TEMA Tend. Mat. Apl. Comput., **8**, No. 2 (2007), 211-218.

Reordenamento Estático na Atualização da Decomposição LU no Método Simplex[1]

D.R. CANTANE[2], Faculdade de Engenharia Elétrica e de Computação (FEEC), UNICAMP, 13083-852 Campinas, SP, Brasil

A.R.L. OLIVEIRA[3], Instituto de Matemática, Estatística e de Ciência da Computação (IMECC), UNICAMP 13081-970 Campinas, SP, Brasil

C. LYRA FILHO[4], Faculdade de Engenharia Elétrica e de Computação (FEEC), UNICAMP 13083-852, Campinas, SP, Brasil.

Resumo. Neste trabalho são desenvolvidas técnicas de atualização da decomposição LU da base no método Simplex, utilizando um reordenamento estático nas colunas da matriz. Uma simulação do método Simplex, realizando troca de colunas básicas obtidas pelo MINOS e verificando sua esparsidade é implementada. Somente os elementos afetados pela mudança de base são considerados para obter uma atualização da decomposição LU eficaz. As colunas da matriz são reordenadas de acordo com o número de elementos não nulos, em ordem crescente e na forma bloco triangular. Assim, obtém-se uma decomposição esparsa para qualquer base sem esforço computacional para obter a ordem das colunas, pois o reordenamento da matriz é estático e as colunas da base obedecem esta ordem. O objetivo deste trabalho é comparar a simulação desenvolvida que considera a esparsidade da matriz com os resultados obtidos pelo MINOS. Resultados computacionais em Matlab para problemas da Netlib mostram que esta idéia é muito promissora, pois não são necessárias refatorações periódicas como nos métodos de atualização tradicionais.

1. Introdução

A solução eficiente de sistemas lineares de grande porte é de importância fundamental na resolução de problemas de otimização. A solução destes sistemas pode ser obtida através de métodos genéricos como, por exemplo, via a decomposição das matrizes que formam a base e sua atualização, ou através da exploração da estrutura da classe de problemas a ser resolvida, como no problema de fluxo em redes [5].

[1]Este trabalho tem apoio da FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo) e CNPq (Conselho Nacional Brasileiro de Desenvolvimento Científico e Tecnológico)
[2]dcantane@densis.fee.unicamp.br, aluna de doutorado
[3]aurelio@ime.unicamp.br, orientador
[4]chrlyra@densis.fee.unicamp.br, orientador

Em [2] e [3] o reordenamento das colunas no problema combinado de dimensionamento de lotes e corte de estoque [8] é feito de modo que a matriz resultante adquire formato bloco triangular, facilitando a decomposição das bases, diminuindo assim o preenchimento da matriz no problema de corte e empacotamento. Através do reordenamento das colunas, é possível obter uma decomposição esparsa para qualquer base sem nenhum esforço computacional para obter a ordem das colunas, pois o reordenamento da matriz de restrições é estático, e o das colunas da base obedece esta ordem.

Na atualização, as operações de ponto flutuante causadas pela coluna que sai da base são desfeitas. Para tanto, as operações são efetuadas na ordem inversa da decomposição, sempre considerando a esparsidade. As operações para decomposição da coluna que entra na base são então realizadas. Finalmente, é necessário efetuar as operações da decomposição das colunas localizadas após a coluna que entra na base, sempre considerando a esparsidade.

Para o problema de lotes e cortes, os resultados obtidos em [2] mostram que esta abordagem além de computacionalmente barata, também é robusta introduzindo erros de arredondamento insignificantes em situações de pior caso, após milhares de iterações.

2. Métodos de Atualização da Decomposição

A técnica de atualização da decomposição da matriz básica com pivoteamento parcial foi proposta em [1]. Duas variantes deste algoritmo propostas em [9] procuram manter tanto a esparsidade existente na decomposição quanto a estabilidade numérica. A segunda variante é um aperfeiçoamento da primeira.

Com o objetivo de manter a esparsidade original dos dados, algumas formas de decomposição da matriz básica foram implementadas em [10], concluindo-se que o número de atualizações da decomposição realizadas nas iterações seguintes influencia consideravelmente no tempo de resolução dos problemas, nos preenchimentos ocorridos e até mesmo nos números de iterações do método simplex.

Em [7], a atualização da decomposição LU e sua aplicação na atualização da base no método simplex proposta por [6] são descritas em detalhe. Além disso, são apresentadas idéias de uma implementação eficiente proposta por [11], que é uma combinação de uma decomposição simbólica e numérica e apresenta um bom compromisso entre esparsidade e estabilidade numérica. Outras técnicas de atualização são descritas em [4].

3. Detalhes da Implementação

O objetivo da implementação é simular as iterações do método Simplex usando reordenamento estático e a atualização da decomposição LU que considera a estrutura esparsa da matriz adotados em [2] para problemas de otimização linear em geral. A simulação é realizada com as seqüências de bases geradas pelo MINOS.

Dois tipos de reordenamentos da matriz de restrições são usados e o número de elementos não nulos de cada um deles é comparado. Primeiramente, as colunas são

reordenadas de acordo com o número de elementos não nulos, em ordem crescente, e uma decomposição esparsa para a base sem esforço computacional para obter a ordem das colunas é realizada, pois o reordenamento da matriz é estático e as colunas da base obedecem esta ordem. Este tipo de reordenamento é denominado R_1. No segundo reordenamento, denominado R_2, a matriz é da forma bloco triangular.

Duas formas de atualizações das colunas foram utilizadas na implementação e o número de elementos não nulos de cada uma delas é comparado. Quando a coluna e, que entra na base, respeita um dos reordenamentos, a forma de atualização é denominada AC_1. Na segunda forma de atualização, AC_2, a coluna e entra na base na posição da coluna que sai da base. A notação AC_{MINOS} é utilizada para os resultados obtidos pelo MINOS.

Na atualização AC_1, uma decomposição LU da base, denominada F_1, é realizada considerando sua esparsidade. A operação é realizada somente nas colunas realmente afetadas pela troca de base devido à estrutura esparsa da matriz. Observe que pode-se não ter alteração na coluna k situada após a coluna que entra e/ou a que sai da base se $LU(k, e) = 0$ ou $LU(k, s) = 0$ devido à estrutura esparsa das colunas envolvidas.

Na decomposição LU, se $s < e$, as operações que foram causadas pela coluna que saiu da base são desfeitas, efetuando as operações na ordem inversa da decomposição, das colunas de $s+1$ até $e-1$, considerando a esparsidade. O próximo passo é verificar quais as colunas a partir de $e + 1$ são afetadas pela entrada da coluna e na base. A decomposição será feita apenas nestas colunas, após a decomposição da própria coluna e. Um procedimento similar é realizado se $s > e$.

Foi implementada uma função para encontrar a base inicial, a base "*Crash*" descrita em [7] e as colunas que entram e saem da base do arquivo de saída gerado pelo MINOS.

Na atualização AC_{MINOS}, uma decomposição LU completa da base, denominada F_{MINOS}, é realizada para simular a decomposição do MINOS.

Com o objetivo de verificar a robustez do método de decomposição LU considerando a estrutura esparsa, é calculada a estimativa do erro introduzido pela decomposição. Todas as operações de decomposição são completamente desfeitas em todas as iterações do método Simplex. Assim, efetuando as operações na ordem inversa da decomposição LU, obtém-se uma matriz que simulará a atualização da decomposição da base ao ser decomposta desde a primeira coluna. Dessa forma, a base original é obtida com algum erro numérico e simula-se o pior caso em termos do erro de arredondamento, após cada iteração.

4. Experimentos Numéricos

Foram realizados experimentos numéricos para verificar a eficiência das formas de atualização das colunas na base apresentadas na seção anterior. A Tabela 1 mostra o número de iterações (fase 2) e o número de decomposições para alguns problemas de otimização linear da Netlib obtidos com o MINOS. Na Tabela 2 encontra-se o número de elementos não nulos dos reordenametos R_1 e R_2.

Problema	Dimensão	Elementos Não Nulos	Atualizações	Número de Iterações	
				Com *Crash*	Sem *Crash*
kb2	43 × 68	313	3	55	82
adlittle	56 × 138	424	3	67	67
sc205	205× 317	665	2	52	208
israel	174× 316	2443	3	235	193
scsd1	77 × 760	2388	8	314	223
bandm	305× 472	2494	3	297	197
scfxm1	330× 600	2732	2	138	173
beaconfd	173× 295	3408	1	26	27
scsd6	147×1350	4316	14	917	972

Tabela 1: Dados dos problemas de otimização linear.

Elementos	Máximo		Mínimo		Média	
Não Nulos	R_1	R_2	R_1	R_2	R_1	R_2
kb2	331	357	86	86	231	241
adlittle	353	399	267	255	324	348
sc205	1150	1273	411	411	712	767
israel	1949	1994	625	625	1594	1641
scsd1	1148	859	433	407	738	611
bandm	6016	5287	3923	3963	4853	4572
scfxm1	2228	2316	1774	1782	2056	2067
beaconfd	1471	1615	1410	1442	1443	1482
scsd6	2527	1744	945	709	1674	1064

Tabela 2: Número de elementos não nulos não utilizando *Crash*.

Pode-se verificar que para os maiores problemas testados, o reordenamento R_2 é mais esparso que o reordenamento R_1. Dessa forma, será utilizado o reordenamento R_2 para realizar os testes de atualização de colunas a seguir.

Nas Tabelas 3 e 4 encontram-se os números de elementos não nulos da decomposição da base para os mesmos problemas de otimização linear, utilizando e não utilizando as bases *Crash*, respectivamente.

Elementos	Máximo			Média		
Não Nulos	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
kb2	353	460	406	271	332	257
adlittle	406	501	501	348	393	367
sc205	1242	1795	897	898	1109	644
israel	2187	6116	3536	1957	4382	2173
scsd1	689	1274	907	537	881	602
bandm	5508	9674	4163	4439	6422	3076
scfxm1	2250	4721	2059	2096	4006	1692
beaconfd	1423	1845	1217	1396	1610	1194
scsd6	1618	2997	1934	1053	1968	1168

Tabela 3: Número de elementos não nulos utilizando *Crash*.

Elementos	Máximo			Média		
Não Nulos	AC_1	AC_2	AC_{MINOS}	AC_1	AC_2	AC_{MINOS}
kb2	357	612	387	241	407	230
adlittle	399	697	528	348	521	374
sc205	1273	2094	1221	767	917	560
israel	1994	5840	2933	1641	4020	1796
scsd1	859	1215	1013	611	820	625
bandm	5287	22422	4173	4572	20415	3121
scfxm1	2316	4718	2090	2067	4227	1654
beaconfd	1615	3744	1485	1482	3655	1458
scsd6	1744	3644	1975	1064	2426	1194

Tabela 4: Número de elementos não nulos não utilizando *Crash*.

A forma de atualização de colunas AC_1 obtém uma base mais esparsa em relação à forma AC_2 em todas as simulações realizadas, onde a matriz da base chega a ser 52% mais esparsa para o problema scfxm1 e 78% para o problema bandm nas Tabelas 3 e 4, respectivamente. A forma de atualização de colunas AC_1 é utilizada para realizar a decomposição LU proposta neste trabalho. A atualização AC_1 é um pouco mais densa que a atualização AC_{MINOS}, mas como pode-se verificar mais adiante, nesta abordagem não são necessárias decomposições periódicas da base e provavelmente ela será mais rápida que outras estratégias de atualização.

A Tabela 5 mostra o número de flops (número de operações em ponto flutuante) de cada decomposição apresentada na seção anterior. A opção de *Crash* do MINOS foi desligada.

Número de Flops	Máximo		Média		Total	
	F_1	F_{MINOS}	F_1	F_{MINOS}	F_1	F_{MINOS}
kb2	2289	3624	1462	1693	119873	138836
adlittle	2492	5358	2103	2646	140909	177309
sc205	7655	12486	4590	4663	954724	969842
israel	12585	30456	10010	12039	1922010	2311507
scsd1	5279	11829	3635	4346	810552	969254
bandm	41178	88358	27961	21238	5508392	4183852
scfxm1	13919	21071	12436	11979	2151433	2072447
beaconfd	9701	9948	8898	9785	240233	264184
scsd6	10298	23511	6199	8158	6025711	7930057

Tabela 5: Número de flops da decomposição LU.

A decomposição F_1, proposta neste trabalho, reduz o número de flops em relação à decomposição F_{MINOS}.

Na Tabela 6, verifica-se a estimativa do erro introduzido por estas operações calculando a norma entre a base obtida a partir da função que desfaz as operações da decomposição e a base reordenada obtida diretamente da matriz de restrições.

Erro	Máximo	Mínimo	Média
kb2	1.7e-14	0	1.0e-14
adlittle	3.4e-16	1.1e-16	2.5e-16
scs205	3.5e-16	0	1.2e-16
israel	4.6e-12	0	2.4e-13
scsd1	7.3e-16	1.4e-16	3.3e-16
bandm	1.9e-13	5.7e-14	1.1e-14
scfxm1	1.1e-14	2.1e-15	7.5e-15
beaconfd	2.4e-16	2.8e-17	7.0e-17
scsd6	1.0e-15	1.4e-16	3.2e-16

Tabela 6: Estimativa do erro da atualização da base não utilizando a base *Crash*.

Pode-se concluir que o método proposto de atualização da base é extremamente robusto, pois o erro absoluto da base acumulado é da ordem de 10^{-12} no pior caso em relação às colunas originais, isto é, nesta atualização não é preciso decompor periodicamente a base como é geralmente feito nos métodos tradicionais devido à robustez e à esparsidade considerada.

5. Conclusões

Neste trabalho, é proposto um reordenamento estático das colunas básicas, obtendo uma base Simplex com decomposição esparsa sem nenhum esforço computacional para obter a ordem das colunas. O reordenamento não possui custo de inicialização ou atualização, pois não é necessário reordenar as colunas na decomposição.

A robustez do método de atualização da base é verificada através dos resultados apresentados na Tabela 6, pois a estimativa do erro introduzido por estas operações calculando a norma entre a base obtida a partir da função que desfaz as operações da decomposição e a base reordenada obtida diretamente da matriz é 10^{-12} no pior caso. Assim, para estes problemas não é preciso decompor periodicamente a base como nos métodos tradicionais [1].

Apesar de obter uma base um pouco mais densa que a do MINOS, a atualização aqui proposta não necessita de decomposições periódicas sendo provavelmente mais rápida. Nosso próximo objetivo é integrar a atualização da decomposição esparsa a outros códigos computacionais existentes, como MINOS e GLPK.

Abstract. In this work techniques of the simplex basis *LU* factorization update are developed using a static reordering in the matrix columns. A simulation of the Simplex method is implemented, carrying through the change of basis obtained from MINOS and verifying its sparsity. Only the factored columns actually modified by the change of the base are carried through to obtain an efficient *LU* factorization update. The matrix columns are reordered according to the number of nonzero entries, in increasing order and in a block triangular form. Thus, sparse factorizations are obtained for any base without computational effort to determine the order of columns, since the reordering of the matrix is static and base columns follow this ordering. The objective of this work is to compare the developed simulation that considers the sparsity of the matrix with the results obtained from MINOS. Computational results in Matlab for Netlib problems show that this is a very promising idea, since there is no need of periodic refactorization as in the traditional updating methods.

Referências

[1] R. Bartels, A stabilization of the simplex method, *Numer. Math.*, **16** (1971), 414-434.

[2] G. Bressan, A. Oliveira, Fast iterations for the combined cutting-stock and lot-sizing problems, *Anais da IX International Conference on Industrial Enginnering and Operations Management*, Arq. TI0601, (2003), 1-8.

[3] G. Bressan, A. Oliveira, Reordenamento eficiente das colunas básicas na programação de lotes e cortes, *Pesquisa Operacional*, **4** (2004), 323-337.

[4] I. Duff, A. Erisman, J. Reid, "Direct methods for sparse matrices", Clarendon Press, Oxford, 1986.

[5] J.L. Kennington, R.V. Helgason, "Algorithms for Network Programming", Wiley, New York, 1980.

[6] H. Markowitz, The elimination form of the inverse and its applications to linear programming, *Management Science*, **3** (1957), 255-269.

[7] I. Maros, "Computational Techniques of the Simplex Method", Kluwer Academic Publishers, 2003.

[8] S.L. Nonas, A. Thorstenson, A combined cutting-stock and lot-sizing problem, *Operations Research*, **120** (2000), 327-342.

[9] J. Reid, A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases, *Mathematical Programming*, **24** (1982), 55-69.

[10] C.T.L. Silva, "Problemas de Otimização Linear Canalizados e Esparsos", Dissertação de Mestrado, ICMC, USP, São Carlos, 2002.

[11] U. Suhl, L. Suhl, A fast lu update for linear programming, *Annals of Operations Research*, **43** (1993), 33-47.