Progress Report on the Early-Career Research Project: Energy-Efficient Parallel Graph and Data Mining Algorithms

Principal Investigator: Dr. Aydın Buluç
Lawrence Berkeley National Laboratory
One Cyclotron Road, MS 50A-1148
Berkeley, CA 94720
abuluc@lbl.gov

Reporting Period: 7/13-9/14

Data are fundamental sources of insight for experimental and computational sciences. The Department of Energy acknowledges the challenges posed by fast-growing scientific data sets and more complex data. The graph abstraction provides a natural way to represent relationships among complex fast- growing scientific data sets. On future exascale systems, power consumption is of primary concern yet existing graph algorithms consume too much energy per useful operation due to their high communication costs, lack of locality, and inability to exploit hierarchy. This project explores methods to increase the energy efficiency of parallel graph algorithms and data mining tasks. A new family of algorithms will be developed to drastically reduce the energy footprint and running time of the graph and sparse matrix computations that form the basis of various data mining techniques. This project will also exploit the well-known duality between graph and sparse matrices to develop communication- avoiding graph algorithms that consume significantly less power. This project is relevant to DOE mission-critical science including bioinformatics and genomics with particular emphasis on plant genomics that can result in better biofuels through efficient genetic mapping, climate science where recent graph- based methods show increased accuracy in hurricane predictions, and combustion science where graph search techniques are used to analyze extreme-scale simulation data.

1 Summary

The early-career research project has made significant progress in its first year. Among the three significant research thrusts that were originally proposed, we made advances in two of them already. They are:

- Higher-level graph kernels that avoid communication.
- Exploitation of filter-induced subgraphs.

The progress for graph kernels that avoid communication is detailed in Section 3.1, while the progress for filtered graph processing is detailed in Section 3.2. In addition, we are developing new graph-theoretical parallel algorithms for the solution of scientific problems. Our collaboration with MANTISSA (ASCR - Applied Math) and DEGAS (ASCR - Computer Science) projects created a novel fine grained parallel algorithm for the construction and travel of de Bruijn graphs for de novo genome assembly, which is explained in detail in Section 3.3. DOE relevance of each research accomplishment is mentioned within the corresponding section.

Our next year plans to further advance both of these research thrusts, as well as our plans to move towards energy measurements to quantify benefits, are explained in Section 4. Overall, our project so far resulted in six peer-reviewed publications [19, 3, 2, 12, 14, 1] and one position paper [15]. The report concludes with lists of software artifacts, presentations, community service, and references.

2 Personnel

In terms of personnel, we successfully filled our postdoctoral fellow position(s). Ariful Azad from Purdue University joined in February and is working 100% on this project. Harsha Simhadri from CMU joined at the beginning of the fiscal year and working about 40% on this project while spending the rest of his time in another DOE Office of Science (ASCR - Computer Science) funded project, DEGAS. In contrast to the original proposal plan, we opted for hiring 1.5 postdocs at the expense of lowering PI's percentage to 50-55%. This allows for having both an algorithms and a partial systems researcher in our team, which is instrumental for successful execution of our research plan. We also temporarily hired an undergraduate summer intern, Eric Lee from UC Berkeley, to help with the implementation of sparse all-to-all [1] operations that are performance critical to the implementation of communication-avoiding sparse matrix-matrix multiplication algorithms.

[1] in the sense of data sparsity, not in the sense of sparse collective neighborhoods

3 Progress and Accomplishments

3.1 Reduced-communication graph algorithms

We designed new parallel algorithms that reduce the communication costs of graph computations. Most graph algorithms have very low computational intensity, hence the execution time is bounded by the communication requirements of the algorithm. In addition to improving the running time drastically, reducing communication will also help improve the energy consumption of graph algorithms in exascale. This is because the power required to transmit data also depends on the length of the wire: the farther the data, the higher the power usage.

All-pairs shortest paths (APSP) is a computationally intensive graph algorithm. For dense graphs, we developed a distributed memory algorithm that is communication optimal. This work-efficient algorithm is based on the recursive version of Kleene's algorithm (as opposed to the more popular Floyd-Warshall algorithm), and uses the ideas from 2.5D algorithms that replicate the input data c times to reduce communication by a factor of $\sqrt{c}$. The algorithm enables much better strong scaling, and we were able to solve a 65K vertex dense APSP problem in about two minutes [19].

APSP itself has important applications in data analysis that are becoming increasingly important to DOE. One of the most popular non-linear dimensionality reduction (i.e., manifold learning) algorithms, IsoMap [20], requires geodesic distances between all pairs of vertices. Another important application is the metric nearness problem [7], which finds the smallest metric approximation to a set of non-metric distances. The decrease-only version of this problem, where only decreasing changes to the input distances are allowed, turns out to be equivalent to the APSP problem.

Sparse matrix-matrix multiplication (SpGEMM) enables efficient parallelization of various graph algorithms. It is the workhorse of a scalable distributed-memory implementation of betweenness centrality, an algorithm that finds influential entities in networks. Existing parallel algorithms for SpGEMM spend the majority of their time in inter-node communication on large concurrencies. We investigated communication-optimal algorithms for SpGEMM. Our new theoretical paper [2] proves new communication lower bounds, presents two new communication-optimal algorithms, and provides a unified communication analysis of existing and new algorithms.

Breadth-first search (BFS) is a fundamental building block of many higher-level graph algorithms. We designed a new distributed memory algorithm that incorporates bottom-up search [3]. The new direction-optimizing

distributed-memory algorithm reduces communication volume by a factor of 8x in theory, for graphs with power-law degree distribution. Our large scale weak-scaling experiments showed that this reduction in theory translates into a reduction in the running time where the new algorithm beats the state-of-the-art classical 2D algorithm by up to 7.9x. The new BFS algorithm has higher latency, hence its strong scaling behavior is not as good as its weak scaling. Even in this context, however, the direction-optimizing algorithm is *energy efficient* because much fewer processors are needed to process a given graph within a fixed time constraint. The performance results are shown in Figure 1.

With many other leading experts in the field, we co-authored a position paper on the development of BLAS-like primitives for graph operations [15], which will include the reduced-communication algorithms described in this section. This effort has implications for exascale as well: graph algorithms will need to be redesigned to address the joint issues of complexity, resilience, and scalability. The translation of every individual graph algorithm to exascale platforms will be prohibitively difficult due to the complexity of hardware architectures and the diversity of graph operations. Primitives allow algorithm designers to think on a higher level of abstraction, and reduce duplication of implementation efforts. Our conjecture is that with the large body of experience with sparse linear algebra in extreme scale computing, the basic graph algorithm primitives we have proposed will be an effective way to manage the exascale challenge. In particular, if we can manage the exascale issues (scalability and reliability) inside the primitives, then graph algorithm developers can explore algorithms for these machines using the same approaches used on modest sized parallel systems. This would help assure that the graph-based software needed for these machines exists as exascale computers emerge.

3.2 Filtered and masked operations of graphs

Execution of complex analytic queries on massive semantic graphs is a challenging problem in big-data analytics that requires high-performance parallel computing. In a semantic graph, vertices and edges carry attributes of various types and the analytic queries typically depend on the values of these attributes. Thus, the computation must view the graph through a filter that passes only those individual vertices and edges of interest. In many applications it is prohibitively expensive to run a filter across an entire graph data corpus, and produce ("materialize") a new filtered graph as a temporary object for analysis. In addition to the obvious storage prob-

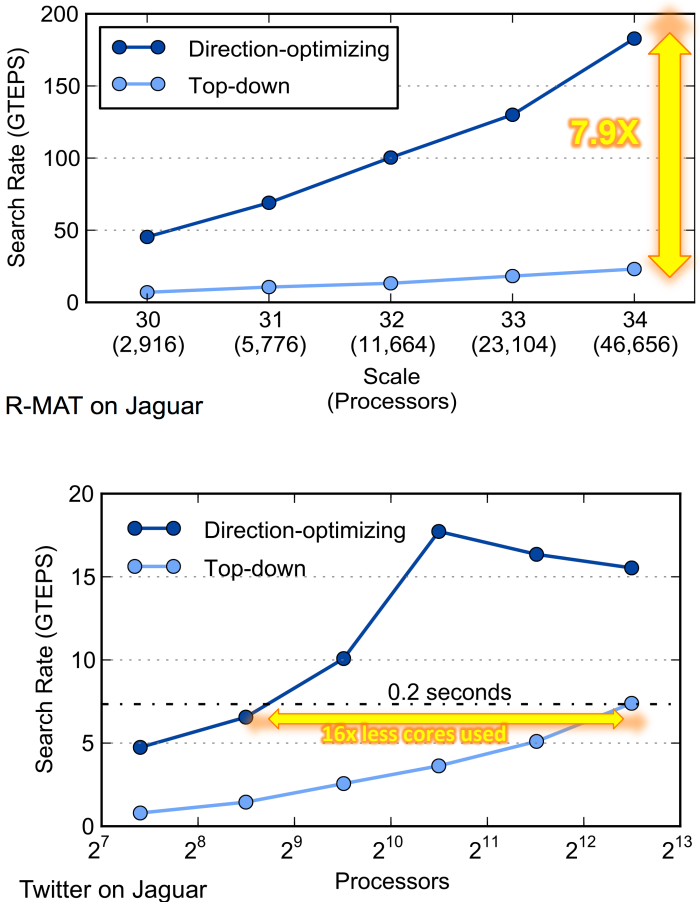

Figure 1: **Top:** Weak scaling of BFS traversal on the ORNL/Jaguar (Titan minus the GPUs), showing the 7.9X performance improvement at scale. **Bottom:** For a fixed-sized real input (twitter), direction-optimizing algorithm finishes at the same time using 1/16th of processors (and energy).

lems with materialization, the time spent during materialization is typically not amortized by many graph queries because the user modifies the query (or just the filter) during interactive data analysis. The alternative is to filter edges and vertices "on the fly" during execution of the complex graph algorithm.

Our manuscript in press [14] provides a solution to efficient on-the-fly processing of filtered semantic graphs and shows how to efficiently handle

nesting of filters without re-processing the whole graph. In other words, if a second filter is applied to an already filtered graph, the algorithm only processed the edges that passes the first filter. Furthermore, our approach is generalized to handle arbitrary algorithms, and supports efficient filtering on run-time definable edges/vertices which is more general than precompiled edge types. We demonstrated the generality of our approach by specializing two different graph algorithms: breadth-first search and maximal independent set.

The same work also provides a new Roofline performance model for high-performance graph traversals, suitable for evaluating the performance of filtered semantic graph operations. The experimental results show excellent performance scaling to graphs with tens of millions of vertices and hundreds of millions of edges.

3.3 Graph analysis to address scientific challenges

Genome assembly: De novo whole genome assembly reconstructs genomic sequence from short, overlapping, and potentially erroneous fragments called reads. In a recent paper to be presented at the Supercomputing conference this November [12], we studied optimized parallelization of the most time-consuming phases of Meraculous, a state-of-the-art production assembler. We developed a new parallel algorithm for k-mer (genome segments of length k) analysis, characterized by intensive communication and I/O requirements, and successfully reduce the memory requirements by 6.93×. We also focused on the efficient parallelization of de Bruijn graph construction and traversal, which necessitates a distributed hash table and is a key component of most de novo assemblers. We provided a novel algorithm that leverages one-sided communication capabilities to facilitate the requisite fine-grained parallelism and avoidance of data hazards. We also analytically proved the scalability of the algorithms. Overall our results show unprecedented performance and efficient scaling on up to 15,360 cores of a Cray XC30, using real human genome data as well as the recently-sequenced massive DNA of the hexaploid wheat genome — with performance improvement from *days* for the original serial code to *seconds* on our highly-optimized parallel version.

Our main performance result is summarized in Figure 2, showing unprecedented scalability to several thousand cores at which point we can go from raw sequencing data to contigs in less than three minutes for a human genome. One of the contributions of this work is a new parallel algorithm for k-mer analysis, which successfully reduced the memory requirements using Bloom filters and probabilistic counting techniques, and attained remark-

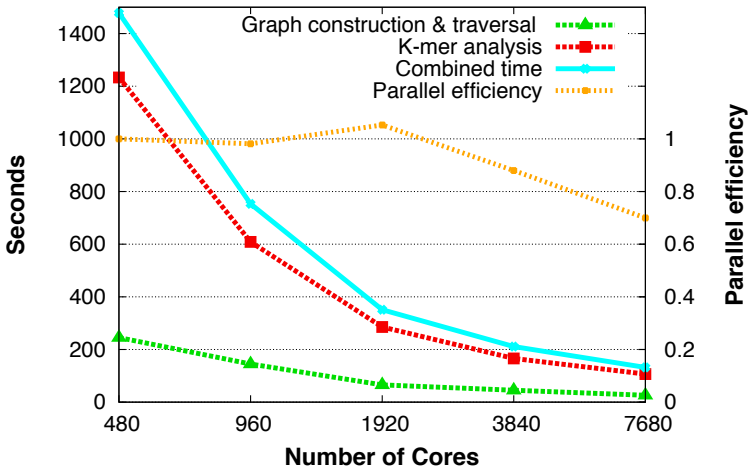

Figure 2: Performance and strong scaling of the new de Bruijn graph construction & traversal and k-mer analysis steps on NERSC/Edison for the human genome. The top three timing curves are with respect to the first y-axis (left) whereas the parallel efficiency curve is with respect to the second y-axis (right). The x-axis uses a log scale.

able scaling for this I/O- and communication-intensive computation. We are planning to apply similar Bloom filter techniques to *filtered and masked graph operations* thrust of this project.

Graph matching algorithms: Matching problems are fundamental in graph theory where they come in many flavors. Maximum cardinality matching in bipartite graphs as well as weighted perfect matching in general and bipartite graphs has fundamental applications in the parallel solution of systems of sparse linear equations. Maximum cardinality matching in bipartite graphs is used to permute a matrix to its block triangular form [16], which is crucial for fast solution of sparse matrices arising in circuit simulation [10].

We are finishing work on a new algorithm and its shared-memory parallelization for computing maximum cardinality matching in bipartite graphs. Our algorithm finds augmenting paths via parallel breadth-first search (BFS) from multiple source vertices, hence exposing more parallelism than single source algorithms. Our novel tree-grafting method that eliminates most of the redundant edge traversals resulting from increased parallelism. We employ the recently proposed direction-optimized BFS algorithm as a subroutine to speed up augmenting path discoveries. Our algorithm generally

compares favorably in terms of the average augmenting path length, the number of iterations, and the number of edges traversed. We provide a proof of correctness of our algorithm. Our NUMA-aware implementation is scalable to 80 threads. For important classes of graphs, such as citation networks, road networks, and web graphs, our algorithm runs 4-15 times faster than the fastest competitor. We will be submitting our results within the next few months.

Sparse matrix-multiple vector multiplication: In collaboration with the FASTMath Institute, and the NUCLEI and the SUPER projects of SciDAC, we utilized the Compressed Sparse Blocks (CSB) format improve the performance of the sparse matrix-multiple vector multiplication (SpMM) kernel, which provides more opportunities for performance optimization than the single vector (SpMV) case. The integration of CSB-based SpMM and its transposed variant into a large- scale distributed-memory block eigensolver resulted in 40% runtime improvement and the results are published in IPDPS this year [1]. In general, block eigensolvers provide an attractive alternative to Lanczos-based eigensolvers as they are sufficiently compute-intensive to relegate DRAM bandwidth to a secondary bottleneck.

While we showcased our SpMM results on MFDn, the implications of improved SpMM performance are broader. For example, spectral clustering, one of the most promising data clustering techniques, uses the eigenvectors associated with the smallest eigenvalues of the Laplacian of the data similarity matrix, to partition the graph into various clusters [21]. For a k-way clustering problem, k eigenvectors are needed, where typically $10 \leq k \leq 100$, an ideal range for block eigensolvers. Block methods that rely on SpMM is also used to solve large-scale sparse singular value problems [4], with most popular methods being the subspace iteration and block Lanzcos. Singular value decomposition can be used to perform dimensionality reduction tasks such as latent semantic indexing [11].

4 Plans for the next fiscal year

Graph matching algorithms: For the parallel solution of unsymmetric systems of linear equations (i.e. LU decomposition), *static pivoting* [13] is often used to avoid dynamic pivoting that becomes prohibitively expensive in distributed memory. In this scheme, the row permutations are chosen before the factorization, in a preprocessing step. Static pivoting is accomplished by finding the *maximum weighted matching* of the bipartite graph $G = (R, C, E)$ where R corresponds to the rows, C corresponds to the columns,

and E corresponds to the nonzeros of the sparse matrix. In general, this matching has to be "perfect", or "maximum cardinality" (since an algorithm that finds a maximum cardinality matching will find the perfect matching if it exists) [17]. Another important advantage of static pivoting, even in serial, is that it allows pre-application of a separate fill-reducing ordering on the permuted matrix. The challenge for distributed-memory environments is to find scalable parallel algorithms for maximum weight perfect matching problem, which we will be working on during the next fiscal year.

Sparse matrix-matrix multiplication: We will build on our theoretical communication-avoiding SpGEMM work [2] and develop high-performance implementations. We will leverage the sparse all-to-all primitive jointly implemented by Eric Lee. In a separate thrust, new multicore algorithms will also be developed in order to fully exploit the increasing intra-node parallelism. In addition to demonstrating performance in graph computations such as betweenness centrality and Markov clustering, we plan to collaborate with different groups in LBNL that need a high performance implementation of SpGEMM and cater to their needs as well:

- PDSLin (Parallel Domain decomposition Schur complement based Linear Solver) library has SpGEMM as one of its performance bottlenecks [23]. We will collaborate with Sherry Li.
- For linear-scaling electronic structure calculations, and specifically for solving the self consistent field (SCF) equations that arise in Kohn-Sham or Hartree-Fock theory, SpGEMM becomes a critical component [5, 6]. NWChem currently does not support sparse methods but has plans to do so in the future. We will collaborate with Bert de Jong, Chao Yang, and Lin Lin.
- The setup phase of algebraic multigrid (AMG) methods perform the so-called Galerkin product, RAP where R and P are sparse restriction matrices and A is the sparse matrix representing the fine grid. This operation becomes a bottleneck in high concurencies. We will collaborate with Sam Williams and Mark Adams.

Filtered and masked operations for graphs: Many graph algorithms can be significantly accelerated if the known structure of the output is exploited. This is true for triangle counting, adaptive PageRank, and even BFS based graph traversals. Formally, this masked operation can be described as *Multiply(Operand1, Operand2, Mask)* where *Mask* is of the same dimension as the output. We will be working on developing a theory of masked sparse linear algebra operations that can avoid communication.

Energy-efficiency measurements: We will start building infrastructure for performing fine-grained power consumption measurements of our algorithms. We plan to wait until NERSC/Cori receives its test subset in Spring 2015, which will be composed of Intel Haswell processors whose cores are capable of independently scaling their frequency and voltages.

Software artifacts

Combinatorial BLAS: The PI, Aydın Buluç, has continued to lead the development of the Combinatorial BLAS (CombBLAS) library [8, 9] along both the functionality and the performance axes. Ariful Azad has familiarized himself with the library and he is now developing distributed memory graph matching algorithms using custom filters and semirings. We have integrated our novel distributed memory direction optimizing BFS algorithm [3] to the version 1.4 of CombBLAS, which makes the library the fastest available code for the Graph500 benchmark on Cray supercomputers.

Our work has been acknowledged and our performance claims has been verified by independent researchers. In a recently published independent study by Intel Research [18], CombBLAS was the fastest among all tested graph processing frameworks on 3 out of 4 benchmarks. This work also demonstrates that the linear algebra abstraction for graph algorithms, which is a main focus of our project, enables high performance and comes close to hand-optimized performance for important data mining algorithms such as PageRank and Collaborative filtering.

A similar benchmarking effort [22] concludes that "the implementation built on the CombBLAS library results in the best performance on multi-CPU systems with a reasonable amount of effort. Advantages of this approach are that it provides a number of data structures necessary for many graph computations and hides from the developer the complexity of the management of multithreading and distributing these data structures throughout a system". It also acknowledges that "the linear algebra implemented by the CombBLAS library may represent the correct level of abstraction to separate the concerns of implementing graph algorithms (on top of the library) from the concerns of the hard- ware and implementing memory-efficient abstractions and data structures (within the library)".

Presentations

1. College of Engineering Distinguished Seminar, Montana State University, 2013
2. *Three goals in parallel graph computations: High performance, high productivity, and reduced communication.* Workshop on Parallel and Distributed Algorithms for Inference and Optimization, Simons Institute, Berkeley, CA, 2013.
3. *High-productivity and high- performance analysis of filtered semantic graphs.* SIAM Conference on Parallel Processing for Scientific Computing, Portland, OR, 2014
4. *Communication-avoiding linear-algebraic primitives for graph analytics.* Graph Algorithms Building Blocks (GABB), IPDPS Workshops, Phoenix, AZ, 2014.
5. *Reducing communication in parallel graph computations.* Workshop on Algorithms for Modern Massive Data Sets (MMDS), Berkeley, CA, 2014.
6. *The Graph BLAS effort and its implications for Exascale.* SIAM Workshop on Exascale Applied Mathematics Challenges and Opportunities (EX14), Chicago, IL, 2014.

Community Service (PI only)

- *Associate Editor*: ACM Transactions on Parallel Computing
- *Minisymposia Co-organizer*: "Graph Analysis for Scientific Discovery" at SIAM Conf. on Parallel Processing for Scientific Computing, 2014.
- *Program Committee*: IEEE International Parallel & Dist. Processing Symp. (IPDPS), 2014
- *Program Committee*: ACM/IEEE Conference on High Performance Computing (SC), 2014
- *Program Committee*: WWW Workshop on Big Graph Mining, 2014
- *Steering Committee*: Graph Algorithms Building Blocks (GaBB), IPDPS workshop.

References

[1] H. M. Aktulga, A. Buluç, S. Williams, and C. Yang. Optimizing sparse matrix-multiple vectors multiplication for nuclear configuration interaction calculations. In *Proceedings of the IPDPS*. IEEE Computer Society, 2014.

[2] G. Ballard, A. Buluç, J. Demmel, L. Grigori, B. Lipshitz, O. Schwartz, and S. Toledo. Communication optimal parallel multiplication of sparse random matrices. In *SPAA 2013: The 25th ACM Symposium on Parallelism in Algorithms and Architectures*, Montreal, Canada, 2013.

[3] S. Beamer, A. Buluç, K. Asanović, and D. Patterson. Distributed memory breadth-first search revisited: Enabling bottom-up search. In *Proceedings of the IPDPSW*. IEEE Computer Society, 2013.

[4] M. W. Berry. Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.

[5] N. Bock, M. Challacombe, and L. V. Kalé. Solvers for $\mathcal{O}(n)$ electronic structure in the strong scaling limit. *CoRR*, abs/1403.7458, 2014.

[6] U. Borštnik, J. VandeVondele, V. Weber, and J. Hutter. Sparse matrix multiplication: The distributed block-compressed sparse row library. *Parallel Computing*, 40(5):47–58, 2014.

[7] J. Brickell, I. S. Dhillon, S. Sra, and J. A. Tropp. The metric nearness problem. *SIAM Journal on Matrix Analysis and Applications*, 30(1):375–396, 2008.

[8] A. Buluç and J. R. Gilbert. The Combinatorial BLAS: Design, implementation, and applications. *International Journal of High Performance Computing Applications (IJHPCA)*, 25(4):496–509, 2011.

[9] Combinatorial BLAS, release v1.4.0, January 2014.

[10] T. A. Davis and E. Palamadai Natarajan. Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Transactions on Mathematical Software (TOMS)*, 37(3):36, 2010.

[11] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[12] E. Georganas, A. Buluç, J. Chapman, L. Oliker, D. Rokhsar, and K. Yelick. Parallel de bruijn graph construction and traversal for de novo genome assembly. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'14)*, 2014.

[13] X. S. Li and J. W. Demmel. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Transactions on Mathematical Software (TOMS)*, 29(2):110–140, 2003.

[14] A. Lugowski, S. Kamil, A. Buluç, S. Williams, E. Duriakova, L. Oliker, A. Fox, and J. Gilbert. Parallel processing of filtered queries in attributed semantic graphs. *Journal of Parallel and Distributed Computing (JPDC))*, 2014 (in press).

[15] T. Mattson, D. Bader, J. Berry, A. Buluç, J. Dongarra, C. Faloutsos, J. Feo, J. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. Leiserson, A. Lumsdaine, D. Padua, S. Poole, S. Reinhardt, M. Stonebraker, and S. Walla. Standards for graph algorithm primitives. In *High Performance Extreme Computing Conference (HPEC '13)*. IEEE, Sept. 2013.

[16] A. Pothen and C.-J. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.*, 16:303–324, December 1990.

[17] E. J. Riedy. *Making static pivoting scalable and dependable*. PhD thesis, University of California, Berkeley, 2010.

[18] N. Satish, N. Sundaram, M. A. Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin, and P. Dubey. Navigating the maze of graph analytics frameworks using massive graph datasets. In *SIGMOD*, 2014.

[19] E. Solomonik, A. Buluç, and J. Demmel. Minimizing communication in all-pairs shortest paths. In *Proceedings of the IPDPS*. IEEE Computer Society, 2013.

[20] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[21] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[22] E. Werner, S. McMillan, and J. Chu. Patterns and practices for future architectures. Technical Report CMU/SEI-2014-TN-001, Software Engineering Institute, Carnegie Mellon University, August 2014.

[23] I. Yamazaki and X. S. Li. On techniques to improve robustness and scalability of a parallel hybrid linear solver. In *High Performance Computing for Computational Science–VECPAR 2010*, pages 421–434. Springer, 2011.