Combinatorial Algorithms for Computational Science and Engineering

Erik G. Boman[1]**, Doruk Bozdag**[2]**, Umit V. Catalyurek**[2]**,**
Karen D. Devine[1]**, Assefaw H. Gebremedhin**[3]**, Paul D. Hovland**[4]**,**
and Alex Pothen[3]

[1] Discrete Algorithms and Math Department, Sandia National Laboratories
[2] Biomedical Informatics, and Electrical and Computer Engineering, Ohio State University
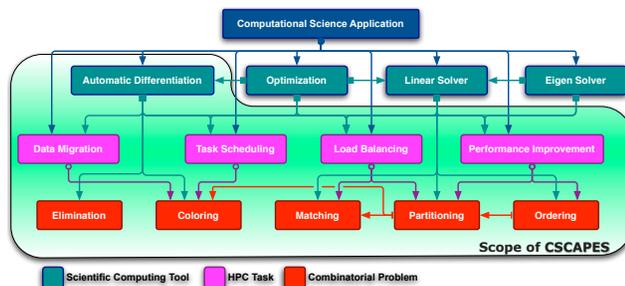[3] Department of Computer Sciences and Computing Research Institute, Purdue University
[4] Mathematics and Computer Science Division, Argonne National Laboratory

Abstract. We discuss recent activities in the SciDAC Applied Math Institute for Combinatorial Scientific Computing and Petascale Simulations (CSCAPES).

1. Introduction

CSCAPES was established in 2006 to help contribute to the SciDAC mission by developing infrastructural algorithmic and software technologies for supporting high-performance computing. Its scope encompasses four areas: load-balancing and related tasks in parallel processing (such as data migration and task scheduling); automatic differentiation (AD), a technology for transforming computer source code for computing a function into code for computing its derivatives; basic ingredients in linear solver technology; and runtime data and iteration reordering to improve performance in irregular computation.

Figure 1. Key research areas in CSCAPES and their relationship to a typical SciDAC application. An arrow from A to B indicates that A in some sense uses B.

The focus in CSCAPES in each of these areas is on the formulation (modeling) and solution (algorithms) of the underlying fundamental combinatorial problems, which are often phrased using graphs or hypergraphs. Load-balancing, the task of distributing data and work in a large-scale computation among the processors of a machine to minimize total execution time, can be phrased as a partitioning problem on a graph, or with more accuracy, on a hypergraph. When computational dependency among subtasks is modeled using a graph, a distance-1 coloring of the graph can be used to determine a scheduling of the tasks for concurrent computation. Distance-2 coloring and several other specialized variants are used to model matrix partitioning problems that arise in the efficient computation of sparse Jacobians and Hessians using AD. In AD, exploitation of associativity and commutativity in the chain rule of differential calculus

leads to a variety of vertex or edge elimination problems in the directed acyclic graph used to represent the computation of a function and its derivative. Various types of matchings in graphs are used in the solution of sparse linear systems (numerical preprocessing and block triangular decomposition) and in the coarsening phase of multilevel methods for graph partitioning. Work and storage reduction in direct solvers for large sparse systems calls for vertex ordering problems in graphs. Graph and hypergraph based models are used to capture the needs in reordering the nodes and elements within an unstructured mesh to improve runtime performance.

CSCAPES researchers are developing sequential and parallel algorithms and software for the combinatorial problems outlined in the previous paragraph. The load-balancing and parallelization capabilities are being deployed through the Sandia-housed Zoltan software toolkit. Similarly, the AD capabilities are being deployed via the tools being developed at Argonne within the OpenAD framework. In addition, serial as well as parallel codes (for coloring, matching, and re-ordering) are being separately developed at the academic institutions within CSCAPES, with the ultimate goal of being integrated with the major software outlets, Zoltan and OpenAD. Both Zoltan and the tools in OpenAD pre-date CSCAPES and have been under continuous development for several years with sustained support from DOE; similarly, the coloring work in CSCAPES was previously supported by NSF. Zoltan's internal interface is currently being redesigned to improve support for integration of third party libraries. For example, it has recently acquired an interface to the parallel ordering functionality available in PT-Scotch.

The load-balancing and partitioning facilities in Zoltan have been used to enable a wide variety of applications in structural mechanics, chemical engineering, groundwater flow, biological systems, electronic circuit simulations, and molecular dynamics. Likewise, the tools in OpenAD have been applied to a broad range of applications: modeling breast cancer, climate, weather, semiconductor devices, power networks, and groundwater; atmospheric chemistry; computational fluid dynamics; the network-enabled optimization server (NEOS); water reservoir simulation; and chemical kinetics.

In the remainder of this short paper we provide a few "nuggets" of our recent works, to give a flavor of the ongoing research and development activities within CSCAPES. For more information, consult the references listed in this paper and the website *http://www.cscapes.org*.

2. Dynamic Hypergraph Repartitioning and 2D Matrix Partitioning
In parallel adaptive computations, processor work loads may become unbalanced over time due to changes in the computations. In adaptive mesh refinement, for example, the refinement and coalescing of elements may cause significant load imbalance. This kind of application requires *dynamic* load balancing. Dynamic load balancing has several goals with complicated trade-offs: load balance in the new data distribution; low communication cost within the application; low cost to migrate data from the old distribution to a new one; and short repartitioning time.

CSCAPES researchers have recently developed a hypergraph-based repartitioning method [5]. In this method, new vertices are added to represent each part. Then, each of these vertices is connected to the objects in the same part with new (weighted) edges to model the data migration cost. The resulting hypergraph accurately accounts for both the communication volume and the migration cost to move the data, and can be partitioned using existing algorithms and software.

In Figure 2, results from applying this hypergraph repartitioning method (HG-Repart) to an adaptive mesh refinement problem from the ALEGRA shock physics code [2] is shown. HG-Repart is compared with other connectivity-based approaches—"scratch-remap" hypergraph partitioning (HG-Scratch) and graph-based partitioning (G-Scratch) and repartitioning (G-Repart)—and the coordinate-based methods Recursive Coordinate Bisection (RCB) and Hilbert Space-Filling Curve partitioning (HSFC). Scratch-remap methods compute partitions without accounting for existing part assignments, and then remap parts to try to minimize migration costs. While the coordinate-based methods do not model communication and migration
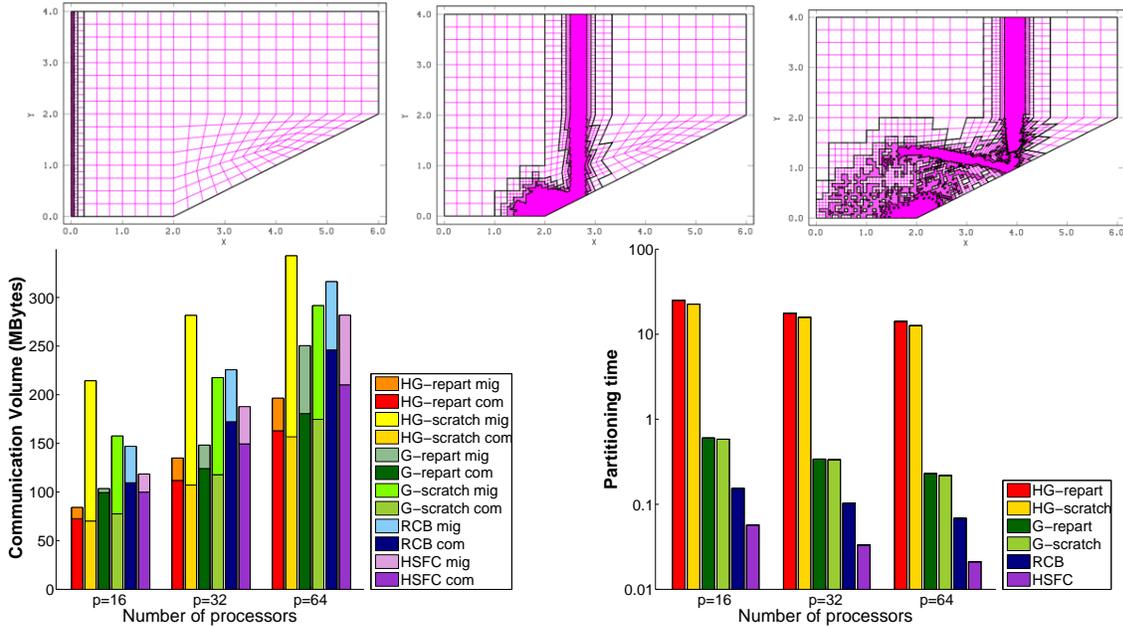
Figure 2. *(Top) Sample adaptive finite element meshes at time-steps 0, 54, and 108; the smallest mesh has 132,209 nodes, and the largest has 1,380,266 nodes. (Bottom) A comparsion of communication volume and execution time over 109 time-steps of an adaptive finite element simulation on 16, 32 and 64 processors of a Linux cluster (dual 2.2GHz AMD Opterons with 4 GB RAM and a Myrinet network).*

costs explicitly, they are implicitly incremental, making them viable for repartitioning mesh-based problems. The repartitioning hypergraph method HG-repart produced lower total cost (application communication volume plus data migration communication volume) than all other methods. The coordinate-based methods were much faster than graph- and hypergraph-based methods, but their total cost was high. Execution time for HG-repart was greater than G-repart, indicating the need for faster heuristics in the hypergraph implementation for applications with relatively low and homogeneous connectivity.

The multiplication of a sparse matrix by a vector is an important kernel in scientific computing. CSCAPES researchers have long studied how to optimize the performance of this operation in parallel by reducing communication volume [6]. Recently, new partitioning algorithms have been developed for novel two-dimensional data distributions that can be computed efficiently using current partitioning software [1, 11].
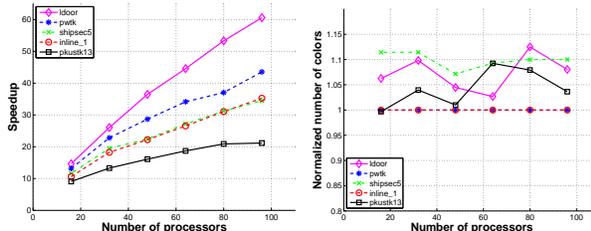
3. Parallel Coloring

CSCAPES researchers have recently developed a framework for parallelizing greedy distance-1 coloring algorithms on distributed-memory computers [4]. Formulated in a generic manner, the techniques employed in the framework include: careful exploitation of features of the initial data distribution; speculation—maximizing concurrency by tentatively tolerating inconsistencies and then detecting and resolving them; randomization; and infrequent, coarse-grain communication among processors as opposed to frequent, fine-grained communication. In [4], several specialized algorithms designed using the framework have been presented and evaluated; experiments carried out on moderate size machines demonstrated good scalability.

In a work that is currently being finalized [3], the distance-1 coloring framework has been extended for the distance-2 coloring problem, an archetypal model for partitioning problems that arise in the efficient computation of *sparse* Jacobian and Hessian matrices using AD (see Figure 3). To address the challange of accessing color information of distance-2 neighbors,

Figure 3. *Structurally orthogonal column partition of a derivative matrix A and its representation as a distance-2 coloring in the adjacency graph $G_a(A)$ when A is symmetric, and bipartite graph $G_b(A)$ when A is nonsymmetric.*
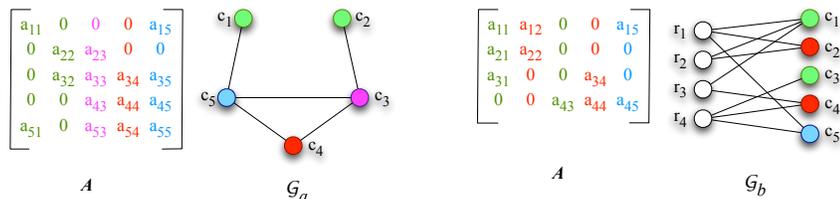
graph name	$|V|$	$|E|$	Degree max	colors (seq)
pkustk13	94,893	3,260,967	299	303
shipsec5	179,860	4,966,618	125	140
pwtk	217,918	5,708,253	179	180
inline_1	503,712	18,156,315	842	843
ldoor	952,203	22,785,136	76	112

Figure 4. *Performance of the parallel distance-2 coloring algorithm on select application graphs. The maximum degree in a graph is a lower bound on the optimal number of colors needed for distance-2 coloring. A greedy sequential algorithm gave near optimal (or perhaps optimal) solution for these test graphs. The parallel algorithm used in the worst case only 12% more colors.*

efficient means of information exchange between processors hosting a pair of vertices that are two edges away from each other are introduced. Experiments run on up to a hundred processors have shown that the resulting parallel distance-2 coloring algorithm scales well and gives a solution very close to the sequential variant, which in turn often gives a near optimal solution (see Figure 4). MPI implementations of the distance-2 coloring algorithm as well as the distance-1 coloring algorithm discussed earlier have been incorporated in Zoltan.

In the case of Hessian computation, the distance-2 coloring model does not exploit the available symmetry. Models that exploit symmetry to various degrees exist. The most accurate model for a direct method is *star coloring* and that for a substitution method is *acyclic coloring*. CSCAPES investigators have in a recent work exploited the structures of two-colored induced subgraphs in star and acyclic coloring to design novel sequential algorithms that have been shown to be superior to previously known approaches [9]. The serial software they have developed has been incorporated into the operator-overloading-based AD tool ADOL-C. Experiments using ADOL-C on Jacobian computation in a Simulated Moving Bed process, a method used in chromatographic separation in chemical engineering, and Hessian computation on an electric power grid problem have shown that the use of the coloring techniques in large-scale problems reduces overall run time by several orders of magnitude [7, 8].

4. Automatic Differentiation

Suppose the derivatives of $y = \prod_{i=1}^{n} x_i$ with respect to all x_i need to be computed. One algorithm to compute this function is $y_1 = x_1; y_i = x_i * y_{i-1}, i = 2, \ldots, n$. One can then compute the derivatives using $\bar{y}_n = 1; \bar{y}_i = x_{i+1} * \bar{y}_{i+1}, \bar{x}_{i+1} = y_i * \bar{y}_{i+1}, i = n-1, \ldots, 1$. Then, $\partial y / \partial x_i = \bar{x}_i$. This algorithm to compute the derivatives corresponds to the reverse mode of automatic differentiation. The reverse mode is optimal in this case and within a factor two of optimal for all scalar functions. Now, consider the case of $y = w^T \prod_{i=1}^{n} M_i v$, where w and v are vectors and the M_i are dense, rectangular matrices (y remains a scalar). Then one might compute y using $b_0 = w^T; b_i = b_{i-1} M_i, i = 1, \ldots, n; y = b_n v$. Then, the

derivatives with respect to the elements of the M_i can be computed optimally (or nearly so) using $\bar{b}_n = v; \bar{b}_{i-1} = M_i\bar{b}_i, \overline{M}_i = \bar{b}_ib_{i-1}, = n, \ldots, 1$. However, if one computes the more general sequence $a_1 = w^TM_1v_1$, $a_2 = w^TM_1M_2v_2$, $\ldots$, $a_n = w^TM_1M_2\ldots M_nv_n$, where $y = \sum_{i=1}^{n} a_i$, and then computes the derivatives of each a_0 using the algorithm above (summing to produce the derivatives of y), the result is far from optimal. The problem is that although the chained vector-matrix products from the left can be reused from one a_i to the next, the chained matrix-vector products from the right cannot be reused because the v_i vary. To illustrate the problem, suppose all of the M_i are square matrices of size $n \times n$. Then, the cost of computing the function is $O(n^3)$ while the cost of computing the derivative matrices is $O(n^4)$. If, however, one recasts the computation as $b_0 = w^T; b_i = b_{i-1}M_i, a_i = b_iv_i, i = 1, \ldots, n$; then, one can simply recur backward on the b_k sequence, with an extra term thrown in to account for the inner product with x_k. This brings the cost back down to $O(n^3)$. Thus, in developing methods to compute the derivatives of a function, it is important to consider the full algorithm used to compute the function, not just subcomponents. We identified a way to use a similar technique to reduce the cost of computing the gradient of a function arising in computational chemistry from $O(n^5)$ to $O(n^4)$ [10]. This reduction will have a major impact on the size of problems that can be solved.

References
[1] E. G. Boman. A nested dissection approach to sparse matrix partitioning. In *Proc. of Appl. Math. Mech., to appear*, 2008. (Presented at ICIAM07, Zurich, July 2007).

[2] E. A. Boucheron, K. H. Brown, K. G. Budge, S. P. Burns, D. E. Carroll, S. K. Carroll, M. A. Christon, R. R. Drake, C. G. Garasi, T. A. Haill, J. S. Peery, S. V. Petney, J. Robbins, A. C. Robinson, R. Summers, T. E. Voth, and M. K. Wong. *ALEGRA: User Input and Physics Descriptions Version 4.2*. Sandia National Laboratories, Albuquerque, NM, 2002. Tech. Report SAND2002-2775.

[3] D. Bozdag, U. Catalyurek, A. Gebremedhin, F. Manne, E. Boman, and F. Ozguner. Distributed-memory parallel graph coloring algorithms for Jacobian and Hessian computation. To be submitted to SIAM J. Sci. Comput., 2008.

[4] D. Bozdag, A. Gebremedhin, F. Manne, E. Boman, and U. Catalyurek. A framework for scalable greedy coloring on distributed memory parallel computers. *Journal of Parallel and Distributed Computing*, 68(4):515–535, 2008.

[5] U. Catalyurek, E. Boman, K. Devine, D. Bozdag, R. Heaphy, and L. Riesen. Hypergraph-based dynamic load balancing for adaptive scientific computations. In *Proc. of 21th International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE, 2007. 11 pages.

[6] U. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, 1999.

[7] A. Gebremedhin, A. Pothen, A. Tarafdar, and A. Walther. Efficient computation of sparse Hessians using coloring and automatic differentiation. *INFORMS Journal on Computing (to appear)*, 2008. 28 pages.

[8] A. Gebremedhin, A. Pothen, and A. Walther. Exploiting sparsity in Jacobian computation via coloring and automatic differentiation: a case study in a simulated moving bed process. In *The Fifth International Conference on Automatic Differentiation, August 2008, Bonn, Germany (to appear)*, 2008. 10 pages.

[9] A. Gebremedhin, A. Tarafdar, F. Manne, and A. Pothen. New acyclic and star coloring algorithms with application to computing Hessians. *SIAM Journal on Scientific Computing*, 29(3):1042–1072, 2007.

[10] Personal communication with Ron Shepherd.

[11] M. Wolf, E. Boman, and B. Hendrickson. Optimizing sparse matrix-vector multiplication by corner partitioning. In *Proc. of PARA08, to appear*, 2008.