Hybrid Reordering Strategies for ILU Preconditioning of Indefinite Sparse Matrices *

Eun-Joo Lee † and Jun Zhang ‡

Laboratory for High Performance Scientific Computing and Computer Simulation,
Department of Computer Science, University of Kentucky,
Lexington, KY 40506 - 0046, USA

March 10, 2005

Abstract

Incomplete LU factorization preconditioning techniques often have difficulty on indefinite sparse matrices. We present hybrid reordering strategies to deal with such matrices, which include new diagonal reorderings that are in conjunction with a symmetric nondecreasing degree algorithm. We first use the diagonal reorderings to efficiently search for entries of single element rows and columns and/or the maximum absolute value to be placed on the diagonal for computing a nonsymmetric permutation. To augment the effectiveness of the diagonal reorderings, a nondecreasing degree algorithm is applied to reduce the amount of fill-in during the ILU factorization. With the reordered matrices, we achieve a considerably improved the stability of incomplete LU factorizations. Consequently, we reduce the convergence cost of the preconditioned Krylov subspace methods on solving the reordered indefinite matrices.

1 Introduction

Preconditioning Krylov subspace methods are generally considered one class of the most promising techniques [1, 3, 4, 14] for solving very large sparse linear systems of the form

$$Ax = b, \tag{1}$$

where A is an unstructured sparse matrix of order n. The incomplete LU preconditioning techniques have been successful for solving many symmetric and nonsymmetric matrices.

*Technical Report 436-05, Department of Computer Science, University of Kentucky, Lexington, KY, 2005.

†E-mail:elee3@uky.edu.

‡The corresponding author. E-mail:jzhang@cs.uky.edu; URL: http://www.cs.uky.edu/~jzhang. This author's research work was supported in part by NSF under grants CR-0092532 and ACR-0202934, in part by DOE under grant DE-FG02-02ER45961, and in part by the University of Kentucky Faculty Research Support Program.

But they may encounter difficulty when the matrix to be solved is indefinite, i.e., when the matrix has both positive and negative eigenvalues.

According to Chow and Saad [4], there are at least two reasons which make the ILU factorization approaches difficult in solving indefinite matrices. The first reason can be due to small or zero pivots in an indefinite matrix, which may lead to unstable and inaccurate factorizations [12]. In addition, small pivots are usually related to small or zero entries on the diagonal of the matrix, so an indefinite matrix with zero diagonal entries may have higher chances of encountering zero pivots if the matrix is also nonsymmetric [13, 16]. Secondly, unstable triangular solutions can happen when $||L^{-1}||$ and $||U^{-1}||$ are extremely large while the off-diagonal elements of L and U are reasonably bounded. Such problems are usually caused by very small pivots [4, 11, 16].

Since small pivots are often the origin of stability problems in computing ILU factorization on an indefinite matrix, we can expect a better performance if small pivots would be replaced with some large values on the matrix. Based on the idea mentioned above, several researchers [2, 6, 7, 9, 10] focused on permuting large entries onto the diagonal of the indefinite matrix before using direct or iterative solvers to solve the matrix. For example, Duff and Koster [9, 10] exploited bipartite matching algorithms and scaling techniques for computing permutations of a sparse matrix. Their reordering and scaling algorithms have an effect on the performance of various solution methods for solving sparse matrices. In the experimental studies with the Duff and Koster's [10] algorithms, Benzi *et al.* [2] considered matching algorithms such as maximum transversal, maximum product transversal, and the bottleneck transversal with symmetric permutations. Their preprocessing step still requires complicated matching algorithms with several symmetric permutations.

As a part of the continuous efforts to the approach on replacing small pivots with large values on the diagonal, we present two diagonal reordering algorithms that are computationally inexpensive but efficient in the aspect of reordering performance for indefinite matrices, and also propose new hybrid reordering strategies that are composed of a combination of the two proposed diagonal reorderings and the nondecreasing degree algorithm described in Section 2 [16].

The main idea of the diagonal reordering algorithms lies on searching for entries of (1) single nonzero element and/or (2) the maximum absolute value, to be placed on the diagonal for computing a nonsymmetric permutation. SINGLE-ELEMENT-REORDERING described in Algorithm 2.2 examines the rows and the columns which have only one nonzero element for the permutation so that the diagonal of the permuted matrix has more nonzero entries than the original one. On the other hand, MAXIMUM-VALUE-REORDERING described in Algorithm 2.3 searches for the entries which have the maximum absolute value for each column for the permutation to maximize the absolute value on the diagonal entries for each column.

In order to augment the effectiveness of the nonsymmetric diagonal reorderings, we apply a symmetric nondecreasing degree algorithm to reduce the amount of fill-in during the ILU factorization. This algorithm is used to reorder the rows of the matrix in a decreasing degree fashion; i.e., the rows with smaller degrees are listed first and those with larger degrees are listed last [16].

In Section 3, we present the numerical results to demonstrate the performance of each reordering strategy. A set of sparse indefinite matrices were used in the experiments. We first performed each reordering strategy as a preprocessing step. In order to solve the reordered matrices, we employed several ILU type preconditioners and a preconditioned iterative solver. The comparison factors derived from solving the reordered matrices were nonzero entries on the diagonal, the number of iterations, the amount of fill-in in the preconditioners, and a preconditioner conditioning estimate parameter. As a result, the convergence cost of the preconditioned Krylov subspace methods on solving the reordered indefinite matrices is markedly reduced by using our reordering strategies.

2 Hybrid Reordering Strategy

Let A be a sparse matrix, and $P = \{(i,j) \mid 1 \leq i,j \leq n\}$ be the set of pairs for a permutation ($i = j$ for the identity permutation, i.e., no permutation), where n is the order of the matrix A, and (i,j) means to interchange rows i and j of the matrix A.

ALGORITHM **2.1** VERIFY-PERMUTATION(A, i, k, d)

1. $r \leftarrow$ `FALSE`
2. $\widehat{k} \leftarrow$ COUNT-NONZEROS(A, i, d)
3. $c \leftarrow$ CHECK-PAIR(P, i, d)
4. **If** ($(k = \widehat{k}) \wedge (c =$ `TRUE`))
5. $\quad r \leftarrow$ `TRUE`
6. **Return** r

We describe the VERIFY-PERMUTATION(A, i, k, d) function in Algorithm 2.1 to determine a permutation, where d is the direction (`ROW` or `COL`; i.e., row-wise or column-wise), i is the index for the direction d, and k is the number of nonzero elements to be compared. In line 2, the COUNT-NONZERO(A, i, d) function counts the number of nonzero elements in the i-th row or column of the direction d. In line 3, the CHECK-PAIR(P, i, d) function determines whether the pair (i, d) is used for the permutation set P. At last, line 4 verifies whether the given position on the matrix A can be a member of the set P.

2.1 Single-Element-Reordering

ALGORITHM **2.2** SINGLE-ELEMENT-REORDERING

1. **Do** $i = 1, ..., n$
2. $\quad$ **If** (VERIFY-PERMUTATION(A, i, 1, `ROW`) = `TRUE`)
3. $\quad\quad j \leftarrow$ Find the column index of the nonzero element in row i.
4. $\quad\quad$ Replace (i,i) with (i,j) in P
5. **Do** $i = 1, ..., n$
6. $\quad$ **If** (VERIFY-PERMUTATION(A, i, 1, `COL`) = `TRUE`)
7. $\quad\quad j \leftarrow$ Find the row index of the nonzero element in column i.
8. $\quad\quad$ Replace (j,j) with (j,i) in P.

Algorithm 2.2 describes the SINGLE-ELEMENT-REORDERING algorithm that finds rows and columns which have only one nonzero element for the permutation. The main role of this algorithm is to increase the number of nonzero elements on the diagonal. The number of elements in a row or in a column can be computed very inexpensively if we use the compressed sparse row (or column) format. Since both rows and columns are needed, the most efficient implementation needs the matrix to be stored in both formats. We can temporarily use the storage space allowed for the preconditioner to be used in a later phase to store an additional copy of the matrix in another format. With the availability of both rows and columns, the SINGLE-ELEMENT-REORDERING algorithm costs $O(n)$ non-numerical operations. It is computationally more efficient than the maximum transversal algorithm [2, 9, 10], which costs $O(n * q)$, where q is the number of nonzero entries of the matrix.

2.2 Maximum-Value-Reordering

ALGORITHM **2.3** MAXIMUM-VALUE-REORDERING

1. $\alpha \leftarrow 0$
2. **Do** $i = 1, ..., n$
3. $\quad \widehat{k} \leftarrow$ COUNT-NONZEROS(A, i, d)
4. $\quad$ **If** $(\alpha < \widehat{k}\ )\ \alpha \leftarrow \widehat{k}$
5. **Do** $k = 2, ..., \alpha$
6. $\quad$ **Do** $i = 1, ..., n$
7. $\quad\quad$ **If** (VERIFY-PERMUTATION(A, i, k, `COL`)=`TRUE`)
8. $\quad\quad\quad j \leftarrow$ Find the row index of the maximum value element in column i.
9. $\quad\quad\quad$ Replace (j, j) with (j, i) in P.

As described in Algorithm 2.3, the MAXIMUM-VALUE-REORDERING algorithm searches for an element which has the maximum absolute value in each column for a permutation to place that entry on the diagonal of each column of the matrix. In lines 1-4, we find the maximum number of α nonzero elements in the columns. In line 8, finding the maximum absolute value in the columns takes $O(n)$ time. Thus, it is obvious that the time complexity of MAXIMUM-VALUE- REORDERING is $O(n^2 * \alpha)$, where α is the maximum number of nonzero elements in a column.

Duff and Koster [9, 10], and Benzi *et al.* [2] considered the maximum product transversal in which the product of the absolute value of the entries on the diagonal is maximized with a complexity $O(n * q * \log n)$. Since the matrices under our consideration are very sparse, α is usually smaller than $\log n$ and $n \ll q$. The MAXIMUM-VALUE-REORDERING algorithm is also more efficient in terms of the time complexity than the maximum product transversal [2, 10].

2.3 Nondecreasing degree algorithm

We denote by $\deg(v_i)$ the degree of the node v_i, which equals the number of nonzero elements of the i-th row minus one, i.e., $\deg(v_i) = Nz(A_i) - 1 = Nz(v_i) - 1$, where

$Nz(A_i)$ represents the number of nonzero elements in row i of a matrix A. The matrix A is reordered by a symmetric permutation such that $\deg(v_i) \leq \deg(v_{i+1}), i = 1, \ldots, (n-1)$. To be more precise, we reorder the nodes in a decreasing degree fashion; i.e., the nodes with smaller degrees are listed first and those with larger degrees are listed last. The purpose of the nondecreasing degree reordering strategy is to reduce the amount of fill-in during the ILU factorization [16].

3 Experimental Results

We conducted numerical experiments to test our reordering strategies. Here we report some interesting experimental results that we obtained from solving sparse matrices with our reordering algorithms and preconditioners. The preconditioned iterative solver that we employed was GMRES(20), with several ILU type preconditioners, such as ILU(0), MILU(0), ILUT, and ILUTP [13]. For all linear systems, the right-hand side was generated by assuming that the solution is a vector of all ones. The initial guess was a zero vector. The iteration was terminated when the l_2-norm of the initial residual was reduced by at least seven orders of magnitude, or when the number of iterations reached 500. The computations were carried out in double precision arithmetics. The set of sparse matrices that we used for our experiments are nonsymmetric matrices from the Harwell-Boeing Sparse Matrix Test Collection [8] and from the sparse matrix collection at the University of Florida [5]. A simple description of the matrices is given in Table 1.

Table 1: Description of the test matrices.

Name	Description
mahindas	Economic model of Victoria, Australia, 1880 data
shl_0	Simplex method basis matrix
shl_200	Simplex method basis matrix
shl_400	Simplex method basis matrix
fpga_trans_01	Circuit simulation matrices
fpga_trans_02	Circuit simulation matrices
shermanACa	Matrices from Kai Shen, UCSB
shermanACd	Matrices from Kai Shen, UCSB
adder_dcop_15	Circuit simulation matrices
eg4	Primarily chemical process simulation matrices

For simplicity, we define some notations as follows: nnz is the number of nonzero entries in the original matrix, org is the original matrix, $nnzD(org)$ is the number of nonzero entries on the diagonal of the original matrix; SER is Single-Element-Reordering, MVR is Maximum-Value-Reordering, and SMR is SER followed by the MVR algorithm.

As we can see from Table 2, in most cases, our reordering strategies place more nonzero entries on the diagonal than the original matrix has. The matrices shl_0, shl_200, and shl_400 have 5, 6, and 3 nonzero diagonal entries in the original matrices, respectively.

Table 2: Comparison of nonzero entries on the diagonal.

Name	n	nnz	$nnzD(org)$	$nnzD$(SER)	$nnzD$(MVR)	$nnzD$(SMR)
mahindas	1258	7682	106	519	907	1152
shl_0	663	1687	5	406	352	663
shl_200	663	1726	6	360	366	645
shl_400	663	1712	3	353	371	641
fpga_trans_01	1220	7382	1154	1220	1158	1220
fpga_trans_02	1220	7382	1154	1220	1158	1220
shermanACa	3432	25220	3432	3432	3408	3432
shermanACd	6136	53329	6136	6136	6110	6130
adder_dcop_15	1813	11246	1801	1813	1786	1797
eg4	5860	46842	5806	5860	5806	5860

They have 406, 360, and 353 nonzero diagonal entries after the reordering strategy SER, and 663, 645, and 641 nonzero diagonal entries after the combined reordering strategy SMR, respectively. In the case of shl_0, all diagonal entries become nonzero after the application of the reordering strategy SMR. We should point out, however, that a few matrices lost several nonzero diagonal entries due to the application of the two reordering strategies. For example, the number of nonzero diagonal entries of the shermanACd matrix is reduced from 6136 to 6130 after the application of the two reordering strategies. The loss of nonzero diagonal entries may happen in the reordering strategy MVR. However, such losses are seen as minor, compared to the number of total nonzero diagonal entries in these matrices.

Table 3: Comparison of the number of preconditioned GMRES iterations.

Name	τ	p	precond	org	ND	SER-ND	MVR-ND	SMR-ND
mahindas	10^{-3}	15	ILUTP	12(U)	12	12	11	6
shl_0			ILU(0)	-7	-7	-7	-7	7
shl_0	10^{-4}	5	ILUT	-3	-3	-1	-7	2
shl_200	10^{-5}	10	ILUT	-3	-3	-3	-1	31
shl_400	10^{-3}	10	ILUT	-3	-3	-3	-1	53
fpga_trans_01			ILU(0)	-7	-7	105	-7	105
fpga_trans_02			ILU(0)	-7	-7	105	-7	105
shermanACa	10^{-3}	5	ILUT	-3	-3	-3	-1	9
shermanACd	10^{-6}	15	ILUT	-3	-3	-3	-1	83
adder_dcop_15			ILU(0)	-7	-7	299	-7	-7
meg4			ILU(0)	-7	-7	7	-7	7

Table 3 reports the number of preconditioned GMRES iterations with different reordering strategies and different preconditioners. We performed SER, and SMR followed by a symmetric nondecreasing degree (ND) permutation (we use the following acronyms SER-ND, and SMR-ND) to reduce the amount of fill-in in the preconditioner. The "τ" and "p" columns in Table 3 are the two dropping parameters used in the ILUT factorization [13]. Moreover, the "precond" column specifies the particular preconditioners. The value "-1" indicates the failure of convergence within the maximum number (500) of allowed

iterations. The value "-3" represents a failure due to breakdown of the GMRES solver. The value "-7" indicates that the preconditioner was not constructed due to zeros on the main diagonal (zero pivot). The symbol "U" stands for a failure due to unstable preconditioning results, which only happened with the original mahindas matrix. In this case, the computed results are not reliable, although the preconditioned solver terminated in 12 iterations. Overall, the SMR algorithm seems to be the most robust one, which helped the ILU preconditioners solve all but one matrices used in the experiments.

Table 4: Comparison of the number of nonzero entries in the preconditioners.

Name	precond	*org*	ND	SER-ND	SMR	SMR-ND
mahindas	ILUTP	20234	9374	9087	25864	9520
shl_0	ILUT	3946	3647	2857	1898	1409
shl_200	ILUT	5839	5707	5399	2568	1970
shl_400	ILUT	5885	5353	5120	2589	1967
shermanACa	ILUT	42094	43395	43395	29163	23277
shermanACd	ILUT	191446	165574	362596	231520	84770

In Table 4, we compare the number of nonzero entries in the preconditioners with SER and SMR followed by the ND algorithm with the same dropping tolerance parameters as listed in Table 3. Since ILU(0) uses the same sparsity pattern of the matrix A, its number of nonzero entries is the same as that of A and is not listed in Table 4. We can see that the nondecreasing degree strategy reduces the amount of fill-in entries during the ILUT and ILUTP factorizations. It is interesting to note that the SMR-ND strategy seems to have the minimum amount of nonzero entries, or close to the minimum amount of nonzero entries, in many cases. This is especially true for the shl matrices and the sherman matrices.

Table 5: Comparison of the preconditioner conditioning estimate parameter.

Name	precond	*org*	ND	SER-ND	SMR	SMR-ND
mahindas	ILUTP	Unstable	4.30(+3)	1.61(+5)	1.03(+6)	5.54(+5)
shl_0	ILU(0)	N/A	N/A	N/A	4.71(+3)	4.51(+3)
shl_0	ILUT	Infinity	Infinity	Unstable	5.44(+3)	5.44(+3)
shl_200	ILUT	Infinity	1.00	Unstable	3.21(+13)	9.73(+10)
shl_400	ILUT	Infinity	Infinity	Unstable	3.65(+11)	7.19(+14)
fpga_trans_01	ILU(0)	N/A	N/A	1.92(+1)	4.89(+2)	1.92(+1)
fpga_trans_02	ILU(0)	N/A	N/A	1.92(+1)	5.28(+2)	1.92(+1)
shermanACa	ILUT	2.87(+115)	Infinity	Infinity	2.07(+4)	2.07(+4)
shermanACd	ILUT	2.83(+3)	2.83(+3)	2.83(+3)	3.29(+19)	4.67(+11)
adder_dcop_15	ILU(0)	N/A	N/A	5.00(+11)	N/A	N/A
meg4	ILU(0)	N/A	N/A	1.97(+5)	1.97(+5)	1.97(+5)

Finally, we present some statistics information concerning the condition number estimate of the preconditioners. Here we used the statistics parameter "condest", as suggested by Chow and Saad [4]. For the preconditioner constructed in the form of LU where L is a lower triangular matrix and U is an upper triangular matrix, the condest parameter is

defined as $\|(LU)^{-1}e\|_\infty$, where e is a vector of all ones. A very large value of condest, e.g., on the order of 10^{16} for double precision arithmetics, indicates that the constructed preconditioner is unstable. The condest parameter results are listed in Table 5, which are concerned with the results listed in Table 3. Under "N/A", we report that the preconditioner was not defined, due to zeros on the diagonal (zero pivot). Also, "Infinity" and "Unstable" represent the unstable triangular solvers. The statistics of the "condest" shows that many matrices benefit greatly from the proposed SMR algorithm in this paper to enhance the stability of the ILU factorization.

4 Conclusion

We have proposed hybrid reordering strategies for indefinite matrices in the ILU preconditioning techniques. The strategies are combinations of the diagonal reorderings to construct stable preconditioners and nondecreasing degree reordering to reduce the amount of fill-in. The diagonal reorderings were used to increase the number of nonzero elements on the diagonal and/or to maximize the absolute value on the diagonal entries. It should also be mentioned that the strategies run efficiently in low computational time cost. For the performance aspect, we showed various experimental results from the proposed reordering strategies. The results demonstrated were quite promising and effective in helping construct stable ILU type preconditioners for solving indefinite sparse matrices. In addition to that, the convergence cost of the preconditioned Krylov subspace methods on solving the reordered indefinite matrices was greatly decreased. However, at present, it is still somewhat less clear that there would be a universal strategy that performs well in solving many classes of general sparse matrices. Hence, in future work, we expect to develop more effective algorithms for permuting general sparse matrices and compare them with the maximum product transversal algorithm.

References

[1] M. Benzi. *Preconditioning techniques for large linear systems: a survey.* J. Comput. Phys., 182:418–477 (2002).

[2] M. Benzi, J.C. Haws, and M. Tuma. *Preconditioning highly indefinite and nonsymmetric matrices.* SIAM J. Sci. Comput., 22:1333–1353 (2000).

[3] T.F. Chan, and H.A. van der Vorst. *Approximate and incomplete factorizations.* Parallel Numerical Algorithms, ICASE/LaRC Interdisciplinary Series in Science and Engineering, 4:91–118 (1997). (http://www.math.uu.nl/people/vorst/publ.html#publ94)

[4] E. Chow, and Y. Saad. *Experimental study of ILU preconditioners for indefinite matrices.* J. Comput. Appl. Math., 86:387–414 (1997).

[5] T.A. Davis. *University of Florida sparse matrix collection.* http://www.cise.ufl.edu/research/sparse/matrices, 1997.

[6] I.S. Duff. *Algorithm 575: Permutations for zero-free diagonal.* ACM Trans. Math. Software, 7:387–390 (1981).

[7] I.S. Duff, G.A. Erisman, and J.K. Reid. *Direct Methods for Sparse Matrices.* Claredon Press, New York, 1986.

[8] I.S. Duff, R.G. Grimes, and J.G. Lewis. *Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I).* Technical Report RAL-92-086, Rutherford Appleton Laboratory, Oxfordshire, England, 1992.

[9] I.S. Duff, and J. Koster. *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices.* SIAM J. Matrix Anal. Appl., 20:889-901 (1999).

[10] I.S. Duff, and J. Koster. *On algorithms for permuting large entries to the diagonal of a sparse matrix.* SIAM J. Matrix Anal. Appl., 22:973-996 (2001).

[11] H.C. Elman. *A stability analysis of incomplete LU factorization.* Math. Comput., 47:191–217 (1989).

[12] D.S. Kershaw. *On the problem of unstable pivots in the incomplete LU-conjugate gradient method.* J. Comput. Phys., 38:114–123 (1980).

[13] Y. Saad. *ILUT: a dual threshold incomplete LU factorization.* Numer. Linear Algebra Appl., 14:387–402 (1994).

[14] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS, New York, 1996.

[15] H.D. Simon. *Incomplete LU preconditioners for conjugate-gradient-type iterative methods.* in Proceedings of the 1985 SPE Reservoir Simulation Symposium, Richardson, TX, Society of Petroleum Engineers, pp. 387–396 (1985).

[16] J. Zhang. *A multilevel dual reordering strategy for robust incomplete LU factorization of indefinite matrices.* SIAM J. Matrix Anal. Appl., 22:925–947 (2001).

[17] J. Zhang. *Preconditioned Krylov subspace methods for solving nonsymmetric from CFD applications.* Comput. Methods Appl. Mech. Engrg., 189:825–840 (2000).

Short Biography

Eun-Joo Lee received a Ph.D. in Mathematics from Chonnam National University, South Korea, in 1997.. She is currently studying for a Ph.D. in Computer Science at the University of Kentucky. Her research interests include large scale scientific computing, and iterative and preconditioning techniques for large sparse linear systems.

Jun Zhang received a Ph.D. from The George Washington University in 1997. He is an Associate Professor of Computer Science and Director of the Laboratory for High Performance Scientific Computing and Computer Simulation at the University of Kentucky.

His research interests include large scale parallel and scientific computing, numerical simulation, iterative and preconditioning techniques for large scale matrix computation. Dr. Zhang is associate editor and on the editorial boards of four international journals in computer simulation and computational mathematics, and is on the program committees of a few international conferences. His research work is currently funded by the U.S. National Science Foundation and the Department of Energy. He is recipient of the U.S. National Science Foundation CAREER Award and several other awards.