

Position Paper:

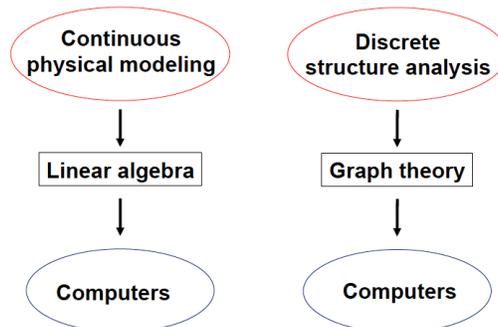
Algorithmic Primitives for Exascale Computational Discrete Mathematics

John R. Gilbert
UC Santa Barbara
gilbert@cs.ucsb.edu
May 7, 2013

Problem Statement

Computation with discrete, combinatorial structures has become essential to the DOE mission in many areas of data analysis, complex systems modeling, and scientific computation, most of which now require extreme-scale computation in both the numerical and discrete domains [Ale2011, Bro2008, Keg2008]. However, the field of high-performance combinatorial computing is in its infancy, and current algorithms for combinatorial computation do not scale well on high-performance parallel computers.

In the well-developed field of numerical computing, programmers possess standard algorithmic primitives, high-performance software libraries, powerful rapid-prototyping tools, and a deep understanding of effective mappings of problems to high-performance computer architectures. A key challenge of the transition to exascale is to replicate these achievements for computational discrete mathematics.



Linear algebra has played a crucial (one might almost say magical) role as “middleware” between continuous physical models to be computed and the simple arithmetic operations implemented by actual computer hardware. From a mathematical point of view, graph theory occupies a similar position as middleware between models of discrete structures and actual computers. However, for the most part computational graph theory still lacks the above-mentioned standard primitives, software libraries, rapid-prototyping tools, and connections to machine architecture.

We recommend research leading to the development of mathematical tools and algorithmic primitives for exascale graph computation that play roles analogous to those of the BLAS (standard primitives) and of LAPACK and ScaLAPACK (high-performance libraries for key algorithms).

State of the Art

What would it mean to complete the analogy between linear algebra as middleware for continuous physical modeling, and graph theory as middleware for analysis of discrete structures? This asks us to reflect on the steps that evolved numerical linear algebra from mathematical linear algebra (“numerical” here means “computational”), and to similarly develop a computational graph theory.

Among the questions we must ask are: What is a good set of primitive representations and operations for computational graph theory? How can such primitives be implemented so that both functionality and performance are portable across a range of modern computers? How should such primitives be presented to the user as APIs, programming patterns, languages, interactive environments? How should we evaluate these tools from the points of view of performance, expressivity, and productivity? Here we propose specific directions to explore for the answers to these questions.

A sparse graph can be represented by a sparse array of edges (indexed by head and tail vertices), which in turn can be considered as a sparse matrix data structure for the graph's adjacency matrix. Linear algebraic operations on this matrix correspond to some kinds of graph operations; for example, multiplying a sparse matrix by a dense vector systematically explores all the neighbors of each vertex. We have reported [KepGil2011] the results of a study showing that sparse linear algebraic operations on semirings are a powerful set of primitives that can be used to implement a surprisingly wide variety of graph computations. The key is to allow arbitrary user-defined semirings, including the familiar (+,*) ring, but also (min,+), (and,or), and others.

The most powerful algebraic primitives include: SpGEMM (sparse matrix-matrix multiplication); SpAdd (sparse matrix-matrix addition); SpMV (sparse matrix-dense vector multiplication); SpRef (selection of a subarray of a sparse matrix); SpAsgn (assignment to a subarray of a sparse matrix) and SpCat (concatenation of sparse arrays), all with the possibility of substituting different objects for numerical array elements and different operations for scalar addition and multiplication. SpGEMM has not been studied extensively by the numerical linear algebra community, although sequential SpGEMM algorithms appear in Matlab [GilMolSch1992, Dav2007] and as far back as Gustavson [Gus1978].

We have developed high-performance scalable parallel implementations of SpGEMM and related primitives [BulGil2008, BulGil2012], as part of our Combinatorial BLAS library [BulGil2011]. We have used the Combinatorial BLAS as the computational kernel of our high-level Knowledge Discovery Toolbox [Bul2013] for analysis of attributed semantic graphs. This latter work uses just-in-time specialization to achieve high performance on graph computation with arbitrary semirings specified in a high-level language.

Recommendations

- Develop mathematical and algorithmic primitives for sparse linear algebraic computation on semirings, including representations of graphs and hypergraphs in terms of edge-vertex incidence matrices, allowing computation in a mixture of different semirings.
- Establish a robust research effort in mathematically informed co-design of graph algorithms and architectures, with a view to enabling high-performance low-power execution of crucial primitive operations.
- Extend the existing set of algebraic graph primitives into a common framework that coherently integrates algebraic, visitor, and map-reduce patterns in a concise library of primitives that will interoperate cleanly for edge-based, vertex-based, and traversal-based algorithms.
- Combine combinatorial and numerical functionality in an integrated set of algebraic primitives, enabling the seamless combination of graph computation and numerical computation in such techniques as spectral clustering, belief propagation, hidden Markov methods, support vector machines, latent Dirichlet analysis, etc.
- Bring together the research community working on graph algorithms expressed as linear algebra to define a common API for the "BLAS 1/2/3"-level primitives that can be used for research and development of higher-level graph algorithms.

References

- [Ale2011] F. Alexander et al. A multifaceted mathematical approach for complex systems. DOE Office of Science, 2011.
- [Bro2008] D. L. Brown et al. Applied mathematics at the U. S. Department of Energy: Past, present, and a view to the future. DOE Office of Science, 2008.
- [BulGil2008] A. Buluç and J. R. Gilbert. On the representation and multiplication of hypersparse matrices. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008)*, pages 1-11, April 2008.
- [BulGil2011] A. Buluç and J. R. Gilbert. The Combinatorial BLAS: design, implementation, and applications. *Intl. J. High Performance Computing Applications* 25(4): 496-509, 2011.
- [BulGil2012] A. Buluç and J. R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: implementation and experiments. *SIAM Journal of Scientific Computing* 34(4):170-191, 2012.
- [Bul2013] A. Buluç, E. Duriakova, A. Fox, J. R. Gilbert, S. Kamil, A. Lugowski, L. Oliker, and S. Williams. High-productivity and high-performance analysis of filtered semantic graphs. *27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, to appear May 2013.
- [Dav2007] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006.
- [GilMolSch1992] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in Matlab: Design and implementation. *SIAM J. Matrix Analysis and Applications*, 13: 333-356, 1992.
- [Gus1978] F. G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Transactions on Mathematical Software*, 4(3):250-269, 1978.
- [Keg2008] P. Kegelmeyer et al. Mathematics for analysis of petascale data. DOE Office of Science, 2008.
- [KepGil2011] J. Kepner and J. Gilbert, eds. *Graph Algorithms in the Language of Linear Algebra*. SIAM, 2011.
- [Kep2013] J. Kepner et al. "Standards for graph algorithm primitives." White paper, 2013.