# Brief Announcement: Hypergraph Partitioning for Parallel Sparse Matrix-Matrix Multiplication

Grey Ballard
Sandia National Laboratories
gmballa@sandia.gov

Alex Druinsky
Lawrence Berkeley National Laboratory
adruinsky@lbl.gov

Nicholas Knight
University of California, Berkeley
knight@cs.berkeley.edu

Oded Schwartz
Hebrew University, Jerusalem, Israel
odedsc@cs.huji.ac.il

## ABSTRACT

The performance of parallel algorithms for sparse matrix-matrix multiplication is typically determined by the amount of interprocessor communication performed, which in turn depends on the nonzero structure of the input matrices. In this paper, we characterize the communication cost of a sparse matrix-matrix multiplication algorithm in terms of the size of a cut of an associated hypergraph that encodes the computation for a given input nonzero structure. Obtaining an optimal algorithm corresponds to solving a hypergraph partitioning problem. Our hypergraph model generalizes several existing models for sparse matrix-vector multiplication, and we can leverage hypergraph partitioners developed for that computation to improve application-specific algorithms for multiplying sparse matrices.

## 1. INTRODUCTION

Sparse matrix-matrix multiplication (SpGEMM) is a fundamental computation in applications ranging from linear solvers to graph algorithms and data analysis (see [5] and references therein). SpGEMM algorithms are typically communication bound, spending much more of their time moving data than performing additions and multiplications. Furthermore, the computation is usually irregular and depends on the nonzero structures of the input matrices.

Hypergraphs are used to model and optimize parallel algorithms for sparse matrix-dense vector multiplication (SpMV); see, e.g., [7, 8]. More recently, hypergraph models have been proposed for the outer-product algorithm for SpGEMM [1]. In this work, we present a hypergraph model that generalizes the "fine-grain" SpMV model [7] and considers a more general class of SpGEMM algorithms than [1] to minimize communication. Hypergraphs have been used to model more general computations than SpGEMM [10]: we differ from this work by optimizing interprocessor communication rather than disk I/O.

In particular, in this work we consider the *critical-path communication cost* of a parallel algorithm, which is the number of words communicated between processors along a critical path of the algorithm. This cost is usually correlated with (but can exceed) the maximum number of words communicated by any processor during the algorithm, and it differs from the communication volume, which is the total number of words communicated by all processors.

The main contributions of this work are (1) showing that SpGEMM can be modeled by a hypergraph so that its communication cost corresponds to the size of a cut induced by a partition of the vertices and (2) demonstrating that hypergraph partitioners can be used to determine efficient SpGEMM algorithms for an algebraic multigrid application.

## 2. THEORETICAL MODEL

Let $\mathbf{A}$ and $\mathbf{B}$ be $I$-by-$K$ and $K$-by-$J$ matrices over $X$, a set closed under two binary operations denoted by addition (commutative and associative with identity element 0) and multiplication (with absorbing element 0). SpGEMM is $(\mathbf{A}, \mathbf{B}) \mapsto \mathbf{C}$, where $\mathbf{C}$ is an $I$-by-$J$ matrix over $X$ defined entrywise by $c_{ij} = \sum_{k \in [K]} a_{ik} b_{kj}$ ($[K]$ denotes the set $\{1, \ldots, K\}$). We let $S_\mathbf{A} \subseteq [I] \times [K]$, $S_\mathbf{B} \subseteq [K] \times [J]$, and $S_\mathbf{C} \subseteq [I] \times [J]$ denote the nonzero structures of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. Here we consider algorithms that evaluate and sum all *nontrivial* multiplications $a_{ik} b_{kj}$, where $a_{ik}$ and $b_{kj} \neq 0$, and thus depend only on $S_\mathbf{A}$ and $S_\mathbf{B}$. We do not consider algorithms that exploit additional structure on $X$ or more general relations on the entries of $\mathbf{A}$ and $\mathbf{B}$: in particular, we ignore numerical cancellation, so $S_\mathbf{A}$ and $S_\mathbf{B}$ induce $S_\mathbf{C}$.

Definition 1. *Given input matrices* $\mathbf{A}$ *and* $\mathbf{B}$*, let the* SpGEMM hypergraph *be* $\mathcal{H}(\mathbf{A}, \mathbf{B}) = (\mathcal{V}, \mathcal{N})$*, with vertices*

$$\mathcal{V} = \{ v_{ikj} : (i,k) \in S_\mathbf{A} \wedge (k,j) \in S_\mathbf{B} \}$$

*and nets* $\mathcal{N} = \mathcal{N}^\mathbf{A} \cup \mathcal{N}^\mathbf{B} \cup \mathcal{N}^\mathbf{C}$*, where*

$$\begin{aligned} \mathcal{N}^\mathbf{A} &= \{ n^\mathbf{A}_{ik} : (i,k) \in S_\mathbf{A} \} \ \textit{with}\ n^\mathbf{A}_{ik} = \{ v_{ikj} : j \in [J] \}, \\ \mathcal{N}^\mathbf{B} &= \{ n^\mathbf{B}_{kj} : (k,j) \in S_\mathbf{B} \} \ \textit{with}\ n^\mathbf{B}_{kj} = \{ v_{ikj} : i \in [I] \}, \\ \mathcal{N}^\mathbf{C} &= \{ n^\mathbf{C}_{ij} : (i,j) \in S_\mathbf{C} \} \ \textit{with}\ n^\mathbf{C}_{ij} = \{ v_{ikj} : k \in [K] \}. \end{aligned}$$

In $\mathcal{H}(\mathbf{A}, \mathbf{B})$, each vertex corresponds to a nontrivial multiplication, and each net $n^\mathbf{A}_{ik}$, $n^\mathbf{B}_{kj}$, and $n^\mathbf{C}_{ij}$ corresponds to a

*SPAA'15,* June 13–15, 2015, Portland, OR, USA.
ACM 978-1-4503-3588-1/15/06.
http://dx.doi.org/10.1145/2755573.2755613.

nonzero of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and contains all nontrivial multiplications in which that nonzero participates.

We now consider performing SpGEMM with input matrices $\mathbf{A}$ and $\mathbf{B}$ on a parallel machine with $p$ processors with disjoint memories. A *parallelization* is a $p$-way partition of $\mathcal{V}$ (assigning multiplications to processors) and a *data distribution* is a triple of $p$-way partitions of $S_\mathbf{A}$, $S_\mathbf{B}$, and $S_\mathbf{C}$ (assigning nonzeros to processors). The communication proceeds in two phases: the *expand* phase, where the processors exchange nonzero entries of $\mathbf{A}$ and $\mathbf{B}$ (initially distributed according to the partitions of $S_\mathbf{A}$ and $S_\mathbf{B}$) in order to perform their multiplications (assigned according to the partition of $\mathcal{V}$), and the *fold* phase, where the processors communicate to reduce partial sums for nonzero entries of $\mathbf{C}$ (finally distributed according to the partition of $S_\mathbf{C}$).

The model proposed in Definition 1 is a generalization of the fine-grain SpMV model [7], but it also has an important intuitive distinction. In the fine-grain SpMV model, nets correspond to rows and columns of the matrix and vertices correspond to entries of the matrix. In $\mathcal{H}(\mathbf{A},\mathbf{B})$, if $\mathbf{B}$ is a dense vector, then all of the nets in $\mathcal{N}^\mathbf{A}$ collapse to singletons, the $K$ nets of $\mathcal{N}^\mathbf{B}$ correspond to columns of the matrix $\mathbf{A}$, and the $I$ nets of $\mathcal{N}^\mathbf{C}$ correspond to rows of the matrix $\mathbf{A}$. The vertices of $\mathcal{H}(\mathbf{A},\mathbf{B})$ correspond to scalar multiplications, which in this case coincide with entries of $\mathbf{A}$. Thus, $\mathcal{H}(\mathbf{A},\mathbf{B})$ reproduces the SpMV model.

However, we emphasize that the vertices of $\mathcal{H}(\mathbf{A},\mathbf{B})$ correspond to computation rather than data, which in the case of general SpGEMM do not coincide (as they do for SpMV). Therefore, the principal partitioning problem is that of assigning work to processors. Data distribution, or partitioning the entries of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ among processors, is our secondary concern, much like the partitioning of the input and output vectors in the case of SpMV.

The model proposed in Definition 1 is also distinct from the ones proposed in [1]. The approach in [1] considers a restricted class of parallelizations, known as outer-product algorithms, which leads to hypergraphs with fewer vertices and nets, while our model encompasses 1D (which include outer-product), 2D, and 3D parallelizations as defined in [2]. Another difference in the models is that the approach in [1] simultaneously partitions scalar multiplications and output matrix data, while we partition only the multiplications, as described above.

Definition 2. *Given a partition $\{\mathcal{V}_1,\dots,\mathcal{V}_p\}$ of $\mathcal{V}$, for each $i \in [p]$, we define $Q_i$, the $i$th cut of $\mathcal{H}$, to be the subset of $\mathcal{N}$ having nonempty intersections with both $\mathcal{V}_i$ and $\mathcal{V}\setminus\mathcal{V}_i$.*

Lemma 1. *Given an SpGEMM computation with parallelization $\{\mathcal{V}_1,\dots,\mathcal{V}_p\}$ and any data distribution, the number of words each processor $i$ sends or receives is at least $|Q_i|$, and the critical-path cost is at least $\max_{i\in[p]}|Q_i|$.*

Proof. For each processor $i$, for each net in $Q_i$, processor $i$ must either receive or send the corresponding nonzero, since at most one processor owns each nonzero at the start and end of the computation. (While singleton nets may not uniquely correspond to a nonzero in $\mathbf{A}$, $\mathbf{B}$, or $\mathbf{C}$, they are never cut.) The bound on the critical-path communication cost is obtained by maximizing over processors. $\square$

Lemma 1 yields a lower bound over all parallelizations, subject to a computational load balance constraint.

Definition 3. *For any $\epsilon \in [0,p-1]$, let $\Pi_\epsilon$ be the set of all partitions $\{\mathcal{V}_1,\dots,\mathcal{V}_p\}$ of $\mathcal{V}$ where $|\mathcal{V}_i| \le (1+\epsilon)|\mathcal{V}|/p$ for each $i \in [p]$. We say an SpGEMM computation with parallelization $\{\mathcal{V}_1,\dots,\mathcal{V}_p\} \in \Pi_\epsilon$ is* $\epsilon$-load balanced.

Note that the sets $\Pi_\epsilon$ are nested: $\Pi_0$ contains only perfectly balanced partitions while $\Pi_{p-1}$ contains every partition, including trivial parallelizations with no interprocessor communication where one processor performs the whole computation.

Theorem 1. *For any $\epsilon$-load balanced SpGEMM computation, its critical-path communication cost is at least*

$$\min_{\{\mathcal{V}_1,\dots,\mathcal{V}_p\}\in\Pi_\epsilon} \max_{i\in[p]} |Q_i|.$$

The next result shows that this critical-path lower bound is tight up to a logarithmic factor.

Theorem 2. *For an SpGEMM computation with parallelization $\{\mathcal{V}_1,\dots,\mathcal{V}_p\}$, there exists a data distribution such that the number of words processor $i$ sends/receives is $O(|Q_i|)$ and the critical-path communication cost is $O(\log p \cdot \max|Q_i|)$.*

Proof. We construct a data distribution by assigning the nonzero corresponding to each net in $\mathcal{N}^\mathbf{A}$, $\mathcal{N}^\mathbf{B}$, and $\mathcal{N}^\mathbf{C}$ to one of the processors owning a vertex in that net. The expand phase proceeds in at most $O(\log p)$ steps. Each nonzero of $\mathbf{A}$ and $\mathbf{B}$ is associated with a binary-tree broadcast among the processors whose parts $\mathcal{V}_i$ intersect the corresponding nets. Processor $i$ receives each of its nonzeros at most once and sends each at most twice, for a total cost of $O(|Q_i|)$. Further, at each step $j$, processor $i$ performs its assigned sends or receives from all the broadcast trees in which it is involved at level $j$ (at most $|Q_i|$), and so each step involves at most $O(\max_{i\in[p]}|Q_i|)$ sends/receives along the critical path. The fold phase is similar, using binary-tree reductions. $\square$

## 3. EXPERIMENTAL RESULTS

Theorems 1 and 2 together reduce the problem of obtaining an optimal (to within a logarithmic factor) algorithm for the multiplication for a particular pair of sparse matrices to the problem of hypergraph partitioning. Unfortunately, this would need to be solved for every instance of SpGEMM (or at least for every pair of sparsity patterns), and it is an NP-hard problem [11]. However, we propose considering representative matrices for specific application areas in order to gain intuition for algorithmic design. In addition, efficient software, such as the PaToH [6] and Zoltan [4] libraries, exists for solving the problem approximately. The main limitations of the software are that (1) there are no guarantees for the quality of approximation and (2) the typical objective function is communication volume rather than critical-path communication cost. In this section, we focus on two SpGEMM computations arising from the setup phase of algebraic multigrid.

We experimentally study the communication costs of computing the product $\mathbf{P}^T\mathbf{A}\mathbf{P}$, formed to produce the grid hierarchy of an algebraic multigrid PDE solver. Here, $\mathbf{A}$ is the adjacency matrix of a graph $G_\mathbf{A}$ of $N = n^3$ vertices arranged as a 3D lattice, where every non-boundary vertex is adjacent to itself and its 26 closest neighbors. The matrix $\mathbf{P}$ corresponds to a bipartite graph $G_\mathbf{P} = (U,V,E)$, where $U$ is the vertex set of $G_\mathbf{A}$ and $V$ is the set of $(n/3)^3$ disjoint

**Table 1: Maximum per-processor communication cost of forming $\mathbf{P}^T\mathbf{AP}$.**

| | | **AP** | | $\mathbf{P^T(AP)}$ | |
|---|---|---|---|---|---|
| $N$ | $p$ | row | fine | row | fine |
| 19,683 | 27 | 5,528 | 4,649 | 10,712 | 964 |
| 91,125 | 125 | 5,528 | 5,823 | 10,712 | 1,324 |
| 250,047 | 343 | 5,528 | 6,160 | 10,712 | 1,444 |
| 531,441 | 729 | 5,528 | 6,914 | 10,712 | 1,491 |
| 970,299 | 1,331 | 5,528 | 6,679 | 10,712 | 1,548 |

3-by-3-by-3 subcubes of $U$. Every subcube $v \in V$ is adjacent to each vertex $u \in U$ that belongs to $v$ or is one of $v$'s neighbors.

We consider two approaches for multiplying these matrices. In the *fine-grained* approach, we use our hypergraph model: we populate data structures that represent $\mathcal{H}(\mathbf{A}, \mathbf{P})$ and $\mathcal{H}(\mathbf{P}^T, \mathbf{AP})$ as given in Definition 1, and use the PaToH library to obtain a good partition, i.e., one with a small cut. Theorems 1 and 2 indicate that parallelizing the computation according to this partition can minimize communication.

The other approach we consider is the *row-wise* approach that corresponds to the way multigrid matrices are usually multiplied in practice [9]. In this approach we partition the output matrix rows among the processors manually, exploiting the multigrid matrices' correspondences with 3D lattices. Consider the rows of $\mathbf{A}$ as the $n^3$ points of a 3D lattice, partition them into $p$ equal subcubes, and assign the rows that belong to each subcube to a distinct processor. We use each processor to compute the corresponding rows of $\mathbf{AP}$, and then similarly partition the rows of $\mathbf{P}^T$ and compute $\mathbf{P}^T(\mathbf{AP})$. This approach corresponds to choosing one specific partition among those considered by the hypergraph partitioner and therefore it can yield a suboptimal cut.

We compare the two approaches by computing the maximum number of words sent or received by any one processor, or $\max_i |Q_i|$, using the notation of the previous section. The results are shown in Table 1, where we follow a weak-scaling scheme that increases $N$ and $p$ proportionally. The table shows that in the $\mathbf{AP}$ step, the cost of the row-wise approach is comparable to or less than that of the fine-grained approach. We believe the slight advantage to the row-wise approach is due to the fact that PaToH minimizes communication volume rather than maximum per-processor cost. In the $\mathbf{P}^T(\mathbf{AP})$ step, the cost of the row-wise approach is 7 to 11 times greater than that of the fine-grained approach. Although this gap slightly narrows as we scale, likely because of the mismatch in cost functions, the fine-grained approach's advantage is dramatic. Our intuitive explanation of this result is based on the observation that the row-wise approach communicates the rows of the second input matrix. When we multiply $\mathbf{AP}$, the matrix $\mathbf{P}$ is quite sparse. However, $\mathbf{AP}$ is less sparse and so for $\mathbf{P}^T(\mathbf{AP})$, another algorithm derived from the fine-grained approach is a better choice.

The results demonstrate that better algorithms for $\mathbf{P}^T\mathbf{AP}$ should be developed; improvements are described in a separate paper [3].

## Acknowledgments

This research is supported by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000; the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Applied Mathematics program under contract number DE-AC02-05CH11231; Department of Energy grant DE-SC0008700; grants 1878/14 and 1901/14 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities); grant 3-10891 from the Ministry of Science and Technology, Israel; the Einstein Foundation; and the Minerva Foundation.

## 4. REFERENCES

[1] K. Akbudak and C. Aykanat. Simultaneous input and output matrix partitioning for outer-product–parallel sparse matrix-matrix multiplication. *SISC*, 36(5):C568–C590, 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 '13*, pages 222–231. ACM, 2013.

[3] G. Ballard, J. Hu, and C. Siefert. Reducing communication costs for sparse matrix multiplication within algebraic multigrid. Technical Report SAND2015-3275, Sandia Natl. Labs., 2015.

[4] E. Boman, K. Devine, L. Fisk, R. Heaphy, B. Hendrickson, C. Vaughan, Ü. Çatalyürek, D. Bozdag, W. Mitchell, and J. Teresco. Zoltan 3.0: parallel partitioning, boad-balancing, and data management services; user's guide. Technical Report SAND2007-4748W, Sandia Natl. Labs., 2007.

[5] A. Buluç and J. R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: implementation and experiments. *SISC*, 34(4):C170–C191, 2012.

[6] Ü. Çatalyürek and C. Aykanat. PaToH: a multilevel hypergraph partitioning tool, version 3.0. Technical report, Dept. of Computer Engineering, Bilkent Univ., 1999.

[7] Ü. Çatalyürek and C. Aykanat. A fine-grain hypergraph model for 2D decomposition of sparse matrices. In *IPDPS '01*, pages 118–123, 2001.

[8] Ü. Çatalyürek, C. Aykanat, and B. Uçar. On two-dimensional sparse matrix partitioning: models, methods, and a recipe. *SISC*, 32(2):656–683, 2010.

[9] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. ML 5.0 Smoothed Aggregation User's Guide. Technical Report SAND2006-2649, Sandia Natl. Labs., 2006.

[10] S. Krishnamoorthy, Ü. Çatalyürek, J. Nieplocha, A. Rountev, and P. Sadayappan. Hypergraph partitioning for automatic memory hierarchy management. In *SC '06*, pages 34–46, 2006.

[11] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Wiley, 1990.