Institute of Physics Publishing Journal of Physics: Conference Series **46** (2006) 453–457
doi:10.1088/1742-6596/46/1/062 SciDAC 2006

Combinatorial algorithms enabling computational science: tales from the front

Sanjukta Bhowmick[1,4], Erik G. Boman[2], Karen Devine[2], Assefaw Gebremedhin[3], Bruce Hendrickson[2], Paul Hovland[1], Todd Munson[1] and Alex Pothen[3]

[1] Mathematics and Computer Science Division, Argonne National Laboratory
[2] Discrete Algorithms and Math Department, Sandia National Laboratories
[3] Computer Science Department, Old Dominion University
[4] Department of Applied Physics and Applied Mathematics, Columbia University

E-mail: bah@sandia.gov

Abstract. Combinatorial algorithms have long played a crucial enabling role in scientific and engineering computations. The importance of discrete algorithms continues to grow with the demands of new applications and advanced architectures. This paper surveys some recent developments in this rapidly changing and highly interdisciplinary field.

1. Introduction

Combinatorial scientific computing (CSC) refers to the development, analysis and application of discrete algorithms in support of scientific and engineering computations. Although the CSC name is of recent vintage, combinatorial algorithms have long been an important, albeit often underappreciated part of scientific computing. For example, graph algorithms have been a key aspect of sparse linear algebra since the 1960s [1]. Graph coloring algorithms for Jacobian and Hessian evaluations in optimization were first proposed in the 1970s [2, 3].

Recent years have witnessed a blossoming of activity in CSC in a wide variety of areas within scientific computing. Geometric algorithms play a critical role in unstructured mesh generation. Graph, hypergraph and geometric partitioning algorithms are essential to the utilization of parallel computers. Graph algorithms play an important role in computational chemistry and biology. Genomic and proteomic analysis is rich in string algorithms. Graphs have long been a key tool for statistical physics. The list goes on and on.

This paper will briefly introduce several recent results in CSC. The goal is not to provide a detailed overview of the breadth of the field, or even to describe the included vignettes in full detail. A more comprehensive introduction to CSC can be found in [4], and the examples discussed below all contain citations to more detailed discussions. Rather, the goal of this paper is to provide a quick introduction to some aspects of CSC, to showcase its deeply interdisciplinary nature, and to highlight the broad enabling role that CSC plays in scientific computing.

2. Partitioning for parallel sparse matrix-vector multiplication

An important problem in parallel scientific computing is how to distribute work (load) among processors. We wish to minimize the communication cost while maintaining approximate load

balance. This problem is often modeled as graph (or hypergraph) partitioning, where the vertices are computational tasks and edges represent data dependencies.

We focus on a common kernel in many numerical algorithms: multiplication of a sparse matrix by a vector. For example, this operation is the most computationally expensive part of iterative methods for linear systems and eigensystems. The question is how to distribute the nonzero matrix entries (and the vector elements) among processors.

The most common matrix distribution is a one-dimensional (1D) decomposition of either matrix rows or columns. The communication needed for matrix-vector multiplication with a 1D distribution is shown in Figure 1. It has been shown [5] that this problem can be modeled as hypergraph partitioning, which is a generalization of graph partitioning. The hypergraph model is superior to the graph model because it more accurately represents communication volume. The hypergraph model has another advantage in that it works for nonsymmetric matrices while the graph model assumes symmetry.

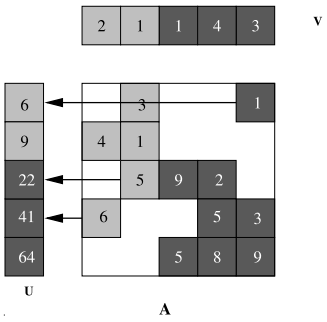

Figure 1. 1D column distribution of a sparse matrix for multiplication $u = Av$. There are only two processors, indicated by dark and light shading, and communication between them is shown with arrows.

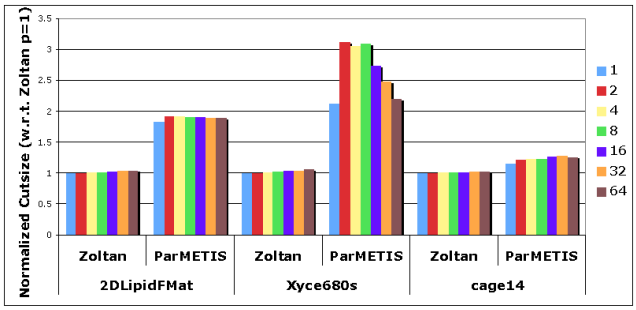

Figure 2. A comparison of partition quality (cuts) for Zoltan (hypergraph partitioning) and ParMETIS (graph partitioning) on $p = 1$ to 64 processors.

The Zoltan toolkit was developed at Sandia National Labs and contains a wide range of partitioning and load-balancing methods, including a parallel hypergraph partitioner [6]. We show results from three different application areas: the Tramonto DFT code for nanoscale fluid simulation (2DLipidFmat), circuit simulation (xyce680s), and DNA electrophoreses (cage14). All produce structurally symmetric sparse matrices. We computed row partitions using graph partitioning (ParMetis) and hypergraph partitioning (Zoltan). Figure 2 shows the estimated communication volume in the applications. The reduction in communication volume (cuts) depends on the matrix structure; up to a factor three in these tests. The actual run-time in applications is also reduced, but less dramatically than the communication volume indicates. The measured performance increase for the matrix-vector kernel in the Tramonto and Xyce examples was 13% and 18%, respectively, compared to graph partitioning on a 64-processor cluster.

3. Coloring in Parallel Processing and Automatic Differentiation

Coloring is an abstraction for partitioning a set of objects into few "independent sets". The notion of independence and the associated coloring rules vary from context to context. In the simplest case, adjacent vertices in a graph are required to receive different colors (distance-1 coloring). In parallel scientific computing, a distance-1 coloring is used to identify computational subtasks that can be performed concurrently. The goal is to use a small number of colors in order to reduce execution time. A greedy coloring algorithm is suitable for such a purpose.

Recently, Boman et al. [7] have developed an efficient scheme for parallelizing greedy coloring algorithms on distributed-memory computers.

The scheme has the following three key ingredients: (A) *Partitioning.* A graph is evenly partitioned among available processors. Each processor is responsible for coloring its assigned vertices. (B) *Speculative coloring.* Conflicts are tentatively tolerated—to maximize concurrency—and then detected and resolved. The scheme involves a few such iterations. (C) *Coarse-grained communication.* To reduce communication cost, a processor communicates only after coloring a *group* of vertices using current local information. Several variants of the scheme have been implemented using MPI. The codes are integrated with the Zoltan toolkit. Computational results on a PC cluster show that the scheme yields near-linear speedup. The goal is to use this work as a stepping-stone to develop coloring algorithms for petascale machines.

Another application area of graph coloring is automatic differentiation (AD). One efficient way of computing sparse Jacobians and Hessians relies on a partitioning of columns into a small number of groups. The idea is to reduce the AD computational effort by calculating *sums* of columns at a time, instead of calculating each column separately. The specific criterion used to partition columns depends on whether the nonzero entries of a matrix are to be retrieved from its *compressed* representation directly or indirectly, via substitution. Partitioning criteria for direct methods are stricter than those for substitution methods. Thus the latter require fewer groups and typically result in more efficient overall computation.

Structural orthogonality is a basic partitioning criterion used for direct methods. Two columns are structurally orthogonal if they do not have a nonzero at the same raw position. Gebremedhin et al. [8] showed that a structurally orthogonal partition of the columns of a Jacobian or a Hessian can be modeled using a distance-2 coloring of an appropriate graph—a bipartite graph for a Jacobian and an adjacency graph for a Hessian.

Symmetry exploitation in Hessian computation gives rise to less restrictive coloring variants: star coloring (direct method) and acyclic coloring (substitution method). A star coloring is a distance-1 coloring in which every path on four vertices uses at least three colors. An acyclic coloring is a distance-1 coloring in which every cycle uses at least three colors. The names here are due to the structure of two-colored induced subgraphs: in the first case the structure is a collection of stars, in the latter it is a forest. Gebremedhin et al. [9] have exploited these structures to design effective heuristic algorithms. Their algorithm for acyclic coloring is the first practical solution known for the problem. For some sparsity structures, computing a Jacobian by partitioning both columns and rows is more effective than a computation based on unidirectional partitioning. Bidirectional computation furnishes yet other coloring variants: star and acyclic *bicoloring*. These require colors for row-vertices to be disjoint from colors for column-vertices. For a comprehensive review of the role of coloring in derivative computation, see [8].

4. Performance improvements in a mesh quality optimization application

FeasNewt [10] is a mesh quality optimization application. Optimizing the quality of a mesh can significantly improve the performance of PDE simulations. We are using combinatorial techniques to improve the performance of FeasNewt itself. FeasNewt uses inexact Newton to optimize the average element quality of a mesh. To improve the performance of this code, we have applied several techniques: using a block sparse matrix data structure to store the Hessian to reduce memory bandwidth requirements, reordering the problem data to improve cache performance, and preconditioning the iterative method. These techniques can significantly reduce the computational time, especially for large problem instances.

Modern microprocessors are highly sensitive to the spatial and temporal locality of data sets. Therefore, reordering the vertices and elements in a mesh can have a significant impact on performance. We have developed metrics and models for mesh reordering and have investigated the performance of several reordering algorithms [11]. Modeling the mesh as a hypergraph is

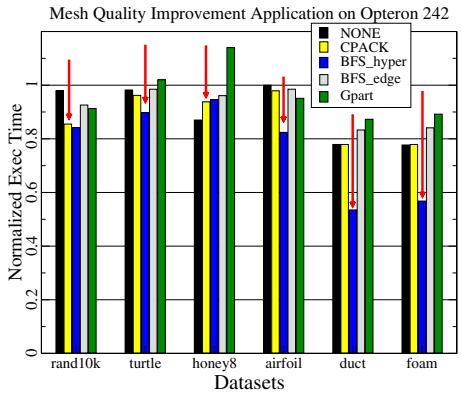

Figure 3. Effects of data reordering applied to the mesh quality optimization application. Arrows indicate the reordering heuristic with best predicted performance according to our metrics.

critical and can lead to performance improvements of nearly 50 percent. Figure 3 shows the effects of the data-reordering algorithms on execution time.

We compute the gradient and Hessian required for the inexact Newton algorithm using automatic differentiation techniques. Exploiting symmetry can reduce the cost of computing the Hessian by nearly half. However, to exploit this symmetry we must detect symmetry in the directed acyclic graph (DAG) used to represent a computation in automatic differentiation algorithms. This requires matching the vertices and edges with their duals and identifying duplicate mirror operations. By performing only one of these duplicate operations we can greatly reduce the computation costs.

Although testing symmetry of a general graph is an NP-complete problem, Hessian computational graphs provide more information about the structure, such as the direction and weight of the edges. Furthermore, we are interested only in the axis of symmetry perpendicular to the direction of the edges. Taking this extra information into consideration, we have developed a polynomial time, complexity $O(V^2)$, symmetry detection algorithm for Hessian computational graphs. The algorithm has two steps,

(i) *Step 1:* Group vertices such that for two vertices u and v are the same group if either of the two conditions are true
- *indegree(v)=indegree(u)* AND *outdegree(v)=outdegree(u)*
- *outdegree(v)=indegree(u)* AND *indegree(v)=outdegree(u)*

It is easy to see that if vertices a and b are symmetric pairs, then they must be in the same group.

(ii) *Step 2:* Subdivide each group further, based on edge weights and the neighbors of the vertices until there are only two vertices left in each group. The vertices in each group represent a symmetric pair of vertices.

5. Conclusions

The vignettes from the previous sections illustrate a few of the areas in which combinatorial algorithms are impacting and enabling scientific computing. For several reasons, we believe that the influence of CSC will continue to grow in both breadth and depth.

- High performance computers continue to grow in complexity with more processors and more complex memory hierarchies. Combinatorial tools are already crucial for obtaining high performance, and new and refined tools will continue to be needed.

- High fidelity scientific and engineering simulations require advanced algorithms and data structures for unstructured meshes, adaptive methods or advanced numerics. Discrete algorithms play important roles in enabling these kinds of capabilities. The enormous data sets produced by emerging applications will also require combinatorial sophistication in data management and analysis.
- Although not discussed in this paper, discrete algorithms are particularly pervasive in biology and information science. As these fields grow in importance, CSC will grow with them.

Like any interdisciplinary field, CSC faces challenging questions about publication venues, reward structures and (most importantly) education. But it is often the case that the most significant breakthroughs occur at boundaries between established scientific domains. We believe that CSC lies upon one of these rich interfaces, where theoretical insights are turned into practical advances.

Acknowledgments
Sandia is a multiprogram laboratory operated by Sandia Corporation, a LockheedMartin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. The work at Argonne was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy under Contract W-31-109-Eng-38.

References

[1] Seymour V. Parter. The use of linear graphs in Gaussian elimination. *SIAM Review*, 3:119–130, 1961.
[2] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.
[3] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20(1):187–209, February 1983.
[4] B. Hendrickson and A. Pothen. Combinatorial scientific computing: The enabling power of discrete algorithms in computational science. In *7th Intl. Mtg. High Perf. Comput. for Computational Sci. (VECPAR'06)*, Lecture Notes in Computer Science. Springer-Verlag, 2006. Invited paper, to appear.
[5] Ü. Çatalyürek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.
[6] K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, and U.V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Proc. IPDPS'06*. IEEE, 2006.
[7] E.G. Boman, D. Bozdağ, U. Catalyurek, A.H. Gebremedhin, and F. Manne. A scalable parallel graph coloring algorithm for distributed memory computers. In *Proc. of Euro-Par 2005*, volume 3648 of *Lecture Notes in Computer Science, Springer*, pages 241–251. Springer, 2005.
[8] A.H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.
[9] A.H. Gebremedhin, A. Tarafdar, F. Manne, and A. Pothen. New acyclic and star coloring algorithms with application to computing Hessians. Technical report, Old Dominion University, 2006. Submitted to SIAM J. Sci. Comput.
[10] T. S. Munson. Mesh shape-quality optimization using the inverse mean-ratio metric. Preprint ANL/MCS-P1136-0304, Argonne National Laboratory, Argonne, Illinois, 2004.
[11] Michelle Mills Strout and Paul D. Hovland. Metrics and models for reordering transformations. In *Proceedings of the Second ACM SIGPLAN Workshop on Memory System Performance (MSP)*, pages 23–34, June 2004.