COMBINATORIAL for

Combinatorial Scientific Computing and Petascale Simulations (CSCAPES) is a SciDAC Institute established to harness terascale and petascale performance for scientific simulations critical to DOE's mission. The Institute will develop and deploy fundamental enabling technologies in high-performance computing that employ algorithms from combinatorial or discrete mathematics. CSCAPES is also committed to educating the next generation of researchers who will develop and apply combinatorial techniques to problems in computational science.

The CSCAPES Institute

We live in a discrete universe. People, molecules, and bits come in integral numbers. It is not surprising, then, that although computational models in science and engineering are expressed using the language of continuous mathematics, such as differential equations and linear algebra, techniques from discrete or combinatorial mathematics play an important role in solving these models efficiently.

Combinatorial scientific computing (CSC) is the name for the interdisciplinary field in which researchers identify combinatorial subproblems that arise in computational science and engineering (CSE), design graph and hypergraph algorithms to solve these subproblems, and create high-performance software implementing the algorithms. CSC plays a crucial enabling role in applications requiring parallelization, differential equations, optimization, eigenvalue computations, and management of large-scale datasets.

While work has been ongoing in CSC since the 1970s, it is only in the past few years that CSC researchers, realizing their common intellectual and aesthetic interests, have organized to form a community of their own. Three international workshops in CSC have been held since 2004, with broad participation from academia, government laboratories, and industry from several countries. Scientists from the DOE laboratories have been leaders in this effort, organizing the meetings, presenting talks and posters, and engaging in international collaborations.

Combinatorial scientific computing is the name for the interdisciplinary field in which researchers identify combinatorial subproblems that arise in computational science and engineering, design graph and hypergraph algorithms to solve these subproblems, and create high-performance software implementing the algorithms.

The Combinatorial Scientific Computing and Petascale Simulations (CSCAPES, pronounced "seascapes") Institute is a natural outgrowth of the research activities undertaken by several members of the CSC community. Researchers from Old Dominion University, Ohio State University, Colorado State University, and two national laboratories—Sandia National Laboratories (SNL) and Argonne National Laboratory (ANL)—serve as the co-principal investigators of the CSCAPES Institute. CSCAPES is led by Dr. Alex Pothen, a professor of computer science and computational science at Old Dominion University.

The major research tasks to be undertaken by the CSCAPES Institute are illustrated in figure 1. A typical CSE application might require a sequence of numerical optimization problems to be solved, eigenvalues and eigenvectors to be computed, and/or nonlinear and linear systems of equations to be solved. Solving this application on a parallel computer would require the computational work to be equally distributed on the processors of the parallel machine. In an adaptive computation, the work distribution would change during the computation, and the work and data would need to be redistributed among the processors to enable it to be solved quickly. Several computational tasks would need to be scheduled to maximize the utilization of the processors and to reduce the idle time processors spend waiting for data or synchronizing. The memory access times needed by CSE codes working with unstructured meshes and

Algorithms PETASCALE Science

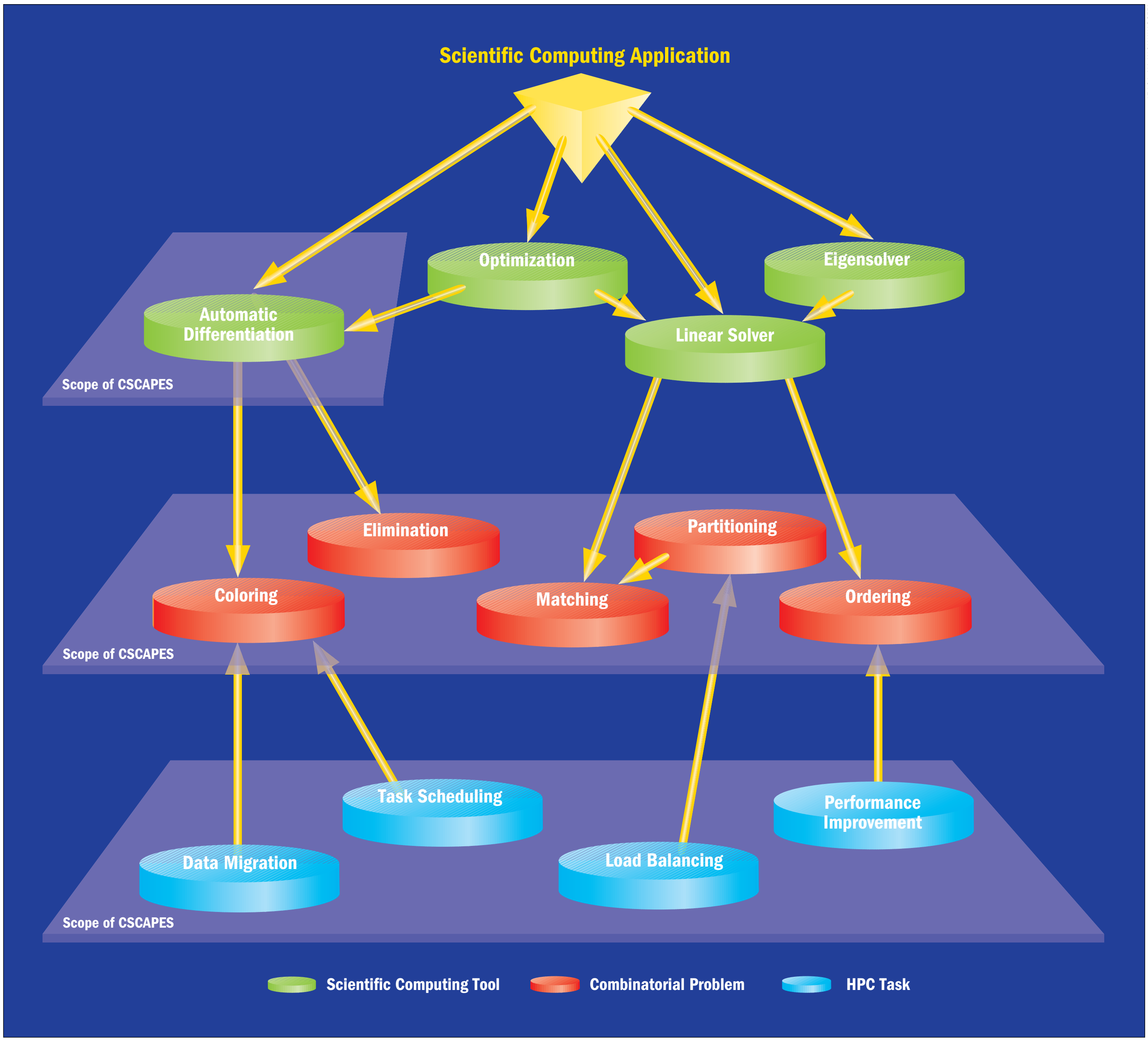

Illustration: A. Tovey

Figure 1. *This diagram shows the key problems in combinatorial scientific computing that the CSCAPES Institute researchers will work on. The solution of a SciDAC application on a parallel computer requires scientific computing tools (the second row of the figure), and involves high-performance computing tasks (the fourth row). The fundamental combinatorial problems that need to be solved by these tools and tasks are shown in the third row. An arrow from A to B indicates that A in some sense uses B.*

ILLUSTRATION: A. TOVEY SOURCE: S. ATTAWAY, SNL

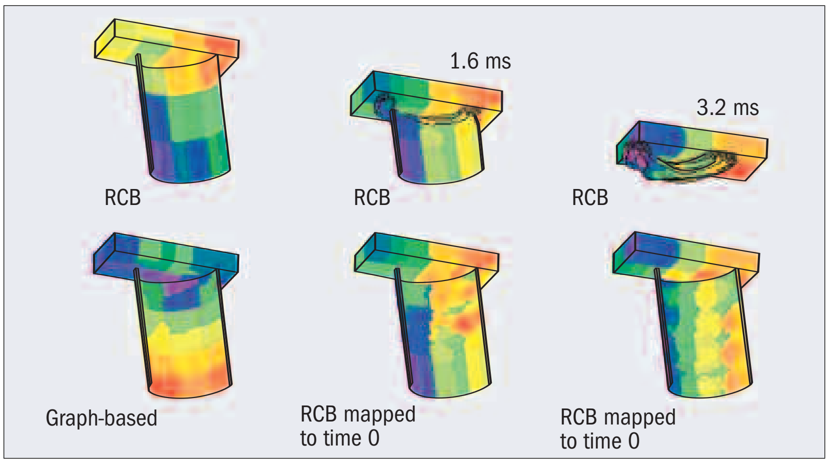

Figure 2. *A simulation of a can crushed by a falling brick. This is a multiphysics simulation where two different partitioning methods are used. Graph partitioning is applied to the mesh to reduce communication in the finite element computations, while a geometric method (RCB) is used for faster contact detection. The colors show how parts of the mesh are assigned to processors. The top row shows the RCB partitioning at three different times. The bottom left figure shows the graph partitioning decomposition. The bottom middle and right figures show the RCB decomposition on the original geometry (without the deformation). Some data must be mapped back and forth between the two data decompositions in the crash simulation. This simulation was performed at SNL using the PRONTO code.*

sparse data structures could be reduced by reordering the data accesses and computations. Several combinatorial problems, namely, graph and hypergraph partitioning, vertex and edge reordering, vertex and edge coloring, and graph matching arise in these contexts.

Similarly, automatic differentiation (AD) tools are needed to ease the task of computing derivative matrices of multivariate functions in nonlinear models. The computation of such functions can be represented as the composition of unary or binary arithmetic operations and intrinsic functions in a mathematical software library, governed by a directed acyclic graph (DAG). The function can then be differentiated using the chain rule with the help of the graph representation, by composing the derivatives of the elementary arithmetic operations. Vertices and edges of these graphs can be eliminated from the DAG representations to reduce storage and computational costs. Several graph coloring models help to exploit sparsity and symmetry inherent in the derivative matrices to compute their elements quickly.

In the remainder of this article, we provide brief overviews of the research areas alluded to in figure 1. For each, we describe the research problems, current capabilities, some applications, and future goals.

Automatic differentiation (AD) tools are needed to ease the task of computing derivative matrices of multivariate functions in nonlinear models.

Load Balancing

Large-scale scientific simulations are typically run on high-performance parallel computers with thousands of processors. Even desktop computers now have multiple cores, and petascale machines will consist of thousands of multicore processors. An important task is to distribute data and work of a large-scale computation among the processors to minimize total execution time. This problem is known as "load balancing" or "partitioning." We assume that the "owner computes" strategy is used, which is common in large-scale computational science problems, since the datasets are huge, and communication costs are larger than computation costs on current distributed-memory architectures. In this scheme, each data item is assigned to a unique processor, the owner, which performs the computations associated with it. Most data items are mapped only to the owner, but items on the boundaries of a partitioned data structure could be mapped to more than one processor. Communication is required when a computation needs access to data items that reside on different processors.

There are two goals, often conflicting, in load balancing. First, the work that can be performed concurrently should be evenly distributed among the processors in order to avoid processors that finish early having to wait for the slowest processor to finish its task. Second, communication between processors is relatively slow compared to computation, and so it should be as small as possible. The communication requirements are dictated by the data dependencies in the application at hand. These goals are conflicting since the first drives the data to be distributed among the processors, while communication costs are lowest if the data reside on one processor.

The load balancing problem is pervasive in parallel scientific computing, and is critical for achieving high performance in many SciDAC applications. Examples include structural mechanics, chemical engineering, groundwater flow, biological systems, electronic circuit simulations, and molecular dynamics (examples in figures 2 and 3).

Most simulations are based on some underlying geometry and mathematical equations. The mathematical model typically uses a mesh-based discretization, such as finite differences or finite elements. One can then use the geometry in the load balancing. Alternatively, the discretized system of equations can be considered as a sparse matrix and the load balancing can be applied to operations on this matrix.

For static problems where the structure does not change during the simulation, the load balancing can be accomplished by a single partitioning step. This may be done on one processor if the description of the problem fits in the memory. However, large problems need to be partitioned in parallel.

For dynamic or adaptive problems, the situation is more complex. Often the problem changes over

S. PLIMPTON, SNL

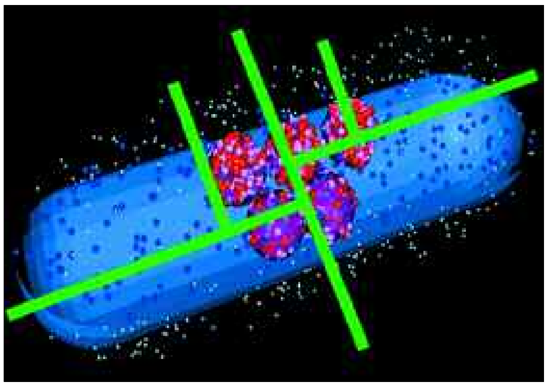
Figure 3. *A particle-based simulation of organelles in a rod-shaped* Synechococcus *cyanobacterium. The inner clumps of particles are carboxysome organelles where carbon fixation reactions take place. This is a snapshot of a metabolic cycle that converts inorganic carbon to organic sugar. A mesh is used to triangulate the outer membrane (not shown here). The domain is partitioned into six regions using a geometric method, shown by green lines, to balance the number of particles per processor. This simulation was performed with ChemCell and Zoltan.*

time, for example by adaptive mesh refinement. In such cases, the load balance may deteriorate over time making rebalancing necessary. The idea is to periodically call the load balancer to see if a better data distribution can be achieved. If so, the application data are moved to the new distribution and the simulation continues. This procedure is known as dynamic load balancing. Used in this context, each load balancing (partitioning) step must be performed at run time, so the partitioner should be fast. Furthermore, in order to reduce data migration costs, the old and new data distributions should be similar.

Several different algorithms have been designed for load balancing. Roughly, they can be divided into two groups: geometric- and connectivity-based methods. The geometric methods use geometric coordinates to partition space into regions, while the connectivity methods use a graph or hypergraph to represent data and their dependencies. Figure 2 provides an example that uses both approaches. Geometric partitioning algorithms are fast, but graph/hypergraph algorithms typically reduce the communication volume more. CSCAPES members have shown in earlier work that hypergraph models are more expressive than graph models for load balancing, and can model communication costs more accurately. Zoltan is a software toolkit for load balancing and parallel data management, developed mainly at SNL, and supported in part by the CSCAPES SciDAC Institute.

Zoltan contains implementations of the most important algorithms for partitioning and load balancing, relieving application developers from the burden of implementing their own load balancing, and making it possible to try different strategies with minimal effort. A recent addition to Zoltan is a parallel hypergraph partitioner, which can replace traditional graph partitioners in many applications, and often reduce communication volume and execution time. We anticipate the hypergraph partitioner to be particularly effective for applications with more irregular structures than unstructured meshes, such as circuit simulation. Zoltan also contains several other useful tools for parallel data management, such as an unstructured communication library to do coarse-grain communication and a distributed data directory to keep track of parallel data. Zoltan uses a flexible, data-structure-neutral interface that may also be used for other graph and hypergraph algorithms, like coloring.

CSCAPES researchers and colleagues at Lawrence Livermore National Laboratory (LLNL) are currently evaluating the scalability and limitations of partitioning tools like ParMETIS and Zoltan. Preliminary results indicate that these load balancing tools work well on hundreds and maybe up to a few thousand processors, depending on the method being used and the computing platform. A challenge that CSCAPES researchers will address in future years is to provide robust and scalable load balancing tools for massively parallel petascale platforms. There is a critical need for further research into better models and algorithms for load balancing. For example, while current graph and hypergraph models minimize communication volume, in practice other factors, such as latency and number of messages, are also important.

Several different algorithms have been designed for load balancing. Roughly, they can be divided into two groups: geometric, and connectivity-based methods.

Performance Improvement

Unstructured meshes are popular because they discretize space more flexibly than do structured meshes. However, one drawback to unstructured meshes is that computational performance is poor relative to structured meshes because the mesh data structure does not directly map to a data array of the same dimension. Instead, the data associated with the nodes and the elements of an unstructured mesh must be linearized in some way. When a computation visits an element in the mesh, indirect memory references are needed to access the data associated with each node in the element. The indirect memory references lead to twice as many memory loads. Additionally, since the accesses to the data array are not sequential, they have poor spatial locality.

Previous work has shown that reordering the nodes and elements within a mesh so that neighboring nodes and elements are local within the linearized layout improves the serial performance of such computations. Reorderings based on space

CSCAPES researchers are pursuing new combinatorial heuristics to further reduce the cost of gradient, Jacobian, and Hessian computations.

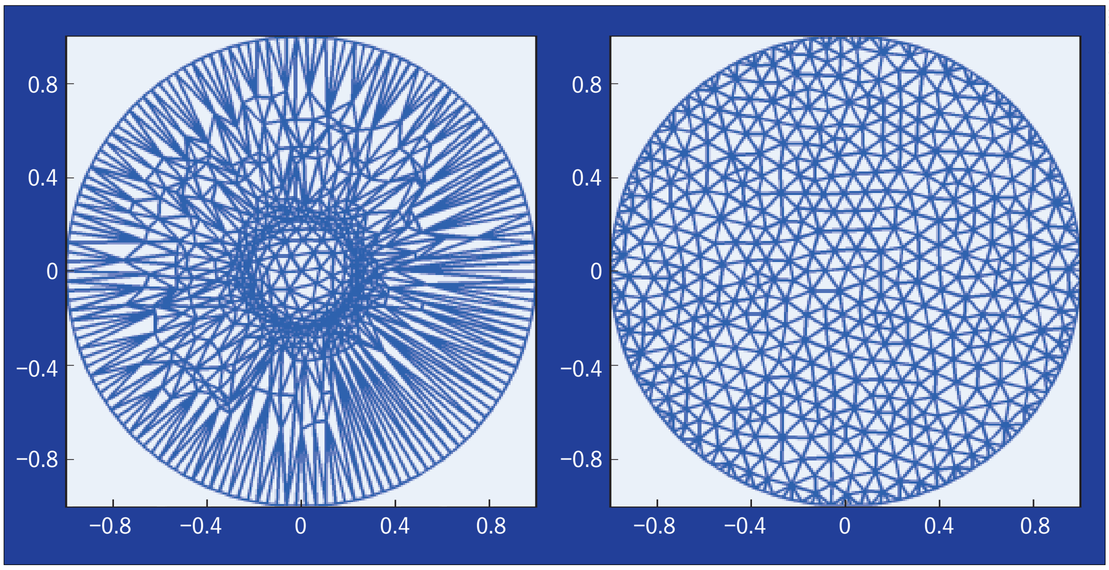

Figure 4. *A mesh of poor quality (left), and an improved mesh (right), using the optimization tool FeasNewt developed by Dr. Todd Munson of ANL. Good reorderings of the mesh data structures enable FeasNewt to run twice as fast relative to an ordering provided by the mesh generator.*

filling curves, breadth-first-search on the graph, and graph partitionings have been employed earlier to improve performance. CSCAPES researchers have created hypergraph models for the problem of reordering nodes and edges of the mesh to enhance data locality. One application where reordering the nodes and elements in a mesh improves performance is a mesh optimization application called FeasNewt developed by Dr. Todd Munson at ANL. FeasNewt improves mesh quality by repositioning the nodes of a mesh geometrically, so that each simplex in the mesh more closely matches the ideal simplex of the same dimension. Figure 4 shows an example mesh before and after mesh improvement. A good reordering of the nodes and simplices in the mesh enables FeasNewt to run twice as fast relative to an ordering provided by the mesh generator.

We are continuing to investigate reordering heuristics for graph and hypergraph models for spatial and temporal locality in unstructured mesh computations. Our goal is to determine the mesh and machine characteristics that can guide the reordering strategy.

Automatic Differentiation

AD is a technique for transforming subprograms that compute some mathematical function into subprograms that compute the derivatives of that function. The resulting derivatives are used for uncertainty quantification, optimization algorithms, nonlinear solvers for discretized differential equations, and solution of inverse problems using nonlinear least squares. AD techniques combine rules for differentiating the functions intrinsic to a given programming language with strategies for applying the chain rule while respecting the control flow of the original program.

The computation of a function and its derivatives via the chain rule can be represented by a DAG; vertices of this DAG represent variables or intermediate expressions, and the edge weights correspond to partial derivatives. The associativity of the chain rule of differential calculus leads to exponentially many possible “modes,” sequences for combining intermediate operands to compute partial derivatives. This choice of modes can be used to reduce the number of operations and storage needed for computing the partial derivatives. Choosing a mode can be interpreted as selecting an order in which vertices and edges are “eliminated” (corresponding to partial evaluation of the associated variables) on the DAG. Finding an optimal mode—one that minimizes the number of operations—for evaluating a Jacobian matrix is intractable (NP-complete). In practice, heuristics are used to identify accumulation strategies with low operations counts and/or storage requirements. CSCAPES researchers are pursuing new combinatorial heuristics to further reduce the cost of gradient, Jacobian, and Hessian computations. Coloring algorithms, discussed later in this article, will also be used to reduce the computational effort associated with evaluating Jacobians and Hessians.

The AD tools developed by ANL researchers over the years with support from DOE’s Office of Science, and now supported in part by the CSCAPES Institute, 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-

Toolkit for Advanced Optimization

AD is used to compute gradients and Hessians for the parallel solution of optimization problems in the Toolkit for Advanced Optimization (TAO) project at ANL. TAO focuses on scalable software for optimization problems, including nonlinear least squares, unconstrained minimization, bound constrained optimization, and general nonlinear optimization.

Many of the algorithms employed by TAO require first-order, and sometimes second-order, derivatives, and AD has been used to compute these derivatives. Figure 5 shows a sequence of solutions and their deviation from the optimal solution for a bound-constrained minimization problem.

Read more about the TAO project online at: http://www.mcs.anl.gov/tao

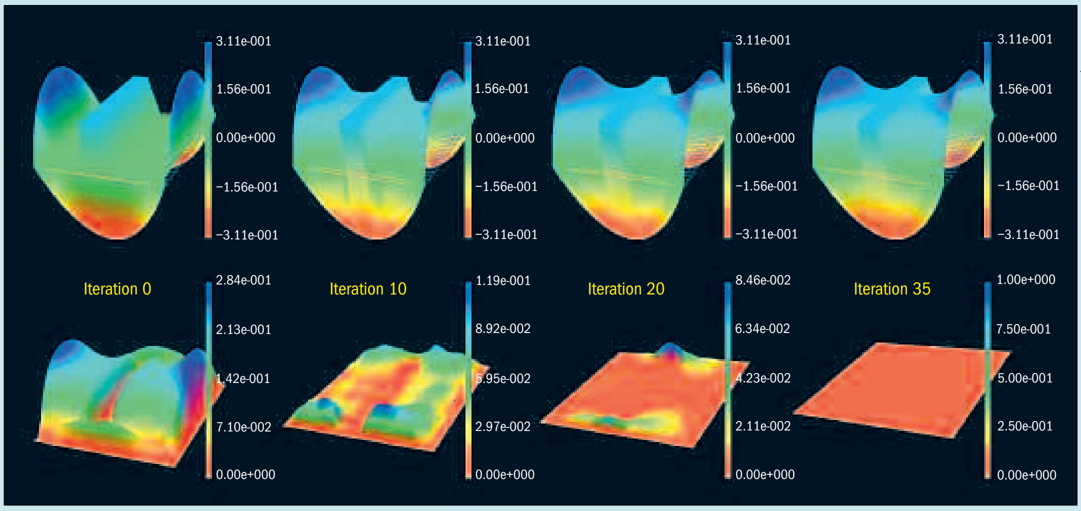

Figure 5. *A sequence of solutions (top) and their deviation from the optimal solution (bottom) for a bound constrained minimization problem. The objective is the surface with minimal area that satisfies Dirichlet boundary conditions and is constrained to lie above a solid plate.*

enabled optimization server (NEOS); water reservoir simulation; and chemical kinetics. The widespread use of the tools helps ensure their robustness and guides future work on theory and implementation. AD software has played an enabling role with both the Toolkit for Advanced Optimization (TAO) project (sidebar "Toolkit for Advanced Optimization") and the MIT General Circulation Model (sidebar "The MIT General Circulation Model," p32).

Graph Coloring

In many large-scale CSE problems, the Jacobian and Hessian matrices that need to be computed are typically sparse, meaning that many of the matrix entries are zero. This inherent sparsity (and when applicable, symmetry) available in the derivative matrices can be exploited to compute the nonzero entries efficiently. One efficient way of computing a sparse Jacobian or Hessian using AD is computation via "compression." The idea is to reduce the computational effort in AD by calculating sums of columns at a time, instead of calculating each column separately. Columns that are to be computed together are determined by exploiting structural properties of the matrix; in particular, the structural information is used to partition the set of columns into a small number of groups of columns. The specific criterion used to partition columns depends on whether the nonzero entries of the 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 row position. 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. In a distance-*k* coloring, vertices in every path of length *k* edges receive distinct colors. In the simplest case, distance-1 coloring, every pair of adjacent vertices is required to receive different colors. Figure 7 illustrates how distance-1 and distance-2

The idea is to reduce the computational effort in AD by calculating sums of columns at a time, instead of calculating each column separately.

The MIT General Circulation Model

AD is being applied to the MIT General Circulation Model (MITgcm). Figure 6 displays a map of sensitivities of zonal volume transport through the Drake Passage to changes in bottom topography everywhere in a barotropic ocean model. The model is based on the shallow water model used by Dr. Martin Losch and Dr. Carl Wunsch, and extended to a global configuration at 2 x 2 degree horizontal resolution with realistic topography. Enhanced sensitivities are manifested both locally and remotely, for example, over the Kerguelen Plateau, over the South Pacific Ridge, and in the Indonesian Throughflow. Sensitivities are mediated through the flow field represented by the model dynamics.

This sensitivity map was achieved through a single adjoint model integration. Generating this map using only the forward mode (one of the modes of computing partial derivatives, as discussed in the main article) would have required over 23 cpu-days. A naive adjoint computation (another mode) required approximately 2.5 hours. An improved adjoint computation, incorporating combinatorial heuristics based on problem structure and domain-specific compiler analysis to reduce storage requirements, lowered the runtime to 22 minutes.

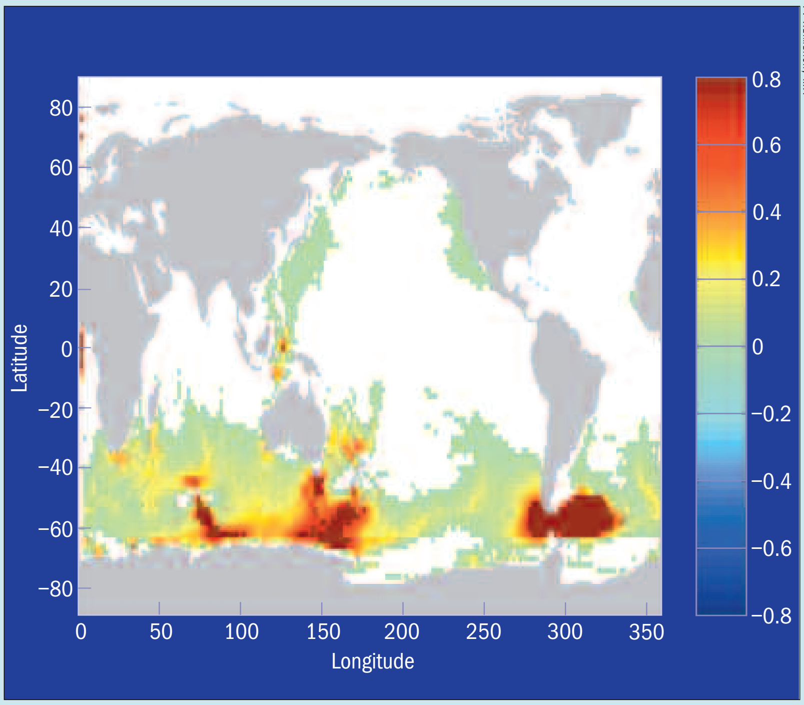

Figure 6. *Sensitivities of zonal volume transport through the Drake Passage with respect to changes in ocean depth.*

Investigators in CSCAPES have exploited these structures to design novel algorithms that have been shown to be superior to previously known approaches.

colorings can be used to evaluate Jacobians.

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. These variant colorings are illustrated in figure 8. 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. Investigators in CSCAPES (in an effort that began in an earlier National Science Foundation-funded project) have exploited these structures to design novel algorithms that have been shown to be superior to previously known approaches. Deploying these algorithms to compute sparse Hessians using an AD tool reduced its execution times by three orders of magnitude over a computation that ignores sparsity.

For some sparsity structures, computing a Jacobian by partitioning both columns and rows (bidirectional partitioning) is more effective than a computation based on either columns or rows exclusively (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. In a recent journal article, CSCAPES members have provided a comprehensive review of graph coloring models and algorithms in derivative computation. Serial coloring software they have developed has in part been incorporated into an operator-overloading-based AD tool. CSCAPES is currently engaged in developing parallel versions of the coloring software and its integration with source-transformation-based automatic differentiation tools.

Graph coloring is also a useful model in the discovery of concurrency in parallel scientific computing. In particular, when computational dependency among subtasks is modeled using a graph, a distance-1 coloring can be used to identify the subtasks that can be computed simultaneously. The number of colors used would then correspond to the computational steps required and therefore is desired to be as small as possible. Since the graph to be colored in such a context is often already distributed among the processors of a machine, the coloring needs to be performed in parallel as well. CSCAPES researchers have recently developed a framework for parallelizing greedy distance-1 coloring algorithms on distributed-mem-

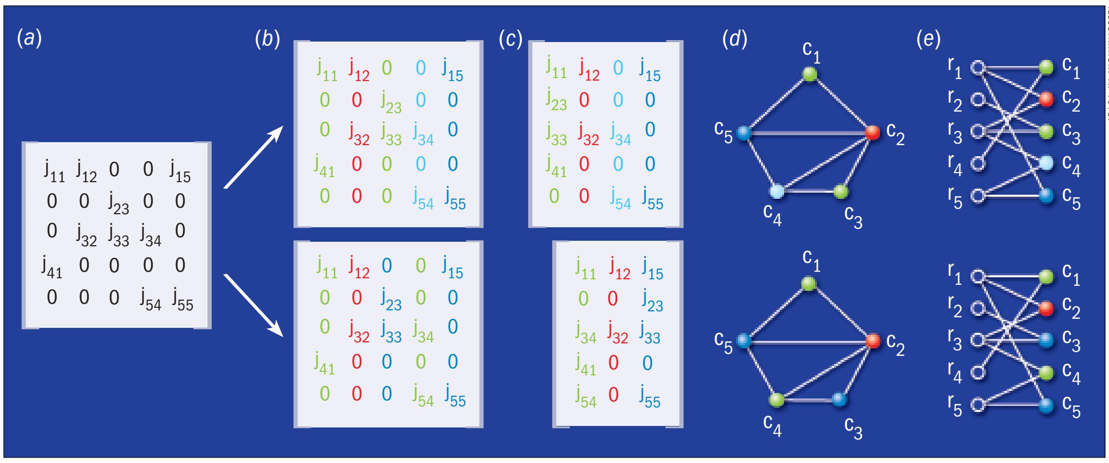

Figure 7. *The evaluation of a sparse Jacobian can be made efficient by computing several structurally orthogonal columns together. Out of this simple observation arises the need for partitioning the columns of a large Jacobian into as few groups of structurally orthogonal columns as possible. This partitioning problem can be modeled and effectively solved using graph coloring, an idea introduced by Tom Coleman and Jorge Moré in the 1980s. In the illustration, the original Jacobian (a) is partitioned into groups of structurally orthogonal columns in two different ways (b), the lower being better since it uses fewer groups. The corresponding compressed representations, where columns in the same group are put together, are shown in (c). Each partition is also represented as a distance-1 coloring in the column intersection graph (d), and as a partial distance-2 coloring in the bipartite graph (e). The distance-1 coloring formulation is due to Coleman and Moré; CSCAPES members introduced and advocate the use of the distance-2 coloring formulation, since it is more flexible and computationally more economical.*

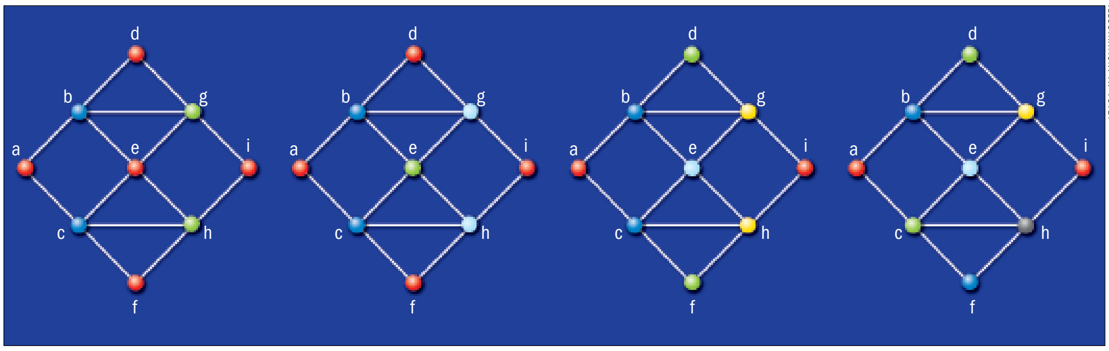

Figure 8. *From left to right: a distance-1, an acyclic, a star, and a distance-2 coloring of a graph. In a distance-1 coloring, a vertex receives a color distinct from its neighbors; in a distance-2 coloring, every path on three vertices uses three colors; in a star coloring neighbors get different colors and every path on four vertices uses at least three colors; in an acyclic coloring, neighbors get different colors and every cycle uses at least three colors.*

ory parallel computers. Message passing interface (MPI) implementations of the framework have been incorporated in the Zoltan parallel data management and load balancing library. Computational results on modest-sized PC clusters show good scalability, and future plans include extending the work to petascale machines.

Matching

A matching in a graph is a pairing of the two endpoints of some edges. Each vertex can be paired exclusively (matched) with one other vertex, or not paired at all. A matching is illustrated by the green edges in the graph in figure 9 (p35). An important task in many scientific computing problems is to compute a matching that is largest according to some objective function. The simplest objective is to find a matching in a graph with the maximum number of edges, known as the maximum cardinality matching problem. A variant of the matching problem has non-negative weights associated with the edges; the weight of a matching is the sum of the weights of the edges in the matching. In the maximum-edge-weighted matching problem, we are required to compute a matching with the maximum weight. A less studied variant assigns weights to the vertices, with the weight of a matching defined as the sum of the weights on the vertices that are endpoints of the matched edges. In a maximum-vertex-weighted matching, the goal is to compute a matching of maximum vertex weight.

Computational results on modest size PC clusters show good scalability, and future plans include extending the work to petascale machines.

Collaborations, Outreach, and Education

As part of our outreach mission, the CSCAPES Institute will collaborate with other SciDAC applications and enabling technology groups that use CSC software tools, as well as create new algorithms and software for combinatorial problems that arise in their work. We are collaborating with enabling technology groups working on solvers, meshing, performance, and other areas, to provide integrated software tools for SciDAC applications. We have a few ongoing conversations with some of the SciDAC application groups; other groups desirous of working with us can contact us directly and through the SciDAC Outreach Center.

CSCAPES also works with partners from Europe and Asia to fulfill our mission of being an international center for excellence in CSC research. In parallel sparse matrix software, we collaborate with colleagues from the European Center for Research and Advanced Training in Scientific Computation (CERFACS) in Toulouse, France, and Tel Aviv University in Israel. Researchers from the Universities of Bergen (Norway) and Utrecht (The Netherlands) work with us on load balancing, coloring, and matching problems. We also interact with leading groups in AD from the Universities of Dresden and Aachen in Germany.

CSCAPES researchers led the organization of the Society for Industrial and Applied Mathematics (SIAM) Workshop on CSC in 2007 (CSC07), and two earlier international workshops on CSC in 2004 and 2005, as part of their outreach. Two of the workshops organized in the U.S. were supported by the DOE, and a third was organized at CERFACS, in Toulouse, France. Participation from scientific and engineering communities that make use of CSC algorithms and software to solve problems in CSE has been a significant hallmark of these past workshops. We will continue to organize international workshops in this area, offer tutorials at the annual SciDAC meeting and at other venues, and work with applications researchers to ease the adoption of combinatorial software in the CSE community. The CSCAPES Institute will work through the SciDAC Outreach Center in these efforts to make other research communities aware of the SciDAC program.

Visit the SciDAC Outreach Center online at: http://outreach.scidac.gov/

All three maximum matching problems can be solved in polynomial time in the size of the graph, and hence from a theoretical perspective these problems are considered to be "well-solved." However, the best-known algorithms compute optimal matchings in superlinear time, and these can be prohibitively slow for massive graphs with millions of vertices and edges. Hence, in the past few years several approximation algorithms that compute matchings, which can be proved to be within some factor, say half or two-thirds, of the weight of an optimal matching have been designed. These approximation algorithms are theoretically and practically much faster than the algorithms for computing optimal matchings. Furthermore, they are more amenable to parallelization, and CSCAPES members propose to develop parallel algorithms for matching on petascale machines.

Computing a matching in an edge-weighted graph is a key computational step in the load balancing problems that CSCAPES members will work on. Matching is a fundamental problem with many other applications in computer science and computational science.

One application of matching that we highlight is in the context of a circuit simulation problem. Dr. Rob Hoekstra, Dr. David Day, and colleagues on the Xyce project at SNL have used matchings in analog models of circuits, in which networks of devices are coupled via Kirchhoff's current and voltage laws. The nonlinear devices generate stiff, coupled differential algebraic equations, which lead to linear systems of equations with ill-conditioned coefficient matrices. However, for steady-state solutions, the coefficient matrix can be decomposed into a collection of smaller submatrices, obtained from the block triangular form (btf) of the matrix. The btf of a sparse matrix is illustrated in figure 9. These researchers found that the computational work to solve the sparse system of linear equations is reduced by two orders of magnitude when the btf is employed. As an example, on a circuit problem with 682,712 rows and columns, the KLU solver developed by Dr. Tim Davis of the University of Florida took 1,918 seconds without the btf, but the system was solved in 9 seconds with the btf. The KLU solver is part of the Trilinos software package being developed at SNL by Dr. Mike Heroux and colleagues. Also, the condition numbers of the diagonal blocks in the btf are much smaller than that of the original matrix, enabling the circuit model to be solved when methods that work with the entire matrix break down numerically. In the future, the Xyce group needs the capability of modeling circuits with 100 million rows and columns; parallel matching algorithms will become critical for such problems.

As part of our outreach mission, the CSCAPES Institute will collaborate with other SciDAC applications and enabling technology groups that use CSC software tools, as well as create new algorithms and software for combinatorial problems that arise in their work.

Training the Next Generation

CSCAPES members will train the next generation of interdisciplinary computational scientists in CSC algorithms and software at pre-doctoral and post-doctoral levels. In the Fall 2006 and Spring 2007 semesters, CSCAPES has offered a weekly Access Grid research seminar to familiarize our students and collaborators with current research in CSC. Through the Access Grid and teleconference links, participants from all five CSCAPES institutions were able to gain a better understanding of the work of the multiple groups within the institute. Students working on CSCAPES projects at the universities will be co-mentored by laboratory scientists, and will spend one or more summers at the participant laboratories, to facilitate broad multidisciplinary training. Currently only a handful of universities in the U.S. offer training for students in CSC research; with our efforts, we hope to increase the number of computational scientists trained in CSC.

Currently only a handful of universities in the U.S. offer training for students in CSC research; with our efforts, we hope to increase the number of computational scientists trained in CSC.

ILLUSTRATION: A. TOVEY

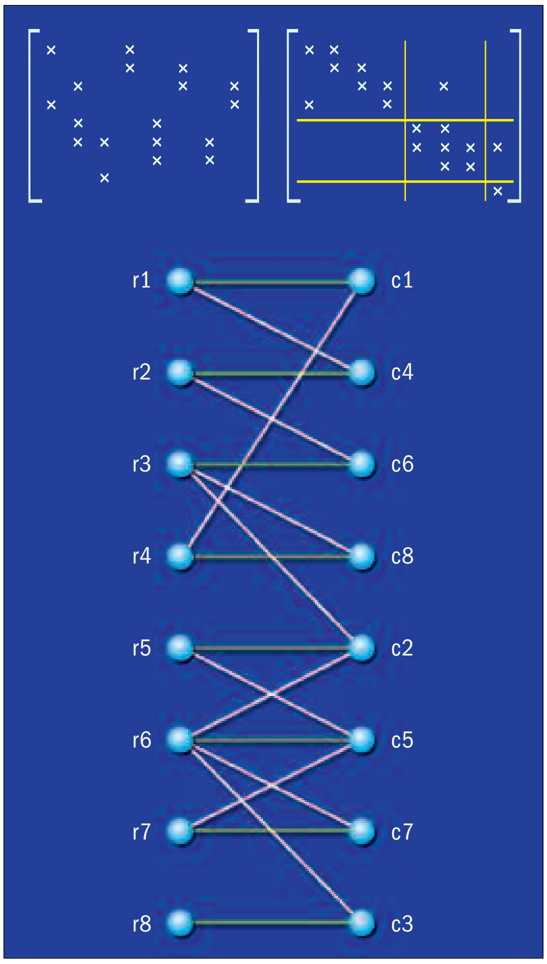

Figure 9. *A sparse matrix (top left) and a permutation of its columns that yields its block triangular form (top right). The permutation is obtained using a maximum matching in the bipartite graph of the matrix (bottom). Row vertices are on the left and column vertices are on the right of the graph, and edges corresponding to a maximum matching are drawn in green. This example has three diagonal blocks as shown. The block triangular form reduces the work in solving linear systems of equations with the matrix, since only its diagonal blocks need to be factored in sparse Gaussian elimination; it also ameliorates ill-conditioning in the matrix.*

Summary

CSCAPES focuses on algorithm and software development in three specific areas for complex SciDAC applications: load balancing and parallelization toolkits, automatic differentiation capabilities, and parallel graph and sparse matrix computations. As the era of petascale computing looms, these areas are becoming increasingly critical for achieving high performance for many SciDAC applications involving irregular computations and large datasets. The Institute is also committed to developing new collaborations and outreach opportunities (sidebar "Collaborations, Outreach, and Education"), as well as educating the next generation of researchers who will ultimately develop and apply combinatorial techniques to future problems in computational science (sidebar "Training the Next Generation").

Many scientific breakthroughs occur at the boundaries between fields where ideas and techniques cross-fertilize each other; we believe that CSC lies at one of these fruitful boundaries. As SciDAC applications grow in complexity and size, optimal algorithmic efficiency becomes paramount, and research in exact and approximation algorithms for CSC problems offers the potential for dramatic advances in simulation capability. ●

This article is dedicated to the memory of Ken Kennedy.

Contributors: Dr. Alex Pothen (PI), Dr. Assefaw H. Gebremedhin, and Dr. Florin Dobrian at Old Dominion University; Dr. Erik G. Boman, Dr. Karen D. Devine, and Dr. Bruce A. Hendrickson at SNL; Dr. Paul Hovland, Dr. Boyana Norris, and Dr. Jean Utke at ANL; Dr. Umit V. Catalyurek at Ohio State University; and Dr. Michelle Mills Strout at Colorado State University

Further Reading
http://www.cscapes.org