An Approach for Parallelizing any General Unsymmetric Sparse Matrix Algorithm *

Tariq Rashid† Timothy A. Davis‡

Technical Report TR-94-036, Computer and Information Sciences Department, University of Florida, 1994. Presented at the 7th SIAM Conference on Parallel Processing for Scientific Computing, February 1995, San Francisco, CA.

Abstract

In many large scale scientific and engineering computations, the solution to a sparse linear system is required. We present a partial unsymmetric nested dissection method that can be used to parallelize any general unsymmetric sparse matrix algorithm whose pivot search can be restricted to a subset of rows and columns in the active submatrix. The submatrix is determined by the partial unsymmetric dissection. We present results of this approach, applied to the unsymmetric-pattern multifrontal method.

1 Introduction

The importance of efficient large sparse matrix manipulation is well known for scientific and engineering applications. It is the need for accuracy and speed in scientific and engineering computation that drives the need for higher performance. Thus parallel computers have become in great demand for such computation. The solution one chooses depends critically on the degree of parallelism sought and the model of parallel computation used. For a given problem to be solved in parallel, the message passing programming model involves distributing the data and the computation among the processors. While this can be easily done for well structured problems, the irregular structure of large problems, for example unsymmetric pattern sparse matrices, makes it difficult to solve, in general. Therefore, a good ordering of rows and columns of a sparse matrix can significantly reduce the storage and execution time required by an algorithm for factoring the matrix on parallel computers.

In this paper, we consider the solution of the n-by-n linear system

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

where A is large, sparse, and unsymmetric. We consider the direct solution of (1) by means of LU factorization. In such a procedure, we apply a graph partition based ordering approach (partial unsymmetric nested dissection) to decompose the computations for the solution so that it can be efficiently done in parallel.

*This project is supported by the National Science Foundation(ASC–9111263 and DMS–9223088)

†phone: (904) 392–1482, email: tariq@cis.ufl.edu

‡phone: (904) 392–1481, email: davis@cis.ufl.edu

2 Design Considerations

Graph partitioning is a method which divides the vertices of the graph into several sets of roughly equal size such that the number of edges connecting the vertices in different sets is kept small. Partitioning is one of the key ideas for generating efficient orderings for sparse matrix calculations. Since finding an optimal partition of a graph is known to be **NP**–complete [7], the nature of the problem has inspired a variety of heuristics.

Consider the undirected bipartite graph of the matrix A, with $2n$ nodes (n row nodes and n column nodes). Let $\mathcal{R}$ and $\mathcal{C}$ be the sets of n rows and n columns and $H = (V, E)$ be the bipartite graph whose node set V is divided into sets $\mathcal{R}$ and $\mathcal{C}$ ($V = \mathcal{R} \cup \mathcal{C}$). The sets $\mathcal{R}$ and $\mathcal{C}$ are disjoint, and every edge in H has one end point in $\mathcal{R}$ and the other in $\mathcal{C}$. An example of the matrix and its associated bipartite graph is shown in Fig. 1 where,

$$(r_i, c_j) \in E,\ r_i \in \mathcal{R}, c_j \in \mathcal{C} \text{ and } r_i \neq c_j \tag{2}$$

Suppose a single vertex separator S is found for H such that there are two disconnected subgraphs H_a and H_b in $H \backslash S$. If the pseudoperipheral node is a row (column), then the level set structures will contain columns (rows) and rows (columns) alternatively. The vertex separator contains either a set of r_s row nodes or a set of c_s column nodes. This occurs because there is no edge between any two row nodes or column nodes in the above defined bipartite graph H, that is, $(r_i, r_j) \notin E$ or $(c_i, c_j) \notin E$ for any i and j. However, in our partial nested dissection algorithm, we perform node improvement (as done in [10]) so that a single vertex separator with r_s row nodes and c_s column nodes is found. In contrast with [10], $r_s \neq c_s$, in general. The resulting unconnected subgraphs H_a and H_b have r_a and r_b row nodes, and c_a and c_b column nodes, respectively. The result is a matrix of the form

$$\begin{bmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

Where A_{11} is r_a-by-c_a, A_{22} is r_b-by-c_b, and A_{33} is r_s-by-c_s (rectangular, in general).

Each of these submatries, A_{11} and A_{22}, are factorized in parallel. Up to $\min(r_a, c_a)$ pivots can be found in A_{11}. Unfactorized rows and columns in A_{11} are then combined with A_{33}. Threshold-based partial pivoting can be used, where the separator A_{31} is updated by the factorization of A_{11}. The method can be applied recursively to generate more parallelism. However, the sizes of the submatries, A_{11} and A_{22}, should not be too small. This is why we call our approach "partial" nested dissection.

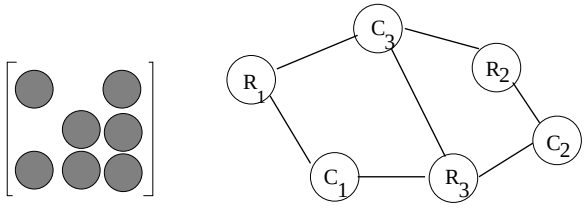

FIG. 1. *Matrix A and its associated bipartite graph.*

3 Constructing a Balanced Separator

In the automatic nested dissection algorithm [9], a level structure rooted at some pseudoperipheral node is generated. A level structure rooted at a pseudoperipheral node is

likely to have many levels. A separator is then selected from the "middle" level such that the connected graph can be divided into two unconnected subgraphs. In our algorithm we perform similar operations to obtain a rooted level structure at some pseudoperipheral node. Let v be the pseudoperipheral node and the sequence of level structure is defined as

$$L_0, L_1, L_2, \cdots, L_h \tag{3}$$

where $L_0 = v$ and $u \in L_i$ such that $(u, w) \in E$ and $u \in L_i$ imply $w \in L_{i+1}$. The separator is selected from the $m = \lceil \frac{(h+1)}{2} \rceil$th level. In our case, $L_m \subseteq \mathcal{R}$ or $L_m \subseteq \mathcal{C}$ but not both for $i = 1, \cdots, |S_m|$. Since the objective is to find a balanced separator with r_s nodes and c_s columns, we either select levels m and $m+1$ and then prune the resulting separator (Method 1) or we select two nodes for L_0 (Method 2).

3.1 Method 1

In this approach we calculate the degrees of all the nodes in level m and $m+1$. Nodes with degree 0 (which can only occur in level m) are discarded. A node with the highest degree is selected for the separator and the degrees of its neighbors are reduced by one. The selection continues until the degrees of all the nodes become zero. Nodes with zero degree are removed from the separator. Ties between two nodes are broken according to their positions in the list. We present the algorithm as follows:

```
Method1()
{
    Calculate the degrees of  ∀v  ∈ H'_m ∪ H'_{m+1} and insert in the list B
    for i = 1, ···, |B|
        if  (deg(b(i)) ≠ 0) then
            S = S ∪ {b(i)}
        endif
        ∀u  ∈ H'_m ∪ H'_{m+1}
            deg(u) = deg(u) − 1 if (v, u) ∈ E'
    endfor
}
```

This algorithm does not always produce a balanced separator. This happens, for example, when the initial degrees of all the nodes are equal. The balance can be improved by breaking ties depending on how many rows and columns are currently in the separator.

3.2 Method 2

In this approach we maintain symmetry similar to [8]. If the pseudoperipheral node is row r_i (column c_i), we also consider column c_i (row r_i) as a pseudoperipheral node and generate level structures starting from both the nodes (L_0 contains both nodes). The nodes for the separator are selected from level m.

4 Experimental Results

The partial unsymmetric nested dissection ordering was incorporated in the UMFPACK [1] package and evaluated using the matrices in Table 1. The matrix **RDIST1** is a chemical engineering matrix [13] and other matrices are from [5] ranging from electrical power flow (**GEMAT11**), fluid flow (**LNS3937**), and oil reservoir simulation (**ORSREG1**) problems. In Fig. 2 we show the patterns of the original, permuted (after dissection), and factorized

RDIST1 matrix. In Table 1 "Nz in LU(1)" and "Nz in LU(2)" represent the number of nonzeros in the LU factors after numeric factorization with Method 1 and Method 2 ordering respectively. We estimate the speedup obtained by partial unsymmetric nested dissection by dividing the total work by the work in the longest (weighted) path in the separator tree. This assumes one processor per node.

Three matrices perform better under Method 2 in terms of number of fill–ins. However, the estimate of speedup does not show reasonable parallelism under this method. We get a factor of speedup 7 in case of RDIST1 matrix when run using Method 1 with higher fill–ins.

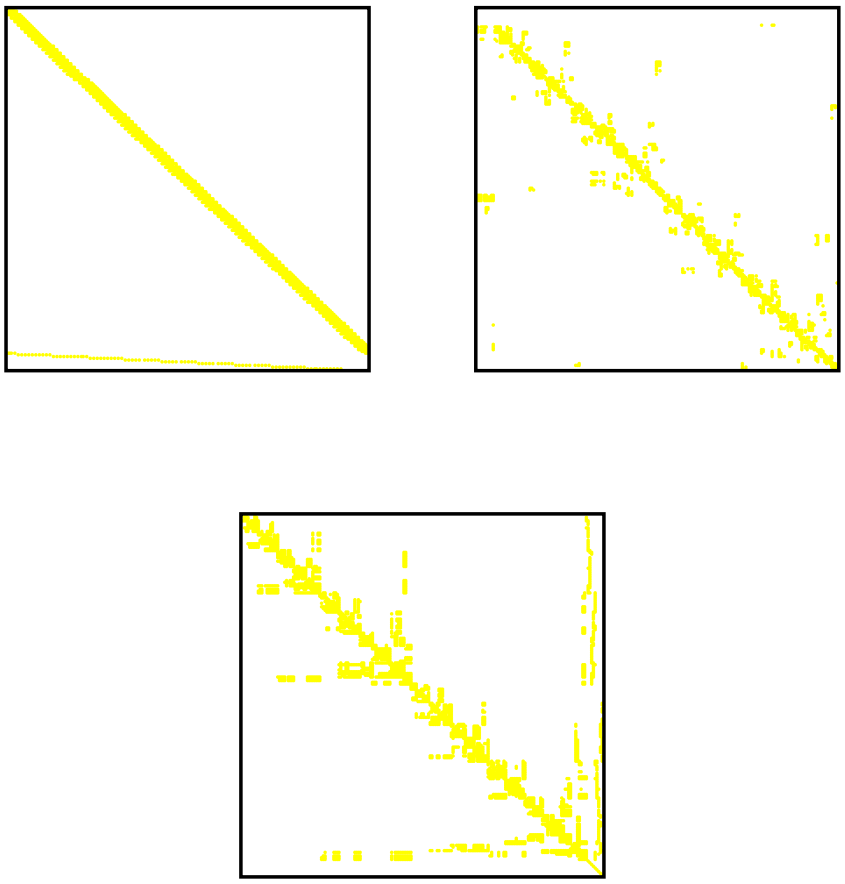

FIG. 2. *Structure of the original matrix, Method 1 ordering, and matrix after factorization.*

5 Conclusions

We have found that an unsymmetric partial nested dissection approach has reasonable parallel potential. The independent blocks can be factorized effectively within a distributed memory environment. Further parallelism can be obtained by applying a parallel factorization algorithm to each node in the separator tree. The method presented here provides a "first–cut" coarse–grain parallelism that should improve the efficiency of the

TABLE 1
Test Unsymmetric Matrices

Matrix	GEMAT11	LNS3937	ORSREG1	RDIST1
Order	4929	3937	2205	4134
Nonzeros in A	33108	25407	14133	94408
Nz in LU(1)	39650	224977	148000	168655
Nz in LU(2)	41961	156573	118518	108216
Speedup(1)	2.3	3.5	2.4	6.8

parallelism within the nodes of the separator tree.

References

[1] T. Davis, *Users' guide to the unsymmetric–pattern multifrontal package (UMFPACK)*, Tech. Report TR–93–020, CIS Dept., Univ. of Florida, Gainesville, Florida, 1993.

[2] T. Davis, *A Combined Unifrontal/Multifrontal Method for Unsymmetric Sparse Matrices.*, Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, Snowbird, Utah, June 15-18, 1994.

[3] I. Duff, *Design features of a frontal code for solving sparse unsymmetric linear systems out–of–core*, SIAM J. Sci. Statis. Comput., 5(1984), pp. 270–280.

[4] I. Duff, A. Erisman, and J. Reid, *Direct Methods for Sparse Matrices*, Claredon Press, Oxford, 1986.

[5] I. Duff, R. G. Grimes, and J. G. Lewis, *User's guide for the Harwell–Boeing sparse matrix collections (Release I)*, Tech. Rep. TRPA9286, Computer Science and Systems Division, Harwell Laboratory, Oxon, U.K., October 1992.

[6] S. Even, *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.

[7] M. Garey and D. Johnson, *Computers and Intractability*, W. H. Freeman and Company, New York, 1979.

[8] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[9] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[10] J. W. H. Liu, *A graph partitioning algorithm by node separators*, ACM TOMS, 15 (1989), pp. 198–219.

[11] ———, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp.134–172.

[12] E. Ng, *A comparison of some direct methods for solving sparse nonsymmetric linear systems*, in Proceedings 5th SIAM conference on Applied Linear Algebra, Lewis, eds., Snowbird, Utah, June 15–18, 1994, SIAM Press, pp.140.

[13] S. E. Zitney and M. A. Stadtherr, *Supercomputing Strategies for the design and analysis of complex separation systems*, Ind. Eng. Chem. Res., 32(1993), pp. 604–612.