Advances in Parallel Partitioning, Load Balancing and Matrix Ordering for Scientific Computing

Erik G. Boman[1], **Umit V. Catalyurek**[2], **Cédric Chevalier**[1], **Karen D. Devine**[1], **Ilya Safro**[3], **Michael M. Wolf**[4]

[1] Scalable Algorithms Dept., Sandia National Laboratories
[2] Biomedical Informatics and Electrical & Computer Engineering Depts., Ohio State Univ.
[3] Mathematics and Computer Science Division, Argonne National Laboratory
[4] Computer Science Dept., Univ. of Illinois at Urbana-Champaign

Abstract. We summarize recent advances in partitioning, load balancing, and matrix ordering for scientific computing performed by members of the CSCAPES SciDAC institute.

1. Introduction

Most large scientific computations are now carried out on parallel computers. As many of these architectures have distributed memory, it is important to distribute the data across the processors in a manner that achieves high parallel efficiency. Computing an effective distribution of the data is known as the load-balancing (or partitioning) problem.

A related problem is finding a good ordering of the unknowns of sparse matrices in simulations that use sparse direct solvers. Indeed, in Cholesky or LU factorizations, the factors are usually less sparse than the input matrix, and the fill depends on the order of the unknowns.

For both load balancing and sparse matrix ordering, we can use combinatorial approachs based on graph or hypergraph partitioning. Zoltan [10] is a toolkit that provides efficient and robust parallel data partitioning, as well as related functionalities like sparse matrix ordering.

A graph $G = (V, E)$ is a set V of vertices and a set E of edges made up of pairs of vertices ($E \subset V \times V$). A hypergraph is an extension of this model that allows each hyperedge to have two *or more* vertices. Each hyperedge is a subset of V.

As the general graph- or hypergraph-partitioning problem is NP-complete, several heuristics have been developed to provide good quality within a short execution time. The most popular scheme is the multilevel framework that coarsens the (hyper)graph to a smaller (hyper)graph with similar topological properties which is easily partitioned; it then projects a partition on the small (hyper)graph back to the original one, refining it to improve partition quality. The phase that reduces the (hyper)graph size is called coarsening; it is followed by the initial partitioning and the uncoarsening phases.

In this paper, we describe an improved coarsening method in multilevel graph partitioning. We discuss application of hypergraph and graph partitioning to distributing large sparse matrices to optimize matrix-vector multiplication. And we discuss progress in parallel sparse matrix ordering for direct solvers and present results from a SciDAC application.

2. Coarsening with a Weighted Aggregation Scheme

Many common multilevel schemes for simple graphs are based on strict coarsening (Strict Aggregation SAG). It is carried out by matching groups (usually pairs) of vertices together and representing each group with a single vertex in the coarsened space (e.g., matching [13, 15], first choice [9]). Another class of multilevel schemes used for several combinatorial optimization problems is based on *algebraic multigrid* (AMG or Weighted Aggregation WAG) [14, 16].

By contrast to SAG, in WAG each node can be divided into fractions, and different fractions belong to different aggregates. As AMG solvers have shown, weighted, instead of strict aggregation is important in order to express the likelihood of nodes to belong together; these likelihoods will then accumulate at the coarser levels, reinforcing each other where appropriate. This enables more freedom at the coarser levels and avoid making hardened local decisions, such as edge matching, before accumulating the relevant global information, while a strict aggregation may lead to conflicts between local and global pictures.

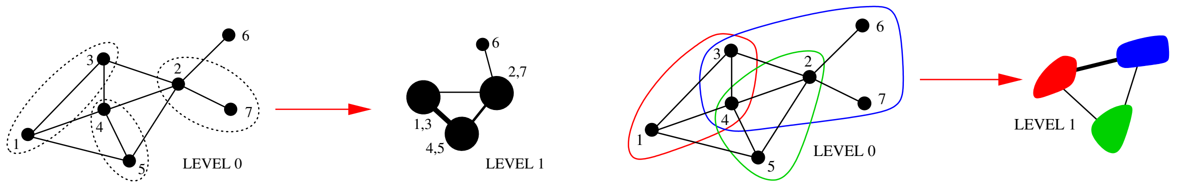

Figure 1. The left figure represents a SAG coarsening; each fine vertex belongs to one coarse vertex. The figure on the right is a WAG coarsening which allows fine vertices to be split across several coarse vertices.

We have shown [8] that the coarsening schemes in AMG can be adapted to graph partitioning; they produce higher quality partitions with less variation (i.e., greater robustness). A summary of these observations may be found in Figure 2. We compare WAG against the most popular technique in the graph partitioning community, Heavy Edge Matching (HEM) [15], which performs a strict pairing of vertices. We use two kinds of refinement during uncoarsening. The simplest is the *gradient* method, a local optimization without any hill-climbing possibility; the other is the more powerful Fidducia-Mattheyses (FM) [11] refinement, which allows hill-climbing.

We believe this approach is also more scalable (since it is similar to the approaches in multigrid codes like ML and Hypre), making it suitable for large problems. We have made steps toward extending the method from graphs to hypergraphs [3] for use in Zoltan's hypergraph partitioners.

3. Sparse Matrix Partitioning

A partitioning problem of particular importance is *sparse matrix partitioning*, where the goal is to reduce the communication in sparse matrix-vector multiplication. Since sparse matrix-vector multiplication is a kernel in many numerical algorithms, any improvement in parallel performance can impact a wide range of applications. We have performed an extensive empirical comparison of established and several new methods [5], demonstrating that 2D methods can reduce both the communication volume and the number of messages compared to traditional 1D row (or column) methods. Aggregated results for a set of 1413 test matrices are shown in Figure 3.

A drawback of 2D methods has been that most are quite expensive to compute. We have developed a new algorithm, nested dissection partitioning [1, 2] (see Figure 4), which is as effective as previous 2D methods to reduce communication but essentially as fast to compute as 1D methods.

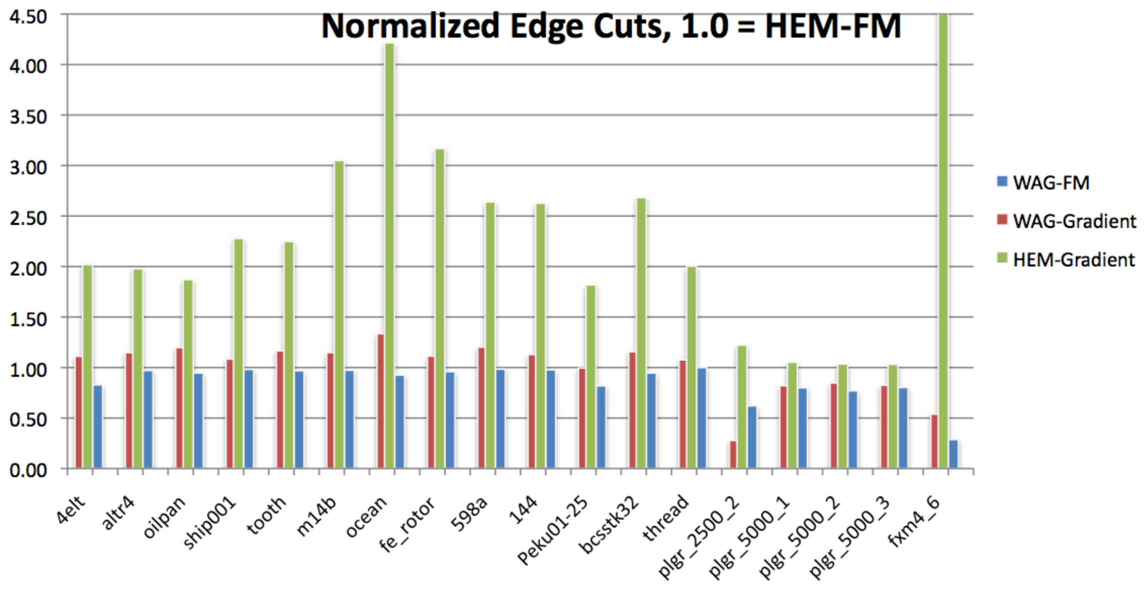

Figure 2. Edge cut for different coarsening schemes, scaled so that HEM with FM refinement is 1.0. In this test, we allow a fine vertex to be split over up to four coarse vertices.

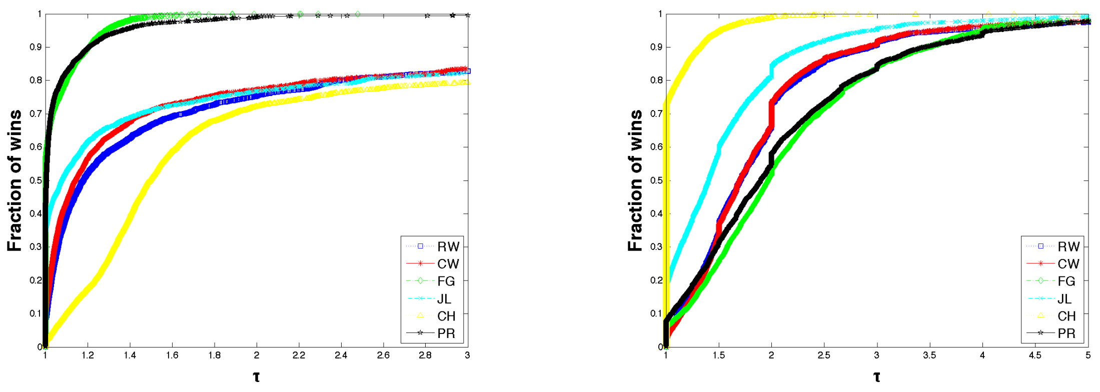

Figure 3. Performance profiles for comparison of common 1D (RW, CW) and some 2D (FG, JL, CH, PR) partitioning methods. We compared communication volume (left) and number of messages (right). Higher is better.

4. Sparse Matrix Ordering for Direct Solvers

Solving large, sparse linear systems of the type $Ax = b$ is a kernel that takes much compute time in scientific computing. Although iterative solvers are often preferred, sparse direct solvers are used for highly ill-conditioned problems or within preconditioners. We have investigated matrix reordering schemes to reduce fill in sparse matrix factorization and, thus, allow larger systems to be solved. We also provide ordering software (nested dissection) in Zoltan. Preliminary results indicate that ordering methods provided by PT-Scotch [7] can be significantly better than previous approaches. In an experiment using a large sparse matrix from a plasma code at Princeton Plasma Physics Lab (95 million nonzeros), we showed that the number of floating point operations in the factorization phase was reduced by up to 35% with Zoltan and PT-Scotch as compared to ParMetis (Figure 5). The memory footprint was also smaller with PT-Scotch. We are currently working with the SuperLU authors so that parallel Zoltan ordering may be

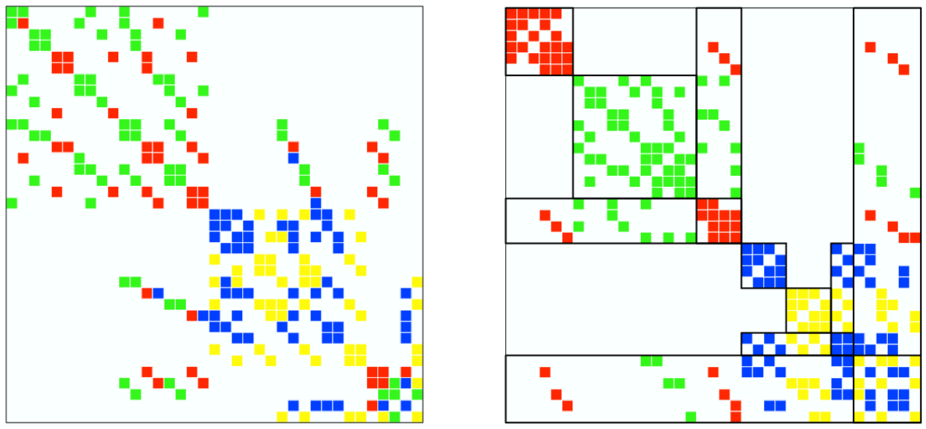

Figure 4. Nested dissection partitioning of a sparse matrix into four parts. Each nonzero is colored to show to which part (processor) it belongs. Original matrix (left) and permuted to reveal the nested structure (right).

supported in future versions of SuperLU_dist, which would make this new capability more easily accessible to a wide range of applications.

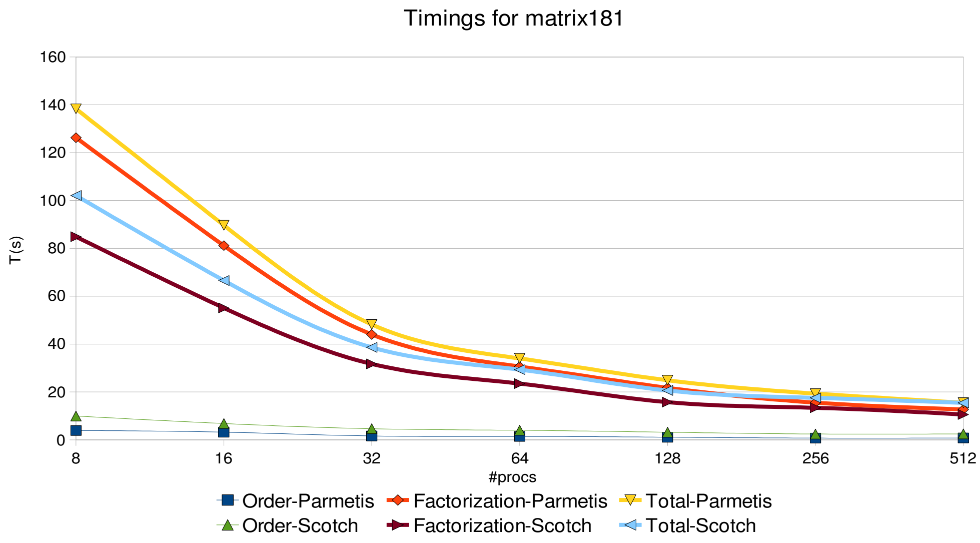

Figure 5. SuperLU_Dist 2.3 runtimes to factorize *matrix181*, for a Parmetis or a Scotch ordering of the unknowns. Typical use on this matrix is up to 64 processors.

We are also developing novel sparse matrix ordering approaches based on hypergraph partitioning [6, 4, 12]. For symmetric matrices, we have continued work on a novel hypergraph partitioning-based nested dissection approach that leverages structural decompositions of the form AA^T or ADA^T for achieving better orderings [6, 4]. Direct solutions of the systems in the form of $ADA^Tx = b$ requires triangular factorization of ADA^T, where A is a sparse and possibly rectangular matrix, and D is a diagonal matrix. We have proposed a new ordering method that applies hypergraph partitioning directly to A [4], as opposed to the standard approach of nested dissection (by vertex separator) on ADA^T. We also developed a simple, yet effective, structural factorization method that allows us to apply our ordering method to square symmetric matrices [4]. Furthermore, our hypergraph-based approach is better suited for the multilevel framework by accurately representing the separator at each level of the hierarchy.

For unsymmetric matrices, the reordering problem is even more challenging. Ordering is generally done as a preprocessing step, but in the unsymmetric case pivoting is needed at run-time to maintain numerical stability. Therefore, a column permutation is used to heuristically reduce fill, though row interchanges from pivoting may cause fill later. There are currently two approaches: Symmetrization (order $A + A^T$), or use a local ordering like COLAMD directly on A. Neither approach is satisfactory for large problems in parallel, because the first creates extra work and loss of concurrency, while the second is inherently serial. We have therefore developed a new algorithm, HUND [12] (Hypergraph Unsymmetric Nested Dissection), that is based on hypergraph partitioning and matching, both of which CSCAPES researchers are already working on. Major advantages of HUND over previous methods are that (i) it produces orderings of consistently high quality, (ii) it uses the structure of the original matrix (not symmetrized), and (iii) it is possible to parallelize. Parallel implementation of HUND in Zoltan is in progress.

Acknowledgments
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U. S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. The CSCAPES Institute is supported by the U.S. Department of Energy's Office of Science through the SciDAC program.

References

[1] E. G. Boman. A nested dissection approach to sparse matrix partitioning. *Proc. of Applied Math. and Mechanics*, 7(1):1010803 – 1010804, 2007. ICIAM07, Zurich, Switzerland.

[2] E. G. Boman and M. M. Wolf. A nested dissection approach to sparse matrix partitioning for parallel computations. Technical Report 2008-5482J, Sandia National Laboratories, NM, 2008.

[3] A. Buluc and E. Boman. Towards scalable parallel hypergraph partitioning. In *Proc. of Computer Science Research Institute (CSRI) 2008*, Albuquerque, NM, 2008. Sandia National Labs.

[4] U. V. Catalyurek, C. Aykanat, and E. Kayaaslan. Hypergraph partitioning-based fill-reducing ordering. Technical Report OSUBMI-TR-2009-n02 and BU-CE-0904, The Ohio State University, Dept. of Biomedical Informatics and Bilkent University, Computer Engineering Dept., 2009. Submitted for publication.

[5] U. V. Catalyurek, C. Aykanat, and B. Uçar. On two-dimensional sparse matrix partitioning: Models, methods, and a recipe. Technical Report OSUBMI_TR_2008_n04, The Ohio State University, Department of Biomedical Informatics, 2008. Submitted for publication.

[6] U. V. Çatalyürek. *Hypergraph Models for Sparse Matrix Partitioning and Reordering.* PhD thesis, Bilkent University, Computer Engineering and Information Science, Nov 1999.

[7] C. Chevalier and F. Pellegrini. PT-SCOTCH: A tool for efficient parallel graph ordering. *Parallel Computing*, 34(6–8):318–331, 2008.

[8] C. Chevalier and I. Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *Proc. of 3rd Workshop on Learning and Intelligent Optimization (LION-3)*, Trento, Italy, 2009.

[9] J. Cong and J. R. Shinnerl, editors. *Multilevel Optimization and VLSICAD.* Kluwer, 2003.

[10] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.

[11] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181. IEEE, 1982.

[12] L. Grigori, E. Boman, S. Donfack, and T. Davis. Hypergraph unsymmetric nested dissection ordering for sparse LU factorization. Technical Report 2008-1290J, Sandia National Labs, 2008. Submitted to SIAM J. Sci. Comp.

[13] B. Hendrickson and R. W. Leland. A multi-level algorithm for partitioning graphs. In *Supercomputing*, 1995.

[14] Y. F. Hu and J. A. Scott. A multilevel algorithm for wavefront reduction. *SIAM J. Scientific Computing*, 23:2000–031, 2001.

[15] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, University of Minnesota, June 1995.

[16] I. Safro, D. Ron, and A. Brandt. Multilevel algorithms for linear ordering problems. *Journal of Experimental Algorithmics*, 13:1.4–1.20, 2008.