Trabalho apresentado no XXXIX CNMAC, Uberlândia - MG, 2019.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Uma Generalização do Método de Redução por Pesos

Carla T. L. S. Ghidini [1]
Faculdade de Ciências Aplicadas - UNICAMP
Domingos B. S. Lukamba [2]
Faculdade de Ciências Aplicadas - UNICAMP
Aurelio R. L. Oliveira[3]
Instituto de Matemática, Estatística e Computação Científica da Unicamp - UNICAMP
Jair Silva[4]
Campus Avançado de Jandaia do Sul - UFPR

Resumo. Neste trabalho apresentamos uma generalização do método de redução por pesos proposto na literatura, o qual foi desenvolvido com base no método de von Neumann. Ambos métodos resolvem problemas de programação linear e têm como vantagens a simplicidade e a convergência inicial rápida, porém são lentos para atingir a otimalidade. Com o objetivo de melhorar a eficiência desses métodos propomos o método de redução por pesos para p coordenadas, o qual leva em consideração p variáveis do problema ao determinar a direção em que o resíduo será movido. Os resultados dos experimentos computacionais realizados com problemas de livre acesso mostraram que o algoritmo proposto teve um melhor desempenho.

Palavras-chave. Programação Linear, von Neumann, Redução de Pesos.

1 Introdução

O método de von Neumann foi apresentado à Dantzig em 1948, por ele divulgado no início da década de 90 [1] e mais tarde estudado por Epelman e Freund [2]. Este método possui uma convergência inicial rápida, porém ele não é muito prático para resolver problemas de programação linear, dado que converge muito lentamente.

Em Gonçalves et al. [4], três novos métodos baseados no método de von Neumann foram propostos numa tentativa de obter métodos mais eficientes e entre eles está o método de redução por pesos. Este método herda as mesmas vantagens do método von Neumann, mas não é garantido que uma iteração deste método seja melhor que uma iteração do algoritmo de von Neumann. Quando isso não acontece, o algoritmo de redução por pesos pode ser facilmente modificado para que uma iteração sua seja substituída por uma iteração do algoritmo de von Neumann.

[1]carla.ghidini@fca.unicamp.br.
[2]domingos.lukamba@fca.unicamp.br.
[3]aurelio@ime.unicamp.br.
[4]jairsilva@ufpr.br.

DOI: 10.5540/03.2020.007.01.0459

Mesmo com as melhorias alcançadas por esses métodos, eles continuam apresentando algumas desvantagens que são o grande número de iterações para obter uma boa solução para o problema , dificuldade em determinar o melhor valor de p e convergência até a otimalidade lenta. Com essa motivação, desenvolvemos o método de redução por pesos generalizado, o qual é também baseado no método de von Neumann e nas ideias do método de redução por pesos, porém levando em consideração p variáveis do problema para determinar a direção em que o resíduo deve ser movido.

Neste trabalho, trataremos do problema de encontrar uma solução factível para o seguinte conjunto de restrições lineares:

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

em que $P \in \Re^{m \times n}$, $x \in \Re^n$ e $e \in \Re^n$ (vetor unitário). As colunas de P tem norma um, isto é, $||P_j|| = 1$, para $j = 1, ..., n$.

Geometricamente, as colunas P_j podem ser vistas como pontos sobre a hiperesfera m-dimensional com raio unitário e centro na origem. Logo, o problema (1) pode ser descrito como o problema de atribuir ponderações x_j não negativas às colunas P_j de modo que, depois de reescalado, o seu centro de gravidade seja a origem. Vale ressaltar que qualquer problema de programação linear pode ser transformado no problema (1).

1.1 Método de von Neumann

O método de von Neumann consiste em encontrar a coluna P_s de P, 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 unindo b^{k-1} a P_s, conforme ilustrado na Figura 1 (a).

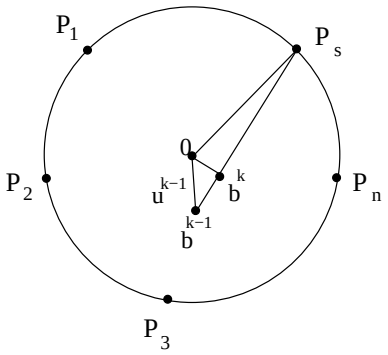

Figura 1: Método de von Neumann

O critério de parada usado no algoritmo do método de von Neumann é $\frac{||b^{k-1} - b^k||}{||b^k||} < \epsilon$, em que ϵ é uma tolerância previamente especificada.

1.2 Método de Redução por Pesos

O método de redução por pesos foi proposto com objetivo de melhorar a eficiência do método de von Neumann e é baseado na ideia de que o resíduo b^{k-1} pode ser movido de forma a se aproximar da origem 0, aumentando um peso x_j de alguma coluna P_j e reduzindo um peso x_i de uma outra coluna P_i. A ideia é que o novo resíduo b^k esteja mais próximo da origem 0 que o resíduo b^{k-1}.

O peso é aumentado da coluna P_{s^+} que forma maior ângulo com o resíduo b^{k-1} e reduzido da coluna P_{s^-} que forma o menor ângulo com o mesmo resíduo. Isto equivale a fazer o resíduo b^{k-1} se mover na direção $d = P_{s^+} - P_{s^-}$. O novo resíduo b^k é o ponto que minimiza a distância da origem 0 a esta linha, mas isto está restrito ao decréscimo máximo possível de x_{s^-}. Veja Figura 1 (b),

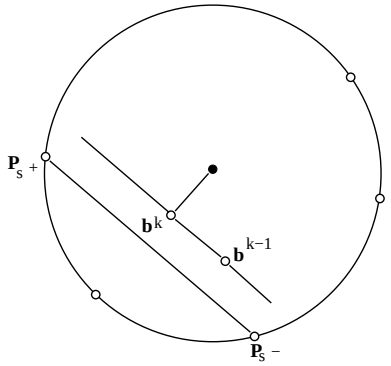

Figura 2: Método de redução por peso

O critério de parada para o algoritmo do método de redução por pesos é o mesmo do algoritmo de von Neumann. O esforço por iteração do algoritmo de redução por pesos é dominado por multiplicação de matriz e vetor para selecionar as colunas P_{s^+} e P_{s^-}, que é $O(mn)$. Este é o maior esforço computacional como no algoritmo de von Neumann.

2 Método de Redução por Pesos Generalizado

Utilizando a mesma ideia do método de redução por pesos, generalizamos e construímos o algoritmo de redução de pesos para p coordenadas. Conforme descrito anteriormente, no método de redução por pesos o cálculo da direção d é feito considerando apenas duas coordenadas. Ao generalizar essa ideia, a nova direção para mover o resíduo mais perto da origem é determinada considerando p coordenadas, as quais são determinadas nesse trabalho escolhendo as $p/2$ colunas de P que formam o maior ângulo e as $p/2$ colunas que formam o menor ângulo com o resíduo b^k. Caso p seja um número ímpar, uma unidade de seu valor é subtraída.

A vantagem deste algoritmo é que ao utilizar mais coordenadas temos uma possibilidade de encontrar uma direção melhor e com isso fazer com que o resíduo chegue próximo a origem mais rapidamente, ou seja, o desempenho do algoritmo é melhorado.

Na sequência apresentamos um resumo desse método.

DOI: 10.5540/03.2020.007.01.0459

2.1 Algoritmo Redução por Pesos para p Coordenadas

Dados: p par, $x^0 \geq 0$, com $e^t\ x^0 = 1$, it (número máximo de iterações) e ϵ.

Calcule: $b^0 = Px^0$ e $u^0 = \left\|b^0\right\|$.

Para $k = 1$ até it faça:

1. Determine: $\{P^-_{\eta_1}, ..., P^-_{\eta_{p/2}}\}$, colunas de P que formam o menor ângulo com o resíduo b^{k-1} e tal que $x^{k-1}_j > 0, j = \eta_1, ..., \eta_{p/2}$.

2. Determine: $\{P^+_{\eta_1}, ..., P^+_{\eta_{p/2}}\}$, colunas de P que formam o maior ângulo com o resíduo b^{k-1}.

3. Calcule: $v^{k-1} = argmax P^+_{\eta_j}.b^{k-1}$.

   Se $v^{k-1} \geq 0$, então **PARE**, o Problema (1) é infactível.

4. Calcule a direção: $d = \sum_{n=1}^{p/2}(P^+_{n_j} - P^-_{n_j})$.

5. Calcule: $\lambda = \frac{-d^t.b^{k-1}}{||d||^2}$. Se $\lambda \geq x_{s^-}$ então $\lambda = x_{s^-}$ = mínimo dos pesos das colunas $P^-_{\eta_1}, ..., P^-_{\eta_{p/2}}$, ou seja, $min\{x^-_{\eta_1}, ..., x^-_{\eta_{p/2}}\}$.

6. Atualize:

   Calcule o novo resíduo: $b^k = b^{k-1} + \lambda d$.

   Calcule o novo ponto: $x^k = x^{k-1} + \lambda(\sum_{j=1}^{p/2} e^+_j - e^-_j)$, em que e^+_j e e^-_j são vetores canônicos com 1 na posição das colunas $P^+_{\eta_j}$ e $P^-_{\eta_j}$, respectivamente.

7. Critério de Parada:

   Calcule: $u^{k-1} = \left\|b^{k-1}\right\|$.

   Se $u^{k-1} \leq \epsilon$ ou $\frac{\left\|u^{k-1} - b^k\right\|}{\left\|u^{k-1}\right\|} < \epsilon$ então **PARE**. A solução desejada do Problema (1) foi encontrada.

   Senão $k = k + 1$.

3 Experimentos Computacionais

O método de redução por pesos generalizado foi implementado usando o Matlab R2016a e os testes computacionais foram realizados em um notebook Dell Inspiron i7 e 8GB de RAM utilizando 38 problemas de programação linear de livre acesso pertencentes à biblioteca Netlib [5]. Para analisar o desempenho do algoritmo os valores de p

considerados foram: 2 (equivale ao método de redução por pesos), 4, 10, 20 e n (número de variáveis do problema). Quando $p = n$, todas as colunas podem ser consideradas se tiver $n/2$ variáveis positivas que formam o menor ângulo com o resíduo. Na Tabela 1, apresentamos os resultados. Comparamos o desempenho com relação a norma do resíduo na última iteração, o número de iterações realizadas e o tempo total de execução. A coluna $||b^0||$ traz o valor da norma do resíduo inicial e na última linha da tabela estão os valores médios considerando todos os problemas. Para o critério de parada foi utilizado $it = 500$ e $\epsilon = 10^{-4}$.

De acordo com os resultados obtidos, podemos dizer que, com relação ao resíduo final, para $p = 20$ o valor foi menor em torno de 63% dos problemas, seguido por $p = n$ com aproximadamente 24% dos problemas. Já para $p = 2$ nenhum problema obteve o melhor resultado.

Considerando o número total de iterações realizadas pelo algoritmo, notamos que novamente para $p = 20$ os resultados foram melhores, realizando menos iterações em torno de 16% dos problemas, seguido agora por $p = 10$ com um pouco mais de 13%. Vale ressaltar que em aproximadamente 58% dos problemas o número de iterações atingiu o limite (500) para todos os casos de p.

Quando analisamos o tempo de execução, percebemos que ele foi menor para $p = 2$ em um pouco mais de 39% dos exemplares, seguido de $p = 10$ com quase 24% e $p = 20$ em torno de 18%. Os resultados foram piores para $p = n$, resolvendo mais rápido quase 8% dos problemas.

Porém, de um modo geral, consideramos que o algoritmo é mais eficiente para $p = 20$, pois apesar do valor médio do resíduo final não ter sido melhor que o valor de $p = 10$, o algoritmo se mostrou mais robusto. A redução média no valor do resíduo para $p = 20$ foi de $82,72\%$, sendo menos de 3% maior que o valor de $p = 10$ e aproximadamente 20% maior que $p = 2$ que teve o pior resultado. Além disso, com relação ao número de iterações médio, novamente $p = 20$ foi mais robusto e apresentou melhor resultado, realizando 10% a menos de iterações que $p = 2$. E por fim, considerando o tempo total médio de resolução, $p = 20$ não foi o mais robusto e nem eficiente, porém o valor foi apenas $2,5\%$ maior que $p = 2$, que apresentou o melhor desempenho.

4 Conclusões

Nesse trabalho, foi desenvolvida uma generalização do método de redução por pesos com o objetivo de melhorar o seu desempenho. As vantagens do método proposto são simplicidade, convergência inicial rápida e a possibilidade de usar p coordenadas por iteração para acelerar a convergência. Vale ressaltar que o valor de p é limitado pelo tamanho do problema e o maior esforço computacional deste método é para determinar as p coordenadas que formam o menor e o maior ângulo com o resíduo.

DOI: 10.5540/03.2020.007.01.0459

Tabela 1: Desempenho do Método de Redução por Pesos Generalizado - Variação de p

Problema	Dimensão		Resíduo						Iteração					Tempo (s)				
Nome	Linha	Coluna	$\|\|b^0\|\|$	p=2	p=4	p=10	p=20	p=n	p=2	p=4	p=10	p=20	p=n	p=2	p=4	p=10	p=20	p=n
adlittle	56	138	0,2587	0,0010	0,0009	0,0017	0,0016	0,0072	500	500	500	500	500	0,5320	0,5898	0,7323	0,5693	1,1873
afiro	27	51	0,1908	0,1497	0,1471	0,1421	0,1455	0,1423	6	4	3	2	3	0,0206	0,0816	0,0189	0,0254	0,0147
agg	488	615	0,0888	0,0158	0,0126	0,0110	0,0118	0,0126	500	500	500	500	500	2,6137	2,6597	2,6648	2,7323	6,6142
agg2	516	758	0,1300	0,0194	0,0149	0,0125	0,0118	0,0156	500	500	500	500	500	3,5084	3,5362	3,6069	4,0533	8,4865
bandm	305	472	0,0744	0,0113	0,0092	0,0081	0,0076	0,0095	500	500	500	500	500	1,8991	1,8612	1,9009	2,2138	4,2446
beaconfd	173	295	0,3740	0,0057	0,0051	0,0035	0,0035	0,0099	500	500	500	500	177	1,0171	0,9977	1,0384	1,2086	1,0075
blend	74	114	0,1482	0,0399	0,0401	0,0394	0,0358	0,0375	79	47	37	68	314	0,0918	0,0380	0,0416	0,0830	0,5449
bnl1	643	1,586	0,1718	0,0614	0,0595	0,0516	0,0444	0,0281	404	209	96	55	500	11,3682	6,0249	2,8896	1,8591	32,7357
bnl2	2324	4486	0,0602	0,0290	0,0185	0,0135	0,0135	0,0133	500	500	498	321	500	86,1522	87,3255	88,2025	57,0378	241,8901
bore3d	233	334	0,2524	0,0665	0,0637	0,0640	0,0631	0,0556	126	70	31	18	77	0,3317	0,1808	0,0922	0,0567	0,4914
brandy	220	303	0,1249	0,0093	0,0089	0,0075	0,0074	0,0093	500	500	500	500	500	1,1286	1,0724	1,1647	1,2052	2,8542
czprob	929	3562	0,0780	0,0063	0,0052	0,0050	0,0047	0,0133	500	500	500	500	500	59,4116	59,5749	58,7374	59,4355	126,9072
d2q06c	2171	5831	0,2621	0,0946	0,0170	0,0077	0,0071	0,0082	500	500	500	500	500	191,1391	201,8870	169,5437	161,9716	367,8077
degen2	444	757	0,4305	0,0469	0,0469	0,0445	0,0443	0,0407	236	124	65	40	143	2,0036	1,3330	0,5912	0,3884	2,7627
degen3	1503	2604	0,1989	0,0291	0,0191	0,0188	0,0187	0,0172	500	500	330	213	500	37,3517	41,7198	26,1423	17,0603	85,7926
dfl001	6071	12230	0,5104	0,4696	0,4288	0,3065	0,1030	0,0297	500	500	500	500	5	785,8281	841,6082	786,5545	785,7299	57,3024
e226	223	472	0,1130	0,0134	0,0114	0,0111	0,0107	0,0156	500	500	500	500	500	1,9262	2,1579	2,0094	2,3226	4,8053
etamacro	400	816	0,0910	0,0096	0,0077	0,0074	0,0074	0,0151	500	500	500	500	500	4,1679	4,4364	4,6013	5,1543	10,5901
fffff800	524	1028	0,1243	0,0394	0,0361	0,0354	0,0258	0,0356	335	198	97	386	500	4,7990	2,9220	1,4447	6,4152	17,5342
finnis	497	1064	0,2805	0,0181	0,0146	0,0135	0,0137	0,0258	500	500	500	500	500	6,8716	7,0664	7,0240	7,7288	17,8695
fit1d	24	1049	0,2094	0,0021	0,0018	0,0016	0,0015	0,0061	500	500	500	500	500	6,5112	6,6820	6,4473	6,9602	15,4141
fit1p	627	1677	0,3981	0,3684	0,3657	0,3577	0,3066	0,0312	56	31	16	18	213	1,9416	1,1940	0,6738	0,8162	17,0198
fit2d	25	10524	0,1316	0,0383	0,0013	0,0008	0,0008	0,0023	500	500	500	500	500	541,2793	548,9224	534,5959	537,6229	1212,8763
fit2p	3000	13525	0,5372	0,5007	0,4642	0,3674	0,2354	0,1284	500	500	500	500	17	899,6103	984,2121	935,0862	1026,8914	151,8420
ganges	1309	1706	0,0408	0,0242	0,0237	0,0232	0,0233	0,0216	334	183	79	40	9	6,2550	3,5585	1,6546	0,9823	1,1813
greenbea	2392	5598	0,0639	0,0401	0,0263	0,0134	0,0120	0,0130	500	500	500	500	500	131,4686	132,1585	140,3840	151,1202	363,5764
greenbeb	2392	5598	0,0639	0,0401	0,0263	0,0134	0,0120	0,0130	500	500	500	500	500	131,4467	131,5132	140,7278	150,4803	363,9806
israel	174	316	0,0912	0,0187	0,0166	0,0167	0,0164	0,0194	477	500	383	454	500	1,1777	1,1727	0,7058	1,2076	3,3170
maros	846	1966	0,1075	0,0143	0,0088	0,0059	0,0057	0,0116	500	500	500	500	500	20,4628	19,2083	17,6853	18,5435	48,8505
perold	625	1506	0,0490	0,0144	0,0101	0,0075	0,0074	0,0092	500	500	500	500	500	12,1024	12,5678	12,2818	13,3525	32,2114
pilot	1441	4860	0,0520	0,0237	0,0157	0,0128	0,0125	0,0129	500	500	500	500	500	120,0451	121,3615	121,8176	120,6154	276,4631
pilot4	410	1123	0,0625	0,0171	0,0107	0,0047	0,0045	0,0094	500	500	500	500	500	7,5993	7,9168	8,1567	8,5556	19,0201
pilot87	2030	6680	0,0434	0,0259	0,0175	0,0104	0,0085	0,0107	500	500	500	500	500	227,0995	228,4490	231,3384	227,0774	512,7143
pilot_ja	940	2267	0,0656	0,0150	0,0096	0,0070	0,0059	0,0069	500	500	500	500	500	25,4648	24,7162	24,6340	26,2476	64,6443
pilot_we	722	2928	0,0866	0,0259	0,0116	0,0066	0,0056	0,0074	500	500	500	500	500	39,2676	39,3502	39,5687	40,5803	90,7032
pilotnov	975	2446	0,0553	0,0170	0,0118	0,0088	0,0070	0,0074	500	500	500	500	500	30,7355	31,2635	31,3788	31,3510	77,2072
stocfor2	2157	3045	0,1012	0,0120	0,0109	0,0105	0,0106	0,0126	500	500	500	500	500	28,4988	32,4247	38,3808	39,7393	110,4497
vtp_base	198	346	0,1610	0,0118	0,0104	0,0102	0,0100	0,0525	500	500	261	266	500	1,3562	1,3491	0,8787	1,0141	3,2616
Valor médio			0,1653	0,0617	0,0529	0,0443	0,0333	0,0242	435,6053	417,52632	392	391,6053	406,7895	90,3813	94,6077	90,6684	92,6423	114,6362

DOI: 10.5540/03.2020.007.01.0459

Os testes computacionais realizados mostraram que para a classe de problemas considerada, os melhores valores com relação ao tamanho do resíduo e ao número de iterações foram para $p = 20$ e o pior desempenho foi para $p = 2$ (que é equivalente ao método de redução por pesos). Considerando o tempo de execução, a versão mais eficiente foi para $p = 2$, porém na média foi observado que a superioridade foi pequena comparando com o valor de $p = 20$, diferentemente do que aconteceu com a redução do valor do resíduo como já comentado. Os maiores tempos na maioria dos problemas testados ocorreu para $p = n$.

Portanto, diante dos resultados apresentados, podemos afirmar que o objetivo proposto nesse trabalho foi atingido e concluímos que para $p = 20$ o método é mais eficiente e robusto.

Como trabalhos futuros consideraremos mais problemas testes de outras bibliotecas para verificar se o resultado se mantém e implementaremos outros métodos propostos na literatura para efeito de comparação e análise da qualidade e eficiência do método proposto.

Agradecimentos

Agradecemos à FAPESP e ao CNPq pelo apoio financeiro.

Referências

[1] G. B. Dantzig. Converting a converging algorithm into a polynomially bounded algorithm, Tech. Rep., Stanford University, SOL 91-5, 1991.

[2] M. Epelman and R. M. Freund. Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system, *Math. Program*, 88, 451-485, 2000.

[3] C. T. L. S. Ghidini, A. R. L. Oliveira, J. Silva and M. I. Velazco. Combining a hybrid preconditioner and a optimal adjustment algorithm to accelerate the convergence of interior point methods, *Linear Algebra and its Applications*, 436, 1267-1284, 2014.

[4] J. P. M. Gonçalves, R. H. Storer, and J. Gondzio. A family of linear programming algorithms based on an algorithm by von Neumann, *Optimization Methods and Software*, 24:3, 461-478, 2009.

[5] NETLIB collection LP test sets. Netlib lp repository. Online at http://www.netlib.org/lp/data.

[6] J. Silva. Uma família de algoritmos para programação linear baseada no algoritmo de von Neumann. Tese de Doutorado, IMECC/UNICAMP, 2009.

[7] J. Silva, C. T. L. S. Ghidini, A. R. L. Oliveira and M. I. Velazco. A comparison among simple algorithms for linear programming, *TEMA*, 19:2, 305–326, 2018.

