Report

Communication Lower Bounds for Tensor Contraction Algorithms

Author(s):
Solomonik, Edgar; Demmel, James; Hoefler, Torsten

Publication Date:
2015

Permanent Link:

https://doi.org/10.3929/ethz-a-010350411 →

Communication Lower Bounds for Tensor Contraction Algorithms

Edgar Solomonik
ETH Zürich
solomonik@inf.ethz.ch

James Demmel
University of California, Berkeley
demmel@cs.berkeley.edu

Torsten Hoefler
ETH Zürich
htor@inf.ethz.ch

ABSTRACT

Contractions of nonsymmetric tensors are reducible to matrix multiplication, however, 'fully symmetric contractions' in which the tensors are symmetric and the result is symmetrized can be done with fewer operations. The 'direct evaluation algorithm' for fully symmetric contractions exploits equivalence between terms in the contraction equation to obtain a lower computation cost than the cost associated with nonsymmetric contractions. The 'symmetry preserving algorithm' lowers the cost even further via an algebraic reorganization of the contraction equation. We derive vertical (between memory and cache) and horizontal (interprocessor) communication lower bounds for both of these algorithms. We demonstrate that any load balanced parallel schedule of the direct evaluation algorithm requires asymptotically more horizontal communication for some fully symmetric contractions than matrix multiplication for nonsymmetric contractions of the same size. Instances of such fully symmetric contractions arise in quantum chemistry calculations. Further, we prove that any schedule of the symmetry preserving algorithm requires asymptotically more vertical and horizontal communication than the direct evaluation algorithm for some fully symmetric contractions. However, for the instances of fully symmetric contractions that arise in quantum chemistry calculations, our lower bounds are asymptotically the same for both of these algorithms.

1. INTRODUCTION

Tensor contractions generalize the product of a matrix and a vector and of two matrices as well as provide an algebra that is widely used in chemistry and physics. Tensor contraction computations are reducible to matrix computations, with the exception of operand tensor symmetry and symmetrization of the output tensor, which are quite common in quantum chemistry tensor computations [3]. In this paper, we study the performance characteristics of the classical 'direct evaluation algorithm' and new 'symmetry preserving algorithm' [16] for 'fully symmetric contractions' (symmetrized contractions of symmetric tensors).

We characterize the communication requirements of both of these algorithms by deriving lower bounds on the communication cost on any schedule which executes the algorithms. Our algorithms are defined by a set of multiplications and summations of sets of elements. We allow schedules to execute the multiplications and perform the summations in any order, however, we preclude recomputation of any operation in the algorithm. For the symmetry preserving algorithm, our lower bound proofs require an additional restriction that precludes schedules which reuse partial summations (from a certain viewpoint, such reuse would correspond to a reorganization of the algorithm). Our lower bounds for communication cost of parallel schedules assume that each unique tensor element is initially stored on a unique processor and that the number inputs and outputs is load balanced across processors.

The symmetry preserving algorithm employs an algebraic reorganization to decrease the computation cost with respect to the direct evaluation approach for fully symmetric tensor contractions by a up to a factorial of the order of the tensor contraction (total number of tensor indices). The order of the tensor contraction is typically a constant for a problem, for instance it is two for matrix-vector multiplication. Therefore, it is of interest to consider constant factors in the communication lower bounds. Further, other recent research efforts have considered constant factors in lower bounds for matrix multiplication [2, 8] and sparse-matrix vector multiplication [4]. We derive vertical (between memory and cache) communication lower bounds that take into account constant factors, but limit our analysis to asymptotic horizontal (interprocessor) communication.

We obtain communication lower bounds for nonsymmetric tensor contractions by direct reduction from matrix multiplication. Except for symmetrization, the direct evaluation algorithm for fully symmetric tensor contractions computes a product of matrices that are of smaller dimension that in nonsymmetric contractions and also have some equivalent entries. We obtain all except one of the communication lower bounds for the direct evaluation algorithm via a nontrivial reduction from matrix multiplication. In order to obtain tighter lower bounds for these algorithms, we give an improved constant factor for the vertical communication lower bound of matrix multiplication. We also derive one horizontal communication lower bound for the direct evaluation algorithm that is asymptotically stronger for certain fully symmetric contractions. This new lower bound has interesting practical implications.

We then use the generalized form of the Loomis-Whitney inequality [12, 19] to obtain lower bounds on the communication costs of the new 'symmetry preserving' tensor contraction algorithm. The Loomis-Whitney inequality bounds the size of a set of order d as a function of the size of its order $d-1$ projections [12]. The generalized Loomis-Whitney inequality bounds the size of an order m set as a function of the size of its order r projections, for any $r < m$ [19]. We state the latter inequality at the start

of Section 4, interpreting it conversely as a lower bound on the size of the union of the order r projections with respect to the size of the order m set. Our lower bounds for the 'symmetry preserving' algorithm are to the best of our knowledge the first that need to employ generalized Loomis-Whitney, as the algorithm's communication lower bounds have instances for any m and any $r \in [\lceil m/2 \rceil, m-1]$. Previous work on communication lower bounds via the regular Loomis-Whitney bound have included

- matrix-vector and the vector outer products with $d = 2$ [19],
- matrix multiplication with $d = 3$ [2, 10, 19],
- stencil computations on an order f mesh with $d = f+1$ [15].

As a result of this analysis, we conclude that for certain contractions the symmetry preserving algorithm requires more communication than the direct evaluation algorithm, however, for most contractions of interest to applications (e.g. coupled-cluster [21]), the two algorithms have the same lower bounds within a constant factor. Our conclusions are supported by the fact that most of the lower bounds are asymptotically attainable by existing or proposed algorithms, as we discuss in Section 6.

2. TENSOR NOTATION

We use the notation from the introductory work on the symmetry preserving algorithm [16]. One key restriction we make is considering all elements of tensors to be on the same algebraic ring. More general definitions of elements and element-wise operations in contractions enable the extension of the formalism and algorithms to partially symmetric contractions [16].

DEFINITION 2.1. *We denote a d-tuple of positive integers as* $\mathbf{i}\langle d\rangle \coloneqq (i_1, \ldots, i_d)$.

These tuples will be used as tensor indices, and each will typically range from 1 to n, so $\mathbf{i}\langle d\rangle \in [1,n]^d$. We concatenate tuples using the notation $\mathbf{i}\langle d\rangle\mathbf{j}\langle f\rangle \coloneqq (i_1, \ldots, i_d, j_1, \ldots, j_f)$.

DEFINITION 2.2. *We refer to the space of increasing d-tuples with values between 1 and n as $\trianglelefteq[1,n]^d$, which means $\forall \mathbf{i}\langle d\rangle \in \trianglelefteq[1,n]^d, i_1 \leq \ldots \leq i_d$. We also refer to the space of strictly increasing tuples as $\triangleleft[1,n]^d$, which means $\forall \mathbf{i}\langle d\rangle \in \triangleleft[1,n]^d, i_1 < \ldots < i_d$.*

The number of increasing d-tuples between 1 and n is given by $|\trianglelefteq[1,n]^d| = \left(\!\binom{n}{d}\!\right) \coloneqq \binom{n+d-1}{d}$. The set of increasing tuples will be useful in our algorithms, as the set of tensor entries with increasing indices enumerates the unique tensor entries.

We define symmetric tensors and symmetrized contractions by considering all possible permutations of their indices. For this task, we introduce the following permutation notation.

DEFINITION 2.3. *Let the set of all possible d-dimensional permutation functions be Π^d, where each $\pi \in \Pi^d$ is associated with a unique bijection $\hat{\pi} : [1,d] \leftrightarrow [1,d]$, as $\pi(\mathbf{i}\langle d\rangle) \coloneqq (i_{\hat{\pi}(1)}, \ldots, i_{\hat{\pi}(d)})$. The number of such functions is $|\Pi^d| = d!$. We denote the collection of all permutations of a tuple $\mathbf{i}\langle d\rangle$ as*

$$\Pi(\mathbf{i}\langle d\rangle) \coloneqq (\pi(\mathbf{i}\langle d\rangle) : \pi \in \Pi^d).$$

We denote tensors in bold font letters, but their elements in regular font. We assume all of their elements are on the same algebraic ring R. We refer to the order of the tensor as the number of indices in the tensor and the dimension as the range of each index. We will generally consider tensors with all dimensions equal to n. Given an order d tensor $\mathbf{A}$, we will refer to its elements using the notation $A_{\mathbf{i}\langle d\rangle} = A_{i_1,\ldots,i_d}$.

DEFINITION 2.4. *We say an n-dimensional order-d tensor $\mathbf{T}$ is symmetric if $\forall \mathbf{i}\langle d\rangle \in [1,n]^d, \pi \in \Pi^d, T_{\mathbf{i}\langle d\rangle} = T_{\pi(\mathbf{i}\langle d\rangle)}$.*

While we will define symmetrization in contractions as summing over all possible permutations of the tensor indices (for any $\mathbf{i}\langle d\rangle$ the collection $\Pi(\mathbf{i}\langle d\rangle)$), our algorithms will exploit the equivalence of many of these permutations. As a result, they will need to sum over a set of partitions rather than a full set of permutations, which we define below.

DEFINITION 2.5. *We define the* **disjoint partition** *$\chi_q^p(\mathbf{k}\langle r\rangle)$ as the collection of all pairs of tuples of size p and q, which are disjoint subcollections of $\mathbf{k}\langle r\rangle$ and preserve the ordering of elements in $\mathbf{k}\langle r\rangle$.*

In other words, if k_i and k_j appear in the same tuple (partition) and $i < j$, then k_i must appear before k_j. For example, the possible ordered partitions of $\mathbf{k}\langle 3\rangle = (k_1, k_2, k_3)$ into pairs of tuples of size one and two are the collection,

$$\chi_2^1(\mathbf{k}\langle 3\rangle) = \big[(k_1, (k_2, k_3)), (k_2, (k_1, k_3)), (k_3, (k_1, k_2))\big].$$

The collection $\chi_q^p(\mathbf{k}\langle r\rangle)$ can be constructed inductively [16].

We denote all possible ordered subcollections via projection function χ^d as $\chi^d(\mathbf{k}\langle d+f\rangle) \coloneqq [\mathbf{i}\langle d\rangle : (\mathbf{i}\langle d\rangle, \mathbf{j}\langle f\rangle) \in \chi_f^d(\mathbf{k}\langle d+f\rangle)]$. When it is implicitly clear what partition is needed, we omit the superscript and subscript from χ completely. We frequently employ

$$\mathbf{i}\langle d\rangle \in \chi(\mathbf{k}\langle d+f\rangle) \coloneqq \mathbf{i}\langle d\rangle \in \chi^d(\mathbf{k}\langle d+f\rangle),$$
$$(\mathbf{i}\langle d\rangle, \mathbf{j}\langle f\rangle) \in \chi(\mathbf{k}\langle d+f\rangle) \coloneqq (\mathbf{i}\langle d\rangle, \mathbf{j}\langle f\rangle) \in \chi_f^d(\mathbf{k}\langle d+f\rangle).$$

DEFINITION 2.6. *For any $s,t,v \geq 0$ with $\omega \coloneqq s+t+v$, we denote a* **tensor contraction** *over v indices between tensor $\mathbf{A}$ of order $s+v$ and tensor $\mathbf{B}$ of order $v+t$, into tensor $\mathbf{C}$ of order $s+t$ each with all dimensions equal to n as*

$$\mathbf{C} = \mathbf{A} \times_v \mathbf{B} \coloneqq \forall \mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle \in [1,n]^{s+t},$$
$$C_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \sum_{\mathbf{k}\langle v\rangle \in [1,n]^v} A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle}. \tag{2.1}$$

Throughout further contraction definitions and algorithms we will always denote $\omega \coloneqq s+t+v$ and assume $n \gg \omega$.

DEFINITION 2.7. *For any $s,t,v \geq 0$, a* **fully symmetric contraction** *is a contraction between symmetric tensors $\mathbf{A}$ and $\mathbf{B}$ into $\mathbf{C}$, where the result is symmetrized, i.e.*

$$\mathbf{C} = \mathbf{A} \otimes_v \mathbf{B} \coloneqq \forall \mathbf{i}\langle s+t\rangle \in [1,n]^{s+t},$$
$$C_{\mathbf{i}\langle s+t\rangle} = \sum_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle \in \Pi(\mathbf{i}\langle s+t\rangle)} \left(\sum_{\mathbf{k}\langle v\rangle \in [1,n]^v} A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle} \right). \tag{2.2}$$

The resulting tensor $\mathbf{C}$ satisfying (2.2) is always symmetric. For $(s=1, t=0, v=1)$, (2.2) corresponds to the product of a symmetric matrix $\mathbf{A}$ with a vector $\mathbf{b}$, $\mathbf{A} \otimes_1 \mathbf{b} \coloneqq \mathbf{A}\mathbf{b}$. For $(s=1, t=1, v=0)$ and commutative "$\cdot$", (2.2) becomes the rank-two vector outer product of a column vector $\mathbf{a}$ and a row vector $\mathbf{b}$, $\mathbf{a} \otimes_0 \mathbf{b} \coloneqq \mathbf{a}\mathbf{b} + \mathbf{b}^\mathsf{T}\mathbf{a}^\mathsf{T}$ (our definition of tensors does not distinguish between row and column vectors, as our definition of contractions permits vectors to behave as either, e.g. $\mathbf{a} \otimes_1 \mathbf{b}$ is the inner product). For $(s=1, t=1, v=1)$ and commutative "$\cdot$", (2.2) becomes symmetrized multiplication of symmetric $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{C} = \mathbf{A} \otimes_1 \mathbf{B} \coloneqq \mathbf{A}\mathbf{B} + \mathbf{B}\mathbf{A}$. Our definition of fully symmetric contractions can be extended to scenarios where the operands and/or the result are partially symmetric, by defining formalism for nested contractions and algorithms [16]. We leave the derivation of communication lower bounds of such nested algorithms for partially symmetric contractions as future work.

3. CONTRACTION ALGORITHMS

We first discuss algorithms for evaluating $\mathbf{A} \times_v \mathbf{B}$ (Definition 2.6) for nonsymmetric $\mathbf{A}$ and $\mathbf{B}$ and $\mathbf{A} \otimes_v \mathbf{B}$ for symmetric $\mathbf{A}$ and $\mathbf{B}$ (Definition 2.7) that directly follow from the algebraic definitions in the previous section. Each algorithm we give is a specification of the scalar multiplications which are computed, but not the order of the summations. Such a specification makes it possible to count the number of operations and yet to also lower bound the communication cost required by any parallel schedule using any summation order to compute the algorithm.

In a few cases, we will perform summations over groups of indices of symmetric tensors, by summing only over the unique values and scaling by the following multiplicative factor.

DEFINITION 3.1. *Let* $\rho(\mathbf{k}\langle v\rangle) := v!/\prod_{i=1}^{l} m_i!$ *where* m_i *is the multiplicity of the ith of* $1 \leq l \leq v$ *unique values in* $\mathbf{k}\langle v\rangle$.

The factor $\rho(\mathbf{k}\langle v\rangle)$ corresponds to the number of unique permutations of $\mathbf{k}\langle v\rangle$, i.e. unique values in the collection $\Pi(\mathbf{k}\langle v\rangle)$.

3.1 Nonsymmetric Contraction Algorithm

We first consider the trivial algorithm which contracts nonsymmetric tensors $\mathbf{A}$ and $\mathbf{B}$ by evaluating (2.1).

ALGORITHM 3.1 ($\mathbf{C} = \Upsilon^{(s,t,v)}(\mathbf{A},\mathbf{B})$). *For any tensor contraction* $\mathbf{C} = \mathbf{A} \times_v \mathbf{B}$ *we define* $\Upsilon^{(s,t,v)}(\mathbf{A},\mathbf{B})$ *to evaluate the multiplications ascribed directly by* (2.1) *in Definition 2.6.*

Algorithm 3.1 is equivalent to a matrix multiplication of a matrix $\bar{\mathbf{A}}$ with dimensions $n^s \times n^v$, where each row corresponds to $\mathbf{j}\langle s\rangle \in [1,n]^s$ and the column corresponds to $\mathbf{k}\langle v\rangle \in [1,n]^v$, by a matrix $\bar{\mathbf{B}}$ with dimensions $n^v \times n^t$, where each row corresponds to $\mathbf{k}\langle v\rangle \in [1,n]^s$ and each column corresponds to $\mathbf{l}\langle t\rangle \in [1,n]^s$, yielding a matrix $\bar{\mathbf{C}}$ with dimensions $n^s \times n^t$. The matrix $\bar{\mathbf{C}}$ contains all elements of the matrix $\mathbf{C} = \mathbf{A} \times_v \mathbf{B}$.

This algorithm precludes algebraic reorganizations of (2.1), such as Strassen's algorithm [18] as it forces the evaluation of scalar multiplications as defined.

3.2 Direct Evaluation Algorithm

The nonsymmetric algorithm may be used to compute fully symmetric contractions with only an additional step of symmetrization of the result of the multiplication between $\mathbf{A}$ and $\mathbf{B}$. However, when $\mathbf{A}$ and $\mathbf{B}$ are symmetric, many of the scalar multiplications in (2.2) are equivalent. The following algorithm evaluates $\mathbf{A} \otimes_v \mathbf{B}$ by computing only the unique multiplications and scaling them appropriately.

In particular, since $\mathbf{C}$ is symmetric, it is no longer necessary to compute all possible orderings of the indices $\mathbf{i}\langle s+t\rangle \in [1,n]^{s+t}$ in $\mathbf{A} \otimes_v \mathbf{B}$, but only those in increasing order $\mathbf{i}\langle s+t\rangle \in \trianglelefteq[1,n]^{s+t}$ as these include all unique values of $\mathbf{C}$. Further, permutations of the $\mathbf{k}\langle v\rangle$ index group result in equivalent scalar multiplications due to symmetry of $\mathbf{A}$ and of $\mathbf{B}$. So, in the following algorithm we rewrite (2.2) to sum over only the ordered sets of these indices and scale them by an appropriate prefactor.

ALGORITHM 3.2 ($\mathbf{C} = \Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$). *For any fully symmetric contraction* $\mathbf{C} = \mathbf{A} \otimes_v \mathbf{B}$ *compute*

$$\mathbf{C} = \Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B}) := \forall \mathbf{i}\langle s+t\rangle \in \trianglelefteq[1,n]^{s+t}, C_{\mathbf{i}\langle s+t\rangle} = s!t! \cdot \sum_{(\mathbf{j}\langle s\rangle,\mathbf{l}\langle t\rangle)\in\chi(\mathbf{i}\langle s+t\rangle)} \left(\sum_{\mathbf{k}\langle v\rangle \in \trianglelefteq[1,n]^v} \rho(\mathbf{k}\langle v\rangle) A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle} \right), \tag{3.1}$$

where $\rho(\mathbf{k}\langle v\rangle)$ *is given in Definition 3.1*

The algorithm $\Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$ is algebraically equivalent to (2.2) and is numerically stable [16].

Modulo the scaling by $s!t!$ and $\rho(\mathbf{k}\langle v\rangle)$, Algorithm 3.1 is equivalent to a matrix multiplication of a matrix $\bar{\mathbf{A}}$ with dimensions $\binom{n}{s} \times \binom{n}{v}$ (where each row corresponds to $\mathbf{j}\langle s\rangle \in \trianglelefteq[1,n]^s$ and the column corresponds to $\mathbf{k}\langle v\rangle \in \trianglelefteq[1,n]^v$) by a matrix $\bar{\mathbf{B}}$ with dimensions $\binom{n}{v} \times \binom{n}{t}$ (where each row corresponds to $\mathbf{k}\langle v\rangle \in \trianglelefteq[1,n]^s$ and each column corresponds to $\mathbf{l}\langle t\rangle \in \trianglelefteq[1,n]^s$), yielding a matrix $\bar{\mathbf{C}}$ with dimensions $\binom{n}{s} \times \binom{n}{t}$. The matrix $\bar{\mathbf{C}}$ can be treated as a partially symmetric tensor of order $s+t$ (symmetric in the permutation within the first s indices and within the last t indices) and can be further symmetrized to obtain $\mathbf{C} = \mathbf{A} \otimes_v \mathbf{B}$,

$$\forall \mathbf{i}\langle s+t\rangle \in \trianglelefteq[1,n]^{s+t}, C_{\mathbf{i}\langle s+t\rangle} = s!t! \sum_{(\mathbf{j}\langle s\rangle,\mathbf{l}\langle t\rangle)\in\chi(\mathbf{i}\langle s+t\rangle)} \bar{C}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}.$$

The scaling by $\rho(\mathbf{k}\langle v\rangle)$ can be applied to the elements of $\bar{\mathbf{A}}$ or $\bar{\mathbf{B}}$ (its preferable to apply the scaling to the tensor which has smaller size).

3.3 Symmetry Preserving Tensor Contraction Algorithm

The symmetry preserving tensor contraction algorithm computes fully symmetric contractions with fewer multiplications and in some cases fewer total operations than the direct evaluation algorithm [16]. It has a number of applications in both matrix computations and high-order coupled cluster [21] tensor contractions. The algorithm requires the computation of a number of intermediate tensors, but for brevity we only give the formula for the highest order tensor computed the algorithm, which suffices for our lower bound derivations.

ALGORITHM 3.3 ($\mathbf{C} = \Phi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$). *For any fully symmetric contraction* $\mathbf{C} = \mathbf{A} \otimes_v \mathbf{B}$*, compute an order* ω *symmetric tensor* $\hat{\mathbf{Z}}$ *whose unique elements are each computed by a single scalar multiplication:* $\forall \mathbf{i}\langle \omega\rangle \in \trianglelefteq[1,n]^{\omega}$*,*

$$\hat{Z}_{\mathbf{i}\langle\omega\rangle} = \left(\sum_{\mathbf{j}\langle s+v\rangle \in \chi(\mathbf{i}\langle\omega\rangle)} A_{\mathbf{j}\langle s+v\rangle} \right) \cdot \left(\sum_{\mathbf{l}\langle v+t\rangle \in \chi(\mathbf{i}\langle\omega\rangle)} B_{\mathbf{l}\langle v+t\rangle} \right). \tag{3.2}$$

Accumulate $\hat{\mathbf{Z}}$ *into a tensor* $\mathbf{Z}$*,* $\forall \mathbf{i}\langle s+t\rangle \in \trianglelefteq[1,n]^{s+t}$*,*

$$Z_{\mathbf{i}\langle s+t\rangle} = \sum_{\mathbf{k}\langle v\rangle \in \trianglelefteq[1,n]^v} \rho(\mathbf{k}\langle v\rangle) \hat{Z}_{\mathbf{i}\langle s+t\rangle \mathbf{k}\langle v\rangle}, \tag{3.3}$$

where $\rho(\mathbf{k}\langle v\rangle)$ *is given in Definition 3.1. Compute* $\mathbf{C} = s!t!(\mathbf{Z} - \mathbf{V} - \mathbf{W})$*, where formulas for computing* $\mathbf{V}$ *and* $\mathbf{W}$ *are given by Solomonik and Demmel [16].*

We note that (3.3) accesses all the elements of $\hat{\mathbf{Z}}$ as these are symmetrically equivalent to the ones computed in (3.2). If we desire to access only the elements with indices in increasing order (the elements computed by (3.2)), we can rewrite (3.3) as a set of accumulations (denoted by $\leftarrow$), which reveals the symmetry of the computation with respect to operations on $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{Z}$,

$$\forall \mathbf{i}\langle\omega\rangle \in \trianglelefteq[1,n]^{\omega}, \mathbf{h}\langle s+t\rangle \in \chi(\mathbf{i}\langle\omega\rangle), Z_{\mathbf{h}\langle s+t\rangle} \leftarrow \hat{Z}_{\mathbf{i}\langle\omega\rangle}. \tag{3.4}$$

The $\mathbf{Z}$ tensor contains all terms needed by $\mathbf{C} = \mathbf{A} \otimes_v \mathbf{B}$ as well as some extra terms which are independently computed as tensor $\mathbf{V}$ and $\mathbf{W}$ then subtracted out. The computation of $\mathbf{V}$ and $\mathbf{W}$ can always be done via a low order number of multiplications, but sometimes requires a constant factor more additions than those needed to compute $\mathbf{Z}$. The correctness proof, numerical stability proof,

numerical tests, computation cost analysis, adaptations from symmetric to partially symmetric, antisymmetric, and Hermitian cases, as well as applications are given by Solomonik and Demmel [16]. The algorithm has no correspondence to a matrix multiplication unlike $\Upsilon^{(s,t,v)}(\mathbf{A},\mathbf{B})$ and $\Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$, which makes its communication cost analysis different and interesting.

4. SEQUENTIAL LOWER BOUNDS

We now derive lower bounds for communication between main memory and cache on a sequential computer. We derive lower bounds for all the nonsymmetric and fully symmetric tensor contraction algorithms given in Section 3. The cache size and vertical (intraprocessor) communication costs in this section are implicitly parameterized by tensor element size (all elements are assumed to be part of the same ring R).

The lower bounds in this and the following sections are based on the generalized Loomis-Whitney inequality [12, 19]. We state it in non-standard form, employing the notation from Section 2 and presenting it as a lower bound on size of the union of projections [14] rather than an upper bound on the size of the order m set.

THEOREM 4.1. *Let V be a set of m-tuples, $V \subset [1,n]^m$, consider the projections given by the projections χ^r,*

$$L = \{\mathbf{w}\langle r\rangle : \mathbf{w}\langle r\rangle \in \chi(\mathbf{v}\langle m\rangle), \mathbf{v}\langle m\rangle \in V\},$$

the size of this set of projections is at least $|L| \geq |V|^{r/m}$.

4.1 Sequential Cost Model

To measure the vertical communication cost on a sequential computer, we consider a cache of size H elements and assume all data starts in main memory. We employ an idealized cache model, i.e. we do not consider track/cache-line size or mechanisms such as cache associativity. We refer to this sequential machine model as $\Gamma(H)$. We do not restrict the size of the main memory of $\Gamma(H)$. We do not allow schedules $\mathcal{S}$ on $\Gamma(H)$ to recompute any element computed by the algorithm (although the algorithm can be defined to compute equivalent elements). We allow reads and writes of data between main memory and cache, both of which have unit communication cost (we do not pay attention to latency cost). We assume none of the inputs of the algorithm reside in cache at the start of execution and that all of the outputs must be written to memory. We denote the sequential communication cost of a schedule executed on $\Gamma(H)$ as Q and provide lower bounds for the cost on Q for a given algorithm by considering all valid schedules of this algorithm.

4.2 Matrix Multiplication

We prove the following theorem which yields a lower bound on the communication cost of matrix multiplication. This lower bound result is not new from an asymptotic stand-point (the asymptotic lower bound was first proven by Jia-Wei and Kung [11]) , but constitutes an improvement of the constant factor on the lower bound with respect to the best bound we are aware of. In particular, the first term in the bound is a factor of 16 higher than the lower bound given by Ballard et al. [2], although our derivation technique is not significantly different.

THEOREM 4.2. *Any load balanced parallel schedule of the classical (non-Strassen-like) matrix multiplication algorithm of m-by-k matrix $\mathbf{A}$ with k-by-n matrix $\mathbf{B}$ into m-by-n matrix C on $\Gamma(H)$ has vertical communication cost,*

$$Q_{\mathrm{MM}}(m,n,k,H) \geq \max\left[\frac{2mnk}{\sqrt{H}}, mk+kn+mn\right].$$

Since this theorem simply improves the constant on a previous result, we give the proof in the Appendix.

4.3 Nonsymmetric Contraction Algorithm

We first consider communication lower bounds for the nonsymmetric contraction algorithm.

THEOREM 4.3. *Any schedule of $\Upsilon^{(s,t,v)}(\mathbf{A},\mathbf{B})$ on $\Gamma(H)$ has vertical communication cost,*

$$Q_{\Upsilon}(n,s,t,v,H) \geq \max\left[\frac{2n^{\omega}}{\sqrt{H}}, n^{s+t}+n^{s+v}+n^{v+t}\right].$$

PROOF. As we explained after the statement of Algorithm 3.1, $\Upsilon^{(s,t,v)}(\mathbf{A},\mathbf{B})$ is equivalent to a matrix multiplication of an $n^s \times n^v$ matrix with an $n^v \times n^t$ matrix yielding a $n^s \times n^t$ matrix. Thus, any matrix multiplication may be computed via the direct evaluation nonsymmetric contraction algorithm, so its cost cannot be lower. Therefore, $Q_{\Upsilon}(n,s,t,v,H) \geq Q_{\mathrm{MM}}(n^s,n^t,n^v,H)$, which yields the lower bound above. □

4.4 Direct Evaluation Algorithm

The bound for the direct evaluation algorithm for fully symmetric tensor contractions is somewhat smaller than the nonsymmetric case, as this algorithm requires product of matrices with smaller dimensions. Further, additional symmetric equivalence between elements exists within these matrices, which makes a lower communication cost possible.

THEOREM 4.4. *Let $q \coloneqq \max\left(\binom{s+v}{s}, \binom{v+t}{t}, \binom{s+t}{s}\right)$, any schedule of $\Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$ on $\Gamma(H)$ has vertical communication cost,*

$$Q_{\Psi}(n,s,t,v,H) \geq \frac{1}{q} Q_{\mathrm{MM}}\left(\left(\binom{n}{s}\right), \left(\binom{n}{t}\right), \left(\binom{n}{v}\right), qH\right),$$

or if we expand the above and tighten the input/output lower bound slightly, $Q_{\Psi}(n,s,t,v,H)$

$$\geq \max\left[\frac{1}{q^{3/2}}\frac{2n^{\omega}}{s!t!v!\sqrt{H}}, \frac{n^{s+t}}{(s+t)!}+\frac{n^{s+v}}{(s+v)!}+\frac{n^{v+t}}{(v+t)!}\right].$$

PROOF. The direct evaluation algorithm corresponds to a matrix multiplication with dimensions $\binom{n}{s}$, $\binom{n}{t}$, and $\binom{n}{v}$. The lower bound for reading in the inputs from memory to cache and writing the output from cache to memory is

$$Q_{\Psi}(n,s,t,v,H) \geq \frac{n^{s+t}}{(s+t)!}+\frac{n^{s+v}}{(s+v)!}+\frac{n^{v+t}}{(v+t)!},$$

which is lower than the corresponding lower bound for nonsymmetric matrix multiplication, because the tensors have less unique entries than the corresponding matrices (the largest tensor by a factor of q).

Consider any schedule $\mathcal{S}$ for the direct evaluation symmetric contraction algorithm. We ignore the scaling by $s!t!$ and by $\rho(\mathbf{k}\langle v\rangle)$ done by the algorithm, and we also ignore the accumulation of equivalent terms into entries of $\mathbf{C}$ with equivalent indices, by replacing the collection $\chi_t^s(\mathbf{i}\langle s+t\rangle)$ with a set $\bar{\chi}_t^s(\mathbf{i}\langle s+t\rangle)$ defined the same way except now without repeated entries. Ignoring part of the computation still allows us to obtain a communication lower bound based on the cost of performing the rest of the computation. The leftover operations in $\mathcal{S}$ compute $\forall \mathbf{i}\langle s+t\rangle \in \trianglelefteq[1,n]^{s+t}$,

$$C_{\mathbf{i}\langle s+t\rangle} = \sum_{(\mathbf{j}\langle s\rangle,\mathbf{l}\langle t\rangle)\in\bar{\chi}(\mathbf{i}\langle s+t\rangle)} \left(\sum_{\mathbf{k}\langle v\rangle\in\trianglelefteq[1,n]^v} A_{\mathbf{j}\langle s\rangle\mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle\mathbf{l}\langle t\rangle}\right),$$

where $\mathbf{A}$ and $\mathbf{B}$ are symmetric. We now show that the existence of $\mathcal{S}$ implies the existence of a schedule $\hat{\mathcal{S}}$, which computes $\forall \mathbf{j}\langle s\rangle \in \trianglelefteq[1,n]^s, \mathbf{l}\langle t\rangle \in \trianglelefteq[1,n]^t$,

$$\hat{C}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \sum_{\mathbf{k}\langle v\rangle \in \trianglelefteq[1,n]^v} \hat{A}_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot \hat{B}_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle},$$

for any $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ that are partially symmetric ($\hat{\mathbf{A}}$ in the first s and the last v indices and $\hat{\mathbf{B}}$ in the first v and last t indices). We will show that $\hat{\mathcal{S}}$ requires a factor of q larger cache size and communication cost than $\mathcal{S}$. Then due to the fact that a the partially symmetric contraction above can be used to compute a matrix multiplication, we will obtain the communication lower bound stated in the theorem.

The construction of $\hat{\mathcal{S}}$ from $\mathcal{S}$ works by associating a set of up $\binom{s+v}{s}$ elements of $\hat{\mathbf{A}}$ with each element of $\mathbf{A}$, $\forall \mathbf{i_1}\langle s+v\rangle \in \trianglelefteq[1,n]^{s+v}$,

$$\mathring{A}_{\mathbf{i_1}\langle s+v\rangle} = \{\hat{A}_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} : (\mathbf{j}\langle s\rangle, \mathbf{k}\langle v\rangle) \in \bar{\chi}(\mathbf{i_1}\langle s+v\rangle)\},$$

and a set of up to $\binom{v+t}{v}$ elements of $\hat{\mathbf{B}}$ with each element of $\mathbf{B}$, $\forall \mathbf{i_2}\langle v+t\rangle \in \trianglelefteq[1,n]^{v+t}$,

$$\mathring{B}_{\mathbf{i_2}\langle v+t\rangle} = \{\hat{B}_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle} : (\mathbf{k}\langle v\rangle, \mathbf{l}\langle t\rangle) \in \bar{\chi}(\mathbf{i_2}\langle v+t\rangle)\}.$$

Every result of scalar multiplication and every partial sum of these must contribute to a unique element of $\mathbf{C}$, i.e. $C_{\mathbf{i}\langle s+t\rangle}$ for some $\mathbf{i}\langle s+t\rangle$. Now, in $\mathcal{S}$, each $C_{\mathbf{i}\langle s+t\rangle}$ is computed as a summation of $\left(\!\binom{n}{v}\!\right)$ multiplications, via $\left(\!\binom{n}{v}\!\right) - 1$ addition operations. Thus including $C_{\mathbf{i}\langle s+t\rangle}$ there are a total of $2\left(\!\binom{n}{v}\!\right)$ partial sums in $\mathcal{S}$ associated with each $C_{\mathbf{i}\langle s+t\rangle}$. We enumerate these as $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$ for $g \in [1, 2\left(\!\binom{n}{v}\!\right)]$. Further, any addition operation in $\mathcal{S}$ can be written for some $g, f_1, f_2 \in [1, 2\left(\!\binom{n}{v}\!\right)]$ and some $\mathbf{i}\langle s+t\rangle \in \trianglelefteq[1,n]^{s+t}$ as

$$C^{(g)}_{\mathbf{i}\langle s+t\rangle} = C^{(f_1)}_{\mathbf{i}\langle s+t\rangle} + C^{(f_2)}_{\mathbf{i}\langle s+t\rangle}.$$

Every multiplication in $\mathcal{S}$ may be represented as

$$C^{(g)}_{\mathbf{i}\langle s+t\rangle} = A_{\mathbf{i_1}\langle s+v\rangle} \cdot B_{\mathbf{i_2}\langle v+t\rangle},$$

for some $\mathbf{i_1}\langle s+v\rangle \in \trianglelefteq[1,n]^{s+v}, \mathbf{i_2}\langle v+t\rangle \in \trianglelefteq[1,n]^{v+t}, g \in [1, 2\left(\!\binom{n}{v}\!\right)]$. For each partial sum $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$, we now define a set of $\binom{s+t}{s}$ partial sums in $\hat{\mathcal{S}}$,

$$\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle} = \{\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} : (\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in \bar{\chi}(\mathbf{i}\langle s+t\rangle)\}.$$

For every scalar multiplication in $\mathcal{S}$, $C^{(g)}_{\mathbf{i}\langle s+t\rangle} = A_{\mathbf{i_1}\langle s+v\rangle} \cdot B_{\mathbf{i_2}\langle v+t\rangle}$, there exist unique $\mathbf{j}\langle s\rangle$, $\mathbf{k}\langle v\rangle$, and $\mathbf{l}\langle t$ such that $(\mathbf{j}\langle s\rangle, \mathbf{k}\langle v\rangle) \in \bar{\chi}(\mathbf{i_1}\langle s+v\rangle)$ and $(\mathbf{k}\langle v\rangle, \mathbf{l}\langle t\rangle) \in \bar{\chi}(\mathbf{i_2}\langle v+t\rangle)$. For each such multiplication, schedule $\hat{\mathcal{S}}$ computes $\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \hat{A}_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot \hat{B}_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle}$ and sets all the other elements of the set to which the result belongs, $\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle}$, to zero. For every addition $C^{(g)}_{\mathbf{i}\langle s+t\rangle} = C^{(f_1)}_{\mathbf{i}\langle s+t\rangle} + C^{(f_2)}_{\mathbf{i}\langle s+t\rangle}$, schedule $\hat{S}$ computes $\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle}$ via the additions:

$$\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \hat{C}^{(f_1)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} + \hat{C}^{(f_2)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}$$

for all elements in $\mathring{C}^{(f_1)}_{\mathbf{i}\langle s+t\rangle}$ and $\mathring{C}^{(f_2)}_{\mathbf{i}\langle s+t\rangle}$, respectively. Schedule $\hat{\mathcal{S}}$ manages memory to cache traffic by mimicking $\mathcal{S}$. Whenever an element of the form $A_{\mathbf{i_1}\langle s+v\rangle}$ or $B_{\mathbf{i_2}\langle v+t\rangle}$ or $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$ is read from memory to cache, $\hat{\mathcal{S}}$ reads one of the sets $\mathring{A}_{\mathbf{i_1}\langle s+v\rangle}$ or $\mathring{B}_{\mathbf{i_2}\langle v+t\rangle}$ or $\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle}$, respectively. Writes from memory to cache of $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$ are similarly mimicked by writes of the set $\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle}$.

We prove correctness of schedule $\hat{\mathcal{S}}$ by demonstrating that if an element $C^{(g)}_{\mathbf{i}\langle s+t\rangle} = C_{\mathbf{i}\langle s+t\rangle}$ then

$$\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle} = \{\hat{C}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} : (\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in \bar{\chi}(\mathbf{i}\langle s+t\rangle)\},$$

and by arguing that all operands necessary are always present. It is easy to see that any operands to a multiplication or addition will be present. For each addition with an operand $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$ in $\mathcal{S}$, an addition is done with members of $\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle}$ in $\hat{\mathcal{S}}$ and the communication of this set mimics that of $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$. Each multiplication in $\mathcal{S}$ is also mimicked by a multiplications of $\hat{\mathcal{S}}$, as well as the communication of the operands.

We can represent each partial sum as

$$C^{(g)}_{\mathbf{i}\langle s+t\rangle} = \sum_{(\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in P} \left(\sum_{\mathbf{k}\langle v\rangle \in N_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}} A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle} \right),$$

where $P \subset \bar{\chi}^s_t(\mathbf{i}\langle s+t\rangle)$ and each $N_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} \subset \trianglelefteq[1,n]^v$. For each such $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$, we show by induction that $\forall (\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in P$,

$$\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \sum_{\mathbf{k}\langle v\rangle \in N_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}} A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle}.$$

Each $C^{(g)}_{\mathbf{i}\langle s+t\rangle}$ is either a result of a multiplication (a leaf node in the summation tree) or a summation of two other such partial sums (an internal node in the summation tree). For leaf nodes that are a result of a multiplication, P contains a single element $(\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle)$ for some $\mathbf{j}\langle s\rangle \in \trianglelefteq[1,n]^s$ and $\mathbf{l}\langle t\rangle \in \trianglelefteq[1,n]^v$, and there is a single $\mathbf{k}\langle v\rangle \in N_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}$. This exact multiplication is performed in $\mathcal{S}$ to form $\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}$. For any internal node that is a result of a summation $C^{(g)}_{\mathbf{i}\langle s+t\rangle} = C^{(f_1)}_{\mathbf{i}\langle s+t\rangle} + C^{(f_2)}_{\mathbf{i}\langle s+t\rangle}$, we can assert by induction that the desired property holds for $\mathring{C}^{(f_1)}_{\mathbf{i}\langle s+t\rangle}$ (with P^{f_1} and N^{f_1}) as well as for $\mathring{C}^{(f_2)}_{\mathbf{i}\langle s+t\rangle}$ (with P^{f_2} and N^{f_2}). For each $(\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in P^{f_1} \cap P^{f_2}$, the elements are summed by $\mathcal{S}$, so we obtain the appropriate union,

$$\begin{aligned}\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} &= \hat{C}^{(f_1)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} + \hat{C}^{(f_2)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} \\ &= \sum_{\mathbf{k}\langle v\rangle \in \left(N^{f_1}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} \cup N^{f_2}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}\right)} A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle}.\end{aligned}$$

Thus for each $(\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in P^{f_1} \cap P^{f_2}$, we have $N^g_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = N^{f_1}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} \cup N^{f_2}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}$. It is also easy to see that the property is preserved for $(\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in P^{f_1} \setminus P^{f_2}$, where each $N^g_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}$ is equal to $N^{f_1}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle}$, since the elements are added to zero and similarly for $P^{f_2} \setminus P^{f_2}$. Thus, we have shown the property holds with the appropriate $P^g = P^{f_1} \cup P^{f_2}$ and with N^g as defined above. Whenever $C^{(g)}_{\mathbf{i}\langle s+t\rangle} = C_{\mathbf{i}\langle s+t\rangle}$, this property implies that $P = \bar{\chi}^s_t(\mathbf{i}\langle s+t\rangle)$ and also that $\forall (\mathbf{j}\langle s\rangle, \mathbf{l}\langle t\rangle) \in \bar{\chi}(\mathbf{i}\langle s+t\rangle)$, $N_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \trianglelefteq[1,n]^v$,

$$\hat{C}^{(g)}_{\mathbf{j}\langle s\rangle \mathbf{l}\langle t\rangle} = \sum_{\mathbf{k}\langle v\rangle \in \trianglelefteq[1,n]^v} A_{\mathbf{j}\langle s\rangle \mathbf{k}\langle v\rangle} \cdot B_{\mathbf{k}\langle v\rangle \mathbf{l}\langle t\rangle}.$$

The computation cost of the schedule $\mathcal{S}$ is minimal (for the partially symmetric contraction) so long as we do not actually compute additions of elements with zero. The communication cost of the schedule $\hat{\mathcal{S}}$ is a factor of q greater than that of $\mathcal{S}$, since the three factors in the max defining q correspond to the number of elements in each set: $\mathring{A}_{\mathbf{i}\langle s+v\rangle}$, $\mathring{B}_{\mathbf{i}\langle v+t\rangle}$, and $\mathring{C}^{(g)}_{\mathbf{i}\langle s+t\rangle}$, respectively, which are communicated whenever a corresponding element of $\mathbf{A}$,

$\mathbf{B}$, or a partial sum of an element of $\mathbf{C}$ is. Similarly, the memory consumption of $\mathcal{S}$ is also greater by a factor of q.

Now, the tensor contraction,

$$\hat{C}_{\mathbf{j}\langle s\rangle\mathbf{l}\langle t\rangle} = \sum_{\mathbf{k}\langle v\rangle\in\trianglelefteq[1,n]^v} \hat{A}_{\mathbf{j}\langle s\rangle\mathbf{k}\langle v\rangle}\cdot\hat{B}_{\mathbf{k}\langle v\rangle\mathbf{l}\langle t\rangle},$$

is equivalent to a matrix multiplication of tensors with dimensions $\binom{\binom{n}{s}}{} \times \binom{\binom{n}{v}}{}$ by $\binom{\binom{n}{v}}{} \times \binom{\binom{n}{t}}{}$ by the same argument as given after the definition of Algorithm 3.2. Since $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ can be arbitrary partially symmetric tensors and have the same number of unique entries as these matrices, this partially symmetric contraction can be used to compute a matrix multiplication, and therefore cannot cost more than such a matrix multiplication,

$$Q_\Psi(n,s,t,v,H) \geq \frac{1}{q} Q_{\mathrm{MM}}\left(\binom{\binom{n}{s}}{}, \binom{\binom{n}{t}}{}, \binom{\binom{n}{v}}{}, qH\right).$$

Combining the above lower bound, with the trivial lower bound from reading the inputs and writing outputs, we obtain the bounds stated in the theorem. □

4.5 Symmetry Preserving Algorithm

We now derive a communication lower bound for Algorithm 3.3 ($\Phi_\otimes^{(s,t,v)}(\mathbf{A},\mathbf{B})$). Our proof assumes that partial sums are not reused to compute different operand elements in (3.2) and that partial sums are also not reused to compute contributions of $\hat{\mathbf{Z}}$ to different elements of $\mathbf{Z}$ in (3.3). The reuse of such partial sums would change the number of operations as well as the dependency graph of the algorithm. Further, such partial sums cannot help when $s = 0$ or $t = 0$ or $v = 0$, since in these cases the lower bound is dominated by reading in the inputs or writing the outputs. Additionally partial sums do not help when $\min(s,t,v) = 1$, since the operands/results of the larger tensor in (3.2) (the only one the proof considers) may share only one summand/target in the summation with any other operand/result, so no shared partial sums exist in these cases. Therefore, this assumption is only significant when $s,t,v \geq 2$, but we are not aware of applications for such cases.

THEOREM 4.5. *Let $\kappa \coloneqq \max(s+v, v+t, s+t)$. Any schedule of $\Phi_\otimes^{(s,t,v)}(\mathbf{A},\mathbf{B})$ on $\Gamma(H)$ has vertical communication cost,*

$$Q_\Phi(n,s,t,v,H) \geq \max\left(\left\lfloor \binom{\binom{n}{\omega}}{} / (2H)^{\omega/\kappa} \right\rfloor H, \frac{n^{s+t}}{(s+t)!} + \frac{n^{s+v}}{(s+v)!} + \frac{n^{v+t}}{(v+t)!}\right),$$

under the assumption that partial sums are not reused to compute different operand elements in (3.2) *and that partial sums are also not reused to compute contributions of $\hat{\mathbf{Z}}$ to different elements of $\mathbf{Z}$ in* (3.3)*.*

PROOF. A previous proof of these communication lower bound on algorithm $\Phi_\otimes^{(s,t,v)}(\mathbf{A},\mathbf{B})$ appeared in the dissertation of the first author [14]. It employed the lower bound technique from Christ et al. [6], which relies on the Hölder inequality [9] and its generalization [5]. More specifically, it employed Theorem 6.6 from Section 6.3 of Christ et al. [6], which applies to programs which are loop nests where in the innermost loop arrays are accessed based on subsets of the loop indices. In this version, we simplify the proof by employing generalized Loomis-Whitney [19] (Theorem 4.1).

Any sequential schedule $\mathcal{S}$ must compute $\binom{\binom{n}{\omega}}{} = |\trianglelefteq[1,n]^\omega|$ multiplications involved in forming $\hat{\mathbf{Z}}$ in $\Phi_\otimes^{(s,t,v)}(\mathbf{A},\mathbf{B})$. We subdivide the schedule into $f+1$ intervals, such that during each interval $\mathcal{S}_i$ for $i \in [1,f]$ a set of $\hat{\mathbf{Z}}$ elements is computed with indices in $E_i \subset \trianglelefteq[1,n]^\omega$ such that $|E_i| = (2H)^{\omega/\kappa}$, where $\kappa \coloneqq \max(s+v, v+t, s+t)$. The last remainder interval $\mathcal{S}_{f+1}$ computes the rest of the elements of $\hat{\mathbf{Z}}$ with indices E_{f+1},

$$|E_{f+1}| = \left(\binom{\binom{n}{\omega}}{}\right) - (2H)^{\omega/\kappa}\left\lfloor \left(\binom{\binom{n}{\omega}}{}\right) / (2H)^{\omega/\kappa}\right\rfloor \leq (2H)^{\omega/\kappa}.$$

We obtain a lower bound on the number of reads from cache and writes to cache done by each S_i.

Reads: For each $\mathbf{i}\langle\omega\rangle \in E_i$, an element $\hat{Z}_{\mathbf{i}\langle\omega\rangle}$ is computed via (3.2), which requires the operands $A_{\mathbf{j}\langle s+v\rangle}$ for all $\mathbf{j}\langle s+v\rangle \in L_i^{\mathrm{in}}$, where

$$L_i^{\mathrm{in}} = \{\mathbf{j}\langle s+v\rangle : \mathbf{j}\langle s+v\rangle \in \chi(\mathbf{i}\langle\omega\rangle), \mathbf{i}\langle\omega\rangle \in E_i\}.$$

By Theorem 4.1 with $r = s+v$ and $m = \omega$, the size of this set of dependencies is at least $|L_i^{\mathrm{in}}| \geq |E_i|^{(s+v)/\omega}$. The elements of E_i can also be computed from partial sums, but by assumption, each would require a unique partial sum, so a set of dependencies with partial sums replacing corresponding elements of $\mathbf{A}$ would be of size only greater than $|E_i|^{(s+v)/\omega}$. The argument above can also be made for $\mathbf{B}$ operands to (3.2), so we assert there is a set of dependencies with indices L_i^{in}, such that $|L_i^{\mathrm{in}}| \geq |E_i|^{\max(s+v,v+t)/\omega}$. Now, at the start of the computation of interval $\mathcal{S}_i$, up to H elements with indices $R_i \subset L_i$ may be in cache, the rest of the elements with indices $\breve{L}_i^{\mathrm{in}} = L_i^{\mathrm{in}} \setminus R_i$, must be read from memory during interval $\mathcal{S}_i$. The number of elements read, $Q_i^{\mathrm{in}} = |\breve{L}_i^{\mathrm{in}}|$ is

$$Q_i^{\mathrm{in}} \geq |L_i^{\mathrm{in}}| - H \geq |E_i|^{\max(s+v,v+t)/\omega} - H.$$

Writes: The scalar multiplications corresponding to the set of tuples E_i compute a set of elements of $\hat{\mathbf{Z}}$. Each of these elements $\hat{Z}_{\mathbf{i}\langle\omega\rangle}$ contributes to the summation computing $Z_{\mathbf{h}\langle s+t\rangle}$, for all $\mathbf{h}\langle s+t\rangle \in \chi(\mathbf{i}\langle\omega\rangle)$ in (3.3) (as shown by the reformulation in (3.4)). Since we assume no recomputation, the schedule interval $\mathcal{S}_i$ may only discard the element $\hat{Z}_{\mathbf{i}\langle\omega\rangle}$ if after the interval E_i, it has been accumulated into a partial sum for each such $Z_{\mathbf{h}\langle s+t\rangle}$ (we have also assumed these partial sums cannot be reused for different elements of $\mathbf{Z}$). We now obtain a bound on the set of outputs of interval $\mathcal{S}_i$, L_i^{out}. We assume without loss of generality that all elements of $\mathbf{Z}$ computed in $\mathcal{S}_i$ are accumulated to partial sums $\bar{Z}_{\mathbf{h}\langle s+t\rangle}$ for all $\mathbf{h}\langle s+t\rangle \in L_i^{\mathrm{out}}$ where

$$L_i^{\mathrm{out}} = \{\mathbf{h}\langle s+t\rangle : \mathbf{h}\langle s+t\rangle \in \chi(\mathbf{i}\langle\omega\rangle), \mathbf{i}\langle\omega\rangle \in E_i\},$$

otherwise the number of outputs is only greater, as each $Z_{\mathbf{i}\langle\omega\rangle}$ that is not accumulated in this way, would need to be an output. Therefore, by Theorem 4.1 with $r = s+t$ and $m = \omega$, the size of this set of outputs is at least $|L_i^{\mathrm{out}}| \geq |E_i|^{(s+t)/\omega}$. Now, at the end of the interval $\mathcal{S}_i$, up to H of these partial sums with indices $R_i \subset L_i$ may remain cache, the rest of the partial sums with indices $\hat{L}_i^{\mathrm{out}} = L_i^{\mathrm{out}} \setminus R_i$ must be written to memory during $\mathcal{S}_i$. The number of such elements written, $Q_i^{\mathrm{out}} = |\hat{L}_i^{\mathrm{out}}|$ is

$$Q_i^{\mathrm{out}} \geq |L_i^{\mathrm{out}}| - H \geq |E_i|^{\max(s+t)/\omega} - H.$$

Combining the reads and writes, we conclude that for any interval $\mathcal{S}_i$, $Q_i = \max(Q_i^{\mathrm{in}}, Q_i^{\mathrm{out}})$ elements need to be communicated between memory and cache,

$$Q_i \geq |E_i|^{\max(s+v,v+t,s+t)/\omega} - H = |E_i|^{\kappa/\omega} - H.$$

For $i \in [1,f]$, we can plug in $|E_i| = (2H)^{\omega/\kappa}$ to obtain

$$Q_i \geq H.$$

We can then obtain a lower bound on the communication cost for $\Phi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$ by summing over the communication costs of the first f schedule intervals,

$$Q_{\Phi}(n,s,t,v,H) \geq fH = \left\lfloor \left(\binom{n}{\omega}\right) / (2H)^{\omega/\kappa} \right\rfloor H.$$

We augment this lower bound, by combining it with a lower bound that is based purely on the size of $\mathbf{A}$ and $\mathbf{B}$, each of whose entries must be read into cache at least once and $\mathbf{C}$, whose entries must be written to memory at least once, obtaining the lower bound stated in the theorem. □

We observe that communication cost of the symmetry preserving algorithm achieves asymptotically less reuse than the standard algorithm when $s,t,v > 0$ and s,t,v are unequal, and so $\omega < (3/2)\kappa$. In these cases the exponent on H corresponding to the cache reuse obtained for each element by the symmetry preserving algorithm is $H^{(\omega/\kappa)-1} < H^{1/2}$ (the standard algorithm gets $H^{1/2}$ reuse in these case). In the cases when one of s,t,v is zero, reading in one of the input tensors or writing the output tensor becomes the asymptotically dominant cost for both algorithms. When $s = t = v > 0$, $\Phi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$ achieves the same asymptotic reuse factor of $H^{1/2}$.

Further, when the algorithm is applied in nested form for partially symmetric contractions [16], each scalar multiplication done by the symmetry preserving algorithm can be asymptotically more expensive than each scalar addition (e.g. a matrix multiplication). In such cases, the reduction in the number of multiplications by $\omega!/(s!t!v!)$ could yield a reduction in communication cost by the same factor, since the underlying scalar operation (e.g. matrix multiplication) may achieve a factor of $H^{1/2}$ reuse itself. The communication cost analysis done in this section does not correctly capture the cost in such cases, since it assumes elements of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are all of unit size.

5. PARALLEL LOWER BOUNDS

We now consider on a parallel computer with a fully connected network. Our lower bounds assume that the input, output, and computation are all well load balanced. The memory size and horizontal (interprocessor) communication costs in this section are implicitly parameterized by tensor element size (all elements are assumed to be part of some Abelian group R).

5.1 Parallel Cost Model

In our parallel cost model, we consider a homogeneous parallel computer $\Lambda(p,M)$ with p processors, each capable of storing M elements (i.e. with M memory). We assume that each element of the input to the algorithm exists on a unique processor at the start of the execution of any parallel schedule and that the parallel schedule does not compute any element twice (no recomputation). We allow all processors to communicate with each other (fully connected network) on $\Lambda(p,M)$ via point-to-point messages. We measure the parallel horizontal communication cost W of a schedule on $\Lambda(p,M)$ as the largest number of elements sent and received by any processor throughout the execution of the parallel schedule. This simplistic horizontal communication cost measurement does not consider dependent sequences of executions (i.e. idle time), however, the algorithms we consider have constant depth (short critical path length) and by deriving lower bounds on this basic cost measurement we also obtain lower bounds on the cost of the algorithm in other models. In particular, by obtaining lower bounds on the amount of communication done by any processor for any schedule of an algorithm, we obtain lower bounds on horizontal communication bandwidth cost for LogP [7], LogGP [1], BSP [20], and the α–β critical path cost model [15], in all of which the communication cost of the algorithm is at least the communication cost incurred by any given processor.

We assume that any parallel schedule for an algorithm must be load balanced, i.e. if a tensor has m unique elements, each processor owns $\Theta(m/p)$ elements at the start of execution and no elements are replicated (no unique input entries are stored on multiple processors initially). We assume the outputs are balanced in the same way. However, our lower bounds apply to any possible load balanced initial input and final output redistributions. All of our lower bounds on W treat n and p as asymptotic parameters and assume s,t,v are constants.

5.2 Matrix Multiplication

THEOREM 5.1. *Any load balanced schedule of the classical (non-Strassen-like) matrix multiplication algorithm of m-by-k matrix $\mathbf{A}$ with k-by-n matrix $\mathbf{B}$ into m-by-n matrix C on $\Lambda(p,M)$ has horizontal communication cost, $W_{\mathrm{MM}}(m,n,k,p,M)$*

$$= \Omega\left(W_{\mathrm{O}}(\min(m,n,k), \mathrm{median}(m,n,k), \max(m,n,k), p, M)\right),$$

where

$$W_{\mathrm{O}}(x,y,z,p,M) = \begin{cases} \frac{xyz}{p\sqrt{M}} + \left(\frac{xyz}{p}\right)^{2/3} & : p > yz/x^2. \\ x\left(\frac{yz}{p}\right)^{1/2} & : yz/x^2 \geq p > z/y. \\ xy & : z/y \geq p. \end{cases}$$

Theorem 5.1 was proven by Demmel et al. [8].

5.3 Nonsymmetric Contraction Algorithm

We start with a parallel horizontal communication lower bound for nonsymmetric contractions, which is just that of a matrix multiplication.

THEOREM 5.2. *Any load balanced schedule of $\Upsilon^{(s,t,v)}(\mathbf{A},\mathbf{B})$ on $\Lambda(p,M)$ has horizontal communication cost,*

$$W_{\Upsilon}(n,s,t,v,p,M) = \Omega\left(W_{\mathrm{MM}}\left(n^s, n^t, n^v, p, M\right)\right).$$

PROOF. By the same argument as in the proof of Theorem 4.3, this algorithm can be used to perform a matrix multiplication with dimensions n^s, n^t, and n^v, hence the bound stated in the theorem. □

5.4 Direct Evaluation Algorithm

As in the sequential bounds, a parallel horizontal communication lower bound of the direct evaluation algorithm is again obtained by reduction from matrix multiplication. In this case, the reduction can cause some computation imbalance, but only by a constant factor as shown in the proof.

However, in the parallel case we derive an additional horizontal communication lower bound for cases when exactly one of s,t,v is zero. The additional bound is stronger than the bound obtained via the reduction from matrix multiplication when s,t,v are all unequal (and one is zero). In the sequential scenario, the communication cost of such cases is dominated by reading the inputs from memory to cache or writing the output to memory. In the parallel scenario, the largest tensor can be kept in place, so communication of the second-largest tensor must be considered. When $v = 0$, the new communication lower bound then arises as a consequence of

symmetrization needed to compute $\mathbf{C}$. When $s = 0$ or $t = 0$, this bound is a consequence of the assumption that each unique elements of the largest the symmetric operand is stored on a unique processor. This assumption corresponds to a 'packed' distributed layout [17]. Therefore, using an 'unpacked' layout where the tensor operands are stored as if they were nonsymmetric. would make it possible to have asymptotically lower communication costs in cases when either s or t is zero and $v \neq s$ as well as $v \neq t$.

THEOREM 5.3. *Any load balanced schedule of* $\Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$ *on* $\Lambda(p, M)$ *has horizontal communication cost,* $W_\Psi(n,s,t,v,p,M)$

$$= \Omega\left(W_{\text{MM}}\left(\left(\!\binom{n}{s}\!\right), \left(\!\binom{n}{t}\!\right), \left(\!\binom{n}{v}\!\right), p, qM\right)\right),$$

where $q = \max\left(\binom{s+v}{s}, \binom{v+t}{t}, \binom{s+t}{s}\right)$. *Further, when exactly one of* s, t, v *is zero,*

$$W_\Psi(n, s, t, v, p, M) = \Omega\left((n^\omega/p)^{\max(s,t,v)/\omega}\right).$$

PROOF. Given any parallel schedule $\mathcal{Q}$ for $\Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$, we can construct a schedule $\hat{\mathcal{Q}}$ that transforms $\mathcal{Q}$ to perform a partially symmetric contraction in almost the exact same fashion. The only difference is that the horizontal communication rather than vertical communication needs to be mimicked by $\hat{\mathcal{Q}}$, which again satisfies all the dependencies. Further, each addition,

$$C^{(g)}_{\mathbf{i}\langle s+t\rangle} = C^{(f_1)}_{\mathbf{i}\langle s+t\rangle} + C^{(f_2)}_{\mathbf{i}\langle s+t\rangle},$$

in $\mathcal{Q}$ yields up to $\binom{s+t}{s}$ additions between elements of the sets $\hat{C}^{(f_1)}_{\mathbf{i}\langle s+t\rangle}$ and $\hat{C}^{(f_2)}_{\mathbf{i}\langle s+t\rangle}$ in $\hat{\mathcal{Q}}$. In the sequential case, the total amount of work done by $\hat{\mathcal{Q}}$ was nevertheless minimal, so this did not matter, however, in the parallel case it may make the work imbalanced. Nevertheless, the imbalance is a factor of at most $\binom{s+t}{s}$, which we assume is a constant, so the resulting schedule $\hat{\mathcal{Q}}$ is still asymptotically load balanced.

Therefore, the communication cost of $\mathcal{Q}$ cannot be less than a factor of $1/q$ multiplied by the cost of a matrix multiplication of matrices with dimensions $\left(\!\binom{n}{s}\!\right) \times \left(\!\binom{n}{v}\!\right)$ and $\left(\!\binom{n}{v}\!\right) \times \left(\!\binom{n}{t}\!\right)$ and with a factor of q more memory, as the theorem states.

We now prove the second bound, which applies when exactly one of $s, t, v = 0$. Consider all $\bar{F} = \left(\!\binom{n}{s}\!\right)\left(\!\binom{n}{t}\!\right)\left(\!\binom{n}{v}\!\right)$ multiplications in (3.1), some processor p_i must perform at least $F = \bar{F}/p \geq \frac{n^\omega}{s!t!v!p}$ such multiplication operations.

Without loss of generality assume $v = 0$ and $s \geq t$. The other cases may be proven similarly as we outline below. Consider the largest tensor, in this case $\mathbf{C}$, each entry of $\mathbf{C}$ requires the addition of $\frac{(s+t)!}{s!t!}$ results of multiplications. Now, one of the following two cases holds,

1. at least $F/2$ of the multiplications done by processor p_i contribute to entries of $\mathbf{C}$ which a processor other than p_i outputs: these must contribute to at least $\frac{s!t!}{(s+t)!}F/2$ partial summations to entries of $\mathbf{C}$, so at least $\frac{s!t!}{(s+t)!}F/2 = \Omega(n^\omega/p)$, communication is necessary, which is stronger than the desired bound.

2. at least $F/2$ of the multiplications done by processor p_i contribute to the entries of $\mathbf{C}$ which p_i outputs: there are at least $K \coloneqq \frac{s!t!}{(s+t)!}F/2 = \Omega(n^\omega/p)$ such entries.

Since in the first case the desired bound is obtained immediately, we now focus on the same case, where K of the multiplications processor p_i computes contribute to entries of $\mathbf{C}$ which p_i outputs. We again consider two cases, one of which must hold,

1. for at least $K/2$ entries of $\mathbf{C}$ which p_i outputs, p_i receives at least one partial sum (a single or more accumulated multiplications) from some other processor: the processor must receive at least $K/2 = \Omega(n^\omega/p)$ entries/partial-sums, which is stronger than the desired bound.

2. for at least $K/2$ entries of $\mathbf{C}$ which p_i outputs, p_i computes all the multiplications.

Again, the first case yields the bound immediately, therefore, we focus on the second case, where p_i computes all multiplications necessary for $K/2 = \Omega(n^\omega/p)$ entries of $\mathbf{C}$. Let this set of multiplications correspond to all entries $C_{\mathbf{i}\langle s+t\rangle}$ for $\mathbf{i}\langle s+t\rangle \in V \subset \trianglelefteq[1,n]^{s+t}$. Computation of the $\mathbf{C}$ entries given by this index set require the operands $\mathbf{A}_{\mathbf{j}\langle s\rangle}$ for all $\mathbf{j}\langle s\rangle \in L_{\text{A}}$, where

$$L_{\text{A}} = \{\mathbf{j}\langle s\rangle : \mathbf{j}\langle s\rangle \in \chi(\mathbf{i}\langle s+t\rangle), \mathbf{i}\langle s+t\rangle \in V\}$$

(keeping in mind that $\mathbf{k}\langle v\rangle$ does not exist since $v = 0$). We obtain a lower bound on the number of $\mathbf{A}$ entries $|L_{\text{A}}|$ needed to perform these multiplications by application of Theorem 4.1 with $m = \omega = s + t$ and $r = s$, in particular,

$$|L_{\text{A}}| \leq |V|^{s/\omega} = \Omega\left((n^\omega/p)^{s/\omega}\right).$$

This is asymptotically greater than the $\Theta(n^s/p)$ of entries of $\mathbf{A}$ any processor can own by the load balanced schedule assumption, so most of these entries must be obtained by communication, yielding the communication lower bound for the case $v = 0$ and $s \geq t$,

$$W_\Psi(n, s, t, v, p, M) = \Omega\left((n^\omega/p)^{s/\omega}\right).$$

When $v = 0$ and $t > s$, we can obtain a stronger bound with an exponent t/ω, by considering the $\mathbf{B}$ operands rather than the $\mathbf{A}$ operands to $\mathbf{C}$. When $s = 0$ or $t = 0$ rather than $v = 0$, we can first assert by a similar argument that p_i must compute $F/2$ multiplications with at least $K = \frac{v!t!}{(v+t)!}F/2$ entries of $\mathbf{B}$ (when $s = 0$) or with at least $K = \frac{s!v!}{(s+v)!}F/2$ entries of $\mathbf{A}$ (when $t = 0$), which it starts with initially (otherwise it must receive enough other operands to prove the bound). Then, rather than obtaining a lower bound of $K/2$ on the number of entries of $\mathbf{C}$ some p_i computes all the multiplications for, we would instead obtain a lower bound on the number of entries of $\mathbf{B}$ (when $s = 0$) or $\mathbf{A}$ (when $t = 0$) for which p_i computes all the multiplications each of these entries is an operand of (otherwise the processor must communicate at least $K/2$ operands). This last step is possible as a result of the assumption that each unique operand tensor entry is an input to a unique processor. Lastly, knowing that p_i computes all multiplications with $K/2$ of entries of the largest tensor (of order ω), we would use Theorem 4.1 to assert that the number of entries of the second largest tensor (of order $\max(s,t,v)$) required for these multiplications is $(K/2)^{\max(s,t,v)/\omega}$. Having covered all the orderings of the magnitudes of s, t, v when one of them is zero, we have now shown the desired bound,

$$W_\Psi(n, s, t, v, p, M) = \Omega\left((n^\omega/p)^{\max(s,t,v)/\omega}\right).$$

□

5.5 Symmetry Preserving Algorithm

We now derive a lower bound on the horizontal communication cost of any parallel schedule for the symmetry preserving algorithm, considering only cases when at least two of s, t, v are

nonzero (if two are zero, then $\Phi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B}) \equiv \Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$). As in the sequential case, we assume that partial sums are not reused to compute different operand elements in (3.2) and that partial sums are also not reused to compute contributions of $\hat{\mathbf{Z}}$ to different elements of $\mathbf{Z}$ in (3.3). When $s,t,v > 0$, by the same arguments as in the sequential case the reuse partial sums can only be useful when $s,t,v \geq 2$. However, when exactly one of s,t,v is zero, the parallel cost is not dominated by communicating in the largest tensor as in the sequential case, but rather with the communication cost associated with the second largest tensor, so the reuse partial sums could potentially be useful when the two of s,t,v are greater or equal to two and the third is zero. Such cases can arise in coupled-cluster contractions [3].

THEOREM 5.4. *Any load balanced schedule of* $\Phi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$ *on* $\Lambda(p,M)$ *has horizontal communication cost:*

1. *when* $s,t,v \geq 1$,

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{\kappa/\omega}\right),$$

where $\kappa \coloneqq \max(s+v, v+t, s+t)$,

2. *when exactly one of* s,t,v *is zero,*

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{\max(s,t,v)/\omega}\right),$$

under the assumption that partial sums are not reused to compute different operand elements in (3.2) *and that partial sums are also not reused to compute contributions of* $\hat{\mathbf{Z}}$ *to different elements of* $\mathbf{Z}$ *in* (3.3).

PROOF. Consider the $\binom{n}{\omega}$ multiplications in (3.2), some processor p_i must perform at least $\binom{n}{\omega}/p$ of these multiplications, computing $\hat{Z}_{\mathbf{i}\langle\omega\rangle}$ for each $\mathbf{i}\langle\omega\rangle$ in $\bar{V}$ where $\bar{V} \subset \trianglelefteq[1,n]^{\omega}$. We now count the number of dependencies i.e. elements of $\mathbf{A}$ and $\mathbf{B}$ needed by any set of such multiplications, using the assumption that partial summations of these operands are not reused for multiple elements of $\hat{\mathbf{Z}}$. We also count the number of elements of $\mathbf{Z}$ that are dependent on these multiplications, again using an analogous assumption that partial summations of multiplications contributing to different elements of $\mathbf{Z}$ are not reused. These assumptions imply that for any operand or result of $\hat{\mathbf{Z}}$, it never helps to communicate a partial sum, as a unique one is needed for each element of $\hat{\mathbf{Z}}$. As a result, if for $|\bar{V}|/2$ of the multiplications p_i computes, partial summations are either of the $\mathbf{A}$ or the $\mathbf{B}$ operand are received or the computed element of $\hat{\mathbf{Z}}$ is sent, we immediately get a communication cost of $\Omega(n^{\omega}/p)$, which is higher than the desired lower bounds. Therefore, there is a subset of the elements of $\hat{\mathbf{Z}}$ with indices $V \subset \bar{V}$ of size $|V| \geq |\bar{V}|/2$, for which the operands are either received or stored initially and for which the element of $\hat{\mathbf{Z}}$ is accumulated to a partial sum for each element of $\mathbf{Z}$ to which it contributes (otherwise this element of $\hat{\mathbf{Z}}$ would have had to be communicated).

Having ruled out the use of partial summations, we can now cleanly derive lower bounds for the communication due to each of the operands and the result.

Data Received: The operands to the set of multiplications done by p_i include all $\mathbf{A}_{\mathbf{j}\langle s+v\rangle}$ for $\mathbf{j}\langle s+v\rangle \in L_{\mathrm{A}}$, where

$$L_{\mathrm{A}} = \{\mathbf{j}\langle s+v\rangle : \mathbf{j}\langle s+v\rangle \in \chi(\mathbf{i}\langle\omega\rangle), \mathbf{i}\langle\omega\rangle \in V\}.$$

We can obtain a lower bound on the size of L_{A} by finding the number of unique indices $\mathbf{j}\langle s+v\rangle$ given by Theorem 4.1 with $m=\omega$ and $r = s+v$,

$$|L_{\mathrm{A}}| = |V|^{(s+v)/\omega} = \Omega\left((n^{\omega}/p)^{(s+v)/\omega}\right).$$

When $t > 0$, this is asymptotically larger than the $\Theta\left(\frac{n^{s+v}}{p}\right)$ elements of $\mathbf{A}$, p_i can own initially by the load balance assumption, so the number of elements of $\mathbf{A}$ received by p_i when $t > 0$ is

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{(s+v)/\omega}\right).$$

We can apply the same argument to the other operand $\mathbf{B}$ and assert that when $s > 0$,

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{(v+t)/\omega}\right).$$

Data Sent: The elements of $\mathbf{Z}$ to which the set of multiplications done by p_i contribute (in (3.4)) are $Z_{\mathbf{h}\langle s+t\rangle}$ for all $\mathbf{h}\langle s+t\rangle \in L_{\mathrm{Z}}$, where

$$L_{\mathrm{Z}} = \{\mathbf{h}\langle s+t\rangle : \mathbf{h}\langle s+t\rangle \in \chi(\mathbf{i}\langle\omega\rangle), \mathbf{i}\langle\omega\rangle \in V\}.$$

We can obtain a lower bound on the size of L_{Z} by finding the number of unique indices $\mathbf{h}\langle s+t\rangle$ given by Theorem 4.1 with $m=\omega$ and $r = s+t$,

$$|L_{\mathrm{Z}}| = |V|^{(s+t)/\omega} = \Omega\left((n^{\omega}/p)^{(s+t)/\omega}\right).$$

When $v > 0$, this is asymptotically larger than the $\Theta\left(\frac{n^{s+t}}{p}\right)$ elements of $\mathbf{Z}$, p_i can output by the load balance assumption, so the number of partial summations (multiplications or sums of multiplications contributing to a given element $Z_{\mathbf{h}\langle s+t\rangle}$) when $v > 0$ is

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{(s+t)/\omega}\right).$$

Combining the data sent and data received lower bounds, we notice that when $s,t,v \geq 1$, all three of the above lower bounds apply and we get the best bound by considering the largest tensor, which is of order $\kappa = \max(s+v, v+t, s+t)$ and yields the lower bound,

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{\kappa/\omega}\right).$$

When exactly one of s,t,v is zero, we have no lower bound associated with the largest tensor (of order ω), but we have a lower bound associated with the second largest tensor, which is of order $\max(s,t,v)$,

$$W_{\Phi}(n,s,t,v,p) = \Omega\left((n^{\omega}/p)^{\max(s,t,v)/\omega}\right).$$

□

6. ATTAINABILITY AND CONCLUSION

To draw conclusions from our lower bounds, it helps to first consider their attainability. The matrix multiplication and nonsymmetric tensor contraction vertical communication lower bounds are easily attainable by reading in $\sqrt{H/3} \times \sqrt{H/3}$ blocks of each matrix. The parallel horizontal communication bounds for these nonsymmetric algorithms are also attainable for all dimensions [8].

The vertical communication lower bounds for the direct evaluation algorithm, $\Psi_{\otimes}^{(s,t,v)}(\mathbf{A},\mathbf{B})$, are asymptotically attainable by use of an efficient matrix multiplication algorithm, however, attaining the constant factors is more difficult. When one of s,t,v is zero, they may be attained by exploiting the symmetry of the largest tensor while it is in cache. However, we leave it for future work to determine whether it is possible to attain the lower bound for arbitrary s,t,v or if a stronger communication lower bound exists.

The parallel horizontal communication lower bounds for the direct evaluation algorithm should be asymptotically attainable when $s, t, v > 0$ by unpacking the tensors and performing an efficient matrix multiplication algorithm [13,17]. When exactly one of s, t, v is zero, storing the largest tensor in packed layout and moving the other operands should in theory asymptotically attain the lower bounds presented in this paper, however, we are not aware of an appropriate communication analysis of this case. It is also interesting to ask whether it is sometimes worth storing tensors in unpacked layout when the memory is available, as it may be possible to have asymptotically lower communication cost for certain contractions where $s = 0$ or $t = 0$.

For the symmetry preserving algorithm, $\Phi_{\otimes}^{(s,t,v)}(\mathbf{A}, \mathbf{B})$, the sequential vertical communication cost can be attained within a constant factor by the algorithm given in [14]. We conjecture that the parallel bounds may also be attained asymptotically for both of the cases given. As future work, it is interesting to investigate the reuse of partial sums as a method for lowering the computation and/or the communication cost of the symmetry preserving algorithm.

Our lower bounds demonstrate that the symmetry preserving algorithm has the same asymptotic communication bounds for the types of (anti)symmetry in coupled cluster contractions (which always have one of s, t, v as zero). However, coupled-cluster contractions generally involve partially symmetric tensors, which can be efficiently computed via nested use of the symmetry preserving algorithm [16]. Our analysis provides an important step towards full characterization of the communication costs of such nested algorithms for arbitrary partially symmetric contractions.

7. ACKNOWLEDGEMENTS

Solomonik was supported by a U.S. Department of Energy Computational Science Graduate Fellowship (DE-FG02-97ER25308) and an ETH Zürich Postdoctoral Fellowship. Demmel was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under awards: DE-SC0004938, DE-SC0003959, and DE-SC0010200 and by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, X-Stack program under awards: DE-SC0005136, DE-SC0008700, and AC02-05CH11231, and by DARPA award HR0011-12-2-0016.

8. REFERENCES

[1] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: incorporating long messages into the LogP model – one step closer towards a realistic model for parallel computation. In *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '95, pages 95–105, New York, NY, USA, 1995. ACM.

[2] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing communication in linear algebra. *SIAM J. Mat. Anal. Appl.*, 32(3), 2011.

[3] R. J. Bartlett. Many-body perturbation theory and coupled cluster theory for electron correlation in molecules. 32(1):359–401, 1981.

[4] M. A. Bender, G. S. Brodal, R. Fagerberg, R. Jacob, and E. Vicari. Optimal sparse matrix dense vector multiplication in the I/O-model. *Theory of Computing Systems*, 47(4):934–962, 2010.

[5] J. Bennett, A. Carbery, M. Christ, and T. Tao. The Brascamp–Lieb inequalities: finiteness, structure and extremals. *Geometric and Functional Analysis*, 17(5):1343–1415, 2008.

[6] M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick. Communication lower bounds and optimal algorithms for programs that reference arrays–part 1. *arXiv preprint arXiv:1308.0068*, 2013.

[7] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: towards a realistic model of parallel computation. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPOPP '93, pages 1–12, New York, NY, USA, 1993. ACM.

[8] J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger. Communication-optimal parallel recursive rectangular matrix multiplication. In *IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, 2013.

[9] O. Hölder. Über einen Mittelwertsatz. *Nachrichten von der König. Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttigen*, pages 38–47, 1889.

[10] D. Irony, S. Toledo, and A. Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *Journal of Parallel and Distributed Computing*, 64(9):1017 – 1026, 2004.

[11] H. Jia-Wei and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proceedings of the thirteenth annual ACM Symposium on Theory of Computing*, STOC '81, pages 326–333, New York, NY, USA, 1981. ACM.

[12] L. Loomis and H. Whitney. An inequality related to the isoperimetric inequality. *Bulletin of the AMS*, 55:961–962, 1949.

[13] S. Rajbhandari, A. Nikam, P.-W. Lai, K. Stock, S. Krishnamoorthy, and P. Sadayappan. A communication-optimal framework for contracting distributed tensors. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 375–386, Piscataway, NJ, USA, 2014. IEEE Press.

[14] E. Solomonik. *Provably efficient algorithms for numerical tensor algebra*. PhD thesis, University of California, Berkeley, 2014.

[15] E. Solomonik, E. Carson, N. Knight, and J. Demmel. *Tradeoffs between synchronization, communication, and computation in parallel linear algebra computations*, pages 307–318. SPAA '14. ACM, New York, NY, USA, 2014.

[16] E. Solomonik and J. Demmel. Contracting symmetric tensors using fewer multiplications. Technical report, ETH Zürich, 2015.

[17] E. Solomonik, D. Matthews, J. R. Hammond, J. F. Stanton, and J. Demmel. A massively parallel tensor contraction framework for coupled-cluster computations. *Journal of Parallel and Distributed Computing*, 74(12):3176 – 3190, 2014.

[18] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[19] A. Tiskin. *The design and analysis of bulk-synchronous parallel algorithms*. PhD thesis, University of Oxford, 1998.

[20] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[21] J. Čížek. On the correlation problem in atomic and molecular systems. calculation of wavefunction components in ursell-type expansion using quantum-field theoretical methods. *The Journal of Chemical Physics*, 45(11):4256–4266, 1966.

APPENDIX

We now give the proof for Theorem 4.2.

PROOF. The term $mk + kn + mn$ arises from the need to read the inputs into cache and write the output from cache to main memory.

Consider the mnk scalar multiplications done by any schedule which executes the matrix multiplication algorithm. If the schedule computes a contribution to $\mathbf{C}$ more than once, we only count the contribution which gets written to the output in memory, and ignore any other redundantly computed and discarded computation. We subdivide the multiplications into f chunks of size $H^{3/2}$ for $f = \lceil mnk/H^{3/2} \rceil$, with the last 'remainder' chunk being of size less than $H^{3/2}$ (we assume that $H^{3/2}$ does not divide evenly into mnk, so $mnk/H^{3/2} - \lfloor mnk/H^{3/2} \rfloor > 0$, the other case is strictly simpler). We can label each scalar multiplication $A_{il} \cdot B_{lj}$ with a tuple (i, j, l). We then define three projections $\pi_1(i, j, l) = (j, l)$, $\pi_2(i, j, l) = (i, l)$, and $\pi_3(i, j, l) = (i, j)$, which define the two operands needed by the (i, j, l) multiplication and the entry of the output $\mathbf{C}$ to which the multiplication contributes. Given any set S of tuples of size three, and projecting them onto a set of tuples of size two using the three projections π_m, gives us sets $P_m = \{p : \exists s \in S, p \in \pi_m(s)\}, \forall m \in [1, 3]$. By the regular Loomis-Whitney inequality [12] inequality, we obtain that $(|P_1||P_2||P_3|)^{1/2} \geq |S|$, from which it can be shown that $\sum_m |P_m| \geq 3|S|^{2/3}$. This inequality implies that for each chunk $i \in [1, f]$, which contains $t_i \leq H^{3/2}$ multiplications, at least $3t_i^{2/3}$ total operands of $\mathbf{A}$ and $\mathbf{B}$ plus contributions to $\mathbf{C}$ are necessary (Theorem 4.1 with $m = 3$ and $r = 2$ obtains a slightly smaller constant as it takes the union of the three projections).

Now, let x_i be the number of operands present in cache prior to execution of chunk i (by assumption $x_1 = 0$). The number of operands (elements of $\mathbf{A}$ and $\mathbf{B}$) present in cache at the end of execution of chunk i should be equal to the number of operands available for chunk $i + 1$, x_{i+1}. Let the number of contributions to $\mathbf{C}$ (outputs), which remain in cache (are not written to memory) at the end of chunk i, be y_i (by assumption $y_f = 0$), the rest of the outputs produced by chunk i must be written to memory. In total, the amount of reads from memory and writes to memory done during the execution of chunk i with t_i multiplications is then at least $w_i \geq 3t_i^{2/3} - x_i - y_i, \forall i \in [1, f]$. Now, since the operands and outputs which are kept in cache at the end of chunk i must fit in cache, we know that for $i \in [1, f - 1]$, $x_{i+1} + y_i \leq H$. Substituting this in, we obtain $w_i \geq 3t_i^{2/3} - H - x_i + x_{i+1}, \forall i \in [1, f - 1]$. Summing over all chunks yields a lower bound on the total communication cost, $Q_{\text{MM}}(m, n, k, H)$

$$
\begin{aligned}
&\geq \sum_{i=1}^{f} w_i \geq 3t_f^{2/3} - x_f + \sum_{i=1}^{f-1} (3t_i^{2/3} - H - x_i + x_{i+1}) \\
&= -(f-1)H + 3t_f^{2/3} + \sum_{i=1}^{f-1} 3t_i^{2/3} \\
&= -(f-1)H + 3(mnk - (f-1)H^{3/2})^{2/3} + \sum_{i=1}^{f-1} 3H \\
&= 2(f-1)H + 3(mnk - (f-1)H^{3/2})^{2/3} \\
&= 2H \cdot \lfloor mnk/H^{3/2} \rfloor + 3(mnk - \lfloor mnk/H^{3/2} \rfloor \cdot H^{3/2})^{2/3} \\
&= 2mnk/H^{1/2} - 2H \cdot (mnk/H^{3/2} - \lfloor mnk/H^{3/2} \rfloor) \\
&\quad + 3H \cdot (mnk/H^{3/2} - \lfloor mnk/H^{3/2} \rfloor)^{2/3} \\
&\geq 2mnk/H^{1/2} + 2H \cdot \Big[(mnk/H^{3/2} \\
&\quad - \lfloor mnk/H^{3/2} \rfloor)^{2/3} - (mnk/H^{3/2} - \lfloor mnk/H^{3/2} \rfloor) \Big] \\
&\geq 2mnk/H^{1/2}
\end{aligned}
$$

Therefore, we have

$$Q_{\text{MM}}(m, n, k, H) \geq \max \left[\frac{2mnk}{\sqrt{H}}, mk + kn + mn \right].$$

□

We note that a similar reduction may be used to show that the multiplication of nonsymmetric matrices whose entries may overlap (e.g. $\mathbf{AA}$) has cost no less than $\frac{\sqrt{2}}{4}$ of $Q_{\text{MM}}(n, m, k, H)$ (where it is assumed the entries $\mathbf{A}$ and $\mathbf{B}$ are disjoint/independent).