COMBINANDO UM ALGORITMO DE AJUSTAMENTO ÓTIMO E O MÉTODO DE PONTOS INTERIORES

Carla T. L. S. Ghidini
IMECC - Universidade Estadual de Campinas
Rua Sérgio Buarque de Holanda, 651 - Cidade Universitária
13083-859 - Campinas, SP - Brasil
carla@ime.unicamp.br

Aurelio R. L. Oliveira
IMECC - Universidade Estadual de Campinas
Rua Sérgio Buarque de Holanda, 651 - Cidade Universitária
13083-859 - Campinas, SP - Brasil
aurelio@ime.unicamp.br

RESUMO

O algoritmo de ajustamento ótimo para *p* coordenadas é uma generalização do algoritmo de ajustamento pelo par ótimo para programação linear, o qual é baseado no algoritmo de Von Neumann. Suas principais vantagens são simplicidade e avanço inicial rápido. Com o objetivo de acelerar a convergência do método de pontos interiores, algumas iterações do algoritmo generalizado são realizadas dentro da heurística de Mehrotra, e dessa forma, uma boa solução inicial é determinada. Os experimentos computacionais mostraram que esta abordagem reduz o número total de iterações e o tempo total de resolução em um conjunto de problemas de programação linear, incluindo os de grande porte.

PALAVARAS CHAVE. Algoritmo de Von Neumann, método de pontos interiores, programação linear.

ABSTRACT

Optimal adjustment algorithm for *p* coordinates is a generalization of the optimal pair adjustment algorithm for linear programming, which, in turn, is based on von Neumann's algorithm. Its main advantages are simplicity and fast initial convergence. To accelerate the convergence of the interior point method few iterations of this generalized algorithm are applied into the Mehrotra's heuristic. Thus, a good starting solution is obtained. Computational experiments have shown that this approach reduces the total number of iterations and the running time for a set of linear programming problems, including large-scale ones.

KEYWORDS. Von Neumann's algorithm, interior point methods, linear programming

1. Introdução

O algoritmo Von Neumann foi apresentado por Dantzig no início da década de 90 (Dantzig (1991), (1992)) e mais tarde foi estudado por Epelman e Freund (2000). Este algoritmo possui propriedades interessantes, como simplicidade e avanço inicial rápido. Porém, ele não é muito prático para resolver problemas de programação linear, visto que sua convergência é muito lenta.

Gonçalves (2004), durante seu doutorado, estudou o algoritmo de Von Neumann e apresentou quatro novos algoritmos baseado nele, sendo que o algoritmo de ajustamento pelo par ótimo foi o que obteve melhor desempenho na prática.

O algoritmo de ajustamento pelo par ótimo herda as melhores propriedades do algoritmo Von Neumann. Embora Gonçalves tenha provado que em termos de convergência o algoritmo de ajustamento pelo par ótimo é superior ao algoritmo de Von Neumann, ainda assim, esse algoritmo não é prático para resolver problemas de programação linear, visto que sua convergência também é lenta.

Silva (2009) generalizou a ideia apresentada por Gonçalves, Storer e Gondzio em Gonçalves (2009) ao propor o algoritmo de ajustamento pelo par ótimo e desenvolveu o algoritmo de ajustamento ótimo para p coordenadas, em que p é limitado pela ordem do problema. Este algoritmo também possui como principais vantagens simplicidade e avanço rápido nas iterações iniciais.

Do ponto de vista computacional, a proposta não é resolver problemas de programação linear até a otimalidade aplicando o algoritmo de ajustamento ótimo para p coordenadas, mas sim explorar suas principais características e usá-lo em conjunto com o método de pontos interiores para acelerar convergência dele.

Sabendo que o ponto inicial influencia muito no desempenho do método de pontos interiores, neste trabalho, o algoritmo de ajustamento ótimo para p coordenadas será aplicado dentro da heurística de Mehrotra (Mehrotra (1992)), a qual determina o ponto inicial para o método dos pontos interiores no software PCx, para que pontos iniciais ainda melhores possam ser obtidos.

2. Algoritmos simples de programação linear

Considere o problema de encontrar uma solução factível para o seguinte conjunto de restrições lineares:

$$\begin{aligned} Ax &= 0, \\ e^t x &= 1, \\ x &\geq 0, \end{aligned} \tag{1}$$

em que $A \in \mathrm{R}^{m \times n}$, x e $e \in \mathrm{R}^n$, e é um vetor unitário e as colunas de A tem norma um, isto é, $||A_j||=1$, para $j = 1, ..., n$.

Geometricamente, as colunas A_j podem ser vistas como pontos sobre a hiper-esfera m-dimensional com raio unitário e centro na origem. Dessa forma, o problema acima pode ser descrito como de atribuir ponderações x_j não negativas às colunas A_j de modo que, depois de re-escalado, seu centro de gravidade seja a origem.

Todo problema de programação linear pode ser reduzido ao Problema (1), (veja Gonçalves (2004)) e quando isso é feito a estrutura da matriz A, geralmente, torna-se bastante esparsa, o que é desejável, uma vez que essa característica pode ser explorada para reduzir o custo computacional.

2.1 Algoritmo de Von Neumann

O algoritmo de Von Neumann foi proposto a Dantizg em 1948, divulgado por ele no início dos anos 90 (Dantzig (1991), (1992)) e mais tarde estudado por Epelman e Freund (2000). Este algoritmo é bastante simples e tem avanço inicial rápido, mas não é muito prático para resolver problemas de programação linear, pois sua convergência é muito lenta.

Basicamente, o algoritmo de Von Neumann consiste em encontrar a coluna A_s de A que forma o maior ângulo com o resíduo b^{k-1}, e então o próximo resíduo é a projeção da origem no segmento de reta ligando b^{k-1} a A_s. A Figura 1 ilustra este método.

O esforço por iteração do algoritmo de von Neumann é dominado por multiplicação de matriz por vetor, necessária na seleção da coluna A_s, que é O(mn). O número de operações requeridas nesta multiplicação tem uma redução significativa se a matriz A é esparsa.

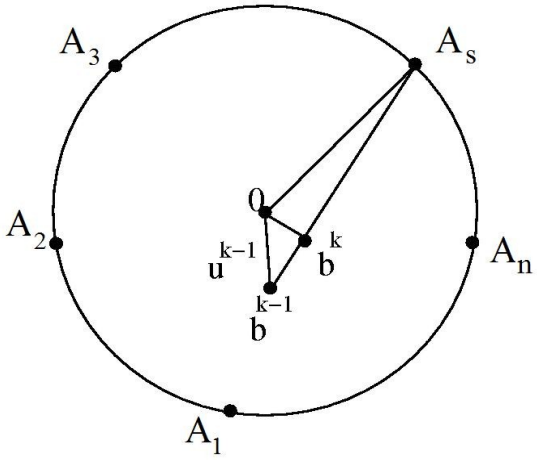

Figura 1: Ilustração do algoritmo de Von Neumann

2.2 Algoritmo de ajustamento pelo par ótimo

O algoritmo de ajustamento pelo par ótimo, proposto por Gonçalves *et al*. (2009), é baseado na idéia de que o resíduo b^{k-1} pode ser movido de tal forma a aproximar-se da origem 0, aumentando um peso x^j de alguma coluna A_j e reduzindo um peso x^i de certa coluna A_i. Espera-se que o resíduo b^k esteja mais próximo da origem que o resíduo b^{k-1}.

De um certo modo, pode-se dizer que o algoritmo de ajustamento pelo par ótimo prioriza duas variáveis em cada iteração, porque encontra o valor ótimo para duas coordenadas e ajusta o restante das coordenadas em função destes valores.

Resumidamente, este algoritmo identifica os vetores A_{s+} e A_{s-} que formam o maior e o menor ângulo com o vetor b^{k-1}, respectivamente. Em seguida, encontra os valores x^k_{s+} , x^k_{s-} e λ, em que $x^k_j = \lambda\, x^{k-1}_j$ para todo $j \neq s+$ e $j \neq s-$, que minimizam a distância de b^k a origem satisfazendo a convexidade e as restrições de não negatividade. Este subproblema de otimização tem solução facilmente obtida analisando as condições de Karush-Kuhn-Tucker (KKT).

2.3 Algoritmo de ajustamento ótimo para *p* coordenadas

Conforme já dito anteriormente, o algoritmo de ajustamento ótimo para p coordenadas é uma generalização do algoritmo de ajustamento pelo par ótimo desenvolvido em Gonçalves (2009). A grande vantagem deste algoritmo é a sua simplicidade, uma vez que a cada iteração é necessário fazer apenas multiplicação de matriz por vetor e resolver um sistema de equações lineares, cuja matriz dos coeficientes é definida positiva e de ordem pequena se comparada com a dimensão dos problemas de grande porte.

O algoritmo começa identificando as s_1 e s_2 colunas que fazem o maior e o menor ângulo com o vetor b^{k-1}, respectivamente, em que $s_1 + s_2 = p$ e p é o número de colunas a ser priorizada. Depois, um subproblema de otimização é resolvido e, finalmente, o resíduo e o ponto corrente são atualizados. Aqui, a resolução do subproblema é feita utilizando método de pontos interiores, uma vez que o número de casos a ser considerado, que satisfazem as condições de KKT, cresce exponencialmente com o valor de p.

Passos:

Dado: $x^0 \geq 0$, com $e^t x^0 = 1$. Calcular $b^0 = Ax^0$.
Para $k = 1, 2, 3, \ldots$:

1) Calcular:

$\left\{A_{\eta_1^+}, \ldots, A_{\eta_{s_1}^+}\right\}$ que formam o maior ângulo com b^{k-1}.
$\left\{A_{\eta_1^-}, \ldots, A_{\eta_{s_2}^-}\right\}$ que formam o menor ângulo com b^{k-1} e tal que $x^{k-1} > 0$,
$i = \eta_1^-, \ldots, \eta_{s_2}^-$, em que $s_1 + s_2 = p$.

$$v^{k-1} = min_{i=1,\ldots,s_1} A_{\eta_i^+}^t b^{k-1} .$$

2) Se $v^{k-1} > 0$, então PARE. O problema (1) é infactível.

3) Resolver o subproblema:

$$\begin{aligned} \text{Min} \quad & \|b\|^2 \\ \text{s.a.} \quad & \lambda_0\left(1 - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1}\right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} = 1 \qquad (2) \\ & \lambda_{\eta_i^+} \geq 0, \text{ para } i = 1, \ldots, s_1, \\ & \lambda_{\eta_j^-} \geq 0, \text{ para } j = 1, \ldots, s_2. \end{aligned}$$

em que, $$b = \lambda_0\left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} A_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} A_{\eta_j^-}\right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} A_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} A_{\eta_j^-}$$

4) Atualizar:

$$b^k = \lambda_0\left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} A_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} A_{\eta_j^-}\right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} A_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} A_{\eta_j^-}$$

$$u^k = \|b^k\|$$

$$x_j^k = \begin{cases} \lambda_0 x_j^{k-1} & j \notin \left\{\eta_{1,.}^+ \ldots, \eta_{s_1}^+, \eta_{1,.}^- \ldots, \eta_{s_2}^-\right\} \\ \lambda_{\eta_i^+}, & j = \eta_i^+ ; i = 1, \ldots, s_1 \\ \lambda_{\eta_j^-}, & j = \eta_j^- ; i = 1, \ldots, s_{2.} \end{cases}$$

$k = k + 1$.

2.3.1 Solução do subproblema

Em cada iteração do algoritmo de ajustamento ótimo para p coordenadas é necessário resolver o Subproblema (2). No caso $p = 2$, que é o algoritmo de ajustamento pelo par ótimo, o subproblema é resolvido verificando todas as possíveis soluções factíveis das condições de KKT, que neste caso são 7. No caso geral, que é o algoritmo para p coordenadas, se o subproblema é resolvido seguindo o mesmo raciocínio, o número de casos possíveis de soluções factíveis cresce exponencialmente com o valor de p e este número é, exatamente, $2^{(p+1)}-1$.

Este fato torna inviável a implementação do algoritmo de ajustamento ótimo para p coordenadas para valores razoavelmente grandes de p. Com a finalidade de contornar este problema, o Subproblema (2) é abordado de outra forma e, então, é resolvido aplicando métodos de pontos interiores. A grande vantagem de usar métodos de pontos interiores é que o custo computacional para resolver um problema de grande porte não é significativo, pois, por exemplo, se quisermos modificar $p = 10$ coordenadas é muito menos custoso fatorar uma matriz definida positiva de ordem 11 do que verificar 2^{11} casos. Para mais detalhes sobre como resolver o Subproblema (2) veja Silva (2009).

3. Ponto inicial no método de pontos interiores

A heurística de Mehrotra é utilizada para determinar um ponto inicial para o método de pontos interiores no código do PCx. Esta heurística consiste em:

Passos:

1) Resolver mínimos quadrados para calcular os pontos:

$$\tilde{y} = \left(AA^t\right)^{-1} Ac \ ,$$
$$\tilde{z} = c - A^t \tilde{y} \ ,$$
$$\tilde{x} = A^t \left(AA^t\right)^{-1} b$$

2) Encontrar os valores δ_x e δ_z tais que $\tilde{x} + \delta_x$ e $\tilde{z} + \delta_z$ sejam não-negativos:

$$\delta_x = \max(-1.5\min\{\ \tilde{x}_i\ \},\ 0),$$
$$\delta_z = \max(-1.5\min\{\ \tilde{z}_i\ \},\ 0).$$

3) Determinar δ_x e δ_z tais que os pontos x^0 e z^0 sejam centralizados:

$$\tilde{\delta}_x = \delta_x + \frac{\left(\tilde{x} + \delta_x e\right)^t \left(\tilde{z} + \delta_z e\right)}{2\sum_{i=1}^{n} \left(\tilde{z}_i + \delta_z\right)}$$

$$\tilde{\delta}_z = \delta_z + \frac{\left(\tilde{x} + \delta_x e\right)^t \left(\tilde{z} + \delta_z e\right)}{2\sum_{i=1}^{n} \left(\tilde{x}_i + \delta_x\right)}$$

4) Calcular os pontos iniciais:

$$y^0 = \tilde{y} \ ,$$
$$z^0 = \tilde{z} + \tilde{\delta}_z e \ ,$$
$$x^0 = \tilde{x} + \tilde{\delta}_x e$$

Normalmente, para melhorar o desempenho dos métodos de pontos interiores são introduzidas modificações após o Passo 4 do algoritmo descrito acima, diferentemente da nossa proposta, que consiste em realizar algumas iterações do algoritmo de ajustamento ótimo para *p* coordenadas antes da etapa de centralização (Passo 2). Uma explicação para isto é que, se o algoritmo for utilizado depois de centralizar os pontos, estes são melhorados, porém a centralidade, que é importante para os métodos de pontos interiores, pode ser perdida. O ponto inicial para o algoritmo de ajustamento ótimo para *p* coordenadas é o ponto determinado ao resolver mínimos quadrados no Passo 1.

Mais detalhes sobre a heurística de Mehrotra podem ser encontrados em Mehrotra (1992) e Czyzyk *et al*. (1999).

4. Experimentos Computacionais

Os experimentos computacionais foram realizados em um Intel Core 2 Duo T7250, 2GB RAM, 2GHz and 250GB hd.

O algoritmo de ajustamento ótimo para *p* coordenadas foi implementado em linguagem C e incorporado ao código do PCx.

Para analisar o desempenho do PCx com a abordagem proposta foram resolvidos 76 problemas, alguns com acesso livre na internet (NETLIB, QAPLIB e Kennington) e outros problemas gentilmente cedidos por Gonçalves.

O critério de parada usado para o algoritmo de ajustamento ótimo para *p* coordenadas foi o número máximo de iterações (100) ou o erro relativo da norma residual menor que 10^{-4}. Aquele que ocorrer primeiro.

Por meio de experimentos numéricos, o seguinte critério para a escolha do valor de *p* foi estabelecido (*m* é o número de linhas do problema):

$$\begin{aligned}
0 < m \leq 100 &\Rightarrow p = 2 \\
100 < m \leq 2000 &\Rightarrow p = 8 \\
2000 < m \leq 15000 &\Rightarrow p = 10 \\
15000 < m \leq 30000 &\Rightarrow p = 20 \\
30000 < m \leq 150000 &\Rightarrow p = 40 \\
150000 < m &\Rightarrow p = 60
\end{aligned}$$

Na Tabela 1, são comparados o número de iterações e o tempo total de resolução da versão do PCx com o algoritmo de ajustamento ótimo (PCxMod) e sem este algoritmo (PCx).

A coluna **Dimensões** traz o número de linhas e colunas dos problemas após o pré-processamento. Na coluna ***p*** estão os valores de *p* usados nos testes e na coluna **It-aux** o número de iterações realizadas pelo algoritmo de ajustamento ótimo.

Problema	Dimensões				Iterações		Tempo (s)		Coleção
	Linha	Coluna	*p*	It-aux	PCx	PCxMod	PCx	PCxMod	
80bau3b	2140	11066	8	2	36	36	**0.4**	0.44	NETLIB
agg2	514	750	4	2	21	21	0.1	**0.07**	NETLIB
agg3	514	750	4	2	19	19	0.12	**0.07**	NETLIB
czprob	671	2779	4	2	26	**25**	0.05	0.05	NETLIB
degen3	1503	2604	4	10	15	15	**0.77**	0.8	NETLIB
dfl001	5984	12143	8	2	45	**41**	77.27	**69.27**	NETLIB
etamacro	334	669	4	10	26	**25**	0.04	0.04	NETLIB
fffff800	322	826	4	2	29	**28**	0.07	0.07	NETLIB
fit2d	25	10524	2	2	22	22	**0.52**	0.55	NETLIB
fit2p	3000	13525	8	2	20	20	**0.3**	0.36	NETLIB
maros-r7	2152	7440	8	2	12	12	**2.25**	2.29	NETLIB
modszk1	665	1599	4	4	20	**19**	**0.04**	0.05	NETLIB
perold	593	1389	4	2	32	32	0.15	**0.12**	NETLIB
pilot	1368	4543	4	2	34	**30**	2.06	**1.93**	NETLIB
pilot87	1971	6373	4	2	25	25	5.19	**5.15**	NETLIB
pilotwe	701	2814	4	6	45	**44**	0.2	**0.16**	NETLIB
scfxm3	915	1704	4	2	19	19	0.08	**0.06**	NETLIB
seba	448	901	4	2	13	13	0.25	**0.24**	NETLIB
ship08l	470	3121	4	2	14	14	0.06	**0.04**	NETLIB
stocfor3	15362	22228	10	2	30	30	**1.16**	1.21	NETLIB
wood1p	171	1718	4	10	22	22	**0.26**	0.3	NETLIB
cre-a	2994	6692	8	2	23	23	0.25	**0.23**	Kennington
cre-b	5336	36382	8	2	35	35	**3.21**	3.34	Kennington
cre-c	2375	5412	8	2	25	25	**0.19**	0.21	Kennington
cre-d	4102	28601	8	2	35	35	**2.73**	2.9	Kennington
ken-11	10085	16740	8	2	20	20	**0.62**	0.72	Kennington
ken-13	22534	36561	10	2	23	23	**1.89**	2.24	Kennington
ken-18	78862	128434	20	2	26	26	**15.96**	18.14	Kennington
osa-07	1081	25030	4	2	22	22	**0.56**	0.6	Kennington
osa-14	2300	54760	8	2	25	25	**1.65**	1.74	Kennington
osa-30	4313	104337	8	4	24	**21**	**3.86**	4.8	Kennington
osa-60	10243	243209	8	5	31	**24**	**15.38**	17.06	Kennington
bl	5729	12462	8	2	32	32	**1.48**	1.54	Gonçalves
bl2	5729	12462	8	7	37	**35**	**1.69**	1.71	Gonçalves
co5	4849	10787	8	2	**47**	48	**1.72**	1.8	Gonçalves

co9	9090	19997	8	20	*	**46**	*	**6.14**	Gonçalves
cq9	8004	19317	8	2	46	**44**	4.67	**4.59**	Gonçalves
ex01	235	1556	4	2	25	25	0.12	**0.1**	Gonçalves
ex02	227	1548	4	3	31	**30**	0.11	0.11	Gonçalves
ex05	832	7805	4	4	32	**31**	0.68	0.68	Gonçalves
ex06	825	7797	4	9	63	**58**	1.17	**1.16**	Gonçalves
ex09	1821	18184	4	2	37	**36**	**1.61**	1.63	Gonçalves
fort45	1152	1582	4	12	17	**15**	0.09	**0.08**	Gonçalves
fort47	1152	1582	4	2	16	16	0.11	**0.07**	Gonçalves
fort48	1152	1582	4	10	15	**12**	0.1	**0.07**	Gonçalves
fort53	1156	1586	4	3	17	**16**	0.11	**0.08**	Gonçalves
fort56	1156	1586	4	3	17	17	0.11	**0.09**	Gonçalves
fort58	1156	1586	4	3	17	17	0.13	**0.09**	Gonçalves
fort59	1156	1586	4	3	16	16	0.12	**0.08**	Gonçalves
fort60	1156	1586	4	19	17	**16**	0.13	**0.11**	Gonçalves
fort61	1156	1586	4	19	18	**16**	0.12	**0.11**	Gonçalves
ge	9150	14990	8	4	43	**37**	2.17	**2.1**	Gonçalves
nl	6665	14680	8	2	33	33	2.62	**2.61**	Gonçalves
x1	1050	1480	4	2	10	10	0.08	**0.04**	Gonçalves
x2	1050	1480	4	10	20	**16**	**0.07**	0.08	Gonçalves
els19	4350	13186	8	10	20	20	145.35	**144.32**	QAPLIB
chr25a	8149	15325	8	10	23	**21**	40.65	**37.08**	QAPLIB
chr22b	5587	10417	8	10	21	21	**15.58**	15.79	QAPLIB
scr15	2234	6210	8	10	16	16	**22.48**	22.69	QAPLIB
scr20	5079	15980	8	2	**14**	15	281.85	**266.86**	QAPLIB
rou20	7359	37640	8	2	13	13	1524.12	**1459.71**	QAPLIB
nug06	372	486	4	4	21	**10**	0.19	**0.08**	QAPLIB
nug07	602	931	4	2	9	9	0.26	**0.23**	QAPLIB
nug08	912	1632	4	3	*	**7**	*	**0.76**	QAPLIB
pds-02	2609	7339	8	2	24	24	**0.25**	0.28	QAPLIB
pds-06	9156	28472	8	10	29	29	**8.18**	8.49	QAPLIB
pds-10	15648	48780	10	2	33	33	42.51	**41.49**	QAPLIB
pds-20	32287	106080	20	2	43	**39**	407.83	**392.01**	QAPLIB
pds-30	47968	156042	20	9	**43**	44	1369.47	**1338.08**	QAPLIB
pds-40	64276	214385	20	2	**50**	51	**4506.78**	4644.72	QAPLIB
pds-50	80339	272513	20	2	52	52	8406.11	**8318.49**	QAPLIB
pds-60	96514	332862	20	10	52	**50**	14293.89	**13687.98**	QAPLIB

pds-70	111896	386238	20	2	55	55	24542.93	**23943.35**	QAPLIB
pds-80	126120	430800	20	10	54	**51**	33795.18	**32035.45**	QAPLIB
pds-90	139752	471538	20	2	50	50	38046.37	**38032.3**	QAPLIB
pds-100	152300	498530	40	2	57	57	**47793.96**	48147.86	QAPLIB

Tabela 1: Comparação do desempenho de PCx e PCxMod
* significa que o método falhou

O uso do algoritmo de ajustamento ótimo para *p* coordenadas dentro da heurística de Mehrotra determinou melhores pontos iniciais que fizeram reduzir o número total de iterações em 38% dos problemas testados e o tempo total de resolução foi menor em 55% dos problemas.

O PCx realizou um número de iterações menor em cerca de 5% dos problemas e resolveu mais rapidamente aproximadamente 38% dos problemas.

Um resultado importante é que dois problemas (co9 e nug08) foram resolvidos somente pelo PCxMod.

Vale ressaltar que o tempo total necessário para obter uma solução para o algoritmo de ajustamento ótimo para *p* coordenadas não é significativo em relação ao tempo total de resolução dos problemas. No pior caso, este tempo foi de aproximadamente 28 segundos para o problema do pds-80.

5. Conclusões e Trabalhos futuros

Neste trabalho, o algoritmo de ajustamento ótimo para *p* coordenadas foi usado em conjunto com o método de pontos interiores para determinar pontos iniciais bons, permitindo que o método tenha um desempenho melhor e, consequentemente, convirja mais depressa. As principais vantagens deste algoritmo são simplicidade e avanço inicial rápido.

Ao incorporar o algoritmo de ajustamento ótimo para *p* coordenadas no código do PCx, mais especificamente, após o Passo 1 da heurística de Mehrotra, e realizar poucas iterações de tal algoritmo para determinar melhores pontos iniciais, uma implementação mais robusta foi obtida. Os experimentos computacionais em um conjunto grande de problemas de programação linear mostraram que, com esta abordagem, foi possível reduzir o número total de iterações em quase 40% dos problemas testados e o tempo total de resolução foi menor em mais de 50% dos problemas. A redução no tempo ocorreu, principalmente, nos problemas de maiores dimensões, o que é mais importante. Além disso, alguns problemas que não são resolvidos pelo PCx sem a incorporação do algoritmo de ajustamento ótimo, utilizando a abordagem proposta passaram a ser solucionados.

Como trabalhos futuros novas estratégias para a escolha de *p* serão desenvolvidas, uma vez que, após vários experimentos foram encontrados valores de *p* que reduzem o número de iterações em cerca de 90% dos problemas de testados e também outros critérios de parada para o algoritmo de ajustamento ótimo para *p* coordenadas serão investigados, pois o número de iterações realizadas por tal algoritmo influencia diretamente na qualidade do ponto inicial determinado.

Agradecimentos

Ao CNPq e CAPES pelo apoio financeiro.

Referências

Czyzyk, J., Mehrotra, S., Wagner, M., Wright, S.J. (1999) PCx an interior point code for linear programming. Optimization Methods & Software, 11-2, 397–430.

Dantzig, G.B. (1991) Converting a converging algorithm into a polynomially bounded algorithm. Tech. rep., Stanford University, SOL 91-5.

Dantzig, G.B. (1992) An ϵ-precise feasible solution to a linear program with a convexity constraint in $1/\epsilon^2$ iterations independent of problem size. Tech. rep., Stanford University, SOL 92-5.

Epelman, M., Freund, R.M. (2000) Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. Mathematical Programming, 88, 451–485.

Gonçalves, J.P.M. (2004) A family of linear programming algorithms based on the von Neumann algorithm. PhD thesis, Lehigh University, Bethlehem.

Gonçalves, J.P.M., Storer, R.H., Gondzio, J. (2009) A family of linear programming algorithms based on an algorithm by von Neumann. Optimization Methods and Software, 24,461–478.

Mehrotra, S. (1992) Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods, ORSA Journal on Computing, 4, 103–118.

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

Silva, J. (2009) Uma família de algoritmos para programação linear baseada no algoritmo de Von Neumann. Tese, IMECC – UNICAMP, Campinas SP.