona.

- Recibido en noviembre de 1996.
- Aceptado en julio de 1997.

1. INTRODUCCIÓN

Desde la aparición del algoritmo de Karmarkar (Karmarkar (1984)) se ha dedicado un gran esfuerzo al estudio de los métodos de punto interior, debido fundamentalmente a la gran eficiencia que han mostrado tener en la resolución de ciertos tipos de problemas de programación lineal. Ver en Monteiro y Adler (1989) un estudio sobre la convergencia de estos métodos para la solución de problemas lineales. En la actualidad existen diversas implementaciones de carácter práctico, tales como el sistema LoQo (Linear Optimization, Quadratic Optimization), ver Vanderbei (1992).

La implementación presentada en este trabajo difiere del sistema LoQo en dos puntos fundamentalmente. Primero, LoQo utiliza técnicas para matrices simétricas casi definidas, dado que resuelve el denominado «sistema aumentado» (que no es más que una simplificación del sistema de ecuaciones de Karush-Kuhn-Tucker del problema). Por su parte, nuestra implementación utiliza las denominadas «ecuaciones normales en forma primal» (consistentes en una simplificación adicional del «sistema aumentado»); ver en Vanderbei (1996) una descripción de ambas técnicas. En segundo lugar, nuestra implementación realiza una clara distinción entre las variables acotadas superior e inferiormente y aquéllas sólo inferiormente. Se realiza un preproceso previo en donde se agrupan las variables pertenecientes a cada tipo de variables. Esta reordenación de variables repercute en una mayor eficiencia computacional.

Este trabajo es una extensión de la implementación de un método primal-dual para problemas lineales descrita en Castro (1995). La diferencia entre ambos algoritmos es la inclusión de un método predictor-corrector para la obtención de direcciones de movimiento de orden superior. Este hecho mejora la convergencia del algoritmo, permitiendo que, a diferencia de la presentada en Castro (1995), la nueva implementación sea tan eficiente como el sistema LoQo en la resolución de problemas lineales.

A su vez, este trabajo también es el primero de una serie de dos, cuyo objetivo es la resolución eficiente de problemas cuadráticos por técnicas de punto interior. Los resultados aquí obtenidos son la base para el desarrollo de un algoritmo de punto interior para problemas cuadráticos tal y como se describe en Castro (1998).

El trabajo tiene la siguiente organización. La sección 2 presenta brevemente el algoritmo primal-dual de punto interior para problemas lineales. La sección 3 detalla el cálculo de la nueva dirección de orden superior. La sección 4 discute los resultados computacionales obtenidos en la solución de una batería de 80 problemas lineales. Finalmente, la sección 5 resume las principales conclusiones de este trabajo.

2. ALGORITMO PRIMAL-DUAL DE PUNTO INTERIOR PARA PROBLEMAS LINEALES

En esta sección se presenta brevemente el método primal-dual para problemas lineal implementado en Castro (1995) (dicha implementación será referida como IP).

El problema de minimización primal considerado es:

$$(1) \qquad (P) \qquad \begin{array}{ll} \min & c_u' x_u + c_l' x_l \\ \text{sujeto a} & A_u x_u + A_l x_l = b \\ & x_u + f = \bar{x}_u \\ & x \geq 0 \\ & f \geq 0 \end{array}$$

donde $c_u, x_u, f \in \mathbb{R}^{n_u}$, $c_l, x_l \in \mathbb{R}^{n_l}$, $c = (c_u', c_l')'$, $x = (x_u', x_l')'$ (por tanto $c, x \in \mathbb{R}^n$, $n = n_u + n_l$), $b \in \mathbb{R}^m$, $A_u \in \mathbb{R}^{m \times n_u}$ y $A_l \in \mathbb{R}^{m \times n_l}$ (por tanto $A = [A_u | A_l] \in \mathbb{R}^{m \times n}$) (observar que todos los elementos $_u$ corresponden a las variables acotadas superior inferiormente, mientras que aquéllos denotados con $_l$ corresponden a variables sól acotadas inferiormente).

El problema dual asociado puede expresarse como:

$$(2) \qquad (D) \qquad \begin{array}{ll} \max & b'y - \bar{x}_u' w \\ \text{subj. a} & A_u' y + z_u - w = c_u \\ & A_l' y + z_l \quad\quad = c_l \\ & y \in \mathbb{R}^m \\ & z_l \geq 0 \quad z_u \geq 0 \quad w \geq 0 \end{array}$$

donde $y \in \mathbb{R}^m$, $z = (z_u', z_l')'$ ($z_u \in \mathbb{R}^{n_u}$, $z_l \in \mathbb{R}^{n_l}$, por tanto $z \in \mathbb{R}^n$) y $w \in \mathbb{R}^{n_u}$.

Las condiciones de primer orden a satisfacer por la solución óptima de los problemas (1) y (2) (denominadas condiciones de Karush-Kuhn-Tucker) una vez eliminados los límites inferiores $x \geq 0$, $f \geq 0$, $z \geq 0$ y $w \geq 0$ mediante la introducción de una barrera logarítmica (ver Castro (1995)), son las siguientes:

$$X_u Z_u e_{n_u} = \mu e_{n_u} \tag{3}$$

$$X_l Z_l e_{n_l} = \mu e_{n_l} \tag{4}$$

$$A_u x_u + A_l x_l = b \tag{5}$$

$$A_l' y + z_l = c_l \tag{6}$$

$$F W e_{n_u} = \mu e_{n_u} \tag{7}$$

$$A_u' y + z_u - w = c_u \tag{8}$$

donde e_l denota el vector l-dimensional de 1's, y X_u, X_l, Z_u, Z_l, W, F son matrices

diagonales, definidas como:

$$
\begin{aligned}
e_l &= (1_1,\ldots,1_l)' \\
X_u &= \mathrm{diag}(x_{u_1},\ldots,x_{u_{n_u}}) \\
X_l &= \mathrm{diag}(x_{l_1},\ldots,x_{l_{n_l}}) \\
Z_u &= \mathrm{diag}(z_{u_1},\ldots,z_{u_{n_u}}) \\
Z_l &= \mathrm{diag}(z_{l_1},\ldots,z_{l_{n_l}}) \\
W &= \mathrm{diag}(w_1,\ldots,w_{n_u}) \\
F &= \mathrm{diag}(\bar{x}_{u_1}-x_{u_1},\ldots,\bar{x}_{u_{n_u}}-x_{u_{n_u}})
\end{aligned} \tag{9}
$$

Los dos siguientes apartados presentan la modificación que se propone al algoritmo original basado en el primal-dual para IP, y los resultados computacionales obtenidos con la nueva variante.

3. INTRODUCCIÓN DE INFORMACIÓN DE SEGUNDO ORDEN

La solución del modelo (3)–(8) presentada en Castro (1995), e implementada en IP, resuelve el sistema anterior mediante el método de Newton. Se resuelve a cada iteración i el sistema $J(\xi_i)d\xi_i = -f(\xi_i)$, siendo $d\xi_i$ la dirección de Newton, y $J(\xi_i)$ y $f(\xi_i)$ el jacobiano del sistema y su evaluación en el punto actual $\xi_i = (x_i', y_i', z_i', w_i')'$. El sistema a resolver es en este caso:

$$
\begin{aligned}
Z_u dx_u + X_u dz_u &= \mu e_{n_u} - X_u Z_u e_{n_u} \\
Z_l dx_l + X_l dz_l &= \mu e_{n_l} - X_l Z_l e_{n_l} \\
-W dx_u + F dw &= \mu e_{n_u} - W F e_{n_u} \\
A_u dx_u + A_l dx_l &= b - Ax \\
A_u' dy + dz_u - dw &= c_u - A_u' y - z_u + w \\
A_l' dy + dz_l &= c_l - A_l' y - z_l
\end{aligned} \tag{10}
$$

Se puede mejorar la efectividad del algoritmo, mediante la resolución del sistema (3)–(8), considerando $f_j(\xi_i + d\xi_i) = 0 \quad j = 1,\ldots,6$ y una aproximación cuadrática para cada $f_j()$, tal que:

$$
f_j(\xi_i) + \nabla f_j(\xi_i)' d\xi_i + \frac{1}{2}\, d\xi_i' \nabla^2 f_j(\xi_i) d\xi_i = 0 \quad j = 1,\ldots,6 \tag{11}
$$

En este caso, puede comprobarse que el sistema de ecuaciones a resolver a cada iteración viene dado por:

$$
\begin{aligned}
Z_u dx_u + X_u dz_u + dZ_u dx_u &= \mu e_{n_u} - X_u Z_u e_{n_u} \\
Z_l dx_l + X_l dz_l + dZ_l dx_l &= \mu e_{n_l} - X_l Z_l e_{n_l}
\end{aligned}
$$

$$
\begin{aligned}
-W dx_u + F dw - dW dx_u &= \mu e_{n_u} - W F e_{n_u} \\
A_u dx_u + A_l dx_l &= b - Ax \\
A_u' dy + dz_u - dw &= c_u - A_u' y - z_u + w \\
A_l' dy + dz_l &= c_l - A_l' y - z_l
\end{aligned} \tag{12}
$$

Se puede observar cómo la única diferencia entre (12) y (10) se debe a la introdu de los términos no lineales $dZ_u dx_u$, $dZ_l dx_l$ y $dW dx_u$ en las tres primeras ecu nes de (12). La forma clásica de solucionar este sistema, propuesta inicialmer Mehrotra (1990) (ver también Lustig *et al.* (1992), Vanderbei (1994)), consis realizar dos pasos, denominados el paso predictor y el paso corrector. En el predictor se obtiene una dirección $\hat{d\xi}_i$ aproximada, mediante la solución del sis (12) eliminando las no linealidades y los términos μ:

$$
\begin{aligned}
Z_u \hat{dx}_u + X_u \hat{dz}_u &= -X_u Z_u e_{n_u} \\
Z_l \hat{dx}_l + X_l \hat{dz}_l &= -X_l Z_l e_{n_l} \\
-W \hat{dx}_u + F \hat{dw} &= -W F e_{n_u} \\
A_u \hat{dx}_u + A_l \hat{dx}_l &= b - Ax \\
A_u' \hat{dy} + \hat{dz}_u - \hat{dw} &= c_u - A_u' y - z_u + w \\
A_l' \hat{dy} + \hat{dz}_l &= c_l - A_l' y - z_l
\end{aligned} \tag{13}
$$

Se utiliza la dirección aproximada $\hat{d\xi}_i$ para aproximar los términos no lineales $\hat{dZ}_u$ $\hat{dZ}_l \hat{dx}_l$ y $\hat{dW} \hat{dx}_u$ de (12). Entonces, la dirección $d\xi_i$ a utilizar viene dada po solución del paso corrector:

$$
\begin{aligned}
Z_u dx_u + X_u dz_u &= \mu e_{n_u} - X_u Z_u e_{n_u} - \hat{dZ}_u \hat{dx}_u \\
Z_l dx_l + X_l dz_l &= \mu e_{n_l} - X_l Z_l e_{n_l} - \hat{dZ}_l \hat{dx}_l \\
-W dx_u + F dw &= \mu e_{n_u} - W F e_{n_u} + \hat{dW} \hat{dx}_u \\
A_u dx_u + A_l dx_l &= b - Ax \\
A_u' dy + dz_u - dw &= c_u - A_u' y - z_u + w \\
A_l' dy + dz_l &= c_l - A_l' y - z_l
\end{aligned} \tag{14}
$$

Hay que destacar que los términos de la izquierda de los sistemas (13) y (14) iguales, y tan sólo varían los términos independientes. En Castro (1995) se mue que este tipo de sistemas de ecuaciones se resuelven mediante:

$$
(ASA')dy = b_3 + ASr \tag{15}
$$

$$
dx = S(A'dy - r) \tag{16}
$$

$$
dw = F^{-1}(b_2 + W dx_u) \tag{17}
$$

$$
dz_u = b_{4_u} + dw - A_u' dy \tag{18}
$$

$$
dz_l = b_{4_l} - A_l' dy \tag{19}
$$

nde el vector r y la matriz S se definen como:

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

$$S = \begin{pmatrix} S_u & \mathbf{0} \\ \mathbf{0} & S_l \end{pmatrix} \quad S \in \mathbb{R}^{n\times n}, \quad S_u \in \mathbb{R}^{n_u \times n_u}, \quad S_l \in \mathbb{R}^{n_l \times n_l u}$$
$$S_u = FX_u(Z_uF + X_uW)^{-1} \qquad S_l = Z_l^{-1}X_l$$

b_{1_u}, b_{1_l}, b_2, b_3, b_{4_u} y b_{4_l} denotan, por este orden, los términos independientes de ecuaciones de (13) o (14) (según el sistema a resolver).

punto más costoso consiste en la resolución de un sistema con la matriz ASA' en), para lo cual es necesario realizar su factorización de Cholesky. Sin embargo, ha factorización es la misma tanto para el paso predictor como para el paso co- ctor. Por tanto, el hecho de introducir un método predictor-corrector, respecto del todo original implementado en IP, implica la resolución de un sistema adicional, ro teniendo ya la factorización de Cholesky realizada (por tanto tan sólo debe rea- arse una sustitución inversa y una directa adicionales). Este coste computacional tra, sin embargo, permite la introducción de información de segundo orden en la solución de (3)–(8).

RESULTADOS COMPUTACIONALES CON EL MÉTODO PREDICTOR-CORRECTOR

ea IPPC (Interior Point Predictor-Corrector) la nueva versión del código IP en el ue se ha implementado la modificación introducida en el apartado anterior. IPPC es emejante a otros códigos que implementan el mismo algoritmo, como LoQo (Vander- ei (1994)). Sin embargo existen algunas diferencias entre IPPC y LoQo. En primer igar, IPPC realiza una clara distinción entre las variables acotadas inferior y supe- ormente y aquéllas sólo acotadas inferiormente, realizando una reordenación previa e variables que agiliza el comportamiento computacional del algoritmo. Además, PPC no considera las variables de holgura f explícitamente (LoQo sí lo hace) y, en ez de ello, usa en todo momento la relación $f = \bar{x}_u - x_u$. Finalmente, cabe destacar ambién que IPPC, como se ha indicado anteriormente, resuelve el sistema simétrico definido positivo ASA' en cada iteración. LoQo, por su parte, resuelve (13) y (14) sando técnicas para matrices simétricas casi-definidas.

El código IPPC (implementado en C) utiliza el procedimiento de IP para la obtención de la longitud de paso primal y dual, actualización del parámetro de la barrera lo- garítmica, obtención del punto inicial de iteración y detección de la optimalidad del punto actual. La única diferencia es la solución de los sistemas (13) y (14) por parte de IPPC, mientras que IP soluciona (10). Hay que hacer notar, sin embargo, que IPPC

implementa una ligera modificación en el método predictor-corrector expuesto en el apartado anterior: realiza un escalado (el factor de escala es siempre menor que 1.0) de los términos no lineales $\hat{dZ}_u\hat{dx}_u$, $\hat{dZ}_l\hat{dx}_l$ y $\hat{dW}\hat{dx}_u$ de (12), a utilizar en (14) una vez obtenidos en (13), en las primeras iteraciones del algoritmo, simulando una disminución de paso de las direcciones $\hat{d\xi}_t$. En la práctica, este escalado se ha mostrado como una opción algorítmica ventajosa y ha permitido agilizar el comportamiento del algoritmo, e incluso, en algunos casos, solucionar problemas de estabilidad numérica. Este escalado tan sólo se aplica en las primeras iteraciones del algoritmo, lejos todavía de obtener la factibilidad primal y dual.

Se han utilizado 80 problemas de la colección Netlib (Gay (1985)) para comprobar la eficiencia de IPPC. Las características de estos problemas se muestran en la tabla 1 donde m, n y nel son el número de condiciones, variables y elementos no nulos de la matriz de restricciones.

La tabla 2 recoge el número de iteraciones (niter) y el tiempo computacional (t), en segundos de CPU, requerido por los códigos IPPC, IP y LoQo para resolver los problemas Netlib cuyas dimensiones se han recogido en la tabla 1. El valor óptimo alcanzado no se muestra, dado que no hubo discrepancias entre los tres códigos (el criterio de parada para los tres era de 8 dígitos de igualdad en las funciones primal y dual). Todas las ejecuciones han sido realizadas sobre una SunSparc 10/41 de 64 Mbytes de memoria (32 reales y 32 mapeadas en disco) y aproximadamente 10 Mflops.

Hay que destacar que las versiones actuales desarrolladas de IP e IPPC no permiten solucionar problemas con variables libres (no acotadas), y no realizan ningún tratamiento específico de columnas densas de la matriz de restricciones A (LoQo sí trata ambas situaciones). (En próximas versiones de IPPC se prevé que estos problemas sean solventados.) El hecho de no realizar un tratamiento específico de columnas densas provoca que IP e IPPC no puedan solucionar algunos problemas por falta de memoria (como «fit2p» y «maros-r7»). LoQo, por su parte, ha podido solucionar todos los problemas (es el más robusto de los tres códigos, seguido de IPPC). El tiempo de computación requerido por IPPC es equiparable al de LoQo, y ligeramente inferior en algunos problemas de dimensión elevada que requieren un tiempo de cálculo considerable (p.e. «d2q06c», «pilot», «pilot87»). La última fila nos muestra el valor promedio (número de iteraciones y tiempo de CPU) para cada código. No se han considerado en este promedio los problemas «fit2p» y «maros-r7», dado que ni IP ni IPPC han podido solucionarlos por problemas de memoria. Si comparamos IPPC, IP y LoQo, se observa como IPPC realiza un número de iteraciones equivalente al de LoQo, y es ligeramente más eficiente que aquél. IP, por su parte, realiza muchas más iteraciones y tiene un rendimiento claramente inferior. Este hecho ya fue notado en Castro (1995), donde se indicaba que, a pesar de que el tiempo por iteración de IP era menor que el de LoQo, aquél era más ineficiente debido a que realizaba un mayor número de iteraciones.

Tabla 1. *Dimensiones de los problemas Netlib*

Problema	*m*	*n*	*nel*
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

Problema	*m*	*n*	*nel*
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

Tabla 2. *Efectividad de los sistemas IPPC, IP y LoQo*

	IPPC		IP		LoQo	
Problema	niter	t	niter	t	niter	t
25fv47	27	16.6	45	23.9	29	20.6
80bau3b	33	41.4	71	61.3	44	52.2
adlittle	15	0.1	20	0.1	14	0.1
afiro	10	0.0	13	0.0	13	0.0
agg	29	5.2	43	6.8	26	1.8
agg2	26	7.5	37	9.5	22	4.3
agg3	36	9.5	39	9.8	22	4.3
bandm	21	1.2	34	1.5	20	1.5
beaconfd	13	1.1	18	1.3	14	1.0
blend	15	0.1	20	0.1	14	0.2
bnl1	28	5.2	51	7.5	35	7.0
bnl2	39	85.9	61	124.7	40	122.5
boeing1	28	3.2	44	4.0	28	2.8
boeing2	25	0.8	32	0.8	28	0.9
bore3d	16	0.7	29	0.9	17	0.8
brandy	24	1.1	29	1.2	22	1.3
czprob	45	11.9	56	12.8	38	10.2
d2q06c	34	167.2	58	263.6	36	249.0
d6cube	43	72.5	51	80.8	23	75.6
degen2	14	4.9	26	6.7	14	4.6
degen3	24	136.9	38	165.6	17	74.8
e226	24	1.2	39	1.7	22	1.3
etamacro	29	3.8	49	5.6	30	10.0
fffff800	53	13.1	78	17.8	36	11.2
finnis	25	2.0	42	2.7	26	2.3
fit1d	28	2.9	56	4.8	21	4.8
fit1p	16	258.4	26	381.6	26	4.5
fit2d	24	28.2	48	44.7	24	195.0
fit2p	[a]		[a]		24	38.5
ganges	26	11.7	41	15.2	23	11.1
gfrd-pnc	38	1.6	55	1.9	19	1.4
greenbea	62	93.6	94 [b]	126.8	47	87.1
grow15	19	1.9	24	2.0	23	2.8
grow22	19	3.0	25	3.3	28	5.1
grow7	17	0.7	23	0.8	20	1.1
israel	45	8.4	71	12.3	28	1.2
kb2	19	0.1	28	0.1	21	0.2
lotfi	21	0.5	29	0.5	25	0.7
maros	49	18.7	65	22.2	28	14.2
maros-r7	[a]		[a]		22	3603.3

[a] Memoria insuficiente. [b] Problemas de convergencia.

Tabla 2. (cont.) *Resultados de los problemas Netlib lineales con IPPC, IP y LoQo*

	IPPC		IP		LoQo	
Problema	niter	t	niter	t	niter	t
nesm	42	14.9	65	19.4	38	23.3
pilot	48	365.5	72	520.5	44	489.6
pilot87	54	1650.7	79	2312.7	46	1976.0
pilotnov	31	33.0	40	40.1	29	40.6
recipe	11	0.1	18	0.1	13	0.2
sc105	12	0.1	15	0.1	14	0.1
sc205	13	0.2	17	0.2	17	0.3
sc50a	11	0.0	15	0.0	14	0.1
sc50b	10	0.0	12	0.0	13	0.0
scagr25	21	0.9	47	1.4	18	0.9
scagr7	17	0.1	23	0.2	15	0.2
scfxm1	25	1.4	38	1.7	26	1.9
scfxm2	26	3.5	46	4.8	28	4.0
scfxm3	28	6.7	46	8.4	28	6.8
scorpion	14	0.6	23	0.7	16	0.7
scrs8	22	1.8	37	2.5	23	2.6
scsd1	13	0.4	14	0.3	15	0.9
scsd6	14	0.7	17	0.8	17	1.4
scsd8	13	1.8	17	1.8	17	2.9
sctap1	19	0.7	30	0.8	18	0.8
sctap2	17	5.8	33	7.7	16	4.6
sctap3	18	9.2	36	11.9	16	5.9
seba	26	47.8	40	68.4	19	1.7
share1b	71	1.0	89	1.0	26	0.7
share2b	14	0.2	20	0.2	14	0.2
shell	39	2.1	32	1.6	25	2.8
ship04l	14	2.3	21	2.4	19	2.6
ship04s	15	1.4	24	1.6	19	1.8
ship08l	18	6.1	27	6.9	20	6.8
ship08s	16	2.9	24	3.3	20	3.3
ship12l	16	8.5	25	10.0	24	8.9
ship12s	16	4.1	25	4.8	22	5.2
sierra	21	6.4	25	6.6	21	6.1
standata	19	1.3	29	1.4	23	1.7
standgub	19	1.2	29	1.4	23	1.9
standmps	28	2.5	42	3.7	32	3.6
stocfor1	18	0.2	22	0.2	17	0.2
stocfor2	28	16.1	45	19.6	31	11.8
woodlp	24	43.6	38	57.1	28	55.9
woodw	30	37.4	51	49.1	27	40.5
Promedio	25	42.4	37	59.0	23	47.5

5. CONCLUSIONES

Basados en la experiencia computacional bastante fuerte, cuyos resultados principales se han obtenido en el apartado anterior, el método presentado en este trabajo se ha mostrado como una alternativa eficiente al sistema LoQo para la solución de problemas lineales. De ello se concluye que el motivo por el cual la implementación IP presentada en Castro (1995) era menos eficiente que LoQo, era precisamente la falta de información de segundo orden en el cálculo de la dirección de movimiento. La introducción de dicha información en IPPC hace que éste claramente supere en rendimiento a IP. Los resultados obtenidos garantizan que la extensión del algoritmo al caso cuadrático permitirá obtener una estrategia de resolución eficiente para este tipo de problemas (tal y como se detalla en Castro (1998)).

6. REFERENCIAS

[1] **Castro, J.** (1995). «Implementació d'un algorisme primal-dual de punt interior amb fites superiors a les variables», *Qüestiió*, **19, 1, 2, 3**, 233–257.

[2] **Castro, J.** (1998). «Un algoritmo de punto interior para programación cuadrática a través de problemas equivalentes separables», *Qüestiió*, **22, 1**, 121-146.

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

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

[5] **Lustig, I.J., R.E Marsten** and **D.F. Shanno** (1992). «On implementing Mehrotra's predictor-corrector interior-point method for linear programming», *SIAM Journal on Optimization*, **2(3)**, 435–449.

[6] **Mehrotra, S.** (1990). «On the implementation of a (primal-dual) interior point method», *Technical Report 90-03 Dept. of Industrial Engineering and Management Science*, Northwestern University, Evanston, IL.

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

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

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

[10] **Vanderbei, R.J.** (1996). *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston.