Graph Expansion and Communication Costs of Algorithms

Grey Ballard James Demmel Olga Holtz Oded Schwartz

Abstract

The communication complexity of algorithms is shown to be closely related to the expansion properties of the corresponding computation graphs. We demonstrate this on Strassen's fast matrix multiplication algorithm, and obtain the first lower bound on its communication cost. This bound is optimal.

1 Introduction

The communication of an algorithm (e.g., transferring data between the CPU and memory devices, or between parallel processors) often costs significantly more than its arithmetic[1]. It is therefore of interest to design algorithms minimizing communication[2] on the one hand, and to obtain lower bounds for the communication needed, on the other hand.

While Moore's law predicts an exponential speedup of hardware in general, the annual improvement rate of time-per-arithmetic-operation has over the years consistently exceeded that of time-per-word read/write [GSP04]. The fraction of running time spent on communication is thus expected to increase further.

Previous Work.

Some well-studied problems have seen new faster algorithms, obtained by minimizing communication. Consider, for example, the classic $\Theta(n^3)$ algorithm for matrix-multiplication. While many of its implementations are communication inefficient, communication-avoiding sequential and parallel variants of this algorithm were constructed, and proved optimal, by matching lower bounds [Can69, HK81, FLPR99, ITT04].

In [BDHS09a, BDHS09b] we generalize the results of [HK81, ITT04] regarding matrix multiplication, to attain new communication lower bounds for a much wider variety of algorithms (most of the bounds were shown to be tight). This includes classical algorithms for LU factorization, Cholesky factorization, LDL^T factorization, QR factorization, as well as algorithms for eigenvalues and singular values. Thus we essentially cover all direct methods of linear algebra. The results hold for dense matrix algorithms (most of them are of cubic time), as well as sparse matrix algorithms (whose running time depends on the number of non-zero elements). They apply to sequential and parallel algorithms, to compositions of linear algebra operations (like computing the powers of a matrix), and to certain graph theoretic problems[3].

In [BDHS09a, BDHS09b] we use the approach of [ITT04], based on the Loomis-Whitney geometric theorem [LW49, BZ88], by embedding segments of the computation process into a three dimensional cube. This approach, however, is not suitable when distributivity is used, as is the case in Strassen [Str69] and Strassen-like algorithms [CW90, CKSU05].

Communication Cost of Fast Matrix Multiplication.

Upper bound. The communication bandwidth cost $BW(n)$ of Strassen's algorithm (see Algorithm 1, Appendix), applied to n-by-n matrices on a machine with fast memory of size M, can be bounded above as follows: Run the recursion until matrices are sufficiently small. Then, read the two input matrices into the fast memory, perform the matrix multiplication inside the fast memory, then write the result into the slow memory. We thus have $BW(n) \leq 7 \cdot BW\left(\frac{n}{2}\right) + O(n^2)$

[1] Communication time varies by orders of magnitude, from 0.5×10^{-9} second for L1 cache reference, to 10^{-2} second for disk access. The variation is even more dramatic when communication occurs over networks or the internet [GSP04].

[2] Communication requires much more energy than arithmetic, and saving energy may be even more important than saving time.

[3] See our [BDHS09b] for a detailed list of previously known and recently designed sequential and parallel algorithms that attain these lower bounds.

and $BW\left(\frac{\sqrt{M}}{3}\right) = O(M)$. Thus

$$BW(n) = O\left(\left(\frac{n}{\sqrt{M}}\right)^{\lg 7} \cdot M\right). \tag{1}$$

Lower bound. In this paper, we obtain a tight lower bound:

Theorem 1. *(Main Theorem) The bandwidth cost $BW(n)$ of Strassen's algorithm on a machine with fast memory of size M is*

$$BW(n) = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\lg 7} \cdot M\right). \tag{2}$$

Corollary 2. *The bandwidth cost $BW(n)$ of Strassen's algorithm on a machine with p processors, each with a local memory of size $M \geq \frac{2n^2}{p}$, is*

$$BW(n) \;=\; \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\lg 7} \cdot \frac{M}{p}\right) \;=\; \Omega\left(\frac{n^2}{p^{2-\frac{\lg 7}{2}}}\right).$$

While the lower bound for the sequential algorithm is tight, we are not aware of a matching upper bound for the parallel case. Although our lower bound for the sequential case contradicts the upper bound from FOCS'99 [FLPR99] (and later in [BCG$^+$08]), their upper bound turned out to be erroneously low [Lei08].

The Expansion Approach.

The proof of the main theorem is based on estimating the combinatorial (edge) expansion of the computation graph of an algorithm (where we have a vertex for each input / intermediate / output argument, and edges according to dependencies). This is similar to the approach taken by [HK81], where they use the pebbles model. The bandwidth cost is shown to be tightly connected to the edge expansion properties of this graph. As the graph has a recursive structure, the expansion can be analyzed directly (combinatorially, similarly to what we do in [ASS08]) or by spectral analysis (in the spirit of what was done for the Zig-Zag expanders [RVW02]). There is however, a new technical challenge. While in the replacement and Zig-Zag products a recursive step acts similarly on all vertices, it does not in our case: multiplication and addition vertices are treated differently.

Paper organization Section 2 contains preliminaries on the computation models and notions of graph expansion. In Section 3 we state and prove the connection between communication bandwidth cost and the expansion properties of the computation graph. In Section 4 we analyze the expansion for Strassen's algorithm. We present conclusions and open problems in Section 5.

2 Preliminaries

Communication in Various Computation Models. In the *sequential model* [4], we assume two levels of memory, between which the communication occurs: fast-small and slow-large (e.g.,

[4] Also known as the *two-level I/O model* or *disk access machine (DAM)* model [AV88, BBF$^+$07, CR06].

RAM and disk). The communication *bandwidth cost* measures the number of words communicated, while the communication *latency cost* counts messages (packages of words) sent.

In the *Sequential Model with Memory Hierarchy* we assume multiple levels of memory, (e.g., L1, L2, L3 caches, main memory, and disk). In this case, an optimal algorithm should simultaneously minimize communication between *all* pairs of adjacent levels of memory hierarchy.

In the *Parallel Model* we are interested in the communication among the p computing entities (e.g., processors, computers on a network, or multi-cores on a single processor). Here M stands for the size of local memory of each processor. In this model, no other memory is available, so $M \geq \frac{2n^2}{P}$ as the local memories have to at least hold the entire input. To measure the communication complexity of a parallel algorithm, we count words communicated simultaneously as one word only. This is the bandwidth cost. The latency cost here is when we similarly count messages transferred.

Edge expansion. The edge expansion $h(G)$ of a d-regular undirected graph $G = (V, E)$ is:

$$h(G) \equiv \min_{U \subseteq V, |U| \leq |V|/2} \frac{|E(U, V \setminus U)|}{d \cdot |U|} \tag{3}$$

where $E(A, B) \equiv E_G(A, B)$ is the set of edges connecting the vertex sets A and B. We omit the subscript G when the context makes it clear.

Expansion of small sets. For many graphs, small sets of vertices have better expansion guarantee. Let $h_s(G)$ denote the edge expansion guarantee for sets of size at most s in G:

$$h_s(G) \equiv \min_{U \subseteq V, |U| \leq s} \frac{|E(U, V \setminus U)|}{d \cdot |U|} \ . \tag{4}$$

In many cases, $h_s(G)$ does not depend on $|V(G)|$, although it may decrease when s increases. One way of bounding $h_s(G)$ is by decomposing G into small subgraphs of large edge expansion.

Definition 1 (Graph decomposition). *We say that the set of graphs $\{G'_i = (V_i, E_i)\}_{i \in [l]}$ is an* edge disjoint decomposition *of $G = (V, E)$ if $V = \bigcup_i V_i$ and $E = \biguplus_i E_i$.*

Claim 3. *Let $G = (V, E)$ be a d-regular graph that can be decomposed into edge-disjoint (but not necessarily vertex disjoint) copies of a d'-regular graph $G' = (V', E')$. Then the edge expansion guarantee of G for sets of size at most $|V'|/2$ is $h(G') \cdot \frac{d'}{d}$, namely*

$$h_{\frac{|V'|}{2}}(G) \equiv \min_{U \subseteq V, |U| \leq |V'|/2} \frac{|E_G(U, V \setminus U)|}{d \cdot |U|} \geq h(G') \cdot \frac{d'}{d} \ .$$

Proof. Let $U \subseteq V$ be of size $U \leq |V'|/2$. Let $\{G'_i = (V_i, E_i)\}_{i \in [l]}$ be an edge disjoint decomposition of G, where every G_i is isomorphic to G'. Then

$$\begin{aligned} |E_G(U, V \setminus U)| &= \sum_{i \in [l]} |E_{G'_i}(U_i, V_i \setminus U_i)| \\ &\geq \sum_{i \in [l]} h(G'_i) \cdot d' \cdot |U_i| = h(G') \cdot d' \cdot \sum_{i \in [l]} |U_i| \\ &\geq h(G') \cdot d' \cdot |U| \ . \end{aligned}$$

Therefore

$$\frac{|E_G(U, V \setminus U)|}{d \cdot |U|} \geq h(G') \cdot \frac{d'}{d} \ . \qquad \square$$

When G is not regular. If $G = (V, E)$ is not regular but has a constant maximal degree d, then we can add ($< d$) loops to vertices of degree $< d$, obtaining a regular graph G'. Note that for any $S \in V$, we have $|E_G(S, V \setminus S)| = |E_{G'}(S, V \setminus S)|$, as none of the added edges (loops) contributes to the edge expansion of G'.

3 Bandwidth Cost and Edge Expansion

In this section we recall the computation graph of an algorithm, then show how a partition argument connects between the expansion properties of the graph and the communication bandwidth cost of the algorithm. A similar partition argument already appeared in [ITT04], and then in our [BDHS09b]. In both cases it was used to connect communication bandwidth cost with Loomis-Whitney geometric bound [LW49], which can be viewed, in this context, as an expansion guarantee for the corresponding graphs.

The computation graph. For a given algorithm, we consider the computation (directed) graph $G = (V, E)$, where there is a vertex for each arithmetic operation *(AO)* performed, and for every input element. G contains a directed edge (u, v), if the output operand of the AO corresponding to u (or the input element corresponding to u), is an input operand to the AO corresponding to v. The in-degree of any vertex of G is, therefore, at most 2. The out-degree is, in general, unbounded (i.e., it may be a function of $|V|$). We next show how an expansion analysis of this graph can be used to obtain the communication lower bound for the corresponding algorithm.

The partition argument. Let M be the size of the fast memory. Let O be any total ordering of the vertices that respects the partial ordering of the DAG G, i.e., all the edges are going from left to right. This ordering can be thought of as the actual order in which the computations are performed. Let P be any partition of V into segments $S_1, S_2, ...$, where a segment $S \in P$ is a subset of the vertices which are contiguous in the ordering O.

Let R_S and W_S be the set of read and write operands, respectively (see Figure 1). Namely, R_S is the set of vertices outside S that have an edge going into S, and W_S is the set of vertices in S that have an edge going outside of S. Then the total bandwidth cost due to reads of AOs in S is at least $|R_S| - M$, as at most M of the needed $|R_S|$ operands are already in fast memory when the execution of the segment's AOs starts. Similarly, S causes at least $|W_S| - M$ actual write operations, as at most M of the operands needed by other segments are left in the fast memory when the execution of the segment's AOs ends. The communication bandwidth cost is therefore bounded below by[5]

$$BW \geq \max_P \sum_{S \in P} (|R_S| + |W_S| - 2M) \ . \tag{5}$$

Edge expansion and communication bandwidth cost. Consider a segment S and its read and write operands R_S and W_S (see Figure 1). If the graph G containing S has $h(G)$ edge expansion[6], maximum degree d and at least $2|S|$ vertices, then (by the definition of $h(G)$), we have

[5] One can think of this as a game: the first player orders the vertices. The second player partitions the vertices into contiguous segments. The objective of the first player (e.g., a good programmer) is to order the vertices so that any consecutive partitioning by the second player leads to a small communication count.

[6] The direction of the edges does not matter much for the expansion-bandwidth argument: treating all edges as undirected, changes the bandwidth cost estimate by a factor of 2 at most. For simplicity, we will treat G as undirected.

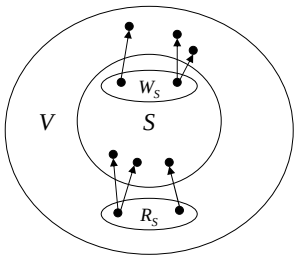

Figure 1: A subset (segment) S and its corresponding read operands R_S, and write operands W_S.

Claim 4. $|R_S| + |W_S| \geq h(G) \cdot |S|$.

Proof. We have $|E(S, V \setminus S)| \geq h(G) \cdot d \cdot |S|$. Either (at least) half of the edges $E(S, V \setminus S)$ touch R_S or half of them touch W_S. As every vertex is of degree d, we have $|R_S| + |W_S| \geq \max\{|R_S|, |W_S|\} \geq \frac{1}{d} \cdot \frac{1}{2} \cdot |E(S, V \setminus S)| \geq h(G) \cdot |S|/2$. □

Combining this with (5) and choosing to partition V into $|V|/s$ segments of equal size s, we obtain:

$$BW \quad \geq \quad \max_s \frac{|V|}{s} \cdot \left(\frac{h(G) \cdot s}{2} - 2M \right) .$$

In many cases $h(G)$ is too small to attain the desired bandwidth cost lower bound. Typically, $h(G)$ is a decreasing function in $|V(G)|$, namely the edge expansion deteriorates with the increase of the input size and running time of the corresponding algorithm. This is the case with matrix multiplication algorithms: the cubic, as well as the Strassen and Strassen-like algorithms. In such cases it is better to consider the expansion of G on small sets only:

$$BW \quad \geq \quad \max_s \frac{|V|}{s} \cdot \left(\frac{h_s(G) \cdot s}{2} - 2M \right) .$$

Choosing[7] the minimal s so that

$$\frac{h_s(G) \cdot s}{2} \geq 3M \tag{6}$$

we obtain

$$BW \quad \geq \quad \frac{|V|}{s} \cdot M . \tag{7}$$

In some cases, the computation graph G does not fit this analysis: it may be non-regular (with vertices of unbounded degree), or its edge expansion may be hard to analyze. In such cases, we may consider some subgraph G' of G instead, to obtain a lower bound on the bandwidth cost:

Claim 5. *Let $G = (V, E)$ be a computation graph of an algorithm Alg. Let $G' = (V', E')$ be a subgraph of G, i.e., $V' \subseteq V$ and $E' \subseteq E$. If G' is d regular and $\alpha = \frac{|V'|}{|V|}$, then the communication bandwidth cost of Alg is*

$$BW \quad \geq \quad \alpha \frac{|V|}{s} \cdot M \quad \textit{where } s \textit{ is chosen so that} \quad \frac{h_s(G') \cdot \alpha s}{2} \geq 3M . \tag{8}$$

[7]The existence of an s that satisfies the condition is not always guaranteed. In the next section we confirm this for Strassen, for sufficiently large $|V(G)|$ (in particular, $|V(G)|$ has to be larger than M). Indeed this is the interesting case, as otherwise all computations can be performed inside the fast memory, with no communication except for reading the input once.

The correctness of this claim follows from Equations (6) and (7), and from the fact that at least an α fraction of the segments have at least $\alpha \cdot s$ of their vertices in G'. We therefore have:

Lemma 6 (Central lemma). *Let Alg be a recursive algorithm with $AO(N)$ arithmetic operations (N being the total input size) and computation graph $G(N) = (V, E)$. Let $G'(N) = (V', E')$ be a regular constant degree subgraph of G, with $\frac{|V'|}{|V|} = \Theta(1)$. Then the bandwidth cost of Alg[8] on a machine with fast memory of size M is*

$$BW = \Omega\left(AO(N) \cdot h(G'(M))\right)$$

or, equivalently,

$$BW = \Omega\left(|V'| \cdot h_s(G'(N))\right) \quad \textit{for } s = AO(\sqrt{M}) \ .$$

4 Expansion Properties of Strassen's Algorithm

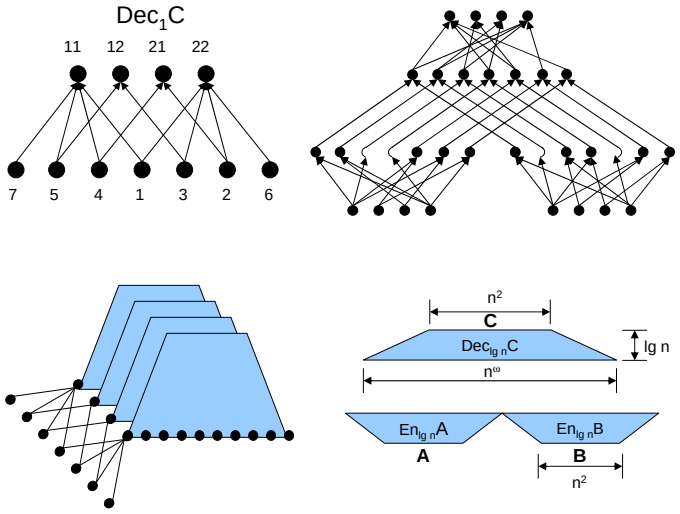

Figure 2: The computation graph of Strassen's algorithm (See Algorithm 1 in Appendix).
Top right: the graph for 2×2 matrices.
Bottom right: schematics of the $n \times n$ matrices.
Top left: decoding the C part of the graph for 2×2 matrices.
Bottom left: schematics of decoding the C part of the graph for $n \times n$ matrices.
Vertices drawn with in-degrees larger that 2 indicate a (weighted) summation. A vertex v with l incoming edges represents a full binary tree (not necessarily balanced) with root v and l leaves.

Recall Strassen's algorithm for Matrix multiplication (see Algorithm 1 in Appendix A) and consider its computation graph (see Figure 2). Let $H_{\lg n}$ be the computational graph of Strassen's algorithm on input matrices of size $n \times n$. $H_{\lg n}$ has the following structure: encode A: generate weighted sums of elements of A. Similarly encode B. Then multiply the encodings of A and B element-wise. Finally, decode C, by taking weighted sums of the multiplications. This is the structure of all the fast matrix multiplication algorithms that were later obtained[9].

[8] In Strassen's algorithm, $N = 2n^2$ is the number of input matrices elements and $T(N) = \Theta\left(n^\omega\right) = \Theta\left(N^{\omega/2}\right)$. G' is the graph $Dec_k C$ for $k = \lg M$, see Section 4 for the definition of $Dec_k C$.

[9] Indeed any arithmetic fast matrix multiplication algorithm can be converted into one with this form [Raz03].

Assume w.l.o.g. that n is an integer power of 2. Denote by $En_{\lg n}A$ the part of $H_{\lg n}$ that corresponds to the encoding of the matrix A. Similarly, $En_{\lg n}B$, and $Dec_{\lg n}C$ correspond to the parts of $H_{\lg n}$ that compute the encoding of B and the decoding of C, respectively.

Given $H_{\lg n}$, one can obtain $H_{\lg n+1}$ as follows: (1) duplicate $H_{\lg n}$ four times, (2) replace every set of four vertices corresponding to a multiplication vertex in $H_{\lg n}$ with a copy of H_1. Similarly, given Dec_iC, one can construct $Dec_{i+1}C$ by duplicating Dec_iC four times and connecting the multiplication layer of vertices with a cross-layer of Dec_1C (one can similarly obtain $En_{i+1}A$ from En_iA and En_1A, and the same for En_iB). In Strassen's algorithm, the graph Dec_1C is bipartite, therefore $Dec_{\lg n}C$ is of constant bounded degree (six). However, En_1A and En_1B have vertices which are both input and output (e.g., A_{11}), therefore $En_{\lg n}A$ and $En_{\lg n}B$ have vertices of out-degree $\Theta(\lg n)$. All in-degrees are at most 2, as an arithmetic operation has at most two inputs.

As $H_{\lg n}$ contains vertices of large degrees, it is easier to consider $Dec_{\lg n}C$: it contains only vertices of constant bounded degree, yet at least one third of the vertices of $H_{\lg n}$ are in it.

Lemma 7. *The edge expansion of Dec_kC is*

$$h(Dec_kC) = \Omega\left(\left(\frac{4}{7}\right)^k\right)$$

Assume w.l.o.g. that n is an integer power of $\sqrt{M}$.[10] Then, $Dec_{\lg n}C$ can be split into edge disjoint copies of $Dec_{\lg M/2}C$. Using Claim 3, we thus deduce the expansion of $Dec_{\lg n}C$ on small sets:

Corollary 8. *$h_s(Dec_{\lg n}C) \geq 3M$ for $s = 9 \cdot M^{\omega/2}$.*

As $Dec_{\lg n}C$ contains $\alpha = \frac{1}{3}$ of the vertices of $H_{\lg n}$, Central Lemma 6 now yields Main Theorem 1.

4.1 Expansion Estimation: Combinatorial Approach

We are now in a position to prove our main lemma:

Proof of Lemma 7. Let $G_k = (V, E)$ be Dec_kC, and let $S \subseteq V, |S| \leq |V|/2$. We next show that $|E(S, V \setminus S)| \geq c \cdot d \cdot |S| \cdot \left(\frac{4}{7}\right)^k$, where c is some universal constant, and $d = 6$ is the constant degree of Dec_kC (after adding loops to make it regular).

The proof works in two steps. We argue (1) that each level of G_k contains about the same fraction of S vertices, or we have many edges leaving S and we are done. We then show that (2) the homogeneity (of a fraction of S vertices) holds between the distinct parts of each level as well, or, again, we have many edges leaving S and we are done. But on the lowest/highest levels, each part of each level is a single vertex, and therefore has fraction 0 or 1 in S. Let us start with (1).

Let l_i be the ith level of vertices of G_k, so $4^k = |l_1| < |l_2| < ... < |l_i| = 4^{k-i+1}7^{i-1} < ... < |l_{k+1}| = 7^k$. Let $S_i \equiv S \cap l_i$. Let σ be the fractional size of S and σ_i the fractional size of S in level i, namely, $\sigma \equiv \frac{|S|}{|V|}$ and $\sigma_i \equiv \frac{|S_i|}{|l_i|}$. Due to averaging, we observe the following:

Fact 9. *There exist i and i' such that $\sigma_i \leq \sigma \leq \sigma_{i'}$.*

From the geometric sum, we now have:

Fact 10. $|V(G_k)| = \sum_{i=0}^{k} |l_k| \cdot \left(\frac{4}{7}\right)^i = |l_k| \cdot \left(1 - \left(\frac{4}{7}\right)^{k+1}\right) \cdot \frac{3}{7}$ *so* $\frac{3}{7} - \left(\frac{4}{7}\right)^{k+1} \leq \frac{|l_k|}{|V(G_k)|} \leq \frac{3}{7}$, *and*

[10] We may assume this, as we are dealing with a lower bound here, so it suffices to prove the assertion for infinite number of n's. Alternatively, in the following decomposition argument, we leave out a few of the top or bottom levels of vertices of $Dec_{\lg n}C$, so that n is an integer power of $\sqrt{M}$ and so that at most $|S|/2$ vertices of S are cut off.

Fact 11. $\left(\frac{3}{7}-\left(\frac{4}{7}\right)^{k+1}\right)\cdot\left(\frac{4}{7}\right)^{k}\leq\frac{|l_1|}{|V(G_k)|}\leq\frac{3}{7}\cdot\left(\frac{4}{7}\right)^{k}$.

Claim 12 (Step (1)). *If there exists i so that $\frac{|\sigma-\sigma_i|}{\sigma}\geq\frac{1}{10}$, then*

$$|E(S,V\setminus S)|\geq c\cdot d\cdot|S|\cdot\left(\frac{4}{7}\right)^{k}.$$

Proof. Let $\delta_i\equiv\sigma_{i+1}-\sigma_i$. A G_1 component connecting l_i with l_{i+1} has no edges in $E(S,V\setminus S)$ if all or none of its vertices are in S. Otherwise it contributes at least one edge. Thus, we proved

Claim 13.

$$|E(S,V\setminus S)\cap E(l_i,l_{i+1})|\geq c'\cdot d\cdot|\delta_i|\cdot|l_i|\qquad\textit{where}\quad c'\equiv\frac{1}{d\cdot|E(G_1)|}\ .$$

By Claim 13, we have

$$\begin{aligned}|E(S,V\setminus S)| &\geq \sum_{i\in[k]}|E(S,V\setminus S)\cap E(l_i,l_{i+1})|\\ &\geq \sum_{i\in[k]}c'\cdot d\cdot|\delta_i|\cdot|l_i|\\ &\geq c'\cdot d\cdot|l_1|\sum_{i\in[k]}|\delta_i|\\ &\geq c'\cdot d\cdot|l_1|\cdot\left(\max_{i\in[k+1]}\sigma_i-\min_{i\in[k+1]}\sigma_i\right).\end{aligned}$$

If $\max_i\sigma_i-\min_i\sigma_i\geq\frac{1}{10}\sigma$, then

$$\begin{aligned}|E(S,V\setminus S)| &\geq c'\cdot d\cdot|l_1|\cdot\frac{1}{10}\sigma\\ &\geq \frac{c'\cdot d}{10}\sigma\cdot\left(\frac{3}{7}\right)^{2}\cdot|V(G_k)|\cdot\left(\frac{4}{7}\right)^{k}\\ &\geq \frac{c'\cdot d}{10}\cdot\left(\frac{3}{7}\right)^{2}\cdot|S|\cdot\left(\frac{4}{7}\right)^{k}\end{aligned}$$

and we are done. □

We next show (2), i.e., that a similar homogeneity argument holds between distinct parts of each level. Consider the graph G_k. If we remove the vertices l_{k+1}, we are left with four copies of G_{k-1}. Let us denote them by $G_{k-1}^{(1)},G_{k-1}^{(2)},G_{k-1}^{(3)}$, and $G_{k-1}^{(4)}$. Let $l_i^{(j)}$ be the ith level of $G_{k-1}^{(j)}$, and $\sigma_i^{(j)}=\frac{|S\cap l_i^{(j)}|}{|l_i^{(j)}|}$. We argue that the fraction of S vertices in the four parts $l_k^{(1)},l_k^{(2)},l_k^{(3)},l_k^{(4)}$ is about the same, or we have many edges leaving S, namely,

Claim 14 (Step (2)).

$$\frac{|E(S,V\setminus S)|}{|l_1|}\geq c'\cdot d\cdot(\sigma_1\cdot(1-\sigma_{k+1})+\sigma_{k+1}\cdot(1-\sigma_1))\ .$$

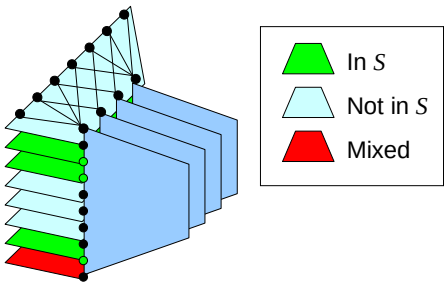

Figure 3: The DAG of a recursive algorithm: Strassen's matrix multiplication

Proof. Let $T_k(\sigma_{k+1})$ denote the minimal possible number of edges leaving S in G_k, given that the fraction of S vertices in l_{k+1} is σ_{k+1}. Observe that the G_1 components connecting l_k with l_{k+1} contributes to $|E(S, V \setminus S) \cap E(l_i, l_{i+1})|$ according to the largest discrepancy between σ_{k+1} and the four σ_k^i (recall the recursive construction of $Dec_k C$ in Figure 3). We therefore have proved

Claim 15.

$$|E(S, V \setminus S) \cap E(l_k, l_{k+1})| \geq c' \cdot d \cdot \max_{i \in [4]} |\sigma_{k+1} - \sigma_k^i| \cdot |l_k^{(i)}| \ .$$

Thus,

$$T_k(\sigma_{k+1}) \quad \geq \quad \sum_{i \in [4]} T_{k-1}\left(\sigma_k^{(i)}\right) + c' \cdot d \cdot \max_{i \in [4]} |\delta_k^i| \cdot |l_k^{(i)}| \ .$$

As G_1 is connected, we have

$$T_1(\sigma_2) \geq \begin{cases} 1 & \text{if } \sigma_2 \notin \{0, 1\} \\ 0 & \text{otherwise.} \end{cases}$$

Consider the recursion tree $T = T_k(\sigma_{k+1})$. It has $|l_1|$ leaves, of which $\sigma_1 \cdot |l_1|$ correspond to vertices in S, namely, with fraction of vertices in S being 1. As the fraction of S vertices changes from σ_{k+1} in the root to 1 in the leaves, these vertices contribute at least $c' \cdot d \cdot (1 - \sigma_{k+1})\sigma_1 \cdot |l_1|$. Similarly, T has $(1 - \sigma_1) \cdot |l_1|$ leaves with fraction of vertices in S being 0. These leaves contribute at least $c' \cdot d \cdot \sigma_{k+1}(1 - \sigma_1) \cdot |l_1|$. Thus

$$\frac{|E(S, V \setminus S)|}{|l_1|} \geq c' \cdot d \cdot ((1 - \sigma_{k+1})\sigma_1 + \sigma_{k+1}(1 - \sigma_1)) \ . \qquad \square$$

By Claim 12, w.l.o.g, $|\sigma - \sigma_1| \leq \frac{\sigma}{10}$, and $|\sigma - \sigma_{k+1}| \leq \frac{\sigma}{10}$. As $\sigma \leq \frac{1}{2}$, by Claim 15 we have $\frac{|E(S,V \setminus S)|}{|l_1|} \geq c' \cdot d \cdot \frac{9}{10} \cdot \frac{8}{10} \cdot \sigma$, so $|E(S, V \setminus S)| \geq \frac{c' \cdot d \cdot 72}{10} \cdot \sigma \cdot |l_1| \geq \frac{72}{120} \cdot \sigma \cdot |S| \cdot \left(\frac{4}{7}\right)^k$. This finishes the proof of Lemma 7. $\square$

5 Conclusions and Open Problems

We obtained a tight lower bound for the communication cost of Strassen's fast matrix multiplication algorithm. This bound is optimal for the sequential model with memory hierarchy.

Our lower bounds, as well as most of the previous lower bounds [HK81, ITT04, BDHS09a, BDHS09b] deal with linear algebra and numerical analysis algorithms[11]. Our new approach, however, is general enough to address communication lower bounds of other recursive algorithms.

[11] With the exception of a few graph theoretic related results, e.g., [MPP02] and in our [BDHS09b].

Consider, for example any Strassen-like fast matrix multiplication algorithms with AOs count $\Theta(n^\omega)$ for some $2 < \omega < 3$ (e.g., the fastest known, in [CW90, CKSU05] with $\omega \approx 2.376$). That is, a recursive algorithm that uses a base case: multiplying two n_0-by-n_0 matrices using $m(n_0)$ multiplications. The running time of the recursive Strassen-like algorithm is then $T(n) = m(n_0) \cdot T\left(\frac{n}{n_0}\right) + O(n^2)$, so $T(n) = \Theta(n^\omega)$ where $\omega = \log_{n_0} m(n_0)$. Using the same argument that leads to Equation (1) we obtain a bandwidth upper bound on Strassen-like algorithms: $BW(n) \leq m(n_0) \cdot BW\left(\frac{n}{n_0}\right) + O(n^2)$ and $BW\left(\frac{\sqrt{M}}{3}\right) = O(M)$. Thus

$$BW(n) = O\left(\left(\frac{n}{\sqrt{M}}\right)^\omega \cdot M\right) . \tag{9}$$

The lower bound obtained using our approach is:

Theorem 16.

$$BW(n) = \widetilde{\Omega}\left(\left(\frac{n}{\sqrt{M}}\right)^\omega \cdot M\right) .$$

Here $\widetilde{\Omega}$ hides a logarithmic factor, arising because of the vertices of high degree in the computation graph. We believe that this loss can be in fact avoided.

For parallel algorithms, using a reduction between the parallel and sequential models (see e.g., [ITT04] or our [BDHS09b]) this yields:

Corollary 17.

$$\begin{aligned} BW(n) &= \widetilde{\Omega}\left(\left(\frac{n}{\sqrt{M}}\right)^\omega \cdot \frac{M}{p}\right) \\ &= \widetilde{\Omega}\left(\frac{n^2}{p^{2-\frac{\omega}{2}}}\right) . \end{aligned}$$

Note that the numerator does not depend on ω. Thus, a change in ω influences only the power of p in the denominator. Finding a parallel algorithm matching this bound remains an open problem.

In many cases, the simplest recursive algorithm for a problem turns out to be communication optimal (e.g., in the cases of matrix multiplication [FLPR99] and Cholesky decomposition [AP00, BDHS09a], but not in the case of LU decomposition [Tol97]). This leads to the question whether the communication optimality of these algorithms is determined by the expansion properties of the corresponding computation graphs.

It is of great interest to construct new models general enough to capture the rich and evolving design space of current and predictable future computers. Such models can be *homogeneous*, consisting of many layers, where the components of each layer are the same (e.g., a supercomputer with many identical multi-core chips on a board, many identical boards in a rack, many identical racks, and many identical levels of associated memory hierarchy); or *heterogeneous*, with components with different properties residing on the same level (e.g., CPUs alongside GPUs, where the latter can do some computations very quickly but are much slower to communicate with).

Some experience has been acquired on such systems (e.g., using GPU assisted linear algebra computation [VD08]). However, there is currently no systematic-theoretic way of obtaining upper and lower bounds for such models. For example, recursive algorithms tend to be cache oblivious and communication optimal for the sequential hierarchy model. Finding an equivalent technique that would work for arbitrary architecture is a fundamental open problem.

Constructing models for complex hardware, that will be simple enough to analyze, yet reflect the actual hardware's properties sufficiently well, is one of our most challenging tasks. Such models

will allow wiser use of available computational resources. Moreover, it is likely to help exposing communication bottlenecks in complex hardware architecture, thus focusing effort on ways of improving its efficiency.

6 Acknowledgment

We would like to thank Eran Rom for helpful discussions.

A Strassen's Fast Matrix Multiplication Algorithm

Algorithm 1 Matrix Multiplication: Strassen's Algorithm

Input: Two $n \times n$ matrices, A and B.

1: **if** $n = 1$ **then**
2: $C_{11} = A_{11} \cdot B_{11}$
3: **else**
4: {Decompose A into four equal square blocks $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ and the same for B.}
5: $M_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$
6: $M_2 = (A_{21} + A_{22}) \cdot B_{11}$
7: $M_3 = A_{11} \cdot (B_{12} - B_{22})$
8: $M_4 = A_{22} \cdot (B_{21} - B_{11})$
9: $M_5 = (A_{11} + A_{12}) \cdot B_{22}$
10: $M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$
11: $M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$
12: $C_{11} = M_1 + M_4 - M_5 + M_7$
13: $C_{12} = M_3 + M_5$
14: $C_{21} = M_2 + M_4$
15: $C_{22} = M_1 - M_2 + M_3 + M_6$
16: **end if**
17: **return** C

References

[AP00] N. Ahmed and K. Pingali. Automatic generation of block-recursive codes. In *Euro-Par '00: Proceedings from the 6th International Euro-Par Conference on Parallel Processing*, pages 368–378, London, UK, 2000. Springer-Verlag.

[ASS08] N. Alon, O. Schwartz, and A. Shapira. An elementary construction of constant-degree expanders. *Combinatorics, Probability & Computing*, 17(3):319–327, 2008.

[AV88] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.

[BBF+07] M. A. Bender, G. S. Brodal, R. Fagerberg, R. Jacob, and E. Vicari. Optimal sparse matrix dense vector multiplication in the I/O-model. In *SPAA '07: Proceedings of the*

nineteenth annual ACM symposium on Parallel algorithms and architectures, pages 61–70, New York, NY, USA, 2007. ACM.

[BCG+08] G. E. Blelloch, R. A. Chowdhury, P. B. Gibbons, V. Ramachandran, S. Chen, and M. Kozuch. Provably good multicore cache performance for divide-and-conquer algorithms. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 501–510, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[BDHS09a] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Communication-optimal Parallel and Sequential Cholesky Decomposition. In *SPAA '09: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 245–252, New York, NY, USA, 2009. ACM.

[BDHS09b] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing Communication in Linear Algebra. Submitted. Available from http://arxiv.org/abs/0905.2485, 2009.

[BZ88] Y. D. Burago and V. A. Zalgaller. *Geometric Inequalities*, volume 285 of *Grundlehren der Mathematische Wissenschaften*. Springer, Berlin, 1988.

[Can69] L. Cannon. *A cellular computer to implement the Kalman filter algorithm*. PhD thesis, Montana State University, Bozeman, MN, 1969.

[CKSU05] H. Cohn, R. D. Kleinberg, B. Szegedy, and C. Umans. Group-theoretic algorithms for matrix multiplication. In *FOCS*, pages 379–388, 2005.

[CR06] R. A. Chowdhury and V. Ramachandran. Cache-oblivious dynamic programming. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 591–600, New York, NY, USA, 2006. ACM.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[FLPR99] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 285, Washington, DC, USA, 1999. IEEE Computer Society.

[GSP04] S. L. Graham, M. Snir, and C. A. Patterson, editors. *Getting up to Speed: The Future of Supercomputing*. Report of National Research Council of the National Academies Sciences. The National Academies Press, Washington, D.C., 2004. 289 pages, http://www.nap.edu.

[HK81] J. W. Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 326–333, New York, NY, USA, 1981. ACM.

[ITT04] D. Irony, S. Toledo, and A. Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *J. Parallel Distrib. Comput.*, 64(9):1017–1026, 2004.

[Lei08] C. E. Leiserson. Personal communication with G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, 2008.

[LW49] L. H. Loomis and H. Whitney. An inequality related to the isoperimetric inequality. *Bulletin of the AMS*, 55:961–962, 1949.

[MPP02] J. P. Michael, M. Penner, and V. K. Prasanna. Optimizing graph algorithms for improved cache performance. In *In Proc. Intl Parallel and Distributed Processing Symp. (IPDPS 2002), Fort Lauderdale, FL*, pages 769–782, 2002.

[Raz03] R. Raz. On the complexity of matrix product. *SIAM J. Comput.*, 32(5):1356–1369 (electronic), 2003.

[RVW02] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1):157–187, 2002.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[Tol97] S. Toledo. Locality of reference in LU decomposition with partial pivoting. *SIAM J. Matrix Anal. Appl.*, 18(4):1065–1081, 1997.

[VD08] V. Volkov and J. Demmel. Benchmarking GPUs to tune dense linear algebra. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–11, Piscataway, NJ, USA, 2008. IEEE Press.

Grey Ballard
ballard@eecs.berkeley.edu
Computer Science Division, University of California-Berkeley. Research supported by Microsoft and Intel Award #20080469 and by matching funding by U.C. Discovery under Award #DIG07-10227.

James Demmel
demmel@cs.berkeley.edu
Department of Mathematics and Computer Science Division, University of California-Berkeley.

Olga Holtz
holtz@math.ias.edu
Departments of Mathematics, University of California-Berkeley and Technische Universität Berlin; School of Mathematics, Institute for Advanced Study. Research supported by the Sofja Kovalevskaja programme of Alexander von Humboldt Foundation and by the National Science Foundation under agreement DMS-0635607.

Oded Schwartz
odedsc@math.tu-berlin.de
Department of Mathematics, Technische Universität Berlin. Research supported by the Sofja Kovalevskaja programme of Alexander von Humboldt Foundation.