Parallel Block-Diagonal-Bordered Sparse Linear Solvers for Electrical Power System Applications

D. P. Koester, S. Ranka, and G. C. Fox
School of Computer and Information Science and
The Northeast Parallel Architectures Center (NPAC)
Syracuse University
Syracuse, NY 13244-4100
dpk@npac.syr.edu, ranka@top.cis.syr.edu, gcf@npac.syr.edu

NPAC Technical Report — SCCS-552

Presented at
The Scalable Parallel Libaries Conference
Mississippi State University, Mississippi
6-8 October 1993

Abstract

Research is on-going that examines parallel direct block-diagonal-bordered sparse linear solvers for irregular sparse matrix problems derived from electrical power system applications. Parallel block-diagonal-bordered sparse linear solvers exhibit distinct advantages when compared to current general parallel direct sparse matrix solvers. Our research shows that actual power system matrices can be readily ordered into block-diagonal-bordered form, although load imbalance becomes excessive beyond 16 processors, limiting scalability for a single parallel linear solver within an application. Nevertheless, other dimensions exist in electrical power system applications that can be exploited to efficiently make use of large-scale multi-processors.

1 Introduction

Solving sparse linear systems practically dominates scientific computing, but the performance of direct sparse matrix solvers have tended to trail behind their dense matrix counterparts [7]. Parallel sparse matrix solver performance generally is less than similar dense matrix solvers even though there is more inherent parallelism in sparse matrix algorithms than dense matrix algorithms. Parallel sparse linear solvers can simultaneously factor entire groups of mutually independent contiguous columns or rows without communications; meanwhile, dense linear solvers can only update blocks of contiguous columns or rows each communication cycle. The limited success with efficient sparse matrix solvers is not surprising, because general sparse linear solvers require more complicated data structures and algorithms that must contend with irregular memory reference patterns. The irregular nature of these problems has aggravated the problems of implementing scalable sparse matrix solvers on vector or parallel architectures: efficient scalable algorithms for these classes of machines require regularity in available data vector lengths and in interprocessor communications patterns [2]. Parallel block-diagonal-bordered sparse linear solvers offer the potential for regularity often absent from other parallel sparse solvers. Nevertheless, when scalability of sparse linear solvers is examined using real irregular sparse matrices, the available parallelism in the sparse matrix can be as much the reason for poor scalability as the parallel algorithm or implementation.

Research is being performed to examine the applicability of parallel direct block-diagonal-bordered sparse solvers for electrical power system sparse matrix problems. This research focuses on real power system applications, consequently, matrix sizes are limited and available parallelism is also limited because the matrix structure defines the available parallelism for a single parallel sparse linear solver. For power

system applications, however, the limited size of the matrices and load imbalance due to limited parallelism in the matrix structure significantly limits scalability for a single parallel linear solver. Our research into specialized ordering techniques has shown that it is possible to order actual power system matrices readily into block-diagonal-bordered form, but load imbalance becomes excessive beyond 16 processors, limiting scalability for a single parallel linear solver within an application. Nevertheless, other dimensions exist in electrical power system applications that can be exploited to efficiently make use of large-scale multi-processors. We believe that this research also has utility for other irregular sparse matrix applications where the data is hierarchical. Other sources of hierarchical matrices exist, for example, electrical circuits, that have the potential for larger numbers of equations than power system matrices.

In this paper we examine the potential for scalability in block-diagonal-bordered direct sparse linear solvers to be incorporated within electrical power system applications. We focus on techniques to order sparse power system matrices into block-diagonal-bordered form, and examine scalability of the sparse linear solver as a function of the available parallelism in the power system network matrix. In section 2, we introduce the electrical power system applications that are the basis for this work. In section 3 we describe the advantages of parallel block-diagonal-bordered sparse matrix solvers. Paramount to exploiting the advantages of this parallel linear solver is ordering the irregular sparse power system matrices into this form. Minimum degree ordering, recursive spectral bisection-based ordering, and node-tearing-based ordering techniques are discussed in section 4. Analysis of the performance of these ordering techniques for actual power system load flow matrices from the Boeing-Harwell series are presented. Lastly, in section 5, we discuss our conclusions concerning these ordering techniques and our conclusions concerning the scalability of block-diagonal-bordered sparse linear solvers for power system applications.

2 Power System Applications

The underlying impetuous for our research is to improve the performance of electrical power system applications to provide real-time power system control and real-time support for proactive decision making. Our research has focused on load-flow and transient stability applications [1, 10]. Sparse linear solvers are employed in both of these applications and linear solvers are responsible for the majority of the floating point operations. Scalability is desired in these applications because they have the potential to be utilized across different sized geographical areas, from single electrical power utilities to regional power authorities.

Load-flow analysis examines steady-state equations based on the network admittance matrix that represents the power system distribution network. Load-flow analysis is used for identifying potential network problems in contingency analyses, for examining steady-state operations in network planning and optimization, and also for determining initial system state in transient stability calculations [10]. Load flow analysis entails the solution of non-linear systems of simultaneous equations, which are performed by repeatedly solving sparse linear equations. Load flow is calculated using the network admittance matrices, which are symmetric positive definite and have sparsity defined by the power system network. The size of these matrices is limited because there are generally less than 2,000 sparse complex equations in the network matrices for individual power systems, while regional power authorities would be limited to less than 10,000 sparse complex equations in the network matrices.

Transient stability analysis is a detailed simulation of the power system, that models the dynamic behavior of the electrical distribution networks, electrical loads, and the electro-mechanical equations of motion of the interconnected generators [1]. Transient stability analysis can be used to perform selective detailed analyses of generator commitment stability, and to support crisis decision making during network recovery. The transient stability problem is modeled by differential algebraic equations (DAEs) with differential equations representing the generators and non-linear algebraic equations representing the power system network that interconnects the generators. The DAEs are in natural non-symmetric block-diagonal-bordered form, with diagonal blocks of generator equations coupled by the power system distribution network. In this representation, there are as many coupling equations as the entire sparse admittance matrix. However, it it possible to order the admittance matrix to block-diagonal-bordered form to order to increase available parallelism. The size of the sparse matrices representing the DAEs have as many as 10,000 complex equations for an individual power system, while regional power authorities could have as many as 50,000 sparse complex equations in the matrix formed from the DAEs.

3 Block-Diagonal-Bordered Sparse Linear Solvers

Research is in progress to examine efficient parallel algorithms for the direct solution of sparse systems of equations

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

where the ordered sparse matrix PAP^T is in block-diagonal-bordered form. Direct block-diagonal-bordered sparse linear solvers exhibit distinct advantages when compared to current general parallel direct sparse linear solvers. All processors can be busy all of the time. Task assignments for numerical factorization on distributed-memory multi-processors depend only on the assignment of mutually independent diagonal blocks to processors and the processor assignments of data in the last diagonal block. In addition, data communications are significantly reduced and those remaining communications are uniform and structured. Figure 1 illustrates the factorization steps for a block-diagonal-bordered sparse matrix with four mutually independent sub-matrices. The mapping of data for the independent sub-matrices to the four processors is included in this figure. Each step involves highly parallel operations on data in the mutually independent diagonal blocks, operations on data in the last block, and operations that couple the data in the mutually independent blocks to data in the last block. This step involves data communications for distributed-memory multi-processor algorithms.

When A is a sparse symmetric positive definite linear system, then a specialized form of LU factorization, Choleski factorization, can be used to determine L such that $PAP^T = LL^T$. All features of parallel block-diagonal-bordered sparse linear solvers are applicable to parallel Choleski factorization, with the only modification to the algorithm requiring limiting calculations to the lower triangular portion of the symmetric matrix and calculating only L, instead of both L and U.

Parallel block-diagonal-bordered sparse linear solvers require modification to the traditional sparse matrix preprocessing phase for parallel Choleski factorization of ordering the matrix to minimize the number of calculations and symbolic factorization to identify the location of all fillin in order to use static data structures [7]. Parallel block-diagonal-bordered sparse linear solvers must include a specialized ordering step coupled to an explicit load balancing step in order to place the original matrix in block-diagonal-bordered form in such a manner as to minimize additional calculations due to fillin during factorization

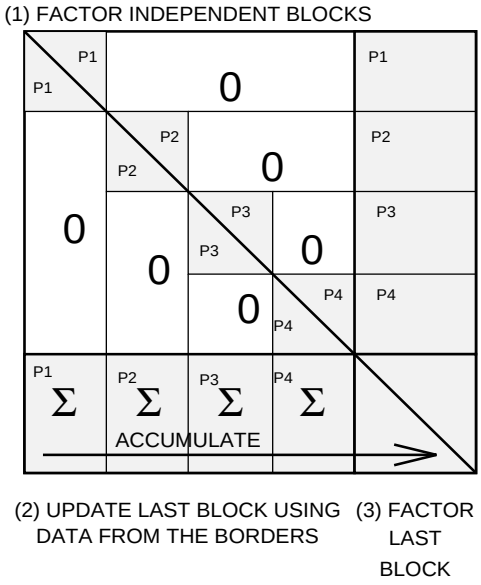

Figure 1: Block Diagonal Bordered Sparse Matrix Factorization for 4 Processors, P1 - P4

and to distribute the workload uniformly throughout a distributed-memory multi-processor. Our research has shown that the lower right-hand diagonal block in a block-diagonal-bordered ordered power system matrix is not extremely sparse, so it should be solved using dense techniques to take advantage of memory access regularity.

4 Ordering Sparse Matrices

Ordering symmetric sparse matrices is simply renumbering the nodes in the graph to modify characteristics of the corresponding matrix [3]. The computational expense of sparse matrix ordering is not unique to block-diagonal-bordered linear solvers, because all sparse linear solvers must consider ordering to minimize fillin [3].

Our research is examining ordering techniques that:

- generate block-diagonal-bordered form matrices
- minimize calculations by minimizing fillin
- minimize the number of coupling equations
- distribute processor workloads uniformly

Unfortunately, it may not be possible to optimize on all of the aforementioned constraints with irregular

matrices, but orderings of sparse power system matrices do well at meeting these criteria because of the unique hierarchical structure and limited number of average network edges to nodes in power system networks. Minimum fillin does not imply optimum parallel performance, because parallel sparse linear solver performance is dependent on load balance and interprocessor communications. Minimizing the number of coupling equations minimizes the number of calculations and also minimizes the size of the nearly dense last block in a parallel block-diagonal-bordered sparse matrix solver; however, the amount of potential scalability may suffer if the workload for factoring the independent blocks cannot be distributed uniformly throughout a multi-processor. A parallel algorithm cannot be scalable if excessive load-imbalance is encountered. When determining the optimal ordering for a sparse matrix, the minimum total number of calculations may be traded for the optimal ordering that yields the most parallelism. While optimal ordering algorithms are NP-complete [5, 7, 9], numerous heuristics exist that can be used to identify network structure to generate block-diagonal-bordered form matrices.

To transform a sparse matrix into block-diagonal-bordered form requires ordering techniques that efficiently identify mutually independent sub-matrices and coupling equations while also minimizing fillin. Fillin are values that become non-zero when factoring the matrix. For symmetric matrices, like power systems network admittance matrices, a graph-theoretical interpretation for independent sub-matrices exists: independent sub-matrices simply have no directly shared edges in their undirected graph. A simple example with four independent portions of the graph connected by nodes that form the couplings equations is presented in figure 2. No subgraph element has edges to any portion of the graph other than within the local subgraph or connecting to the coupling equations.

In addition to ordering the matrix into block-diagonal-bordered form, load balancing is required in the preprocessing phase. Due to the poor correlation of the size of independent sparse diagonal blocks with the workload, the actual number of calculations in each independent block must be determined in a pseudo factorization step during preprocessing. It is possible to load balance nearly all steps in the factorization of block-diagonal-bordered sparse matrices, except the communications step. Communications are regular with long messages, however, the length of these messages and the corresponding updates within the last block, may vary between processors.

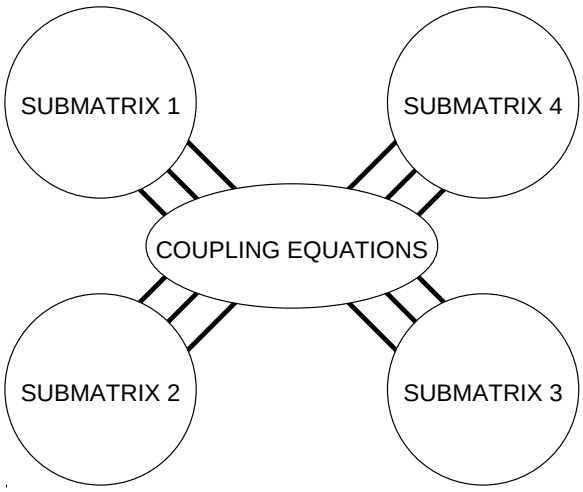

Figure 2: Graph with Four Independent Sub-Matrices

The preprocessing phase required for efficient factorization of block-diagonal-bordered form will be more computationally intensive than the simple algorithms required to prepare a matrix for numerical factorization presented in the recent literature. In the two techniques presented below that order sparse matrices into block-diagonal-bordered form, each has two phases. First, the independent sub-matrices are identified and then the matrix is ordered with this additional constraint to minimize resulting fillin. The additional processing requirements for these steps limit the applicability of this technique to repetitive solutions of static network structures, where the additional effort can be amortized over multiple solutions of similar linear systems. Because the sparse matrices from power system applications are representative of actual physical power distribution networks, the sparse matrices in load-flow analysis and power system transient stability analysis remain static over time periods significantly greater than required to recalculate new orderings. The computational expense of each ordering can be amortized over many applications of the block-diagonal-bordered linear solver.

4.1 Minimum Degree Ordering

Minimum-degree ordering has been used in our research in a two-fold manner:

1. to order symmetric power system admittance matrices to provide baseline orderings with which to

compare the performance of other ordering techniques

2. to order the independent sub-matrices in recursive spectral bisection and node-tearing ordering techniques

Minimum degree ordering is a greedy algorithm that selects a node with a minimum number of connected edges in the graph for factoring next, or the row to be factored next is that row with a minimum number of variables [5]. This algorithm is not optimal because truly efficient techniques do not exist to resolve ties and numerous rows have equal numbers of elements. Various versions of minimum degree ordering exist, with some versions employing heuristics to minimize the number of calculations. Examples of recursive spectral bisection-based ordering and node-tearing-based ordering are presented below. To contrast the performance of those ordering techniques, figure 3 illustrates a minimum degree ordering of the BCSPWR09.PSA matrix from the Boeing-Harwell series [4]. This matrix represents an actual 1723 bus western US power network, and will be used for all ordering comparisons. Note that the matrix is the most sparse in the upper left-hand corner, while the matrix is less sparse in the lower right-hand corner. When factoring this matrix, the number of zero values that become non-zero while factoring the matrix, is 2,168. Original nonzero values are represented in this figure in black, fillin locations are represented in gray, and all remaining zero values are white. A bounding box has been placed around the sparse matrix.

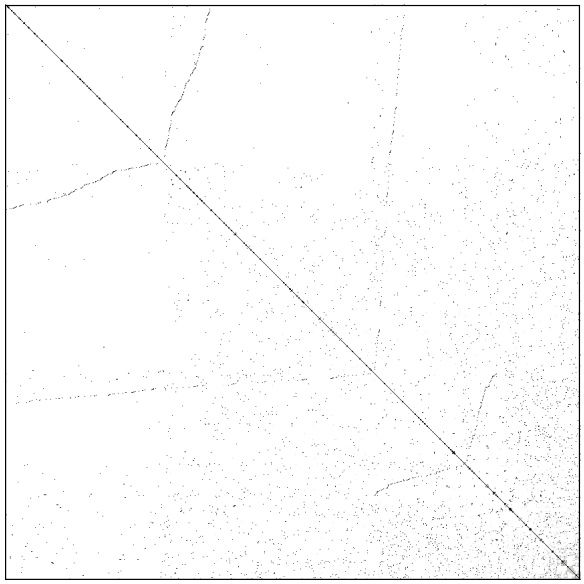

Figure 3: Minimum Degree Ordering

4.2 Recursive Spectral Bisection-Based Ordering

Recursive spectral bisection can be used to produce block-diagonal-bordered matrices for the parallel sparse matrix solvers. The recursive spectral bisection algorithm has been developed to balance computational load in a manner that minimizes interprocessor communications. While this goal is similar to generating block-diagonal-bordered sparse matrices with independent blocks, the ordering technique based on recursive spectral bisection requires two steps to generate matrices in this form. The first step in this ordering technique is to employ a multilevel version of the recursive spectral bisection method for partitioning unstructured graphs [8]. This technique computes the smallest non-trivial eigenvector of the Laplacian matrix associated with a graph to partition a sparse matrix into a predetermined number of subgraphs where the number of edges connecting subgraphs is minimized.

The second step in this ordering technique requires the extraction of the coupling equations to form the block-diagonal-bordered matrix. Those edges that interconnect multiple sub-matrices must be identified and moved to the set of coupling equations. The initial recursive spectral bisection partitioning method yields approximately equal sized sub-blocks, and the algorithm for this second step also considers the requirement to balance the size of the sub-blocks as the coupling equations are removed. Meanwhile, the number of coupling equations must be minimized to ensure good performance of the parallel block-bordered-diagonal sparse matrix solver. A greedy heuristic algorithm has been developed to perform this task while addressing both optimization goals.

The heuristic developed to extract the coupling equations from a graph ordered by recursive spectral bisection is based on repetitively removing all nodes with edges in multiple partitions and placing these nodes in a separate set. To minimize the number of coupling equations, nodes are removed in order of the largest number of inter-partition connections. Moreover, when there is more than one node with the same number of inter-partition connections, these ties are broken by selecting nodes from partitions with the largest number of remaining nodes. This maintains equal numbers of nodes per independent partition, a

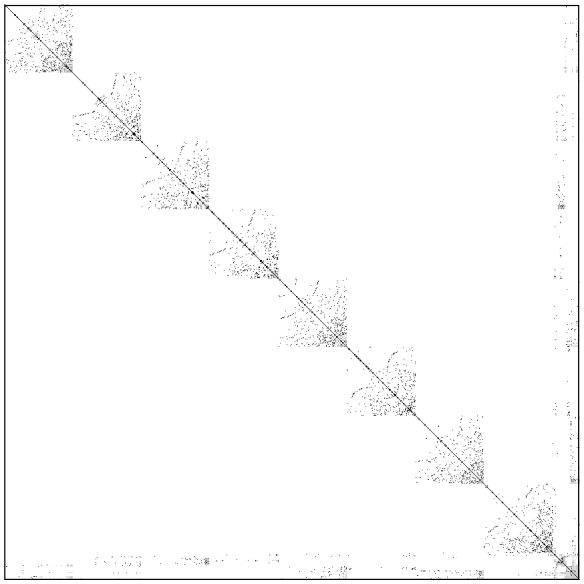

Figure 4: RSB-Based Ordering — 8 Processors

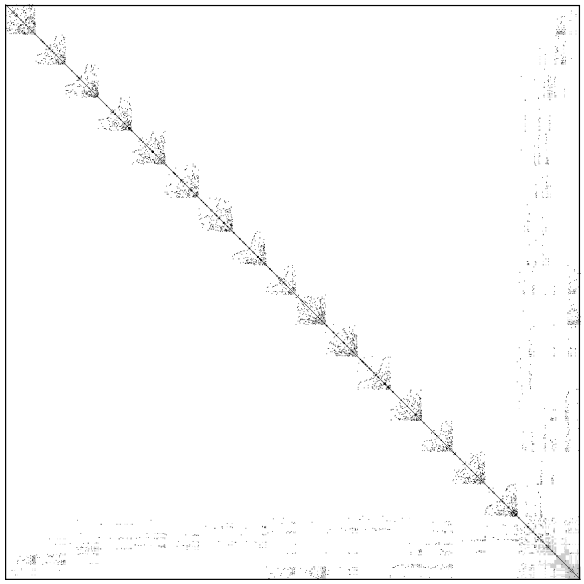

Figure 5: RSB-Based Ordering — 16 Processors

criteria of the first step in the algorithm.

Examples of recursive spectral bisection-based ordering are presented in figures 4 and 5 respectively for eight and sixteen partitions/processors, again using the BCSPWR09.PSA matrix from the Boeing-Harwell series [4]. Each partition could be assigned to a separate processor, with work proceeding independently until communications is required to send data to the last block before factoring the non sparse last block. Note that mutually independent partitions are equal sized, although the number of variables in each equation can vary. The number of fillin for these examples is 3,386 and 4,809 respectively, with a substantial portion of the fillin occurring in the lower right-hand corner.

The general performance of recursive spectral bisection-based ordering is dependent on the number of graph partitions, which is equal to the number of independent partitions. The number of coupling equations and the size of the last block is dependent on the number of processors, which is illustrated in figure 6. As the number of processors increases, so does both the size of the last block and load imbalance. Load imbalance is defined as the ratio of the difference of the maximum and minimum numbers of floating point operations divided by the maximum number of floating point operations per processor and illustrates the percentage of time that the processor with the least amount of work is inactive. Load imbalance for recursive spectral bisection-based ordering as a function of the number of processors is presented in figure 7. Even for only two partitions of the matrix, there is load imbalance. Load imbalance is approximately 20% for two processors and nearly 90% for greater than eight processors. While this ordering technique attempts to maintain equal numbers of elements in the graph partitions, the actual number of floating point operations is dependent on the sub-matrix structure.

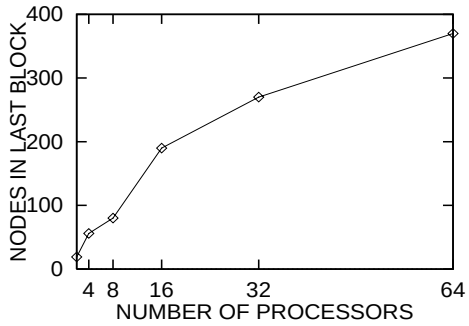

Figure 6: RSB — Nodes in Last Block

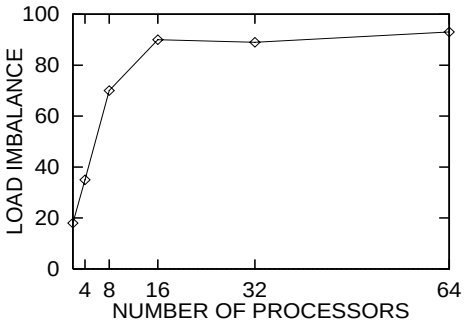

Figure 7: RSB — Load Imbalance

4.3 Node-Tearing-Based Ordering

The node-tearing-based ordering is designed to isolate both independent blocks and coupling equations within a sparse matrix while minimizing the number of coupling equations by examining the natural structure in the matrix. Tearing here refers to breaking the original problem into smaller sub-problems, whose partial solutions can be combined to give the solution of the original problem. Node-tearing nodal analysis is a specialized form of diakoptic analysis [6] that has been developed especially for power system network analysis [9]. Examples in reference [9] illustrate that this technique also has validity for general structural analysis.

The basic goal of node-tearing analysis is to partition a graph into two arbitrary subsets that contain mutually independent sub-blocks and coupling equations. The tearing optimization problem attempts to minimize the number of elements in the coupling equations over all distinct partitions of sub-matrices and coupling equations while also meeting a constraint that limits the size of any sub-matrix. By modifying this parameter, control can be exercised over the shape of the ordered sparse matrix. When the maximum size of the diagonal blocks is small, then the matrix is nearly in diagonal-bordered form. However, when this value is large, the number of coupling equations decreases as some of these equations are moved into diagonal blocks as smaller diagonal blocks are concatenated. Moreover, this technique can be applied recursively to reduce the bandwidth of the diagonal, except for controlled sized sub-borders. Because both the size of the major and minor blocks are user-selectable, the size of these blocks can be chosen to maximize performance for vector processor implementations.

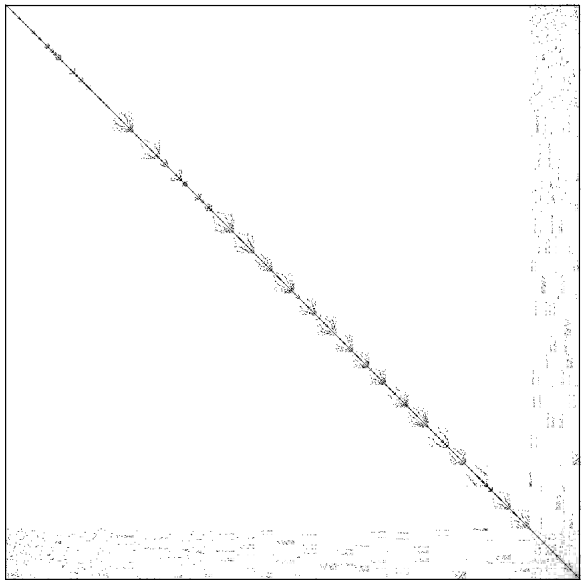

Figure 8: Node-Tearing-Based Ordering — Maximum Blocksize of 64

This graph optimization problem belongs to the family of NP-complete problems, so an efficient heuristic algorithm has been developed, based on examining the contour of the graph [9]. The computational complexity of this algorithm is

$$O(\max_{\forall i} |\mathcal{A}_i| \times n) \tag{2}$$

due to the fact that all nodes in the graph must be examined and for each element in the contour tableau, all elements of the adjacency set, $\mathcal{A}_i$, must be examined for the next node. Because the matrix is sparse, the maximum number in the adjacency set will be substantially less than n. Examples of node-tearing-based ordering are presented in figures 8 and 9 respectively for maximum diagonal block sizes of 64 and 128 nodes. Figure 10 illustrates a recursive application of node-tearing with maximum block size of 128 nodes for the larger blocks and maximum size of 16 for the diagonal sub-blocks within the larger blocks. These examples also use the BCSPWR09.PSA matrix of the Boeing-Harwell series [4]. The number of fillin for these ordered matrices is 3,248, 3,486, and 3,838 respectively, with a substantial portion of the fillin occurring in the lower right-hand corner.

The general performance of node-tearing-based ordering is dependent on the maximum number of nodes in a diagonal block, and the number of processors that

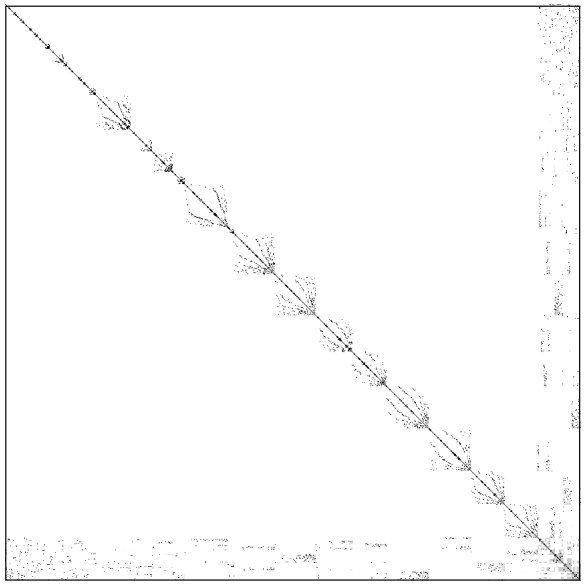

Figure 9: Node-Tearing-Based Ordering — Maximum Blocksize of 128

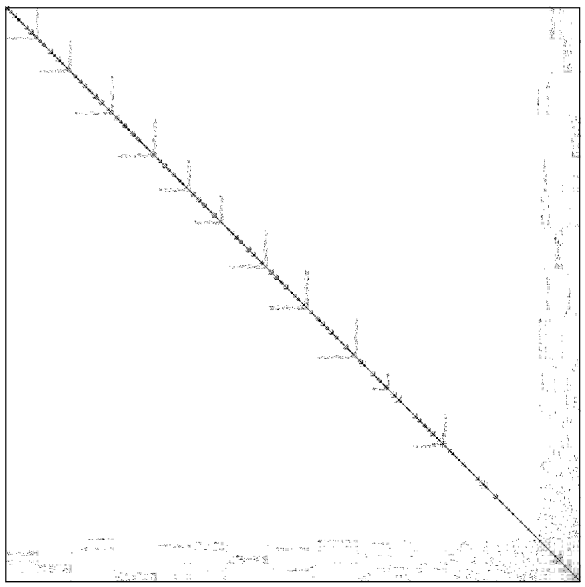

Figure 10: Recursive Node-Tearing-Based Ordering — Maximum Blocksizes of 128 and 16

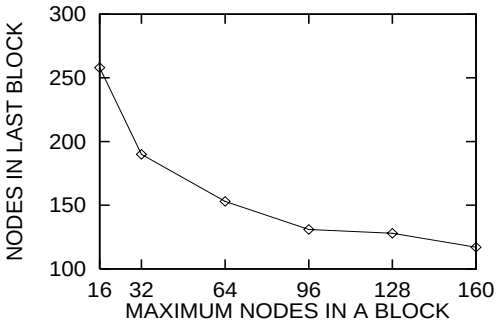

Figure 11: Node-Tearing — Nodes in Last Block

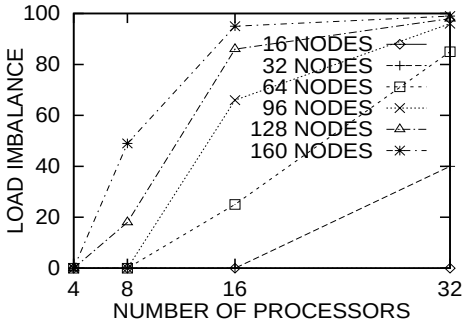

Figure 12: Node-Tearing — Load Imbalance

the independent sub-matrices are spread over. The number of coupling equations and the size of the last block is dependent only on the maximum number of nodes in a diagonal block, which is illustrated in figure 11. It is desirable to minimize the size of the last block, but this can cause load imbalance as the number of processors increases. Families of curves illustrating load imbalance as a function of the number of processors is presented in figure 12. For all ordering with different size diagonal blocks, there is perfect load-balancing for four processors, however, as the number of processors increases, so does load imbalance.

5 Conclusions

We have extensively examined ordering techniques based on node-tearing and recursive spectral bisection. The implementations of these ordering techniques

include simulated factorization of independent submatrices to examine load-imbalance. Node-tearing-based techniques produce numerous diagonal blocks, and the workload assigned to separate processors is determined in an explicit load balancing step. Recursive-spectral bisection attempts to balance the workload by assigning equal numbers of network nodes or matrix rows/columns per processor. However, work in sparse LU factorization is dependent on the number of non-zero and fillin values in the partition rather than the number of nodes.

Node-tearing-based ordering can vary the maximum size of the diagonal blocks by a user-selectable parameter. Varying the size of the diagonal blocks affects load imbalance within the diagonal blocks and borders, while also affecting the size of the last block. It is desirable to minimize the size of the last block, because that portion of the matrix is not sparse and traditional dense factorization techniques are used to minimize indexing overhead. The computational complexity of dense solvers is $O(n^3)$, so even a small reduction in the number of coupling equations can have a significant impact on parallel solver performance.

5.1 Conclusions on Scalability

Node-tearing-based ordering offers more scalability than recursive spectral bisection-based ordering techniques, although the structure in the electrical power system matrices does not permit unlimited scalability. Node-tearing-based ordering produces numerous diagonal blocks that contribute to effective load balancing for four to sixteen processors, while offering the potential to generate matrices in a form that could be factored efficiently by vector processors.

Our research into specialized ordering techniques has shown that load imbalance becomes excessive beyond 16 processors, limiting scalability for a single parallel linear solver. Nevertheless, other dimensions exist in electrical power system applications that can be exploited to efficiently use large numbers of processors. While a limited number of processors can be efficiently applied to a single power system simulation, multiple events can be simulated simultaneously.

5.2 Research Status

This paper has reported on our research into network ordering techniques for block-diagonal-bordered sparse linear solvers. Implementations of parallel block-diagonal-bordered sparse matrix solvers are in progress, and these prototype solvers will be utilized to verify multiprocessor performance. This research is being closely coordinated with parallel differential algebraic equation (DAE) research being performed at Northeast Parallel Architectures Center (NPAC) at Syracuse University.

Acknowledgments

We thank Alvin Leung, Kamala Anupindi, Nancy McCracken, Paul Coddington, and Tony Skjellum for their assistance in this research. This work has been supported in part by Niagara Mohawk Power Corporation, the New York State Science and Technology Foundation, the NSF under co-operative agreement No. CCR-9120008, and ARPA under contract #DABT63-91-K-0005.

References

[1] A. R. Bergen. *Power Systems Analysis.* Prentice-Hall, 1986.

[2] J. J. Dongarra, D. C. Sorensen I. S. Duff, and H. A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM, Philadelphia, 1991.

[3] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, Oxford, 1990.

[4] I. S. Duff, R. G. Grimes, and J. G. Lewis. Users' Guide for the Harwell-Boeing Sparse MAtrix Collection (Release I). Technical Report TR/PA/92/86, CERFACS, 1992.

[5] A. George and J. Liu. The Evolution of the Minimum Degree Ordering Algorithm. *SIAM Review*, 31(1):1–19, March 1989.

[6] H. H. Happ. Diakoptics - The Solution of System Problems by Tearing. *Proceedings of the IEEE*, 62(7):930–940, July 1974.

[7] M. T. Heath, E. Ng, and B. W. Peyton. Parallel Algorithms for Sparse Linear Systems. In *Parallel Algorithms for Matrix Computations*, pages 83–124. SIAM, Philadelphia, 1991.

[8] A. Pothen, H. Simon, and K.. P. Liou. Partitioning Sparse Matrices with Eigenvalues of Graphs. *SIAM J. Mat. Anal. Appl.*, 11(3):pp. 430–452, 1990.

[9] A. Sangiovanni-Vincentelli, L. K. Chen, and L. O. Chua. Node-Tearing Nodal Analysis. Technical Reprot ERL-M582, Electronics Research Laboratory, College of Engineering, University of California, Berkeley,, October 1976.

[10] Y. Wallach. *Calculations and Programs for Power System Networks*. Prentice-Hall, 1986.