Tendências em Matemática Aplicada e Computacional, **2** (2001), 175-184.
Sociedade Brasileira de Matemática Aplicada e Computacional.

Experimentos com Programas Lineares por Partes em Redes

C. PERIN[1], M. MELLO[2], Departamento de Matemática Aplicada, Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, UNICAMP, Cx.P. 6065, 13083-200 Campinas, SP, Brasil

F.A.S. MARINS[3], Departamento de Engenharia de Produção, Campus de Guaratinguetá, Universidade Estadual Paulista, UNESP, Cx.P. 205, 12500-000 Guaratinguetá, SP, Brasil.

Resumo. A estratégia usual de resolução de um problema de fluxo em redes lineares por partes é transformá-lo em um problema de fluxo em uma rede equivalente. Neste caso, uma rede linear por partes com n nós e m arcos com k intervalos por arco é convertida em uma rede linear com n nós e km arcos. Métodos de pontos interiores têm tido um grande sucesso na solução de problemas de fluxo em redes de grande porte devido ao baixo tempo computacional dispendido. Mostramos que é vantajoso especializar o método de pontos interiores para problemas lineares de modo a resolver diretamente problemas de fluxo em redes lineares por partes, ao invés de utilizar sua versão genérica para resolver a forma equivalente de fluxo em redes. Realizamos também testes computacionais para efetuar diversas comparações entre: o uso da estratégia preditor-corretor versus preditor puro, estratégias de inicialização do corretor e dos critérios de parada no preditor e no corretor.

1. Introdução

Uma importante área de estudos e aplicações da Programação Matemática é a Programação Linear por Partes que está relacionada com a otimização de uma função objetivo separável, convexa e linear por partes. Neste trabalho, é dada uma atenção especial à sub-área denominada de Programação Linear por Partes em Redes que apresenta relevantes aplicações no mundo real, tais como em: sistemas de potência, redes hidráulicas, transportes, redes de comunicações, administração hídrica, e mesmo problemas estocásticos de fluxo em redes. Já investigou-se [2] a especialização do método simplex para redes para o caso linear por partes com sucesso.

Damos prosseguimento à pesquisa relatada em [6]. Naquele trabalho foi descrita a implementação de uma adaptação do método de pontos interiores para problemas

[1] clovis@ime.unicamp.br
[2] margarid@ime.unicamp.br
[3] fmarins@feg.unesp.br

de fluxo lineares por partes. Esta implementação é agora modificada, incorporando melhorias propostas por [5], especificamente no que se refere à utilização de preditor-corretor. Cabe notar que as implementações usuais do esquema de resolução preditor-corretor de [5] utilizam a decomposição de Cholesky, como, por exemplo, [1, 4]. Na nossa implementação, no entanto, seguimos [7] e resolvemos os sistemas lineares, produzidos pelo esquema preditor-corretor, pelo método de Gradientes Conjugados, que tira proveito da estrutura da matriz (de incidência) de coeficientes do problema, conforme explicitado em [6].

Os algoritmos são descritos na seção 2. Suas implementações são o objeto da seção 3. Os testes computacionais são relatados e comentados na seção 4.

2. Algoritmos

O problema de fluxo em redes lineares por partes é definido por

$$\min\{f(x) \mid Ax = b, x \geq 0\},$$

onde A é a $n \times m$-matriz de incidência de uma rede com n nós e m arcos, b é o n-vetor de demandas dos nós, x é um m-vetor de fluxos nos arcos, a ser determinado, e $f(x) = \sum_{j=1}^{m} f_j(x_j)$ é uma função convexa, separável e linear por partes. As funções convexas e lineares por partes f_j podem ser especificadas pelo par de ℓ-vetores c, d, onde $\ell = \sum_{j=1}^{m} \ell_j$ é o número total de intervalos, sendo ℓ_j o número de intervalos de f_j.

Este problema pode ser transformado, ver [3], em um problema de fluxo em redes lineares definido por

$$\min\{c'\bar{x} \mid \bar{A}\bar{x} = b, \bar{x} + \bar{s} = d, \bar{x}, \bar{s} \geq 0\}.$$

Neste caso, a $n \times \ell$-matriz $\bar{A}$ é obtida a partir da matriz A com a replicação de cada uma de suas colunas tantas vezes quantos são os intervalos a ela associados. Estamos supondo que esta replicação também é feita no vetor x transformando-o no vetor $\bar{x}$. O problema dual associado é dado por

$$\max\{b'y - d'\bar{w} \mid \bar{A}'y - \bar{z} + \bar{w} = c, \bar{z}, \bar{w} \geq 0\}$$

e as condições de folgas complementares são expressas por

$$\overline{X}\,\overline{Z}\mathbf{1} = 0 \qquad \text{e} \qquad \overline{S}\,\overline{W}\mathbf{1} = 0,$$

onde $\overline{X}, \overline{S}, \overline{Z}, \overline{W}$ são matrizes diagonais com os elementos de $\bar{x}, \bar{s}, \bar{z}, \bar{w}$, respectivamente, e $\mathbf{1}$ é um vetor de 1's.

Foram estudadas duas versões de métodos de pontos interiores: a versão primal-dual usual (preditor puro) e a versão primal-dual com preditor-corretor.

Conforme descrito em [6], o método primal-dual usual de pontos interiores é inicializado a partir de um iterando $\bar{x}, \bar{s}, y, \bar{z}, \bar{w}$ satisfazendo $\bar{x}, \bar{s}, \bar{z}, \bar{w} > 0$ e a cada

iteração é obtida uma direção de deslocamento $\Delta\bar{x}, \Delta\bar{s}, \Delta y, \Delta\bar{z}, \Delta\bar{w}$ em conformidade com um parâmetro de centragem μ de acordo com as equações (2.1) abaixo

$$\begin{aligned} \bar{A}\Delta\bar{x} &= b - \bar{A}\bar{x}, \\ \Delta\bar{x} + \Delta\bar{s} &= d - \bar{x} - \bar{s}, \\ \bar{A}'\Delta y + \Delta\bar{z} - \Delta\bar{w} &= c - \bar{A}'y - \bar{z} + \bar{w}, \\ \overline{X}\Delta\bar{z} + \overline{Z}\Delta\bar{x} &= \mu\mathbf{1} - \overline{X}\bar{z}, \\ \overline{S}\Delta\bar{w} + \overline{W}\Delta\bar{s} &= \mu\mathbf{1} - \overline{S}\bar{w}, \end{aligned} \tag{2.1}$$

que refletem a factibilidade primal, a factibilidade dual, e a complementaridade das soluções quando $\mu = 0$.

O método primal-dual usual de pontos interiores pode ser esquematizado da seguinte forma:

`let` $\bar{x}, \bar{s}, y, \bar{z}, \bar{w}$ `such that` $\bar{x}, \bar{s}, \bar{z}, \bar{w} > 0$
`while` (critério de parada não é satisfeito) `do`
  `update` μ
  `obtain` $\Delta\bar{x}, \Delta\bar{s}, \Delta y, \Delta\bar{z}, \Delta\bar{w}$
  `find` α `such that` $(\bar{x}, \bar{s}, \bar{z}, \bar{w}) + \alpha(\Delta\bar{x}, \Delta\bar{s}, \Delta\bar{z}, \Delta\bar{w}) > 0$
  `update` $(\bar{x}, \bar{s}, y, \bar{z}, \bar{w}) \leftarrow (\bar{x}, \bar{s}, y, \bar{z}, \bar{w}) + \alpha(\Delta\bar{x}, \Delta\bar{s}, \Delta y, \Delta\bar{z}, \Delta\bar{w})$.

Já no método primal-dual com preditor-corretor são resolvidos dois sistemas: o preditor (2.1) e o corretor (2.2) abaixo, que possui a mesma matriz de coeficientes no lado esquerdo:

$$\begin{aligned} \bar{A}\Delta\bar{x} &= b - \bar{A}\bar{x}, \\ \Delta\bar{x} + \Delta\bar{s} &= d - \bar{x} - \bar{s}, \\ \bar{A}'\Delta y + \Delta\bar{z} - \Delta\bar{w} &= c - \bar{A}'y - \bar{z} + \bar{w}, \\ \overline{X}\Delta\bar{z} + \overline{Z}\Delta\bar{x} &= \mu\mathbf{1} - \overline{X}\bar{z} - \delta\overline{X}\delta\bar{z}, \\ \overline{S}\Delta\bar{w} + \overline{W}\Delta\bar{s} &= \mu\mathbf{1} - \overline{S}\bar{w} - \delta\overline{S}\delta\bar{w}, \end{aligned} \tag{2.2}$$

onde $\delta\bar{x}$, $\delta\bar{z}$, $\delta\bar{s}$ e $\delta\bar{w}$ correspondem à solução do sistema (2.1).

O vetor $(\Delta\bar{x}, \Delta\bar{s}, \Delta\bar{z}, \Delta\bar{w})$ do algoritmo é obtido pela resolução de (2.1) na versão preditor puro e de (2.2) na versão preditor-corretor. O parâmetro α é obtido por um simples teste da razão na versão preditor puro e de forma especial, como descrito em [1, 5].

Utilizamos a estratégia de resolver cada um dos sistemas preditor e corretor por meio da resolução de um $n \times n$-sistema SDP pelo método do gradiente conjugado. A matriz deste sistema é utilizada implicitamente percorrendo os arcos da rede do problema. Estes sistemas SDPs são resolvidos pelo método do gradiente conjugado com precondicionamento diagonal nas primeiras seis iterações do método primal-dual e com o precondicionamento da árvore geradora nas iterações subseqüentes. Este procedimento baseia-se em estratégia similar adotada em [7].

Os critérios de parada adotados são: (1) número máximo de iterações é excedido; (2) soluções quase-complementares $\bar{x}'\bar{z} + \bar{s}'\bar{w} \approx 0$ são encontradas; (3) limite inferior para o valor da função objetivo primal é excedido; (4) limite superior para o valor da função objetivo dual é excedido.

3. Implementação

Foi utilizada a linguagem C na programação dos algoritmos. Foram implementados e testados 4 pares de programas com diferentes versões do método de pontos interiores; 4 para programas em redes lineares e 4 para progamas em redes lineares por partes. Um par (*FL9, FP9*) é do tipo usual ou preditor puro e os 3 outros (*FL0–2, FP0–2*) são do tipo preditor-corretor. O sistema corretor é construído de tal forma que a sua solução já componha a direção de deslocamento do iterando. Assim, podemos aplicar o método do gradiente conjugado utilizando a solução do sistema preditor como solução de partida do sistema corretor. A solução de partida do sistema preditor é a solução nula e a solução de partida do sistema corretor é a solução final do primeiro sistema, em *FL0–1, FP0–1*, e é a solução nula, no par *FL2, FP2*. O critério de parada do gradiente conjugado sempre utiliza a tolerância de 10^{-8} para o erro relativo da solução corrente no último sistema. Apenas o par *FL1, FP1* utiliza a tolerância de 10^{-6} para o erro relativo da solução corrente no primeiro sistema. Em resumo, os programas são:

FL9 – preditor puro para redes lineares com tolerância de 10^{-8} no erro relativo da solução corrente para terminar o gradiente conjugado.

FL0 – preditor-corretor com tolerâncias de 10^{-8}, 10^{-8}; utiliza a solução final do sistema preditor como solução de partida do sistema corretor.

FL1 – preditor-corretor com tolerâncias de 10^{-6}, 10^{-8}; utiliza a solução final do sistema preditor como solução de partida do sistema corretor.

FL2 – preditor-corretor com tolerâncias de 10^{-8}, 10^{-8}; utiliza a solução nula na partida do sistema corretor.

FP9 – preditor puro para redes lineares por partes com tolerância de 10^{-8} no erro relativo da solução corrente para terminar o gradiente conjugado.

FP0 – preditor-corretor com tolerâncias de 10^{-8}, 10^{-8}.

FP1 – preditor-corretor com tolerâncias de 10^{-6}, 10^{-8}.

FP2 – preditor-corretor com tolerâncias de 10^{-8}, 10^{-8}; utiliza a solução nula na partida do sistema corretor.

O espaço de memória requerido por estes programas para armazenar os dados da rede, das soluções correntes, da aplicação do gradiente conjugado e de outras variáveis auxiliares pode ser resumido em: (*FL9*) 13 n-vetores e 16 ℓ-vetores; (*FL0–2*) 15 n-vetores e 19 ℓ-vetores; (*FP9*) 13 n-vetores, 4 m-vetores e 13 ℓ-vetores; (*FP0–2*) 15 n-vetores, 4 m-vetores e 16 ℓ-vetores.

As tolerâncias utilizadas nos critérios de parada são: 10^{-8} (para tolerância relativa no teste de complementaridade), -10^{8} (para limite inferior da função objetivo primal) e 10^{8} (para limite superior da função objetivo dual).

4. Testes Computacionais

Os programas foram testados em um micro computador Pentium III, 500MHz de clock e 328MB de memória RAM utilizando problemas em redes conectadas geradas aleatoriamente com até 100 mil nós, 1 milhão de arcos e 1 milhão de intervalos. As redes correspondem a problemas de transporte modificados. À estrutura de digrafo bipartido com igual número de nós de oferta e de demanda acrescentou-se um ciclo orientado contendo os nós de oferta e outro contendo os nós de demanda, com o objetivo de tornar os problemas gerados factíveis. Todos os nós das redes geradas apresentam o mesmo grau. As ofertas/demandas, os coeficientes de custos e os coeficientes de capacidades foram geradas com distribuição uniforme em intervalos fornecidos como parâmetros do programa gerador. As tabelas a seguir contêm os tempos de CPU (em segundos), o número de iterações do método primal-dual e o número total de iterações do método do gradiente conjugado executados em cada programa. Estes dados foram colhidos em 5 testes para cada configuração de rede (mesmo conjunto de parâmetros e sementes diferentes) e as médias dos 5 valores obtidos nestes testes arredondadas para o inteiro mais próximo.

Exemplares iniciais. Primeiramente foram testados exemplares de programas relativamente pequenos, variando os parâmetros fornecidos para o gerador procurando definir direções para este estudo. Os parâmetros envolveram os intervalos de geração das ofertas/demandas nos nós, os intervalos de geração dos ceoficientes de custo, os intervalos de geração das capacidades, o número de nós, o número de arcos e o número de intervalos. Os dados considerados mais promissores correspondem aos exemplares de problemas da Tabela 1 com ofertas/demandas, custos e capacidades gerados aleatoriamente com distribuição uniforme no intervalo (0,100).

A parte superior da Tabela 1 contém as dimensões das redes e sua parte inferior apresenta os tempos de CPU obtidos com os 9 grupos de exemplares que foram resolvidos com as oito versões. Estes dados são também apresentados sob forma gráfica na Figura 1 (*FL0, FL1, FL2, FP0, FP1* e *FP2*) e na Figura 2 (*FL9, FL0, FP9* e *FP0*). Pode-se constatar o desempenho superior das versões especializadas para redes lineares por partes. Em cada uma das classes 0–2 de preditor-corretor as versões do tipo 1 (*FP1, FL1*) são ligeiramente superiores às demais (vide Figura1). Na Figura 2 pode-se verificar que apenas no grupo G9 a versão preditor-corretor (aqui representada pela implementação *FL0, FP0*) deixa de ser superior em relação à versão preditor puro *FP9*.

Variando o número de intervalos. A parte superior da Tabela 2 apresenta os tempos de CPU obtidos com os oito programas em exemplares gerados com 10000 nós, 35000 arcos e número intervalos variando entre 70000 e 1050000. Estes dados constatam um desempenho superior das versões especializadas para redes lineares por partes. Além disto, o programa *FP9* apresenta-se melhor do que os demais para quase todos os exemplares (vide Figura 3).

Variando o número de arcos. A parte central da Tabela 2 apresenta os tempos de CPU obtidos com os oito programas em exemplares gerados com 10000

Grupos	G1	G2	G3	G4	G5	G6	G7	G8	G9
# nós (10^3)	10	10	10	10	10	10	10	10	10
# arcos (10^3)	20	20	20	35	35	35	50	50	50
# inter. (10^3)	40	100	160	70	105	280	100	250	400
FL9	988	859	772	913	674	636	857	649	775
FL0	175	295	468	301	655	1014	447	1002	1576
FL1	169	280	447	291	640	980	432	980	1613
FL2	186	313	502	322	721	1085	479	1097	1740
FP9	846	637	513	744	456	369	670	409	**398**
FP0	125	168	205	200	281	358	284	425	533
FP1	**120**	**154**	**186**	**192**	**258**	**318**	**256**	**367**	478
FP2	130	165	197	202	274	338	281	398	498

Tabela 1: Dimensões das redes e segundos de CPU dos exemplares iniciais, com tempos mínimos em negrito

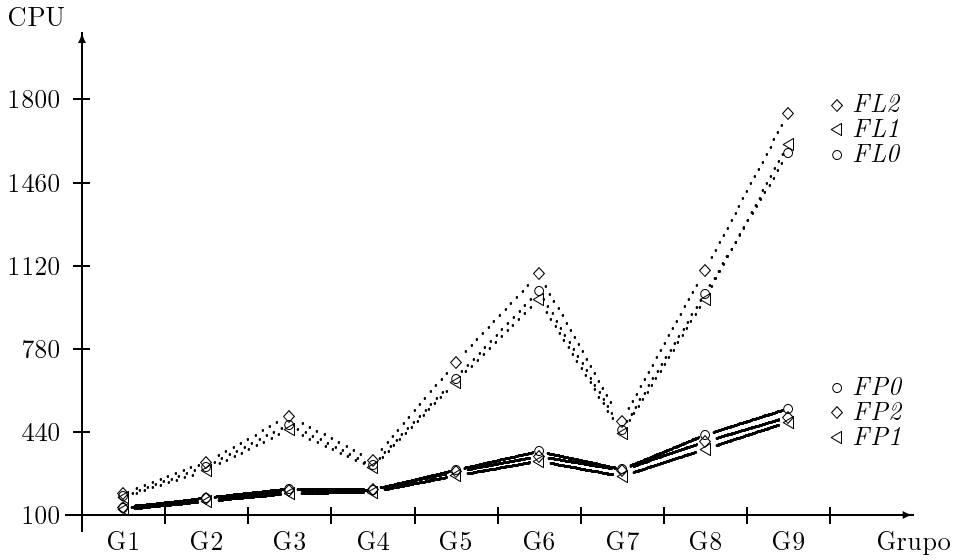

Figura 1: Segundos de CPU para as versões do tipo preditor-corretor

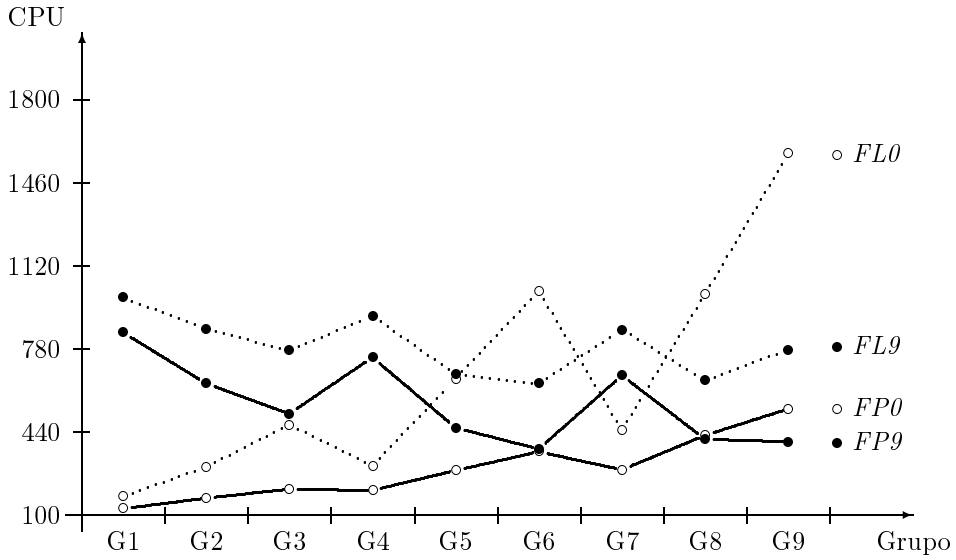

Figura 2: Segundos de CPU para *FL9, FL0, FP9* e *FP0*

Prog	*FL9*	*FL0*	*FL1*	*FL2*	*FP9*	*FP0*	*FP1*	*FP2*
intervalos	10000 nós, 35000 arcos							
70 mil	913	301	291	322	744	200	**192**	202
175 mil	674	655	640	721	456	281	**258**	274
280 mil	636	1014	980	1085	369	358	**318**	338
525 mil	877	1784	1918	1995	**382**	461	457	446
1050 mil	1806	4125	4390	4640	**529**	702	687	705
arcos	10000 nós, 5 intervalos/arco							
20 mil	859	305	280	313	637	161	**154**	165
35 mil	674	697	640	721	456	271	**258**	274
50 mil	649	1077	980	1097	409	398	**367**	398
85 mil	913	2179	1997	2259	**485**	736	675	746
160 mil	1970	5389	4978	5746	**878**	1587	1471	1647
nós	50000 arcos, 250000 intervalos							
5 mil	334	798	737	832	**165**	248	236	249
10 mil	649	1077	980	1097	409	398	**367**	398
12.5 mil	970	1171	1090	1200	650	472	**442**	480
20 mil	3120	1450	1339	1496	2371	735	**695**	739

Tabela 2: Segundos de CPU com variação de parâmetros, com mínimos em negrito

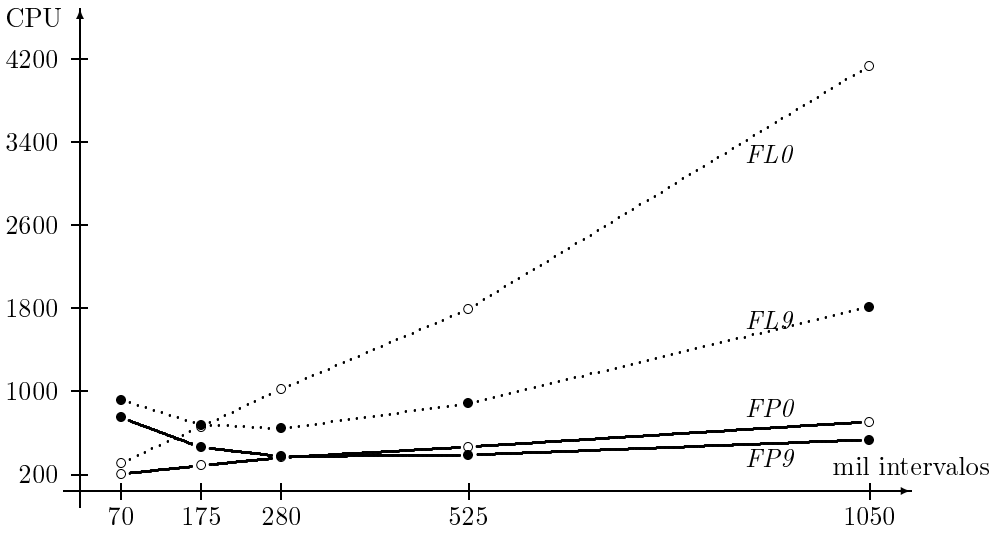

Figura 3: Segundos de CPU variando o número de intervalos

nós e número de arcos variando entre 20000 e 160000. Estes dados constatam um desempenho superior das versões especializadas para redes lineares por partes. Além disto, o programa *FP9* apresenta-se melhor do que os demais para quase todos os exemplares (vide Figura 4).

Variando o número de nós. A parte inferior da Tabela 2 apresenta os tempos de CPU obtidos com os oito programas em exemplares gerados com 50000 arcos, 250000 intervalos e número de nós variando de 5000 a 20000. Estes dados constatam um desempenho superior das versões especializadas para redes lineares por partes. Nestes testes, as versões do tipo preditor-corretor, em especial *FP1*, apresentam-se com um comportamento superior aos das demais versões (vide Figura 5).

5. Comentários Finais

Nos experimentos foram também contabilizados o número de iterações do método primal-dual e o número total de iterações do método do gradiente conjugado com precondicionamento diagonal e com precondicionamento da árvore geradora máxima. De um modo geral, o número de iterações do método primal-dual na versão usual ou preditor puro foi superior a 2 vezes o número de iterações do método primal-dual nas versões preditor-corretor. Verificou-se que seria melhor utilizar um critério para número de iterações com precondicionador diagonal que dependesse do tamanho da rede (nas redes maiores o precondicionador diagonal seria utilizado em mais do que 6 iterações iniciais). Entre as versões preditor-corretor não houve diferença significativa no número de iterações do método primal-dual. Outros tipos de variações na estrutura e nos coeficientes das redes foram testados; selecionamos para apresentação neste artigo os resultados considerados mais interessantes.

Em resumo, a versão preditor-puro parece ter um comportamento melhor apenas

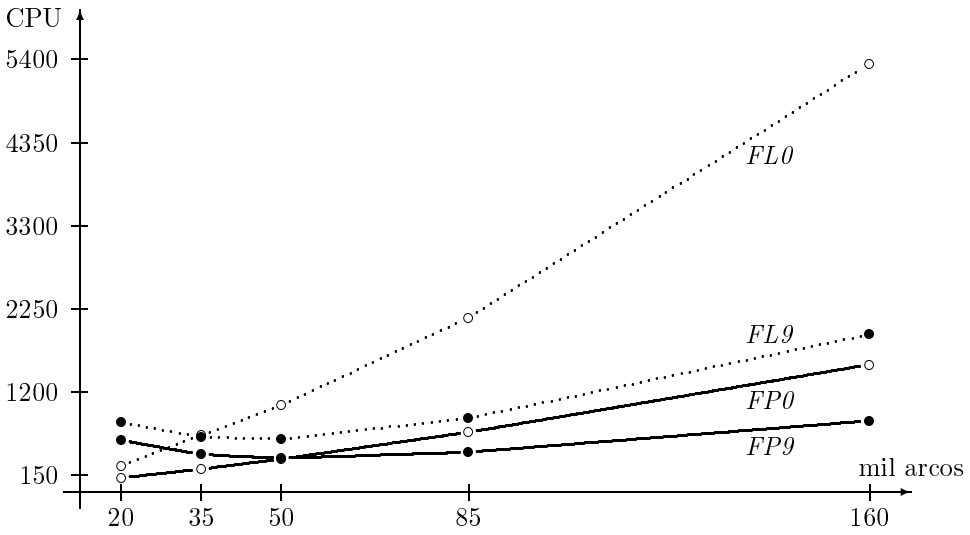

Figura 4: Segundos de CPU variando o número de arcos

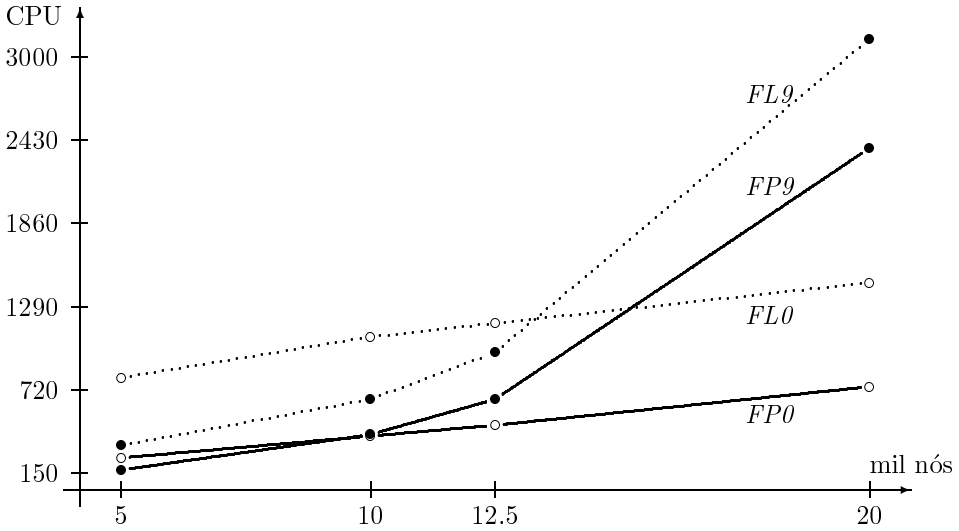

Figura 5: Segundos de CPU variando o número de nós

quando a rede tem um grande número de arcos ou intervalos em comparação com o número de nós.

Abstract. The usual approach to solving a piecewise linear network flow problem is to transform it into an equivalent linear one. In this transformation, a piecewise network with n nodes and m arcs, each with k intervals (corresponding to the linear pieces of the arc cost function), has an equivalent linear one with n nodes and mk arcs. Interior point methods have been proved successful in the solution of linear network flow problems. We show that it is advantadgeous to construct a customized interior point method to solve piecewise network problems directly, instead of applying its generic version to the equivalent linear problem. Two algorithms were implemented and tested: one using predictor-corrector and the other without the corrector step. Comparison between alternative strategies (initialization, stopping criteria) are done by means of several computacional tests.

Referências

[1] J. Czyzyk, S. Mehrotra, M. Wagner e S.J. Wright, PCx user guide, Technical Report OTC 96/01, Optimization Technology Center, 1997.

[2] K. Darby-Dowman, F.A.S. Marins, E. Senne, C. Perin e A. Machado, Algorithms for network piecewise-linear programs: a comparative study, *European Journal of Operational Research* **97** (1997), 183–199.

[3] J.K. Ho, Relationships among linear formulations of separable convex piecewise linear programs, *Mathematical Programming Study* **24** (1985), 126–140.

[4] M. Kojima, N. Megiddo e S. Mizuno, A primal-dual infeasible-interior-point algorithm for linear programming, *Mathematical Programming* **61** (1993), 263–280.

[5] S. Mehotra, On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization* **2** (1989), 575–601.

[6] C. Perin, M.P. Mello e F.A.S. Marins, Uma implementação de pontos interiores para fluxo em redes lineares por partes, in "Seleta do XXII CNMAC", Tendências em Matemática Aplicada e Computacional, Vol. 1, Parte 2, pp. 431-442, SBMAC, 2000.

[7] M. Resende e G. Veiga, An efficient implementation of a network interior point method, Technical Report, AT&T Bell Laboratories, Murray Hill, NJ, USA, 1992.