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

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

Program Manager: Steven Lee
Reporting Period: 10/15-9/16

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 important progress during FY 2015. Specifically, our accomplishments include:

- Standardization of linear-algebraic building blocks for graph computations (GraphBLAS), as described in Section 3.1.
- Development of novel higher-level communication-avoiding matrix kernels for graph algorithms (detailed in Section 3.2).
- New formulations of graph algorithms in the language of linear algebra, as described in Section 3.3.

- Parallel data and graph analytics kernels for computational genomics. This is in collaboration with MANTISSA (ASCR - Applied Math) and DEGAS (ASCR - Computer Science) projects, and the details are explained in Section 3.4.

DOE relevance of each research accomplishment is mentioned within the corresponding section. Our next year plans to further advance both of these research areas, as well as our plans to expand our research into data clustering such as computing Markov Clustering in parallel. Further details can be found in Section 4. In FY16 alone, our project resulted in seven peer-reviewed publications [3, 20, 4, 7, 5, 18, 24]. In addition, we have one paper under review [25].

The report concludes with lists of awards, software artifacts, presentations, community service, and references. In particular, Dr. Buluç was part of the HipMer team that received the Best Use of HPC Application in Life Sciences by the HPCWire Magazine (Readers' Choice) for boosting the assembly of the human genome on the Cray XC30. The development of HipMer was partially supported by this project and more details can be found in Section 3.4.

Dr. Buluç was also an invited speaker at the IEEE International Workshop on High Performance Computational Biology (HiCOMB) at IPDPS'16 and a keynote speaker at the pSalsa (Parallel Software Libraries for Sequence Analysis) Workshop at the ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB'16).

Dr. Buluç co-organized a widely-attended minisymposia on "Combinatorial Scientific Computing" at the SIAM Conference on Parallel Processing for Scientific Computing, 2016 (with Alex Pothen) [1]. In addition, he organized a minisymposia on "GraphBLAS: Graph Algorithms in the Language of Linear Algebra" at the SIAM Annual Meeting 2016 [2].

2 Personnel

The project is lead by Aydın Buluç, a staff scientist, who spent about 50% of his time on this project. In addition, Ariful Azad, a postdoctoral fellow, spent the majority of his time (70%) on this project. Veronika Strnadova-Neeley, a graduate student research assistant (GSRA) from UC Santa Barbara, was funded to develop new similarity measures and data clustering algorithms. Carl Yang, a GSRA from UC Davis, was partially funded to assist with developing GraphBLAS kernels on multi-GPU systems.

3 Progress and Accomplishments

3.1 The GraphBLAS kernels

The GraphBLAS is an effort to define standard building blocks for graph algorithms in the language of linear algebra [2, 18]. The PI Buluç co-leads a multi-institutional group of researchers that are working on defining the right primitives and providing initial reference

[1]Informal report available at `https://cscresearchblog.wordpress.com/2016/05/11/csc-mini-symposium-at-siam-pp16/`

[2]Details can be found at `http://meetings.siam.org/sess/dsp_programsess.cfm?SESSIONCODE=23074`

implementations. Thanks to these efforts, the number of systems that support graph algorithms in the language of matrices has increased steadily [26, 14, 16]. Our own library, Combinatorial BLAS (CombBLAS) [10, 1], remains the gold standard among those efforts due to its pioneering status.

Dr. Buluç has been leading monthly teleconferences the community at-large (with 70 members at the moment). In addition, Dr. Buluç is a member of the five person team (along with Carl Yang who is also partially funded by this project) that works hands on to develop a C language application programming interface (API). The current C API specification [13] is almost final and permissions have been obtained from DOE to release it publicly. It is worth noting that Dr. Buluç is the only person that serves both as a steering committee member and as a member of the C language API specification committee.

We have successfully ran the second peer-reviewed Graph Algorithms Building Blocks (GABB) workshop at IPDPS'16, where the attendance was 30-40 people at any given time. There were 12 submissions that were peer-reviewed and 9 of them got accepted to be part of the proceedings. The call for papers for next version will be out at `http://www.graphanalysis.org/workshop2017.html` soon. Different from previous years, Dr. Buluç will be co-chairing the workshop program in 2017.

GraphBLAS will include the reduced-communication algorithms described in this report. The GraphBLAS 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.

The GraphBLAS website (`http://graphblas.org`) tracks our progress in more detail. Dr. Buluç has also co-authored a paper on the mathematical description of the GraphBLAS [18], along with Carl Yang who is also partially funded by this project. Our team develops new algorithms for GraphBLAS kernels and evaluates relative merits of different approaches. Sparse-sparse matrix multiplication (SpGEMM) and sparse-dense matrix multiplication (SpDM^3), which will be detailed in the next section, are key GraphBLAS kernels.

3.2 Communication-Avoiding Matrix Kernels for Graph Algorithms

Sparse matrix-matrix multiplication (SpGEMM) is a key GraphBLAS kernel (and perhaps the most challenging one), which 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. It is also the most time-consuming step (i.e. Galerkin triple product) of the Algebraic Multigrid setup phase at large concurrencies. The linear scaling electronic structure calculations, which are exponentially faster than their dense counterparts, also rely on SpGEMM [9].

Existing parallel algorithms for SpGEMM spend the majority of their time in inter-node communication on large concurrencies [12]. We had previously investigated communication-optimal algorithms for SpGEMM, and developed two new communication-optimal algorithms (one iterative and one recursive) that both attain the communication lower bounds [8].

Recently, we developed a high-performance implementation of the 3D iterative algorithm with in-node multithreading, which outperforms all the existing implementations at large scale. The percentage of time spent in communication (data movement) is significantly lower in our new implementation compared to a 2D implementation. This is advantageous for multiple reasons. First, the bandwidth for data movement is expected to increase at a slower rate than other system components, providing a future bottleneck. Second, communication costs more energy than computation [21, Figure 5].

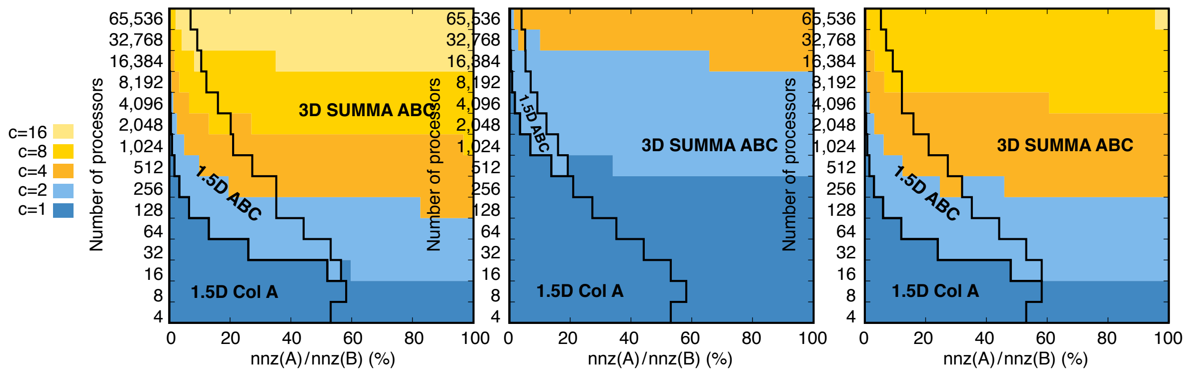
We also developed new multicore algorithms in order to fully exploit the increasing intra-node parallelism. In node performance will be more critical with the upcoming architectures such as NERSC Cori. A new implementation of the column-by-column algorithm of Buluç and Gilbert [11] shows good intra-node scaling. The paper presenting these two results is recently accepted at the SIAM Journal on Scientific Computing [3].

We have also explored multiplication of a sparse matrix with a dense matrix, or SpDM^3 in short, in an IPDPS paper [20]. This problem has applications in diverse domains such the all-pairs shortest-paths problem [27] in graph analytics (which itself has applications in non-linear dimensionality reduction), non-negative matrix factorization [19] for linear dimensionality reduction, a novel formulation of the restriction operation [22] in Algebraic Multigrid, quantum Monte Carlo simulations for large chemical systems [23], interior-point methods for semidefinite programming [15], and sparse inverse covariance selection (ICS) in statistical/machine learning.

We analyzed the communication lower bounds and compared the communication costs of various classic parallel algorithms for SpDM^3. We also presented new communication-avoiding algorithms based on a 1D decomposition, called 1.5D, which — while suboptimal in dense-dense and sparse-sparse cases — outperform the 2D and 3D variants both theoretically and in practice for sparse-dense multiplication. Experiments demonstrate speedups up to 100x over a baseline 3D SUMMA implementation and show parallel scaling over 10 thousand cores.

We plot Figure 1 to help understand which algorithm has the lowest overall bandwidth cost (theoretically) on varying number of processors and $\text{nnz}(A)/\,\text{nnz}(B)$ ratios [3] by relatively comparing the costs of various algorithms developed in that paper (3D Summa ABC, 1.5D Col A, 1.5D Col ABC, and 1.5D Inner ABC) for all possible replication factors. The best replication factors picked for each point are also shown in colors. Since this analysis is based on just $\text{nnz}(A)$, $\text{nnz}(B)$, and $\text{nnz}(C)$, it is trivially applicable to sparse-sparse matrix-matrix multiplication (SpGEMM) of different sparsities and/or sizes or even dense-dense matrix-matrix multiplication (DGEMM) of different sizes. Overall, this work greatly expands our understanding of the interaction between algorithms (1D, 2D, 3D, and variations) and input characteristics (sparsity, aspect ratio, size) for large-scale parallel matrix

[3] $\text{nnz}(M)$ computes the number of nonzeros of M and the multiplication is of the form $C = A\,B$



(a) 1:1 (b) 200,200:17,160 = 11.67:1 (c) 66,185:172,431 = 0.38:1

Figure 1: Illustrating areas that each algorithm has theoretically lowest overall bandwidth cost. X-axis is the ratio of nnz(A) versus nnz(B). Y-axis is the number of processors. There are three subgraphs for three different nnz(C) : nnz(B) ratios. *1.5D ABC* stands for both Col ABC and InnerABC. The area for *1.5D ABC* includes the area for *1.5D Col A*. Best replication factors for each data point are shown in colors. General observation is that ColA is best for sparser matrices or lower concurrency while SummaABC is the opposite. *1.5D ABC* algorithms help improve scalability of ColA.

multiplication, in terms of communication (and indirectly energy) costs.

In FY17, we plan to extend our studies to other GraphBLAS kernels, such as the multiplication of a sparse-matrix with a sparse vector (SpMSpV). This primitive forms the bases of many graph algorithms such as breadth-first search and maximal independent sets. It also serves as the workhorse of some machine learning algorithms such as support vector machines and logistic regression.

3.3 New graph algorithms in the language of linear algebra

Graph matching algorithms: In previous years, we had developed a novel tree-grafting algorithm for computing a maximum cardinality matching on bipartite graph [6]. The shared memory implementation of the tree-grafting algorithm scales up to 80 threads of an Intel multiprocessor (Westmere-EX) and is 10 times faster than current state-of-the-art matching algorithms for several important classes of graphs. In an extension of this work, recently accepted to the IEEE Transactions on Parallel and Distributed Systems (TPDS), we have demonstrated our algorithm's readiness to the Intel MIC architecture [7]. Our code achieved scalability up to 240 threads on an Intel Knights Corner coprocessor, the predecessor of the KNL architecture that will power Cori Phase 2.

Despite good scalability of matching algorithms on shared memory multiprocessors, it remained a challenge to achieve high performance on distributed platforms because these algorithms rely on searching for paths in the graph, and when few long paths pass through multiple processors, there is little concurrency. In this endeavor, we designed matching algorithms on sparse graphs in terms of linear algebra routines such as sparse matrix-vector multiplication (SpMV) and other operations on sparse and dense vectors. We used

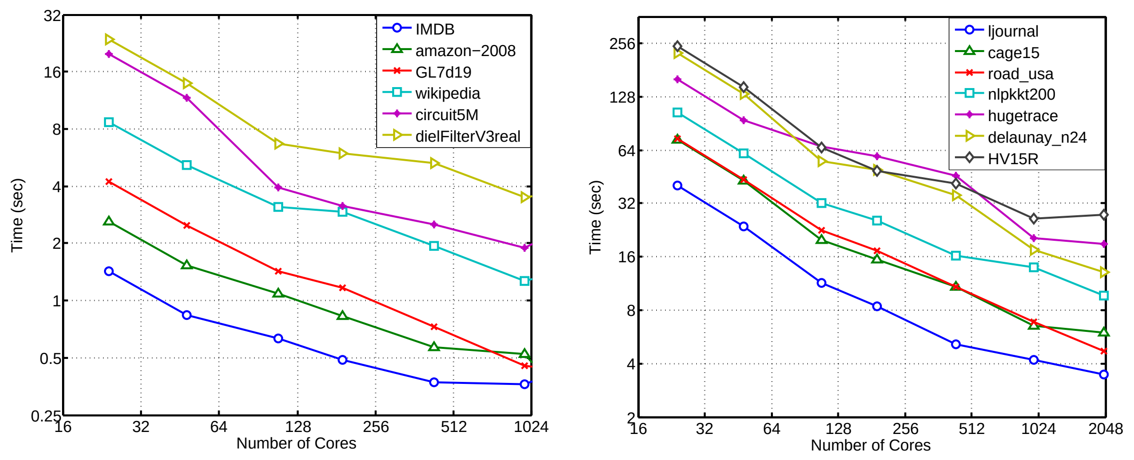


Figure 2: Strong scaling of MCM-Dist when computing maximum matching on real matrices on Edison. The left and right subfigures show the scaling of relatively smaller and larger matrices, respectively. 12 threads are used on all concurrencies except on 24 cores where each process on a 2×2 grid employs 6 threads.

existing functionalities of the CombBLAS library [10] and added several new functions to the library so that we can express the matching algorithms completely by a small number of functions from CombBLAS. By using these functions we implemented maximal and maximum cardinality matching algorithms.

In FY16, we have extended our conference paper on maximal cardinality matching algorithms, where we had used the same matrix-algebraic framework is used to implement three different algorithms with minimal modifications to the underlying semiring operations and data structures, to a more comprehensive journal publication [5].

More importantly, using the same algorithmic framework, we also developed scalable solutions to the significantly more challenging problem of maximum cardinality matching (MCM). MCM is a more challenging problem because the best practical algorithms are based on finding augmenting paths, whose lengths can vary widely. Hence, the computational load of the algorithm is highly dynamic. The number of active vertices, i.e. the size of the frontier during augmenting path searches, changes dramatically as the number of unmatched vertices decreases during execution, posing a significant challenge to harnessing the parallelism available. We use sparse vectors to represent the frontier sets, hence ensuring work efficiency during this dynamic execution. The results of our distributed-memory maximum cardinality algorithm [4] have been published recently. Figure 2 shows a brief strong scaling result on real graphs (i.e., sparse matrices).

In our primary application area of sparse linear solvers, the MCM can be applied to permute a matrix to its block triangular form, potentially leading to faster linear solvers. In addition, MCM is a subroutine for even more important problems, such as the weighted perfect matching problem (WPM). Due to the lack of distributed-memory parallel algorithms, the WPM problem is currently a sequential bottleneck in the celebrated distributed-memory version of the SuperLU sparse direct solver, where it is used for static pivoting.

3.4 Genome assembly

We have used high-performance graph analysis and data engineering to address the challenges of large-scale genome assembly. Recent advances in sequencing technology have made the redundant sampling of genomes extremely cost-effective. Such a sampling consists mostly of shorts reads with low error rates most of which can be aligned to a reference genome in a straightforward way. De novo genome assemblers, on the other hand, reconstruct an unknown genome from a collection of short reads, without requiring a reference genome. De novo assembly is therefore more powerful and flexible than mapping-based approaches, since it can be applied to novel genomes and can in principle discover previously unknown sequences in species for which reference genomes do not exist. This advantage, however, comes at a cost of significantly increased runtime and memory requirements. If de novo assembly could be performed rapidly, it would be preferable to mapping based approaches for large-scale human genome sequencing, and other biomedical model and agricultural species.

The components of the de novo genome assembly pipelines have very different execution profiles and unique challenges. The initial processing and data reduction (k-mer analysis) is bandwidth-bound, the subsequent construction and traversal of the *de Bruijn* graph are latency-bound, and the sequence alignment is compute bound. Many of the graph processing techniques developed for low-diameter graphs are not applicable to de Bruijn graphs, which have very low degree and a high diameter.

Meraculous is a hybrid k-mer/read-based whole genome assembler that avoids explicit error correction steps. While Meraculous is one of the best assemblers available in terms of accuracy, it is known to be too slow for large genomes. HipMer [17] is our high-performance re-engineering of Meraculous for distributed-memory supercomputers, designed and implemented from scratch using advanced data structures and Unified Parallel C (UPC)'s remote atomics capabilities. In FY16, we publicly released the source code of our HipMer genome assembler (`https://sourceforge.net/projects/hipmer/`), in line with ASCR's commitment to open-source software.

4 Other plans for the next fiscal year

Metagenome assembly: Metagenomics is the study of uncultured organisms sampled directly from the environment. Metagenome assembly is the process of combining the short reads produced by high-throughput sequencers into long contiguous genome sequences suitable for analysis. Each microbial genome is typically millions of base pairs long, four orders of magnitude longer than individual raw sequencing reads, and since a metagenome can contain hundreds to thousands of component genomes, its total assembled length can be many gigabases. Metagenome assembly is one of the most computationally demanding bioinformatics challenges, requiring multiple dependent sweeps of single genome techniques to filter errors and maximize quality.

Currently, we are working on leveraging the techniques developed for HipMer to handle the more complex problem of metagenome assembly. Due to differences in coverage of organisms involved, metagenome assembly is more difficult and requires even more computing resources. We have worked on this problem in FY16, where we developed an algorithm that is competitive in quality with the best assemblers but substantially faster. We will continue working on this problem in FY17 in collaboration with the ExaBiome project and finalize our metagenome assembly for publication.

Parallel Clustering: While many graph clustering methods exist, they typically have similar computational challenges, hence we will work on a famous graph clustering algorithm named Markov Cluster algorithm (MCL) [28]. The MCL algorithm is extremely popular among biologists, frequently used as the go-to algorithm for clustering protein and gene sequences. The workhorse of this algorithm is the SpGEMM primitive but a black-box SpGEMM implementation is not usable due to memory limitations. Consequently, we will develop parallel multi-stage SpGEMM implementations that are memory efficient. In addition, we will also develop new scalable algorithms for other phases of MCL, namely finding top-k entries in each column and finding weakly-connected components in the final matrix.

Graph Ordering: Ordering vertices of a graph is key to minimize fill in sparse direct solvers and to maximize locality in iterative solvers. Except for naturally parallelizable ordering methods (e.g., nested dissection), many important ordering methods (e.g., reverse Cuthill-McKee and minimum degree) have similarly not been efficiently mapped to distributed-memory architectures, a problem our team has recently started tackling together with Esmond Ng's group. Due to the long dependency structure, graph ordering problems have challenges similar to graph matching.

Awards

HipMer team, the Best Use of HPC Application in Life Sciences by the HPCWire Magazine (Readers' Choice) for boosting the assembly of the human genome on the Cray XC30.

Software artifacts

Combinatorial BLAS: Both Dr. Buluç and Dr. Azad contributed significantly to the development of the Combinatorial BLAS (CombBLAS) library [10, 1] along both the functionality and the performance axes. We released significantly improved version 1.5 in January 2016. The new features include complete in-node multithreading support, experimental support for 3D distributions, fully parallel I/O, experimental support for the distributed CSC data structure, new maximal and maximum cardinality matching implementations as well as the necessary auxiliary functions.

HipMer: HipMer is an end-to-end high performance de novo assembler designed to scale to massive concurrencies. It is based on the Meraculous assembler, which has been shown to produce high quality assembly results. Large-scale results on the Cray XC30 supercomputer demonstrate efficient performance and scalability on thousands of cores. We publicly released the source code of our HipMer genome assembler (`https://sourceforge.net/projects/hipmer/`), in line with ASCR's commitment to open-source software.

MS-BFS-Graft: Multithreaded OpenMP code for computing maximum cardinality matching on bipartite graphs. Performs multi-source breadth-first search with tree-grafting for exploiting parallelism. The code is developed by Dr. Azad and Dr. Buluç and released to accompany our TPDS paper. The code is available at `https://bitbucket.org/azadcse/ms-bfs-graft`

Selected Talks

- “Parallel de novo Assembly of Complex Genomes via HipMER”, Aydın Buluç, *invited talk* at the IEEE International Workshop on High Performance Computational Biology (HiCOMB), May 23, 2016
- “A tour of contemporary genome assembly algorithms and software”, Aydın Buluç, *keynote talk* at the Workshop on Parallel Software Libraries for Sequence Analysis (pSALSA), October 2, 2016
- “Faster and more scalable sparse matrix-matrix multiplication”, Aydın Buluç, *minisymposium talk* at the SIAM Conference on Parallel Processing for Scientific Computing, Paris, France, 2016.
- ”The GraphBLAS effort: Kernels, API, and parallel implementations”, Aydın Buluç, *minisymposium talk* at the SIAM Conference on Parallel Processing for Scientific Computing, Paris, France, 2016.
- “Distributed-memory algorithms for cardinality matching using matrix algebra”, Ariful Azad, *minisymposium talk* at the SIAM Conference on Parallel Processing for Scientific Computing, Paris, France, 2016.
- “Scalable parallel algorithms for de novo assembly of complex genomes”. Aydın Buluç, *workshop talk* at The Resurgence of Reference Quality Genome Sequence session, Plant & Animal Genome (PAG) Conference, San Diego, CA, 2016.
- “Parallel Sparse Matrix-Matrix Multiplication and Its Use in Triangle Counting and Enumeration”, Ariful Azad, *minisymposium talk* at the SIAM Applied Linear Algebra, Atlanta, October 26, 2015.

Community Service

Aydın Buluç:

- *Associate Editor*: ACM Transactions on Parallel Computing
- *Minisymposia Organizer*:
  - “Combinatorial Scientific Computing” (w/ Alex Pothen) at SIAM Conference on Parallel Processing for Scientific Computing, 2016
  - “GraphBLAS: Graph Algorithms in the Language of Linear Algebra” at SIAM Annual Meeting, 2016
- *Program Committee*:

  - IEEE International Parallel & Distributed Processing Symp. (IPDPS), 2016
  - The ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2016
  - IEEE High Performance Extreme Computing Conference (HPEC), 2016
  - SIAM Workshop on Combinatorial Scientific Computing (CSC), 2016
  - IEEE Workshop on High Performance Computational Biology (HiCOMB), 2016
  - High Performance Graph Processing (HPGP) workshop at HPDC, 2016

- *Steering & Program Committee*: Graph Algorithms Building Blocks (GABB), IPDPS workshop, 2016.

Ariful Azad:

- *Program Committee*: IEEE Int. Parallel & Dist. Processing Symp. (IPDPS), 2016

References

[1] Combinatorial BLAS Library, version 1.5, 2016. `http://graphblas.org/~aydin/CombBLAS/html/`.

[2] The graphblas forum, 2016. `http://graphblas.org/`.

[3] A. Azad, G. Ballard, A. Buluç, J. Demmel, L. Grigori, O. Schwartz, S. Toledo, and S. Williams. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. *SIAM Journal on Scientific Computing (SISC)*, 2016. (to appear).

[4] A. Azad and A. Buluç. Distributed-memory algorithms for maximum cardinality matching in bipartite graphs. In *Proceedings of the IPDPS*, 2016.

[5] A. Azad and A. Buluç. A matrix-algebraic formulation of distributed-memory maximal cardinality matching algorithms in bipartite graphs. *Parallel Computing*, 2016.

[6] A. Azad, A. Buluç, and A. Pothen. A parallel tree grafting algorithm for maximum cardinality matching in bipartite graphs. In *Proceedings of the IPDPS*, 2015.

[7] A. Azad, A. Buluç, and A. Pothen. Computing maximum cardinality matchings in parallel on bipartite graphs via tree-grafting. *IEEE Transactions on Parallel and Distributed Systems (TPDS))*, 2016.

[8] 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.

[9] 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.

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

[11] A. Buluç and J. R. Gilbert. On the representation and multiplication of hypersparse matrices. In *IPDPS'08: Proceedings of the 22nd IEEE International Symposium on Parallel&Distributed Processing*, pages 1–11. IEEE Computer Society, 2008.

[12] 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.

[13] A. Buluç, T. Mattson, S. McMillan, J. Moreira, and C. Yang. Proposal for a GraphBLAS C API. Technical report, September 2016. *Working document from the GraphBLAS Signatures Subgroup.*

[14] K. Ekanadham, W. P. Horn, M. Kumar, J. Jann, J. Moreira, P. Pattnaik, M. Serrano, G. Tanase, and H. Yu. Graph Programming Interface (GPI): A linear algebra programming model for large scale graph computations. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '16, pages 72–81, New York, NY, USA, 2016. ACM.

[15] K. Fujisawa, M. Kojima, and K. Nakata. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. *Mathematical Programming*, 79(1-3):235–253, 1997.

[16] V. Gadepally, J. Bolewski, D. Hook, D. Hutchison, B. Miller, and J. Kepner. Graphulo: Linear algebra graph kernels for nosql databases. In *International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, pages 822–830. IEEE, 2015.

[17] E. Georganas, A. Buluç, J. Chapman, S. Hofmeyr, C. Aluru, R. Egan, L. Oliker, D. Rokhsar, and K. Yelick. HiPMer: An extreme-scale de novo genome assembler. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15)*, 2015.

[18] J. Kepner, P. Aaltonen, D. Bader, A. Buluç, F. Franchetti, J. Gilbert, D. Hutchison, M. Kumar, A. Lumsdaine, H. Meyerhenke, S. McMillan, J. Moreira, J. Owens, C. Yang, M. Zalewski, and T. Mattson. Mathematical foundations of the GraphBLAS. In *IEEE High Performance Extreme Computing (HPEC)*, 2016.

[19] J. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

[20] P. Koanantakool, A. Azad, A. Buluç, D. Morozov, S.-Y. Oh, L. Oliker, and K. Yelick. Communication-avoiding parallel sparse-dense matrix-matrix multiplication. In *Proceedings of the IPDPS*, 2016.

[21] P. Kogge and J. Shalf. Exascale computing trends: Adjusting to the. *Computing in Science & Engineering*, 15(6):16–26, 2013.

[22] M. McCourt, B. Smith, and H. Zhang. Sparse matrix-matrix products executed through coloring. *SIAM Journal on Matrix Analysis and Applications*, 36(1):90–109, 2015.

[23] A. Scemama, M. Caffarel, E. Oseret, and W. Jalby. Quantum Monte Carlo for large chemical systems: Implementing efficient strategies for petascale platforms and beyond. *Journal of computational chemistry*, 34(11):938–951, 2013.

[24] V. Strnadová-Neeley, A. Buluç, J. Chapman, J. R. Gilbert, J. Gonzalez, and L. Oliker. Efficient data reduction for large-scale genetic mapping. In *Proc. 6th ACM Conf. on Bioinformatics, Computational Biology and Health Informatics*, BIBM '15, pages 126–135, 2015.

[25] V. Strnadová-Neeley, A. Buluç, J. R. Gilbert, L. Oliker, and W. Ouyang. LiRa: A new likelihood-based similarity score for collaborative filtering. 2016.

[26] N. Sundaram, N. Satish, M. M. A. Patwary, S. R. Dulloor, M. J. Anderson, S. G. Vadlamudi, D. Das, and P. Dubey. Graphmat: High performance graph analytics made productive. *Proceedings of the VLDB Endowment*, 8(11):1214–1225, 2015.

[27] A. Tiskin. All-pairs shortest paths computation in the BSP model. In *ICALP*, pages 178–189, 2001.

[28] S. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, 2000.