Techniques of the simplex basis LU factorization update

Daniela Renata Cantane[1]
Electric Engineering and Computation School (FEEC), State University of Campinas (UNICAMP), São Paulo, Brazil

Aurelio Ribeiro Leite de Oliveira[2]
Institute of Mathematics, Statistics and Science of Computation (IMECC), State University of Campinas (UNICAMP), São Paulo, Brazil

Christiano Lyra Filho[3]
Electric Engineering and Computation School (FEEC), State University of Campinas (UNICAMP), São Paulo, Brazil

Abstract

The objective of this work is to compare the developed LU factorization update with results from MINOS. This technique, use a matrix columns static reordering and they are rearranged in accordance with the increasing number of nonzero entries and triangularized, leading to sparse basis factorization without computational effort to reorder the columns. Only the columns factorizations actually modified by the change of basis are carried through due to matrix sparse structure. Computational results in Matlab for problems from the Netlib show that this is a very promising idea, since there is no need to refactorize the matrix in the tested problems.

Keywords: linear optimization, sparse matrix, factorization update, simplex.

1. Introduction

The efficient solution of large-scale linear systems is very important for solving linear optimization problems. These systems can be approached through generic methods as, for example, *LU* factorization of the basis and its update, or through the problem specific structure exploitation, as in the network flow problem [5].

In [2] the column reordering in the combined cutting stock and lot sizing problems [8] is done in such a way that the resulting matrix is block diagonal. Therefore, it is easy to factor the base, resulting in little fill-in. Through the static reordering of columns, it is possible to obtain a sparse factorization of the basis, without any overhead to determine the order of the columns, since the base columns follow the given ordering.

In the *LU* update, the floating-point operations originated by the column that leaves the base are undone, in the reverse order of the factorization, always considering sparsity.

Finally, it is necessary to compute the factored columns after the entering column, also considering the sparse pattern.

[1] Av. Albert Einsten, 400, CEP: 13083-852, Campinas, SP, Brazil. E-mail: dcantane@densis.fee.unicamp.br
[2] Sérgio Buarque de Holanda, 651, CEP: 13081-970, Campinas, SP, Brazil. E-mail: aurelio@ime.unicamp.br
[3] Av. Albert Einsten, 400, CEP: 13083-852, Campinas, SP, Brazil. E-mail: chrlyra@densis.fee.unicamp.br

For the combined problem, results obtained in [2] have shown that this approach is fast and robust, introducing insignificant accumulated rounding errors in worst case situations, after thousands of iterations.

2. LU Factorization Update Methods

The LU factorization update technique with partial pivot was proposed by [1]. Two variants of this algorithm proposed in [9] aim to balance the sparsity and numeric stability in the factorization. The latter variant is an improvement over the former.

Several basic matrix LU factorization update was implemented [10] (in portuguese) with the objective to balance data original sparsity, it is conclude that the LU factorization update number carried through in the following iterations it influences considerably in the solved time of the problems, in the fill-ins and even iterations numbers of the simplex method.

In [7], the LU factorization update proposed by [6] and its application in the simplex update basis method is described in detail. Moreover, it presents the ideas of an efficient implementation proposed by [11], which is a good combination of the symbolic and numerical phase of the pivoting and presents a compromise between sparsity and numerical stability. Other techniques updates are describe in [4].

3. Implementation Issues

The objective of the implementation developed here is to simulate simplex iterations using the static reordering and the LU update adopted in [2] for general linear optimization problems.

The base sequence obtained from MINOS is used in the simulation. A procedure for finding the initial basis, “Crash” base described in [7] and the leaving and entering columns of the basis from MINOS' output was implemented. The matrix is triangularized, the columns are reordered according to the number of nonzero entries, in increasing order, leading to sparse factorizations without computational effort to obtain the order of columns, since the reordering of the matrix is static and basis columns follow this ordering.

Two column update approaches are used in the implementation and the number of nonzero entries for each one is compared. When the entering column e in the basis follows the static ordering the approach is called U_1. In the second approach, called U_2, the entering column e entries in the basis in the position of the leaving column l. The notation U_{MINOS} is the used for MINOS results.

In the U_1 approach, a LU factorization of the basis considering its sparsity, called F_1, is performed. Only the factored columns actually modified by the change of the basis are carried through. Notice that there is no changes in column k located after the entering and/or leaving column if $LU(k,e) = 0$ or $LU(k,l) = 0$ due to sparse structure of columns involved. In the U_{MINOS} approach, a complete LU factorization of the basis to simulate the MINOS’ factorization, called F_{MINOS}, is carried through.

The introduced error estimation in the factorization was computed with the objective of verify the robustness of the LU factorization method. Each factorization is completely undone in every simplex method iteration. Thus, the operations are performed in the reverse order of the LU factorization and the matrix that will simulate the basis factorization update is obtained from the one being factored from the very first column. Therefore, the original basis

is obtained with some amount of numerical error and the accumulated errors are a worst case estimation for each iteration.

4. Numerical Experiments

Initially, it was carried through numeric experiments for verifying the efficiency of the basis column update approaches presented in the previous section. Table 1 shows iterations (phase 2) and factorizations number of a subset of Netlib problems. Tables 2 and 3 show the nonzero number entries of the basis factorization of the same linear optimization problems using and not using the Crash basis, respectively.

LO Problems	Updates	Iterations	
		With Crash	Without Crash
kb2	3	55	82
adlittle	3	67	67
sc205	2	52	208
israel	3	235	193
scsd1	8	314	223
bandm	3	297	197
scfxm1	2	138	173
beaconfd	1	26	27
scsd6	14	917	972

Table 1: Linear optimization problems data.

Nonzero entries	Maximum			Minimum			Average		
	U_1	U_2	U_{MINOS}	U_1	U_2	U_{MINOS}	U_1	U_2	U_{MINOS}
kb2	353	460	406	179	210	130	271	332	257
adlittle	406	501	501	297	314	248	348	393	367
sc205	1242	1795	897	676	677	439	898	1109	644
israel	2187	6116	3536	1606	2740	1612	1957	4382	2173
scsd1	689	1274	907	421	577	371	537	881	602
bandm	5508	9674	4163	3240	3818	1977	4439	6422	3076
scfxm1	2250	4721	2059	1988	3397	1364	2096	4006	1692
beaconfd	1423	1845	1217	1396	1610	1194	1396	1610	1194
scsd6	1618	2997	1934	734	1294	580	1053	1968	1168

Table 2: Nonzero entries number with crash basis.

Nonzero entries	Maximum			Minimum			Average		
	U_1	U_2	U_{MINOS}	U_1	U_2	U_{MINOS}	U_1	U_2	U_{MINOS}
kb2	357	612	387	86	86	44	241	407	230
adlittle	399	697	528	255	363	262	348	521	374
sc205	1273	2094	1221	411	411	206	767	917	560
israel	1994	5840	2933	625	749	363	1641	4020	1796
scsd1	859	1215	1013	407	516	342	611	820	625
bandm	5287	22422	4173	3963	18496	2229	4572	20415	3121
scfxm1	2316	4718	2090	1782	3352	1279	2067	4227	1654
beaconfd	1615	3744	1485	1442	3581	1421	1482	3655	1458
scsd6	1744	3644	1975	709	1368	615	1064	2426	1194

Table 3: Nonzero entries number without crash basis.

The U_1 basis update approach reduces up to 52% for the problem scfxm1 and 78% for the problem bandm nonzero entries, in comparison to the U_2 approach in the Tables 2 and 3, respectively. Thus, U_1 is used to carry through the LU factorization proposed in this work. The U_1 update approach is a little more dense that the U_{MINOS} update in a few cases, but as we shall see it not need to refactorize the base. Thus, it probably will be faster than other approaches.

The Table 4 shows the flop number (floating point operation count) of each factorization presented in the previous section. MINOS' Crash option is turned off.

Flop Number	Maximum		Average		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
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

Table 4: Flops number of LU factorization.

The F_1 factorization, proposed in this work, reduces the flop number with respect to the F_{MINOS} factorization.

In Table 5, the error estimate introduced by these operations was verified, computing the norm of the difference between the basis obtained by completely undoing the factorization and the original basis obtained directly from the constraint matrix.

Error	Maximum	Minimum	Average
kb2	1.7e-14	0	1.0e-14
adlittle	3.4e-16	1.1e-16	2.5e-16
sc205	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

Table 5: Error estimate of updated basis without crash basis.

It can be concluded that the factorization update proposed method is very robust. The maximum accumulated error in the worst case is of the order of 10^{-12}, that is, in this approach it is not necessary at all to refactorize the basis any time, as it is usually done by the tradicional methods due to robustness and sparsity considerations.

5. Conclusions

In this work a static reordering of matrix columns is proposed, leading to simplex base with sparse LU factorizations and inexpensive factorization updates. The reordering has no initialization or updating costs since there is no need to reorder the columns in the factorization.

Stable results are present in Table 5. It is safe to conclude that no periodical factorization is needed for these problems as it is usually for updating schemes [4] and [7]. As a result, if the starting basis is the identity, is not necessary to compute any LU factorization at all.

In spite of the new basis update approach obtain a little less sparse basis than MINOS, it does not need to perform periodic refactorizations. Thus, it will probably obtain better running time in the same computational environment since it takes less computational effort. For future work this approach and factorization will be integrated to an implementation such as MINOS or GLPK.

Acknowledgement

This research was sponsored by the Foundation for the Support of Research of the State of São Paulo (FAPESP) and the Brazilian Council for the Development of Science and Technology (CNPq).

References

[1] R. Bartels, *A stabilization of the simplex method*, Numer. Math., 16 (1969), pp. 414–434.

[2] G. Bressan and 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), pp. 1–8.

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

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

[5] J. L. Kennington and 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), pp. 255–269.

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

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

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

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

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