1

Combinatorial Scientific Computing: Past Successes, Current Opportunities, Future Challenges

Bruce Hendrickson
Sandia National Labs

Alex Pothen
Purdue University

1.1 Introduction

As detailed in other chapters in this book, combinatorial techniques have become essential tools across the landscape of computational science. The history of what we now call Combinatorial Scientific Computing (CSC) has involved the steady accretion of new domains where combinatorial theory and algorithms have been used to enable computational science and engineering. The 1970s and 1980s witnessed the flourishing of graph algorithms in sparse direct methods. The 1990s saw the growth of combinatorial algorithms as key enablers of parallel computing. In recent years, graph algorithms have become essential to automatic differentiation, and play a pivotal role in computational biology. Discrete algorithms also play important roles in mesh generation, computational chemistry, performance optimization, and many other fields.

In this chapter, we provide a brief overview of the historical roots of combinatorial scientific computing, the formation of the CSC community in the last decade, and provide our (of course, limited) perspective on likely future directions for CSC. We had provided a longer exposition on our view of Combinatorial Scientific Computing a few years ago [19]. It is our belief that the coming years will see still more applications of combinatorial scientific computing. Computational science remains a young, and rapidly changing discipline. As it evolves, new challenges will arise which discrete algorithms will be able to help solve. Our predictions will certainly be incomplete, and probably wrong—certainly many past trends would have been difficult to foresee. But we hope that by generating curiosity and interest in new CSC applications, our predictions become self-fulfilling.

0-8493-8597-0/01/$0.00+$1.50

We also believe that new applications will enrich the CSC community, leading to new combinatorial problems, abstractions, algorithms, and analysis. As has been the case in the past, we expect this work to have broader impact on the combinatorial theory and algorithms communities.

Since CSC is so pervasive within computational science, we believe that the best way to anticipate the future of CSC is to explore general trends within scientific computing. This is the approach we will take in this paper.

1.2 The CSC Community

1.2.1 The Roots of the CSC Community

Sparse Matrix Computations. One root of the CSC community comes from research in sparse matrix computations. Harry M. Markowitz, the Economics Nobel laureate in 1990, writes about working with sparse matrices in modeling industrial capabilities using linear programming at the RAND Corporation in the 1950s [28].

> Our models strained the computer capabilities of the day. I observed that most of the coefficients in our matrices were zero; i.e., the nonzeros were "sparse" in the matrix, and that typically the triangular matrices associated with the forward and back solution provided by Gaussian elimination would remain sparse if pivot elements were chosen with care. William Orchard-Hayes programmed the first sparse matrix code. ... Sparse matrix techniques are now standard in large linear programming codes.

The *Markowitz scheme* [27] for pivot selection in factoring sparse unsymmetric matrices is still in use today, but research in sparse matrix computations grew gradually in the 1960s before blossoming in the 1970s. Seymour Parter [32] introduced a graph model for Gaussian elimination on sparse matrices in 1961. A number of early sparse matrix conferences were held at I.B.M. from 1969 under the leadership of Ralph Willoughby [39, 36], and a conference was held at Oxford in 1971 [34]. Collections of articles presented at two of these meetings were published as books, and these were followed by further meetings and book collections. Rose [35], in a PhD thesis written at Harvard, showed that the adjacency graph of the Cholesky factor obtained by Gaussian elimination of a sparse symmetric matrix was a chordal graph. George's thesis at Stanford (1971) and Duff's thesis at Oxford (1972) were early contributions to sparse matrix algorithms and their implementations. Duff's comprehensive 1977 survey on sparse matrix research was influential on early researchers in the field. The text book by George and Liu [14], the Waterloo Sparspak, sparse matrix routines in the Harwell library (HSL), and the Yale Sparse Matrix Package (YSMP) enabled the widespread use of sparse matrix computations in many areas of computational science and engineering. The book, Graph Theory and Sparse Matrix Computation (edited by George, Gilbert and Liu), which resulted from a special year on linear algebra held at the Institute for Mathematics and its Applications (IMA) at the University of Minnesota in 1991, provides a snapshot of research current at that time [13]. Much more detail is available in the slides that accompanied Iain Duff's talk on the Development and History of Sparse Direct Methods at the SIAM Conference on Applied Linear Algebra in 2009[1].

[1] Slides are available at `www.siam.org/meetings/la09/talks/duff.pdf`

Automatic Differentiation. Another root of the CSC community comes in the context of derivative computation in optimization. Bauer [3] showed in 1974 how a computational graph could be associated with the evaluation of a mathematical function in terms of intrinsic mathematical operations such as addition, multiplication, and elementary mathematical functions (trigonometric functions, transcendentals, etc.), with vertices corresponding to independent, intermediate, and dependent variables, and with an edge joining an operand of an intrinsic operation to its result. This computational graph is now used in Automatic Differentiation (AD) to compute the derivative of a function specified by a program by applying the chain rule of differentiation along paths between dependent variables and independent variables [15].

Yet another early use of combinatorial methods arose in computing Jacobians and Hessians efficiently with the fewest function evaluations (when finite differencing is used), or with the fewest passes through the computational graph (when AD is used). The essential idea is that of structural orthogonality, i.e., when two columns of the Jacobian have nonzeros in disjoint row indices, the nonzeros in both columns can be evaluated simultaneously by differencing along both column directions. Furthermore, this observation can be extended to a subset of columns with the property that for each row, at most one of the columns in the subset has a nonzero in it. This observation was used by Curtis, Powell, and Reid in 1974 to compute sparse Jacobians; it was then extended to Hessians, where symmetry could be exploited in addition to sparsity, by Powell and Toint. Coleman and Moré [7] observed that the problem of minimizing the number of function evaluations for the Jacobian could be modeled as a vertex coloring problem on a derived graph, thus establishing that the minimization problem was NP-hard, and proposed a greedy heuristic approach with a number of vertex ordering algorithms. Coleman and various coauthors also extended the graph model to Hessian matrices, to relaxed versions of structural orthogonality (leading to a tree-based computation of the nonzeros beginning with the leaves in the tree, and proceeding iteratively after removing the leaves computed), and bidirectional evaluations of Jacobians where both columns and rows are used.

Statistical Physics. Since the 1930s, statistical physicists have been using combinatorial models as simplified representations of complex physical systems (see [17] for a survey). The paradigm for this work is the Ising spin model, which attempts to represent the salient features of ferromagnetism. In the Ising model, atoms are assumed to be located at the points of a two- or three-dimensional lattice, and they only interact with atoms at neighboring lattice points. Each atom has a spin, and the energy is a function of the spins of neighboring atoms. The goal is to understand the characteristics of low energy configurations of spins, subject to various external and boundary conditions. This seemingly simple problem leads to difficult combinatorial optimization problems with rich graph properties. Key combinatorial kernels include counting the number of perfect matchings in a graph, and their connection to matrix Pfaffians. Several Nobel Prizes in physics have been awarded for work that has deep combinatorial underpinnings.

The success of the Ising model in describing the essence of some magnetic phenomena led to the development of a plethora of combinatorial models for other problems. These include protein folding, quantum systems, material science, and orbital mechanics. The statistical physics community is often interested in the asymptotic statistics of optimal solutions to their combinatorial models. This is somewhat different from the traditional focus in CSC on algorithmic performance to enable scientific modeling and simulation. But the two communities are closely aligned in spirit and in interests, and we foresee significant opportunities for beneficial interactions in the future.

Parallel Computing. Partial differential equations model many physical phenomena in the sciences and engineering, and their solutions are often computed through discretizations

on finite difference or finite element meshes. Complex geometries and efficiencies in computation often require the use of unstructured or block-structured meshes. The computational intensity of these simulations necessitates the use of parallel computing. To make effective use of a parallel computer, the problem must be decomposed into subtasks, the subtasks must be assigned to processors, and the resulting communication and computation must be carefully orchestrated. The goals of these steps are to ensure that the work is equally distributed among the processors while keeping the communication costs low, so as to ensure that the parallel computation completes in the least possible time. Algorithms for solving this load-balancing problem use graph models of the computation to compute good partitions and mappings.

Partitioning methods for parallel computing benefited from earlier work on graph partitioning for sparse direct methods. Graph partitioning provides a mechanism for ordering sparse matrices through a divide and conquer paradigm called nested dissection, pioneered by Alan George [12] in 1973. George's work was generalized to sparse and irregular graphs by Lipton, Rose and Tarjan [25], and led to theory and algorithms for finding vertex and edge separators that divide a graph into two subgraphs of roughly equal sizes. By recursive application of graph partitioning, the load balancing problem in parallel computation and the problem of computing fill-reducing orderings for sparse matrices could be solved. This work led to the development of spectral graph partitioning algorithms as well as multi-level graph partitioning in the 1990's.

1.2.2 Organization of the CSC Community

Although the many subcommunities mentioned above were employing combinatorial mathematics, algorithms and software in the context of scientific computing, they remained isolated from each other because they contributed to different fields in scientific computing and computational science and engineering, attending different conferences. By 2000, a group of thirty international researchers began efforts to formally organize a community. An email discussion in 2002 settled on the name *Combinatorial Scientific Computing* to describe the research undertaken by members of this community. An email subscription server was created for the CSC community[2], and a number of minisymposia on CSC were organized at SIAM and ICIAM conferences during this period.

Five CSC workshops have been organized thus far, beginning with a SIAM workshop on CSC in February 2004, co-located with SIAM Conference on Parallel Processing, in San Francisco, CA. The organizers of this meeting were John Gilbert, Horst Simon, Sivan Toledo, and the authors. A special volume of the Electronic Transactions on Numerical Analysis (ETNA) consisting primarily of papers presented at CSC04 was published in 2005 (available as Volume 21 at `www.emis.de/journals/ETNA`). The second CSC Workshop was held in Europe, at CERFACS, Toulouse in 2005, with Iain Duff and Alex Pothen serving as Co-Chairs of the meeting. The following two CSC workshops were organized as SIAM Workshops in 2007 (co-located with SIAM Conference on Computational Science and Engineering at Costa Mesa, CA) and 2009 (co-located with SIAM Conference on Applied Linear Algebra at Seaside, CA). The Dagstuhl Seminar on CSC, which served as the stimulus for the papers in this volume, was organized in January 2009. The next CSC workshop, CSC11, will be held in May 2011 in Darmstadt, Germany, co-located with the SIAM Conference on Optimization, and Uwe Naumann and Ali Pınar serve as Co-Chairs of the workshop.

[2]Subscribe at `https://lists.purdue.edu/mailman/listinfo/csc`

Currently Paul Hovland, Sivan Toledo and the authors serve as members of the Steering committee for this series of CSC workshops. One measure that indicates that CSC is thriving as a community is that the the responsibility of organizing the biennial SIAM CSC Workshops has been passed from the original group of organizers to the 'next generation' in the CSC community.

1.3 Current Opportunities

From the disparate foundations discussed above, combinatorial scientific computing has emerged to be recognized as a key enabler for a wide array of scientific areas. The cross-fertilization of disciplines that has been encouraged by the formation of the CSC community has accelerated progress in both algorithm development and applications. In this section, we briefly sketch some of these areas of current activity. Many are discussed in more detail in other chapters within this book, but several are not.

The diversity of applications of CSC can be daunting, but this seeming complexity masks significant underlying commonality. Several algorithmic themes are at the core of CSC, and recur throughout the chapters of this book. Graph and hypergraph algorithms are central to many (perhaps most) CSC activities. Geometric algorithms and algebraic methods on graphs also appear prominently. CSC researchers span a range of activities including modeling, algorithm development, theoretical analysis, application and empirical performance analysis. Parallel and architecturally-informed algorithms also play a prominent role.

One area of high CSC activity and impact is the development and deployment of techniques to support parallel computing. Since the early work of Simon [37], graph partitioning has been used as a tool for dividing a parallel computation among processors to balance the work while minimizing the communication. Sophisticated multilevel heuristics have been developed for this problem, which have subsequently been adapted for a range of other combinatorial optimization problems. The underlying graph model has been enhanced in a variety of ways, most notably by the generalization to hypergraphs by Çatalyürek and Aykanat [6]. These models are surveyed by Hendrickson and Kolda [18].

A second recurring combinatorial kernel in parallel computing is graph coloring. Consider a graph in which vertices represent entities that require updating and edges indicate conflicts that preclude simultaneous updating of the adjacent vertices. One example of such a problem arises in mesh refinement [21]. In this setting, it is useful to identify a maximal subset of vertices, no two of which are adjacent to each other. All of the vertices in this subset can be updated in parallel, with no concern about conflicts. More generally, one could color all the vertices in such a way that no pair of adjacent vertices has the same color. The fewer colors required, the fewer the number of phases necessary to update all the entities. More detail on parallel computing, partitioning and coloring can be found in the other chapters in this book.

As has been discussed earlier, graph coloring also arises in optimization as a tool for exploiting sparsity structure in the computation of derivative matrices. Several variations of graph coloring, including distance-k vertex coloring (where a vertex must receive a color distinct from every neighbor within a distance k from it), star coloring, acyclic coloring, partial coloring (where only a subset of the vertices need to be colored), edge coloring, and hypergraph coloring, arise as graph models of derivative matrix computation problems. Efficient heuristic algorithms have been developed for these problems, and these algorithms often yield nearly optimal colorings for many classes of graphs and real-life instances of irregular graphs, since they are close to lower bounds which can be computed for the problem. Other chapters in this book discuss these problems in more detail, and a survey

is provided in [11].

As sketched in §1.2, the CSC community has deep roots in sparse direct methods. Graph algorithms are central to the exploitation of sparse structure in the factorization of various classes of matrices, and this continues to be an active area of research as detailed elsewhere in this book. Iterative methods for solving linear systems also rely upon combinatorial algorithms in various ways. Incomplete factorization preconditioners use graph algorithms to reorder the matrix. Algebraic multigrid approaches employ graph matchings and colorings in the construction of coarse representations. Support theory preconditioners make extensive use of graph embeddings in the construction and analysis of preconditioners. Other chapters in this book discuss some of these topics in greater depth.

A key step in many computational problems is the construction of a discrete representation of the geometry of the system being modeled. The mesh might represent the interior of a combustion chamber for a fluid simulation, the components of an automobile for virtual crash tests, or a tokamak for fusion modeling. The mesh must decompose the overall geometry into a set of well-shaped polyhedra (usually tetrahedra or octahedra). Geometric and combinatorial algorithms are central to this task. More detail can be found in subsequent chapters.

Combinatorial algorithms also play a prominent role in scientific applications that are not covered in this book. One of these is modern biology. String algorithms are central to the assembly of genomic data from shotgun sequencing methods, to the analysis of genomic data, and to searching gene and protein databases for close matches to a new sequence [16]. Dynamic programming on the entire sequences is forbiddingly expensive, and clever heuristic algorithms to identify promising substrings to perform exact dynamic programming on are the basis of search techniques such as BLAST, FASTA, PatternHunter, and others. Suffix trees and suffix arrays are critical for solving genome wide assembly problems as well as comparative studies in metagenomics [1]. Phylogenetics, the construction of evolutionary trees from sequence data, relies on multiple sequence alignments and rich combinatorics to reduce the search space of potential trees. Population genomics includes the reconstruction of genotypes from haplotype data (the number of variations in a specific allele in a pair of chromosomes, which can be 0, 1 or 2), and leads to problems in chordal graphs, as discussed in §1.4.4. Biological networks such as gene regulatory networks, protein interaction networks, and metabolic networks lead to a collection of graph problems. Random graph models for generating networks help in understanding the evolutionary relationships of these networks from different organisms. Discovering the modular organization (the decomposition of the network into topological clusters and the overlaps among the clusters) of these networks aids in understanding the biology of the processes of life and help in systematically identifying drug targets.

Another important CSC application area that is not covered in detail in this book is chemistry. The word *graph* as used in CSC was first coined by Cayley in his study of molecular structures, and graph concepts continue to be important in thinking about molecules and materials. The graphs of molecules are used today to characterize potential drug designs. Graph and geometric algorithms are also used in material science to describe complex chemical bond or proximity structure in granular and other complex materials [10, 29, 38].

1.4 Future Challenges

1.4.1 Trends in High Performance Architectures

For the past two decades computer performance has improved steadily due to faster clock speeds. But power consumption and heat generation are proportional to clock speeds, and

these factors are constraining further speedups. Instead, future performance advances will have to come from the increasing use of on-chip parallelism. Current generations of processors are multi-core, and the number of cores on a chip is expected to grow quickly as feature sizes continue to shrink exponentially. Effective use of current and future generations of on-chip multiprocessors will require algorithmic ideas that are closely informed by architectural constraints. The CSC community can play an important role in algorithmic design and implementation, and also in impacting the development of future architectures.

Several different designs of multicore processors are competing for market share right now, so it is difficult to foresee the details of future market leaders. But it is likely that future processors will contain multiple types of cores. These heterogeneous multiprocessors will require careful decomposition of the work amongst the different core types to optimize the utilization of each resource. The resulting scheduling and resource allocation problems will benefit from new load balancing models. The parallel computing community has largely avoided heterogeneity in the past, but will not be able to do so in the future.

Communication between cores on a chip is much more efficient than off-chip communication, and so algorithmic choices that are appropriate for our traditional cluster environments may no longer be optimal. For instance, shared-memory algorithms will be more viable on-chip. Shared variables can be used to store bounds in a branch-and-bound computation, or to facilitate partner selection in a matching algorithm. Lower-latency, more finely grained communication will support more fine-grained and asynchronous parallel algorithms.

Multi-core processors are likely to provide some multithreading capability to tolerate memory latencies. Work on the Cray XMT [26] has showcased the potential for latency tolerant computers to run graph algorithms with high efficiency while using very different notions of parallelism than traditional message passing codes. The development of combinatorial algorithms that perform well on future processors will be a rich area of research problems.

These multi-core processors will be the nodes of our future large parallel machines. For optimal performance this will likely require applications to exploit multiple layers of parallelism – message-passing between nodes but some other model within a node. The extreme scale machines of the future will have hundreds of thousands or millions of cores. To use such machines effectively we will need to improve our models and methods for load balancing, and devise algorithms that require fewer synchronizations. The cost of global collective communication operations on so many cores will penalize synchronous applications. In addition, slight deviations in core performance due to operating system jitter can induce imbalance that will hinder the performance of highly synchronous algorithms. The cost of global collective communication operations will also penalize synchronous applications. Thus, there will be a need for greater asynchrony in key scientific computing algorithms. With so many components, failures will be common occurrences, so fault tolerant and resilient algorithms will be important. In all of these areas, the CSC community will play an important role.

A critical issue with future leadership-class supercomputers will be power consumption [23, 22]. Current petascale machines consume multiple megawatts of power, and deliver fewer than 300 Mflop/sec/Watt. Clearly, dramatic improvements in efficiency are required for exascale computing. Current technology roadmaps indicate that the power consumption in large machines of the future will be dominated by data movement, not by computation. Thus, there will be a need for algorithms that use memory hierarchies efficiently to reduce the overall movement and power consumption. This area of algorithmic analysis is currently immature – the standard measure of algorithmic complexity is based upon computational operations, not data movement. But the theoretical computer science community has generated interesting models for out-of-core algorithms, or cache-oblivious algorithms which

provide new ways of thinking about memory hierarchies [9]. As it has always done, the CSC community can serve as a bridge between theoretical computer science and computational applications and bring some of these ideas to the service of scientific computing.

A very different architectural trend is evident in the emergence of cloud computing — highly distributed machines providing computing as a service. It is too early to assess the likely impact of clouds on scientific computing, but their growing importance in business and data analytics suggests that they will be an important feature of the future computing ecosystem. CSC researchers can contribute to improving the efficiency of clouds and also to studying the utility of such platforms for scientific applications.

1.4.2 Trends in Traditional Applications

Computational modeling and simulation are widely recognized as essential components of the scientific process. Computation allows for the study of phenomena that would be dangerous, expensive, or even impossible to study experimentally. It also allows for the exploration of detailed and complex phenomena that are beyond the reach of theory. The forefront of computational science continues to advance via new mathematical models, algorithms and computer architectures.

As they grow in fidelity and sophistication, state-of-the-art simulations increasingly involve complex geometries, multiple temporal and spatial scales, and multiple physical phenomena. These capabilities rely on unstructured and adaptive meshes, advanced linear and non-linear solvers, and complex coupling between models within the simulation. Design optimization and uncertainty quantification are playing a growing role above and beyond the traditional phenomenological modeling.

Combinatorial scientific computing techniques are essential enablers of all this sophistication. Unstructured mesh generation employs sophisticated geometric and graph theoretic algorithms. Numerical solvers rely upon graph algorithms for reordering and multigrid coarsening. Multiscale and multiphysics problems make use of advanced load balancing techniques that are built upon CSC technologies. Optimization and uncertainty quantification use graph algorithms to exploit structure in computing derivatives.

As discussed in a subsequent chapter derivative computation is of growing importance in computational science for design, optimization and uncertainty quantification. Automatic differentiation (AD) relies on combinatorial models to represent transformations of the computational graph representation of the computation of a mathematical function, through vertex, edge, or face elimination, to compute the derivative with the least amount of work or storage. In the adjoint or reverse mode of AD (where the derivatives are computed by applying the chain rule from the dependent to independent variables), which is often more efficient than the forward mode (computation from the independent to dependent variables), one challenge is to reduce the large storage required by placing checkpoints at chosen steps during the computation, and doing forward computations from them. Choosing the number and placement of these checkpoints to reduce the storage required while keeping the additional computations entailed by the checkpoints is an interesting combinatorial problem. When the functions involved recursively call functions within themselves, the problem becomes richer still.

All of these CSC underpinnings are essential to scientific computing and will become even more so as scientific computing continues to progress. There will undoubtedly be new nuances and variants of existing problems that will require new ideas within existing areas of CSC. But of even greater interest to this current paper are broader changes in the relationship between science and computation. We discuss several of these in the subsequent sections.

1.4.3 Emerging Applications

Data-Centric Scientific Computing

High-throughput experimental facilities are transforming many areas of science including biology, astronomy, earth science, and high energy physics [20]. All of these fields are struggling to make optimal use of a flood of new data. In biology, novel experimental methodologies are generating data that is qualitatively different from before, e.g. sequence data, gene and protein expression data from tissue micro-arrays, functional magnetic resonance imaging, multichannel flow cytometry and more. In other fields, the data are qualitatively similar to what came before, but orders of magnitude more voluminous. New models and abstractions are needed to address this growth in data quantity and diversity. And new computing approaches will be required to extract maximal scientific insight from the available data.

Extracting useful insight from this data is a preeminent scientific challenge. The internet community has shown how the combination of vision, clever algorithms, and diverse data can make data tremendously valuable. It is a certainty that large scientific data sets will enable answers to scientific questions that have not yet been posed.

Advanced techniques for exploring large data sets will require computational tools that are quite different from those that used in traditional scientific computing applications. Mature capabilities already exist for fundamental operations like indexing, range queries, and searching. More advanced analytics will be application-specific, and will involve various data abstractions. Graphs are increasingly popular as a tool for describing entities and their relationships, so graph algorithms will likely be an important component of data-centric scientific computing (see §1.4.3). New geometric algorithms will likely play a role in the analysis of complex spatial data in geoscience and other applications.

Computing in the Social Sciences

An emerging area of opportunity for CSC is the study of networks that arise in natural or human systems. The connectivity structure of links in the world wide web is central to search ranking algorithms. Communication links between people are reflected in blog postings, email traffic, instant messaging, and other social media. These large, complex networks are of keen interest in the social sciences, but techniques of extracting insight from them remain immature. Two recent books that discuss network science from this perspective have been written by Newman [31] and Easley and Kleinberg [8].

As happened recently with biology, the availability of large data sets is beginning to transform social sciences from qualitative to quantitative disciplines. Not long ago, the study of social interactions required direct observation, interviews and surveys. While richly informative for social modeling, public health, and other applications, these labor-intensive tasks limited the scale of the the communities being studied. But these days a well-connected social scientist can make use of the huge volume of data generated by on-line social networks, chat rooms, emails, instant messages, and more. In a recent study, Leskovec and Horvitz studied data from a month of Microsoft's instant-messaging traffic. The data encompassed 240 million people and 30 billion interactions [24]. Tellingly, Leskovec and Horvitz generated some summary statistics of the data, but did no real analysis. In another recent study, Michel et al. studied millions of books digitized by Google to analyze changes in language usage and various social trends [30]. As social science on this scale is in its infancy, no one knows what questions to ask, let alone how best to answer them.

Internet data is not only much larger than traditional social science data, but it is qualitatively different. Internet data is less highly curated, and largely uncontrolled. It comes

from natural, unconstrained human interaction which is attractive, but also greatly complicates its analysis. New graph algorithms and implementations will be needed to address emerging questions in social science. The structure of a social network is quite different from that of a finite element mesh, so existing parallel algorithms may not be appropriate. As with other areas of computational science, the future will emerge from the intersection of what is scientifically interesting and what is computationally feasible.

1.4.4 Biological Applications

Combinatorial techniques have played an essential role in the development of modern biology, with strings, networks, and trees playing prominent roles. Since these topics are not discussed in detail elsewhere in this book, in this subsection we provide a brief introduction to several important combinatorial problems in biology.

Population Genomics

A phylogenetic tree represents the evolutionary history of a set of biological sequences (e.g., proteins) from different organisms, individuals from the same species, etc. Given a matrix of sequences, with columns corresponding to characters, and rows to specific organisms or individuals, the state of a character for an individual is the value of the corresponding matrix element. The problem is to construct an unrooted tree with the given set of sequences at the leaves, generating new sequences at the internal nodes such that the subtree induced by every character-state pair is a subtree (and hence connected). Such a tree has the virtue of not displaying homoplasy, i.e., a state change in a character occurring at more than one place in the tree. The perfect phylogeny problem is to determine if a tree with these properties exists, and to construct such a tree if there is one.

When a fixed number k (with $k > 2$) states are permissible for a character, the perfect phylogeny problem can be posed as a chordal graph completion problem on a k-partite graph. Each character in the sequence matrix corresponds to a vertex part in the k-partite graph, and each character-state pair corresponds to a vertex in this sequence intersection graph. Each sequence is represented as a clique consisting of exactly one vertex from each vertex part. A legal edge in this graph is an edge that joins two vertices from different vertex parts, whereas an illegal edge joins two vertices in the same vertex part. Buneman [5] proved in 1974 that a perfect phylogeny tree exists if and only if the sequence intersection graph can be made a chordal graph by adding only legal edges.

Recently Gusfield (2009, 2010) has considered the perfect phylogeny problem with missing data, where the goal is to determine if missing entries can be completed in the sequence matrix so as to obtain a perfect phylogeny. The solution involves computing minimal separators in the sequence intersection graph, and employs a clique tree representation of a chordal completion.

Several genome-scale phylogeny problems remain to be formulated and solved in this area of population genetics with important consequences for understanding genetic similarities and differences in human populations (the International HapMap project). Since researchers in sparse matrix algorithms have made extensive use of chordal graph theory, clique tree representations of chordal graphs, and computing minimal separators in non-chordal graphs [4, 33], this is an example of an area of research in biology where the CSC community might be able to make significant contributions.

Computational Systems biology

Systems biology studies how large sets of genes and proteins within a cell interact with each other to accomplish the manifold functions of the cell. High-throughput large-scale experimental methodologies have been developed to characterize the interactions, and graphs are used to represent and study them. Examples are protein-protein interaction networks and gene regulatory networks, which are known for humans and model organisms such as yeast and the worm *C. elegans*. The graphs occurring in the biological context have modified power-law degree distributions, vertices have small average distances in the graph,

Topology based graph clustering methods have been used to understand the biological role or function of newly discovered genes and proteins from the roles of proteins or genes in their cluster. The frequency with which motifs, which are subgraphs with specific structures, usually of small size due to the computational complexity, occur in a large graph have been investigated through search algorithms. Network alignment is the problem of aligning several networks of a type in a group of related organisms to study how the networks could have evolved. Currently most of these networks are studied as static objects due to the nature of the data that is gathered. How the networks change due to changing conditions within the cell, or due to disease progression, requires the graphs to be dynamic, with edges or vertices appearing or being deleted, or changing their weights. Furthermore, kinetics of their interactions could be described by differential equation models. Most biological systems that are known to such detail are small-scale networks, but as these data become available, more sophisticated combinatorial and algebraic methods would be needed to study them.

Comparing biological networks across organisms leads to the study of graph alignment problems, where the goal is to identify how a subgraph in one network is present, with modifications, in another network. These lead to integer programming formulations, and often require the use of matching algorithms to solve them. Much remains to be done to develop effective algorithms for large network alignment problems.

Next Generation Sequencing

As of February 2011, the genomes of more than 2700 individuals have been sequenced, and tens of thousands of others are in the process of being sequenced. The discovery that our specific genetic makeup could lead to different outcomes to a drug necessitates the ability to sequence individual genomes at low costs, and the era of personalized medicine. New sequencing methodologies (next generation sequencing or NGS) that lower costs are being actively developed, and these rely on shotgun sequencing where the genomes are cut into small fragments, which are sequenced, and then assembled based on overlaps and known distances between the fragments. The assembly problems are more massive in the NGS methods, since the fragments are smaller, and the coverage of the genome by fragments is larger, to ensure that the sequences can be assembled correctly. Many new algorithms remain to be discovered and implemented efficiently to solve this challenging problem. Furthermore, new sequencing technologies such as single cell sequencing will soon be available, resulting in new algorithmic problems to be solved.

This is a one instance of a recurring theme in bioinformatics and computational biology. A large number of novel, high-throughput experimental methodologies are being developed that result in massive amounts of multidimensional, error-plagued data that need to be analyzed to discover features of interest to life scientists. An example of downstream analysis of high-dimensional flow cytometry data from leukemias analyzed with techniques from graph matching is in [2]. Often, measurements are taken from the samples at multiple times in varying conditions, and one needs to register features in a sample from one time

interval to the next. Researchers in CSC working on these problems need to understand the experimental context well, and need to develop algorithms that can effectively deal with the errors in the data.

1.5 Conclusions

From a diverse set of antecedents, combinatorial scientific computing has emerged as a recognized and essential component of a broad range of scientific disciplines. The range of problems and applications is growing rapidly, and we anticipate further growth in the future due to several trends in science and technology.

First, as parallelism becomes a ubiquitous aspect of all computing platforms, the strong role that CSC has played in facilitating parallelism will impact many more computations. And the unrelenting increase in complexity of leadership class parallel machines will also demand new combinatorial models and techniques for load balancing and scheduling. Second, as traditional scientific computing applications grow in sophistication, they are increasingly embracing complex geometries, adaptivity and multiscale methods. All of these changes require sophisticated combinatorial algorithms to manage mesh generation, and to get optimal performance out of memory hierarchies. Third, emerging computational applications are rich in combinatorial opportunities. Among these applications are data-centric science, computational social science, and new challenges in biology.

For all these reasons, we believe that the breadth and depth of activity in combinatorial scientific computing will continue to grow briskly.

Acknowledgment

Bruce Hendrickson's work was performed at Sandia National Labs, a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. DOE undercontract number DE-AC-94AL85000. Alex Pothen's work was supported by the grant DE-FC02-08ER25864 for the CSCAPES Institute, funded under the Scientific Discovery through Advanced Computing program of the U.S. Department of Energy, and by the U.S. National Science Foundation through grant CCF-0830645.

References

References

[1] Srinivas Aluru. *Handbook of Computational Molecular Biology.* Chapman and Hall, 2005.

[2] Ariful Azad, Johannes Langguth, Youhan Fang, Alan Qi, and Alex Pothen. Identifying rare cell populations in comparative flow cytometry. In *Proceedings of the Workshop on Algorithms in Bioinformatics, Lecture Notes in Bioinformatics*, volume 6293, pages 162–175. Springer Verlag, 2010.

[3] F. L. Bauer. Computational graphs and rounding error. *SIAM Journal on Numerical Analysis*, 11:87–96, 1974.

[4] Jean R. S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph Theory and Sparse Matrix Computation, IMA Volumes in Mathematics and its Applications*, volume 56, pages 1–30. Springer Verlag, 1993.

[5] Peter Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9:205–212, 1974.

[6] Ü. Çatalyürek and C. Aykanat. Decomposing irregularly sparse matrices for parallel matrix-vector multiplication. In *Lecture Notes in Computer Science 1117*, pages 75–86. Springer-Verlag, 1996. Proc. Irregular'96.

[7] T. F. Coleman and J. J. More. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20:187–209, 1983.

[8] David Easely and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010.

[9] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache–oblivious algorithms. In *Proc. IEEE Symp. Foundations of Computer Science*. IEEE, 1999.

[10] R. Garcia-Domenech, J. Gálvez, J. de Julián-Ortiz, and L. Pogliani. Some new trends in chemical graph theory. *J. Molecular Graphics and Modelling*, 19:60–69, February 2001.

[11] Assefaw Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.

[12] Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.

[13] Alan George, John R. Gilbert, and Joseph W. H. Liu, editors. *Graph Theory and Sparse Matrix Computation*. Springer-Verlag, 1993. IMA Volumes in Applied Mathematics, Volume 56.

[14] Alan George and Joseph W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[15] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, second edition, 2008.

[16] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[17] A.H. Hartmann and H. Rieger. *Optimization Algorithms in Physics*. Wiley, 2003.

[18] B. Hendrickson and T. Kolda. Graph partitioning models for parallel computing. *Parallel Comput.*, 26:1519–1534, 2000.

[19] Bruce A. Hendrickson and Alex Pothen. Combinatorial Scientifc Computing: The enabling power of discrete algorithms in computational science. In *Lecture Notes in Computer Science*, volume 4395, pages 260–280. Springer Verlag, 2007.

[20] T. Hey, S. Tansley, and K. Tolle, editors. *The fourth paradigm: Data–Intensive scientific discovery*. Microsoft Research, 2009.

[21] M. T. Jones and P. E. Plassmann. A parallel graph coloring heuristic. *SIAM J. Sci. Comput.*, 14(3):654–669, 1993.

[22] Peter Kogge. The tops in flops. *IEEE Spectrum*, pages 50–54, February 2011.

[23] Peter Kogge (editor). Exascale computing study: Technology challenges in achieving exascale systems. Technical Report TR-2008-13, University of Notre Dame, CSE Dept., 2008.

[24] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proc. World Wide Web Conference*, Beijing, China, April 2008.

[25] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16:346–358, 1979.

[26] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. Berry. Challenges in parallel graph processing. *Parallel Processing Letters*, 17(1):5–20, 2007.

[27] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3:255–269, 1959.

[28] H. M. Markowitz. Biosketch. In Tore Fralingsmyr, editor, Lex Prix Nobel. Nobel Foundation, Stockholm, 199.

[29] S. Martin, D. Roe, and J.-L. Faulon. Predicting protein–protein interactions using signature products. *Bioinformatics*, 21(2):218–226, August 2004.

[30] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, 2011.

[31] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[32] Seymour V. Parter. The use of linear graphs in Gaussian elimination. *SIAM Review*, 3:119–130, 1961.

[33] Barry W. Peyton, Alex Pothen, and Xiaoqin Yuan. Partitioning a chordal graph into transitive subgraphs for parallel sparse triangular solution. *Linear Algebra and its Applications*, 192:329–354, 1993.

[34] John K. Reid. *Large Sparse Sets of Linear Equations*. Academic Press, London, 1971.

[35] Donald J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, New York, 1972.

[36] Donald J. Rose and R. A. Willoughby. *Sparse matrices and their applications*. Plenum Press, New York, 1972. Proceedings of a Symposium at the IBM Research Center, NY, Sep. 9-10, 1971.

[37] H. D. Simon. Partitioning of unstructured problems for parallel processing. In *Proc. Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications*. Pergammon Press, 1991.

[38] M. F. Thorpe, M. Lei, A. J. Rader, D. J. Jacobs, and L. A. Kuhn. Protein flexibility and dynamics using constraint theory. *Chem. Rev.*, 108(3):1127–1169, February 2008.

[39] R. A. Willoughby. Proceedings of the symposium on sparse matrices and their applications. Technical Report Report RAI (No. 11707), IBM, Yorktown Heights, NY, 1969.