# Revisiting the I/O-Complexity of Fast Matrix Multiplication with Recomputations

Roy Nissim
*The Hebrew University of Jerusalem, Israel*
roynissim@cs.huji.ac.il

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

***Abstract*—Communication costs, between processors and across the memory hierarchy, often dominate the runtime of algorithms. Can we trade these costs for recomputations? Most algorithms do not utilize recomputation for this end, and most communication cost lower bounds assume no recomputation, hence do not address this fundamental question. Recently, Bilardi and De Stefani (2017), and Bilardi, Scquizzato, and Silvestri (2018) showed that recomputations cannot reduce communication costs in Strassen's fast matrix multiplication and in fast Fourier transform. We extend the former bound and show that recomputations cannot reduce communication costs for a few other fast matrix multiplication algorithms.**

***Index Terms*—I/O-complexity, Fast Matrix Multiplication, Recomputation, Memory Hierarchy, Parallel Computation**

## I. Introduction

Communication costs between processors or across the memory hierarchy often dominate algorithms run time. The fraction of communication costs in the total run time is expected to further increase since the annual improvement rate of arithmetic-operation, exceeds that of communication. Can one avoid communication by allowing recomputation? Since intermediate computations may be used several times, and by different processors, it may seem that by repeating computations one can save communication, both between processors and across memory hierarchy. Tight communication costs lower bounds are known for a variety of algorithms, such as classical matrix multiplication [1]–[5], direct linear algebra algorithms [6], [7], fast matrix multiplication [1], [8]–[10], fast Fourier transform [2], [5], [11]–[14] and the N-body problem [15]. Yet most of these bounds assume that every computation is performed once, namely there is no recomputation. Can recomputation decrease communication costs?

Savage showed an example where recomputation can asymptotically reduce communication costs for some $CDAG$s, using the "S-span technique" [16]. Bilardi and Peserico [17] showed that some $CDAGs$ admit an optimal schedule only when recomputation is allowed. Bilardi and De Stefani [10] obtain a lower bound that allows recomputations for Strassen's matrix multiplication algorithm [18]. Particularly they showed that recomputation cannot reduce communication costs asymptotically for that algorithm, both on sequential and on parallel machine models. Bilardi, Scquizzato, and Silvestri [13] obtain similar results regarding fast Fourier transformation.

### A. Our contribution

Using the approach of Bilardi and De Stefani [10] we show that recomputation can not reduce communication cost asymptotically for all fast matrix multiplication and alternative basis fast matrix multiplication algorithms, with base case of $2 \times 2$. These lower bounds apply to Winograd's algorithm [19] (with leading coefficient of the arithmetic complexity reduced from 7 to 6) and Karstadt-Schwartz's algorithm [20] (with leading coefficient further reduced to 5). We provide both memory dependent and memory independent lower bounds, extending [1]. Hopcroft and Kerr [21] showed that 7 is the minimum number of multiplication necessary to multiply two matrices of size $2 \times 2$. Thus a fast matrix multiplication algorithm with 2 base case consists of 7 multiplication in his base case.

*Theorem 1.1:* Consider a fast matrix multiplication algorithm using a $2 \times 2$ base case, and let $\omega_0 = \log_2 7$. The I/O-complexity of the algorithm is:

$$\Omega((n/\sqrt{M})^{\omega_0} \cdot M)$$

in the sequential model, and

$$\max\{\Omega((n/\sqrt{M})^{\omega_0} \cdot M/P),\ \Omega(n^2/P^{2/\omega_0})\}$$

in the distributed model, ***regardless of recomputations***, where $n$, $M$, and $P$ denotes the size of the input matrix, the size of the cache of each processor, and the number of processors, respectively.

### B. Paper organization

In Section II we provide some preliminaries and describe our machine models. In Section III we prove lower bounds for fast matrix multiplication. In Section IV we extend the bounds to hold for alternative basis matrix multiplication algorithms. In Section V we discuss some open problems and future work.

[1]Internal computations in the classic matrix multiplication algorithm are used only once, therefore, there is no point in recomputation

[2]This is the bound for most values of $P$, some values of $P$ yield slightly lower bound

TABLE I
KNOWN LOWER BOUNDS

The table presents the known lower bounds on several problems, and the citations of their proofs. We distinguish proof that assume no recomputation with proofs allowing recomputations.

| Algorithm | Lower Bounds | Without recomputation | With recomputation |
|---|---|---|---|
| Classic matrix multiplication | $\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^3 \cdot \frac{M}{P}\right)$ | [2] | Not relevant[1] |
| | $\Omega\left(\frac{n^2}{P^{2/3}}\right)$ | [3] | |
| Strassen's matrix multiplication | $\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} \cdot \frac{M}{P}\right)$ | [8]–[10], | [10] |
| | $\Omega\left(\frac{n^2}{P^{2/\log_2 7}}\right)$ | [1] | [here] |
| Other fast matrix multiplication with $2 \times 2$ base case | $\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} \cdot \frac{M}{P}\right)$ | [8]–[10], | [here] |
| | $\Omega\left(\frac{n^2}{P^{2/\log_2 7}}\right)$ | [1] | [here] |
| Fast matrix multiplication with general base case | $\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\omega_0} \cdot \frac{M}{P}\right)$ | [8]–[10] | - |
| | $\Omega\left(\frac{n^2}{P^{2/\omega_0}}\right)$ | [1] | - |
| Rectangular fast matrix multiplication with $\langle m, n, p; q\rangle$ base case | $\Omega\left(\frac{q^t}{P \cdot M^{\log_{mp} q-1}}\right)$ where $t$ is the exponent of the base case | [22] | - |
| Fast Fourier transform | $\Omega\left(\frac{n \cdot \log n}{P \cdot \log(M)}\right)$ | [12] | - |
| | $\Omega\left(\frac{n \cdot \log n}{P \cdot \log\left(\frac{n}{P}\right)}\right)$ | [5], [11] | [13][2] |

The lower bounds for the sequential model are derived with $P = 1$. $\omega_0$ is the exponent of the arithmetic of the algorithm.

## II. PRELIMINARIES

### A. Definitions

A fast matrix multiplication algorithm with $2 \times 2$ base case, often consists of three steps: (I) Encoding the $n \times n$ input variables of each of the two matrices into $n^{\log_2 7}$ variables, consisting of linear sums of the input. (II) Performing scalar multiplications between the encoded variables of the two matrices. (III) Decoding the $n^{\log_2 7}$ outputs of the multiplication into $n \times n$ output variables using linear sums.

*Definition 2.1 (Computational directed acyclic graph):* A computational directed acyclic graph ($CDAG$) is a directed graph where each of its vertices represents an input, intermediate, or output argument, and each of is edges represents direct dependency. Given a $CDAG$ $G$, we denote by $V_{inp}(G)$ (respectively $V_{int}(G)$, $V_{out}(G)$) the set of input (respectively internal, output) vertices of $G$.

For the arithmetic operations $z = x+y$ for instance, we have edges from $v_x$ and from $v_y$ to $v_z$, where $v_x$, , $v_y$, and $v_z$ are nodes in graph $G$ which represents the variables $x$, $y$, and $z$,

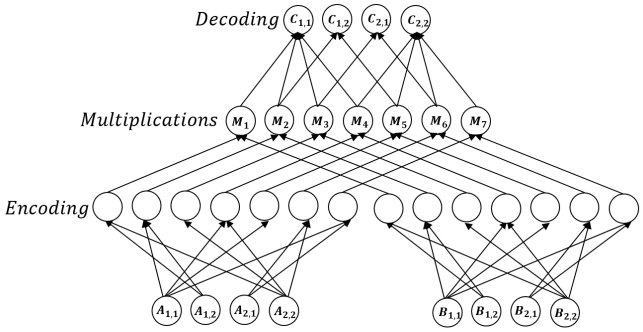


Fig. 1. The $CDAG$ of Strassen's base algorithm

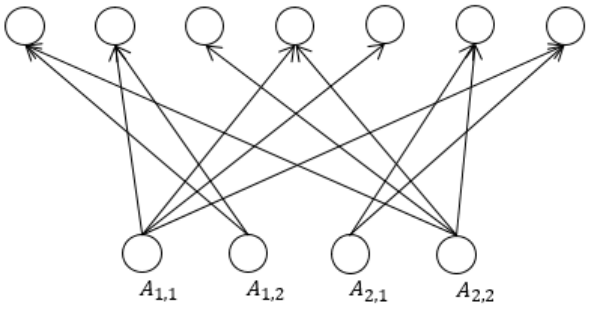


Fig. 2. The $CDAG$ of Strassen's encoder graph for matrix A

respectively. We denote by $H^{n\times n}$ the $CDAG$ that represents the fast matrix multiplication algorithm with base case of size $2 \times 2$ using 7 multiplication. We denote by $SUB\_H^{r\times r}$ the union of all sub-graphs that correspond to intermediate multiplications of matrices of size $r \times r$.

We call the part of the CDAG that corresponds to the encoding of the input (respectively decoding of the output) **Encoder/Enc** (respectively **Decoder/Dec**). See Figures 1,2 and Algorithm 2 for demonstration.

*Lemma 2.2 (Recursive expansion):* The group of sub-CDAGs of size $r \times r$ ($SUB\_H^{r\times r}$) has $(n/r)^{\log_2 7} \cdot r^2$ output vertices.

*Proof:* Proof. The algorithm uses the base matrix multiplication algorithm recursively, with smaller block sizes. At each step of the recursion the size of the input is divided by 2, and the number of sub-problems is multiplied by 7. Therefore, there are $(n/r)^{\log_2 7}$ sub problems of size $r\times r$, i.e. sub-graphs that correspond to intermediate multiplications of matrices of size $r \times r$. Each of the sub graphs have $r^2$ output vertices, thus $|V_{out}(SUB\_H^{r\times r})| = (n/r)^{\log_2 7} \cdot r^2$.

■

*Definition 2.3 (Dominator set):* Given a graph $G = (V, E)$ we say that $\Gamma \subseteq V$ is a dominator set for $V^{'} \subseteq V$ if every path from $V_{inp}(G)$ to $V^{'}$ contains a vertex in $\Gamma$. We also say that $V^{'}$ is dominated by $\Gamma$. A minimum dominator set is a dominator set with minimum cardinality.

*Definition 2.4 (Matching):* Let $G = (X, Y, E)$ be a bipartite graph, and let $X^{'}$ be a subset of $X$. We say that there is a matching for $X^{'} = x_1, ...x_k$ in $G$ if there exists a set $Y^{'} \subseteq Y$, where $Y^{'} = y_1, ..., y_k$, such that $\forall i\ (x_i, y_i) \in E$. We also say that there is a matching between $X^{'}$ and $Y$.

*Theorem 2.5 (Hall's theorem [23]):* Let $G = (X, Y, E)$ be a bipartite graph, and let $X^{'}$ be a subset of $X$. There is a matching for $X^{'}$ if and only if $\forall W \subseteq X^{'},\ |W| \leq |N_G(W)|$, where $N_G(W)$ denotes the set of neighbors of $W$.

*Definition 2.6 (Bilinear matrix multiplication [20]):* Let $A,\ B$ be matrices of size $n \times m,\ m \times k$, respectively. Let $R$ be a ring and let $\phi,\ \psi$, and $\nu$ be automorphisms of $R^{n\cdot m},\ R^{m\cdot k}$, and $R^{n\cdot k}$, respectively. We denote the algorithm which takes $\phi(A), \psi(B)$ as inputs and outputs $\nu(A \cdot B)$ using $t$ multiplication by $\langle n, m, k; t\rangle_{\phi,\psi,\nu}$-algorithm.

*Definition 2.7 (Alternative basis matrix multiplication [20]):* Given a recursive-bilinear $\langle n, m, k; t\rangle_{\phi,\psi,\nu}$-algorithm, $ALG$, alternative basis matrix multiplication works as follows (see Algorithm 1 in Section A):

(I) Perform fast basis transformation on the input matrices $A,\ B$ ($A \to \phi(A),\ B \to \psi(B)$).

(II) Perform the recursive-bilinear algorithm, $ALG$, on the input matrices in the alternative form $(\phi(A), \psi(B) \to \nu(A \cdot B) = \nu(C))$.

(III) Perform fast basis transformation on the result from previous stage to restore to the original base ($\nu(C) \to C$).

*Definition 2.8 (Grigoriev's flow [10]):* A function $f : \mathcal{R}^p \to \mathcal{R}^q$ has a $\omega(u, v)$ Grigoriev flow if for all subsets $X_1$ and $Y_1$ of its $p$ input and $q$ output variables, with $|X_1| \geq u$ and $|Y_1| \geq v$, there is a sub-function $h$ of $f$ obtained by making some assignment to variables of $f$ not in $X_1$ and discarding output variables not in $Y_1$, such that, $h$ has at least $|\mathcal{R}|^{\omega(u,v)}$ points in the image of its domain.

### B. Machine models

We address two machine models: Sequential and Parallel. **The sequential model** is a machine model with two layer memory, a slow memory of unlimited size, and a fast memory of limited size $M$. The input and the output is stored in the slow memory. Computations are performed on arguments that are in the fast memory, and the result of the computation is stored in the fast memory. Reading an argument from the slow memory or writing an argument to the slow memory is considered an I/O operation.

**The parallel model** is a machine model consists of $P$ identical processors, each equipped with a fast memory of size $M$. The input and the output is distributed evenly among all processors. A processor performs computations on arguments in its local memory, and the result of the

computation is stored in his local memory. Exchanging an argument between processors is considered an I/O operation.

## III. Fast matrix multiplication algorithms with 2x2 base case

In this section we show that the known lower bounds for fast matrix multiplication with base case of size $2 \times 2$ ($\Omega((n/\sqrt{M})^{\log_2 7} \cdot M/P)$, $\Omega(n^2/P^{2/\log_2 7})$) hold when recomputation is allowed. Our proof follows that of Bilardi and De Stefani [10]. As in many I/O-complexity lower bounds, they partition the computation into segments, and bound the I/O operations of each segment as a function of the available input and output operands. They do so using a new technique which combines aspects of the Hong-Kung dominator set method, with the concept of Grigoriev information flow of functions. Their technique also precludes the use of repeated computations for reducing the I/O-complexity (an assertion which almost none of the previous techniques can guarantee). In their proof they use a case analysis of the computations of Strassen's algorithm to bound from below the size of the minimal dominator set of the segments computations (see Lemma 7 in [10]). We replace the case analysis part of their proof with a bipartite matching argument, which applies to any fast matrix multiplication algorithms with base case of size $2 \times 2$ (such as Winograd's algorithm [19]). In Section IV we extend the results to hold for alternative basis fast matrix multiplication with base case of size 2x2 as well (such as Karstadt-Schwartz's algorithm [20]). We also apply the technique of [1] to obtain a memory independent lower bound.

The proof involves several inter dependant lemmas. Let us first describe our proof idea and its work-flow in general. As in other lower bound proofs, we partition the computation schedule into segments, bound the I/O-complexity of each segment, and compute the total bound by multiplying the above with the number of segments. The bound on each segment is obtained using the Hong-Kung dominator set method (in the same way Bilardi and De Stefani are utilizing it). Particularly, we show a relation between the number of outputs of intermediate problems (output vertices of sub-CDAGs representing sub-problems of size $2\sqrt{M} \times 2\sqrt{M}$) that a segment computes, and the minimum number of I/O operations required in the segment (Lemma 3.6). We show this relation by bounding from beneath the size of a dominator set of subsets of those output vertices (Lemma 3.7); and the observation that the set of vertices corresponding to the segment should be a dominator set to the outputs which are computed within the segment. Bounding the size of the dominator set is done by a counting argument on the number of paths connecting input vertices with intermediate output vertices (Lemma 3.11).

This is the general framework of the proof, which is similar to [10]. The two main modifications are the proof of Lemma 3.1 (Lemma 7 in [10]), and the addition of memory independent lower bound (for the parallel model). In the proof of Lemma 3.11 (counting argument on the number of paths connecting input vertices with intermediate output vertices) we use a statement on the size of a matching in the encoder graph (Lemma 3.1). This is were we replace the case analysis of Bilardi and De Stefani with a bipartite matching argument, which applies to a broader range of algorithms.

We start with the proof that replaces the case analysis of Bilardi and De Stefani (Lemma 3.1). We then provide the rest of the proof for completeness.

*Lemma 3.1:* Let $G = (X, Y, E)$ denote an encoding bipartite graph of a fast matrix multiplication algorithm with base case of size $2 \times 2$ (see Figure 2 for example). For every set $Y' \subseteq Y$ there exists a set $X' \subseteq X$ of size $|X'| \geq 1 + \lceil (|Y'| - 1)/2 \rceil$, such that there is a matching between $X'$ and $Y'$. Where the set $X$ denotes the 4 input arguments, and the set $Y$ denotes the 7 encoded arguments.

We require two supporting lemmas for the proof of Lemma 3.1:

*Lemma 3.2:* Let $G = (X, Y, E)$ denote an encoder graph of a fast matrix multiplication algorithm with $2 \times 2$ base case. Then, every vertex in $X$ has at least two neighbors, and every two vertices in $X$ have at least 4 neighbors.

*Proof:* Let $T$ be the set of 8 scalar multiplications of the form $a_{i,k} \cdot b_{k,j}$ in the classical matrix multiplication algorithm. Observe that every input argument of matrix $A$ (vertex in $X$) has 2 representations in the set $T$, with different input arguments of matrix $B$, which means that it has at least 2 neighbors. Let the set $X' \subset X$ be of size 2. W.l.o.g., assume that elements of $X'$ correspond to different columns of the input matrix $A$ (otherwise they correspond to different rows and we can switch the roles of the input matrices $A$ and $B$). Observe that every two input arguments of matrix $A$ ($X'$) has 4 representatives in the set $T$ with different input arguments of matrix $B$ (by representative we mean that a vertex of $X'$ multiplied by an input vertex of matrix $B$ appearing in $T$). Thus $X'$ has at least 4 neighbors, as required. ∎

*Lemma 3.3:* Let $G = (X, Y, E)$ denote an encoder graph for a fast matrix multiplication algorithm with $2 \times 2$ base case. There are no two vertices in $Y$ with identical neighbors sets.

We use the following results by Hopcroft and Kerr [21].

*Lemma 3.4:* [21]. If an algorithm for $2 \times 2$ matrix multiplication has k left (right) hand side multiplication from the set $S = [A_{1,1}, (A_{1,2} + A_{2,1}), (A_{1,1} + A_{1,2} + A_{2,1})]$, then

it requires at least $6+k$ multiplications.

*Corollary 3.5:* [21]. Lemma 3.4 also applies for the following definitions of $S$ :

(1) $(A_{1,1}+A_{2,1}), (A_{1,2}+A_{2,1}+A_{2,2}), (A_{1,1}+A_{1,2}+A_{2,2})$

(2) $(A_{1,1}+A_{1,2}), (A_{1,2}+A_{2,1}+A_{2,2}), (A_{1,1}+A_{1,2}+A_{2,2})$

(3) $(A_{1,1}+A_{1,2}+A_{2,1}+A_{2,2}), (A_{1,2}+A_{2,1}), (A_{1,1}+A_{2,2})$

(4) $A_{2,1}, (A_{1,1}+A_{2,2}), (A_{1,1}+A_{2,1}+A_{2,2})$

(5) $(A_{2,1}+A_{2,2}), (A_{1,1}+A_{1,2}+A_{2,2}), (A_{1,1}+A_{1,2}+A_{2,1})$

(6) $A_{1,2}, (A_{1,1}+A_{2,2}), (A_{1,1}+A_{1,2}+A_{2,2})$

(7) $(A_{1,2}+A_{2,2}), (A_{1,1}+A_{2,1}+A_{2,2}), (A_{1,1}+A_{1,2}+A_{2,1})$

(8) $A_{2,2}, (A_{1,2}+A_{2,1}), (A_{1,2}+A_{2,1}+A_{2,2})$

*Proof of Lemma 3.3:* Since Lemma 3.4 and Corollary 3.5 cover all possible linear sums of matrix elements, there must not be two vertices in $Y$ with identical neighbors sets (otherwise the number of multiplications in the base case will exceed 7).

■

We can now complete the proof of Lemma 3.1.

*Proof of Lemma 3.1:* By Hall's Theorem, there exists a matching for $X^{'}$ in $G$ if and only if $\forall w \subseteq X^{'}, |N(w)| \geq |w|$. Observe that $Y^{'} \geq 1 + \lceil (|Y^{'}|-1)/2 \rceil$. Thus if $|N(Y^{'})| \geq 1 + \lceil (|Y^{'}|-1)/2 \rceil$ there exists a set $X^{'} \subseteq X$ where $|X^{'}| = 1 + \lceil (|Y^{'}|-1)/2 \rceil \leq |Y^{'}| = |N(X)|$. By showing this for all possible sizes of $Y^{'}$ we get that $\forall w \subseteq X^{'},\ |W| \leq |N(w)|$ (It follows from previous sizes of $Y^{'}$). Therefore it is sufficient to show that $\forall Y^{'} \subseteq Y,\ |N(Y^{'})| \geq 1 + \lceil (|Y^{'}|-1)/2 \rceil$. We show this for every size of $Y^{'}$:

If $|Y^{'}| = 1$ it is trivial, there must be a vertex in $X$ with an edge to $Y^{'}$.

If $|Y^{'}| = 2$ or $|Y^{'}| = 3$ there must be at least two neighbors since there are no two vertices in $Y$ with the same neighbors set (Lemma 3.3).

If $|Y^{'}| = 4$ or $|Y^{'}| = 5$ we need to show the existence of at least 3 neighbors. Every set $X^{''} \subseteq X$ of size 2 has a neighbor in $Y^{'}$ since it has at least 4 neighbors (Lemma 3.2). Therefore $|N(Y^{'})| \geq 3$ (otherwise there will be a set $X^{''} \subset X$ of size 2 with no neighbor in $Y$, since $|Y^{'}| \geq 4,\ |Y \setminus Y^{'}| \leq 3$, it means that $N(X^{''}) \leq 3$, contradicting Lemma 3.2).

If $|Y^{'}| = 6$ or $|Y^{'}| = 7$ then every vertex in $X$ has a neighbor in $Y^{'}$ (since he has at least 2 neighbors, Lemma 3.2, and $|Y \setminus Y^{'}| \leq 1$). Therefore $|N(Y^{'})| = 4$, as required.

■

For completeness we next provide the rest of the proof which builds on that of [10], with the necessary modifications.

*Lemma 3.6:* Given a positive integer $r$, a segment $S_i$ that computes $r^2$ output vertices of $V_{out}(SUB\_H^{r\times r})$, uses at least $r^2/2 - n_{init}$ I/O operations, where $n_{init}$ denotes the number of values that reside in the cache when segment $S_i$ begins.

We prove Lemma 3.6 later on, but first we show how to use it to prove our main theorem.

*Proof:* (Theorem 1.1). We prove the three bounds (sequential, parallel memory dependent and parallel memory independent) separately.

**Sequential model**. Let $S$ be a computation schedule. Consider only computations that are performed for the first time. Partition $S$ into segments $S_i$, where each $S_i$ (outside perhaps the last one) contains exactly $4M$ output computations of intermediate multiplications of matrices of size $2\sqrt{M} \times 2\sqrt{M}$ (vertices of $V_{out}(SUB\_H^{2\sqrt{M}\times 2\sqrt{M}})$). Denote these vertices by $V_{out}^{i}(SUB\_H^{2\sqrt{M}\times 2\sqrt{M}})$. Let $IO_{s_i}$ denote the number of I/O operation during segment $S_i$. By Lemma 3.6, with $r = 2\sqrt{M}$ and $n_{init} \leq M$ ($n_{init}$ cannot exceed cache size), $IO_{S_i} \geq M$. By Lemma 2.2 there are $(n/2\sqrt{M})^{\omega_0}$ segments, thus:

$$IO(n,M) = \Omega((n/\sqrt{M})^{\omega_0} \cdot M)$$

**Parallel model**. The proof of the parallel model memory dependent lower bound follows a similar strategy. There exists a processor $p_i$ that performs at least $1/P$ of the computations, namely $n^{\omega_0}/P$ computations. By following the proof for the sequential model, there are at least $(n/2\sqrt{M})^{\omega_0}/P$ segments containing $4M$ output computations of intermediate multiplications of matrices of size $2\sqrt{M} \times 2\sqrt{M}$, that performs at least $M$ I/O operations, thus:

$$IO(n,M,P) = \Omega((n/\sqrt{M})^{\omega_0} \cdot M/P)$$

**Memory independent**. The proof of the parallel model memory independent is slightly different. The proof combines the above analysis with the approach of [1]. Assume that at the beginning (respectively at the end) of the algorithm each processor holds the same portion of the input (respectively output) i.e., $2n^2/P$ (respectively $n^2/P$) elements. We partition $S$ into $P$ segments $S_i$, one for computations of each processor, and denote the number of I/O operation in segment $S_i$ by $IO_i$. Let $r = n/P^{1/\omega_0}$. By Lemma 2.2 $|V_{out}(SUB\_H^{r\times r})| = P \cdot (n/P^{1/\omega_0})^2$. Therefore some processor calculates at least $(n/P^{1/\omega_0})^2$ output vertices of $V_{out}(SUB\_H^{r\times r})$. From Lemma 3.6, when assign $r = n/P^{1/\omega_0}$ and $n_{init} = 2n^2/P$, we get that:

$$IO(n,P) = n^2/P^{2/\omega_0} - 2n^2/P = \Omega(n^2/P^{2/\omega_0})$$

■

It remains to prove Lemma 3.6. For this end we require a supporting lemma which we prove later on.

*Lemma 3.7:* Let $Z$ be a subset of $V_{out}(SUB\_H^{r\times r})$ of size $r^2$. Then every dominator set $\Gamma$ of $Z$ in $H^{n\times n}$ satisfies $|\Gamma| \geq |Z|/2$.

*Proof of Lemma 3.6:* Let $\Gamma_i$ denote the vertices of $H^{n\times n}$ that correspond to values that are in the possession of the processor during segment $S_i$. Notice that $IO_{S_i} \geq |\Gamma_i| - n_{init}$ (since argument that are in the possession of the processor during the segment can either reside in the cache when the segment begins, or read during the segment using an I/O operation). Every path from vertex in $V^i_{out}(SUB\_H^{r\times r})$ to vertex in $V_{inp}(H^{n\times n})$ must contain a vertex in $\Gamma_i$, otherwise there will be a value which is needed to the computation but was not in the possession of the processor during the segment. Therefore, by definition, $\Gamma_i$ is a dominator set of $V^i_{out}(SUB\_H^{r\times r})$. From Lemma 3.7 $|\Gamma_i| \geq r^2/2$, therefore:

$$IO_{S_i} \geq r^2/2 - n_{init}$$

■

We now show few more lemmas which required for the proof of Lemma 3.7.

*Lemma 3.8 ( [2]):* $f_{n\times n} : \mathcal{R}^{2n^2} \rightarrow \mathcal{R}^{n^2}$ has a $\omega_{n\times n}(u,v)$ Grigoriev's flow, where:

$$\omega_{n\times n}(u,v) \geq \frac{(v-(2n^2-u)^2/4n^2)}{2}$$
$$for\ 0 \leq u \leq 2n^2,\ 0 \leq v \leq n^2$$

*Lemma 3.9 ( [2]):* Let $G(V,E)$ be a CDAG computing $f : \mathcal{R}^p \rightarrow \mathcal{R}^q$ with Grigoriev's flow $\omega_f(u,v)$. Let $I$ (respectively $O$) denote the set of input (respectively output) vertices of $G$. Any dominator set $\Gamma$ for any subset $I^{'} \subset I$ with respect to any subset $O^{'} \subset O$ satisfies $|\Gamma| \geq \omega_f(|I^{'}|, |O^{'}|)$.

*Proof:* Proof. Given $I^{'} \subset I$ and $O^{'} \subset O$, suppose the values of the input variables corresponding to vertices in $I \setminus I^{'}$ to be fixed. Let $\Gamma$ be a dominator set for $I^{'}$ with respect to $O^{'}$. $(i)$ according to the definition of flow of f, there exists an assignment of the input variables corresponding to vertices in $I^{'}$ such that the output variables in $O^{'}$ can assume $|\mathcal{R}|^{\omega_f(|I^{'}|,|O^{'}|)}$ distinct values. $(ii)$ As there is no path from $I^{'}$ to $O^{'}$ which has not a vertex in $\Gamma$, the values of the outputs in $O^{'}$ are determined by the inputs in $I \setminus I^{'}$, which are fixed, and the values corresponding to the vertices in $\Gamma$; hence the outputs in $O^{'}$ can assume at most $|\mathcal{R}|^{|\Gamma|}$ distinct values. The claimed results follows by a simple combination of observation (i) and (ii).

■

*Lemma 3.10 ( [10]):* Let $G^{q,n\times n}$ be a $CDAG$ composed by q vertex disjoint $CDAGs$ $G^{n\times n}$. Denote I (respectively O) the set of input (respectively output) vertices of $G^{q,n\times n}$. Let $O^{'} \in O$ be a subset of output vertices. For any subset $\Gamma$ of the vertices of $G^{q,n\times n}$ with $|\Gamma| \leq 2|O^{'}|$, the set $I^{'} \subset I$ of input vertices which are not dominated by $\Gamma$ satisfies $|I^{'}| \geq 2n\sqrt{|O'| - 2|\Gamma|}$.

*Proof:* Let $O^{'}_j$ (respectively $\Gamma_j$) denote the subset of $O^{'}$ (respectively $\Gamma$) which corresponds to vertices in the $j$-th sub CDAG $G^{n\times n}_j$ for $j \in 1, 2, \cdots, q$. As, by hypothesis, the sub-CDAGs $G^{n\times n}_q$ are vertex disjoint, $O^{'}_1, O^{'}_2, \cdots, O^{'}_q$ (respectively $\Gamma_1, \Gamma_2, \cdots, \Gamma_q$) constitute a partition of $O^{'}$ (respectively $\Gamma$). Let $I^{''}_j \subset \mathcal{X}$ denote the set of input vertices of $G^{n\times n}_q$ for which $\Gamma_j$ is a dominator set with respect to $O_j$. From Lemma 3.8 and Lemma 3.9 the following condition must hold:

$$|\Gamma_j| \geq \omega_{n\times n} \geq 1/2 \cdot [|O^{'}_j| - (2n^2 - |I^{''}_j|)^2/4n^2]$$

Let $I^{'}_j = I_j \setminus I^{''}_j$ denote the set of input vertices of $G^{n\times n}_j$ for which $|\Gamma_j|$ is a dominator set with respect to $O_j$. Since $|I| = 2n^2$, from Lemma 3.8 we have

$$|I^{'}|^2 = \sum_{j=1}^{q} |I^{'}_j|^2 \geq 4n^2 \sum_{j=1}^{q} (|O^{'}_j| - 2|\Gamma_j|)$$
$$= 4n^2(|O^{'}| - 2|\Gamma|)$$

Notice that all vertices in $I^{'} = \cup^i_{j=1} I^{'}_j$ are not post-dominated by $\Gamma$ (because the sub-CDAGs $G^{n\times n}_j$ are vertex disjoint) and

$$|I^{'}|^2 = \sum_{j=1}^{q} |I^{'}_j|^2 \geq 4n^2 \cdot \sum_{j=1}^{q} (|O^{'}_j| - 2|\Gamma_j|)$$

■

*Lemma 3.11:* Let $\Gamma$ be a subset of $V_{int}(SUB\_H^{r\times r})$, and let $Z$ be a subset of $V_{out}(SUB\_H^{r\times r})$ such that $|Z| \geq 2|\Gamma|$. There exists a set $X \subset V_{inp}(H^{n\times n})$ of size $|X| \geq 2r\sqrt{|Z| - 2|\Gamma|}$, such that vertices in $X$ can be connected to vertices in a set $Y \subset V_{inp}(SUB\_H^{r\times r})$ of size $|Y| = |X|$ via vertex disjoint paths. Furthermore any vertex in $Y$ can be connected to vertex in $Z$ by a directed path which does not include any vertex in $\Gamma$.

*Proof:* The proof is by induction on $n$, the dimension of the input matrices. For the base of the induction we show that the statement holds for $n = r$. The sets $X$ and $Y$ are identical. Therefore, the statement follows immediately from Lemma 3.10 with $q = 1$ and $n = r$. For the inductive step we show that if the statement holds for $H^{n\times n}$ then the statement holds for $H^{2n\times 2n}$ as well. Let $H^{n\times n}_i$ denote the seven sub-graphs of $H^{n\times n}$, corresponding to the seven products (notice that they are vertex disjoint from Lemma 3.3). Let $Z_i$ (respectively $\Gamma_i$) denote the subset of $Z$ (respectively $\Gamma$) in $H^{n\times n}_i$, notice

that these are partition. Let $d_i \equiv \max\{0, |Z_i| - 2|\Gamma_i|\}$, and $d \equiv \sum_{i=1}^{m_0} d_i \geq |Z| - 2|\Gamma|$ (w.l.o.g., assume $d_1 = max_i\ d_i$). Applying the inductive hypothesis on each $H_i^{n\times n}$, there is sets $Y_i \subset V_{inp}(H_i^{n\times n})$ such that $|Y_i| \geq 2r\sqrt{|Z| - 2|\Gamma|}$, and vertices of $Y_i$ can be connected to vertices in $Z_i$ via paths which do not include vertices in $\Gamma_i$. These paths can be connected to sets $K_i \subseteq V_{inp}(H_i^{n\times n})$ such that $|K_i| = |Y_i|$, using vertex disjoint paths. We denote by $K$ the union of all $k_i$. It remains to show that it is possible to extend at least $2r\sqrt{|Z| - 2|\Gamma|}$ of these paths to vertices in $V_{inp}(H^{2n\times 2n})$, preserving vertex disjoint. According to the form of the graph, vertices in $V_{inp}(H^{2n\times 2n})$ are connected to vertices in $K_i$ by $2n^2$ encoder graphs (as defined in Section II). We denote by $Enc_j$ the $j^{th}$ encoder graph. Notice that they do not share any input or output vertices. For each $Enc_j$ consider the vector $y_j \in \{0,1\}^{m_0}$, which satisfied the condition $y_j[i] = 1$ if the $i-th$ output vertex is in $K_i$ and $y_j[i] = 0$ otherwise. Therefore $|y_j|$ is the number of output vertices in $Enc_j$ which are in $K$. From Lemma 3.1 for each $Enc_j$ there exists a subset $X_j \in V_{inp}(Enc_j)$ where each vertex in $X_j$ can be connected to a distinct vertex in $V_{out}(Enc_j)$ by a single edge (and therefore disjoint), and $X_j$ satisfy the condition:

$$|X_j| \geq y_j[1] + \lceil (|y_j| - 1)/2 \rceil$$

The number of vertex disjoint paths connecting vertices in $V_{inp}(H^{2n\times 2n})$ to vertices in $K$ is therefore at least $\sum_{j=1}^{2n^2} |X_j|$. Notice that:

$$|X_j| \geq y_j[1] + 1/2 \sum_{i=2}^{m_0} y_j[i],\ \textit{and}\ \sum_{j=1}^{2n^2} y_j[i] \geq 2r\sqrt{d_i}$$

Since all $Enc_j$ are vertex disjoint, we can sum their contributions and therefore conclude that the number of vertex disjoint paths connecting vertices in $V_{inp}(H^{2n\times 2n})$ to vertices in $K$ is at least:

$$\begin{aligned}\sum_{j=1}^{2n^2} |X_j| &\geq 2r(\sqrt{d_1} + 1/2 \sum_{i=2}^{m_0} \sqrt{d_i}) \\ &\geq 2r\sqrt{(\sqrt{d_1} + 1/2 \sum_{i=2}^{m_0} \sqrt{d_i})^2} \\ &\geq 2r\sqrt{d_1 + \sqrt{d_1} \sum_{i=2}^{m_0} \sqrt{d_i}} \geq 2r\sqrt{d}\end{aligned}$$

Therefore there are at least $2r\sqrt{(|Z| - 2|\Gamma|)}$ vertex disjoint paths connecting vertices in $V_{inp}(H^{2n\times 2n})$ to vertices in $K$. ∎

Now we can prove Lemma 3.7 and conclude our proof of Theorem 1.1.

*Proof of Lemma 3.7:* . Assume by contradiction that $|\Gamma| < |Z|/2$ and let $\Gamma' \subseteq \Gamma \cap V_{inp}(SUB\_H^{r\times r})$. By Lemma 3.11 there exist sets $X \subseteq V_{inp}(H^{n\times n}), Y \subseteq V_{inp}(SUB\_H^{r\times r})$ such that there are at least $2r\sqrt{|Z| - 2|\Gamma'|}$ vertex disjoint paths connecting $X$ to $Y$, and those paths can be extended to $Z$ without passing through $\Gamma'$. Denote such set of paths by $T$. This implies that each of the vertices in $\Gamma \setminus \Gamma'$ can be traversed by at most one of path of $T$. Therefore, the number of paths from $X$ to $Z$ that do not traverse $\Gamma$ vertices is at least:

$$\begin{aligned}2r\sqrt{|Z| - 2|\Gamma'|} - (|\Gamma| - |\Gamma'|) \quad &\geq \\ 2(|Z| - 2|\Gamma'|) - (|Z| - 2|\Gamma'|) \quad &\geq 1\end{aligned}$$

Therefore there is at least one path from $X$ to $Z$ that does not traverse any $\Gamma$ vertices, contradicting $\Gamma$ being a dominator set of $Z$.

∎

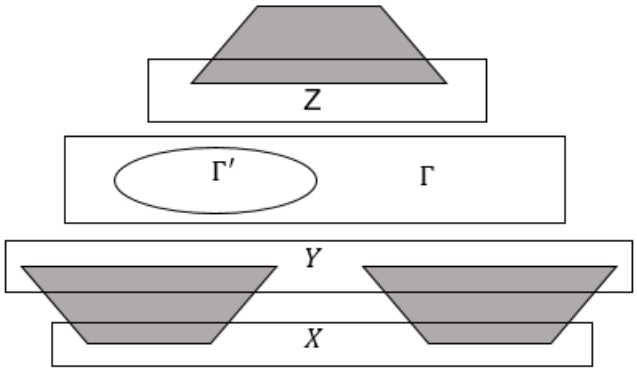


Fig. 3. A graphic representation of Lemma 3.11

## IV. Alternative Basis Matrix Multiplication

Karstadt and Schwartz [20] obtained a method for decreasing the leading coefficient of the arithmetic and I/O-complexity of fast matrix multiplication algorithms. For matrix multiplication algorithm with base case of $2 \times 2$ for instance, they reduced the leading coefficient of the arithmetic computations of Winograd's algorithm from 6 to 5 and that of the I/O-complexity from 10.5 to 9. They did so by introducing fast basis transformation before and after the multiplication part of the algorithm. We show that our bounds in Theorem 1.1 hold for such algorithms as well. The alternative basis matrix multiplication is composed of three parts: Computing $\phi(A), \psi(B)$ using a fast basis transformation algorithm, computing $\nu(C) = \phi(A) \cdot \psi(B)$ using a recursive-bilinear algorithm, and computing $C = \nu^{-1}(\nu(C))$ using a fast reverse basis transformation algorithm. The I/O-complexity of the basis alternation is negligible compared to the multiplication part. Therefore we can deduce the following corollary.

*Theorem 4.1:* Let $ALG$ be a recoursive-bilinear matrix multiplication algorithm of type $\langle 2,2,2;7\rangle_{\phi,\psi,\nu}$. The I/O-complexity of an alternative basis matrix multiplication using $ALG$ (As defined in Section II) is: $\Omega((n/\sqrt{M})^{\log_2 7} \cdot M)$ when running in the sequential model, and

$$\max\{\Omega((n/\sqrt{M})^{\log_2 7} \cdot M/P),\ \Omega(n^2/P^{2/\log_2 7})\}$$

when running in the parallel model, **even when recomputation is allowed**, where $n$, $M$, and $P$ denotes the size of the input matrix, the size of the cache of each processor, and the number of processors, respectively.

*Proof:* Theorem 1.1 apply on recursive matrix multiplication algorithms that their base case is a multiplication of two matrices of size $2 \times 2$ using 7 multiplications. Practically any recursive algorithm that is base case starts with encoding two matrices of size $2 \times 2$ and ends with decoding a matrix of size $2 \times 2$, from 7 arguments, is acceptable. Therefore Theorem 1.1 applies on algorithm $ALG$ as well. Since the I/O complexity of the alternative basis matrix multiplication algorithm is dominated by the I/O complexity of his recursive-bilinear matrix multiplication algorithm ($ALG$), the same hold for such type of algorithms as well. ∎

*Corollary 4.2:* Recomputation can not decrease communication cost (asymptotically) of alternative basis matrix multiplication algorithm with base case of size $2 \times 2$.

## V. DISCUSSION AND OPEN PROBLEMS

Most of the existing lower bounds assume no recomputation. Thus, they do not exclude the possibility of more efficient algorithms, in terms of communication costs, which do take advantage of recomputations. The lower bounds of Bilardi and De Stefani (2017), Bilardi, Scquizzato, and Silvestri (2018), as well as ours, show that recomputation is sometimes useless for asymptotically reducing the I/O-complexity. But this is not always the case. Recomputation can be useful for reducing the I/O-complexity for some $CDAG$s [16]. Moreover, some $CDAGs$ admit an optimal schedule only when recomputation is allowed [17]. We conjecture that recomputation cannot reduce communication cost (asymptotically) for any fast matrix multiplication algorithm, and for direct linear algebra algorithms.

In non volatile memory technologies, a write operation may be much more expensive than a read operation. Therefore algorithms that minimize write operations are likely to be more efficient [24]–[28]. Blelloch et al. [26] showed that for the minimum edit distance problem (and related problems), recomputation can reduce the number of writes (at the cost of additional reads). Whether one can trade recomputation for writing in other problems remains an open question.

## VI. ACKNOWLEDGEMENT

Research was supported by grants 1878/14, and 1901/14 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities). This work was supported by the PetaCloud industry-academia consortium. This research was supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel. This work was supported by The Federmann Cyber Security Center in conjunction with the Israel national cyber directorate.

## APPENDIX

Let $ALG$ be a recursive-bilinear, $\langle n, m, k; t \rangle_{\phi,\psi,\nu}$-algorithm. Alternative basis matrix multiplication [20] works as follows:

**Algorithm 1** Alternative Basis Matrix Multiplication

**Input:** $A \in R^{n \times m},\ B \in R^{m \times k},\ ALG \in \langle n, m, k; t \rangle_{\phi,\psi,\nu}$
**Output:** $C = A \cdot B,\ C \in R^{n \times k}$

**function** ABMM(A,B)
  $\tilde{A} = \phi(A),\ \tilde{B} = \psi(B)$
  $\tilde{C} = ALG(\tilde{A}, \tilde{B})$
  $C = \nu^{-1}(\tilde{C})$
**return** C

Where $\phi(A)$, $\psi(B)$, and $\nu^{-1}(C)$ are computed using fast basis transformation techniques.

A recursive fast matrix multiplication algorithm using Strassen's matrix multiplication [18] works as follow:

**Algorithm 2** Recursive Strassen's Matrix Multiplication

**Input:** $A, B \in R^{n \times n}$ {We assume $n = 2^k$ for some $k$}
**Output:** $C = A \cdot B$ ($C \in R^{n \times n}$)

**function** RSMM($A, B, n$)
  **if** $n = 1$ **then**
    $C = A \cdot B$

  **else**
    Decompose $A,\ B$ into four sub matrices as follow:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix},\ B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$$

$$\begin{aligned}
M1 &= \text{RSMM}(A_{1,1} + A_{2,2},\ B_{1,1} + B_{2,2},\ n/2) \\
M2 &= \text{RSMM}(A_{2,1} + A_{2,2},\ B_{1,1},\ n/2) \\
M3 &= \text{RSMM}(A_{1,1},\ B_{1,2} - B_{2,2},\ n/2) \\
M4 &= \text{RSMM}(A_{2,2},\ B_{2,1} - B_{1,1},\ n/2) \\
M5 &= \text{RSMM}(A_{1,1} + A_{1,2},\ B_{2,2},\ n/2) \\
M6 &= \text{RSMM}(A_{2,1} - A_{2,2},\ B_{1,1} + B_{1,2},\ n/2) \\
M7 &= \text{RSMM}(A_{1,2} - A_{2,2},\ B_{2,1} + B_{2,2},\ n/2)
\end{aligned}$$

$$\begin{aligned}
C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\
C_{1,2} &= M_3 + M_5 \\
C_{2,1} &= M_2 + M_4 \\
C_{2,2} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

    Compose $C$ as follow:

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

  **return** $C$

## REFERENCES


[1] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz, "Strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds," in *Proceedings of the 24th ACM symposium on Parallelism in algorithms and architectures*, 2012, pp. 77–79.

[2] 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*, ser. STOC '81. New York, NY, USA: ACM, 1981, pp. 326–333.

[3] D. Irony, S. Toledo, and A. Tiskin, "Communication lower bounds for distributed-memory matrix multiplication," *Journal of Parallel and Distributed Computing*, vol. 64, no. 9, pp. 1017–1026, 2004.

[4] A. Aggarwal, A. Chandra, and M. Snir, "Communication complexity of prams," *Theoretical Computer Science*, vol. 71, no. 1, pp. 3–28, 1990.

[5] M. Scquizzato and F. Silvestri, "Communication lower bounds for distributed-memory computations," in *Proceedings of the 31st Symposium on Theoretical Aspects of Computer Science (STACS)*, 2014, pp. 627–638.

[6] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Minimizing communication in numerical linear algebra," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 3, pp. 866–901, 2011.

[7] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz, "Communication lower bounds and optimal algorithms for numerical linear algebra," *Acta Numerica*, vol. 23, pp. 1–155, 2014.

[8] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Graph expansion and communication costs of fast matrix multiplication," *Journal of the ACM (JACM)*, vol. 59, no. 6, p. 32, 2012.

[9] J. Scott, O. Holtz, and O. Schwartz, "Matrix multiplication i/o-complexity by path routing," in *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '15. New York, NY, USA: ACM, 2015, pp. 35–45.

[10] G. Bilardi and L. De Stefani, "The I/O complexity of strassen's matrix multiplication with recomputation," *Algorithms and Data Structures*, vol. 10389, pp. 181–192, 2017.

[11] G. Bilardi, M. Scquizzato, and F. Silvestri, "A lower bound technique for communication on bsp with application to the FFT," in *Proceedings of the European Conference on Parallel Processing*, vol. 7484. Springer, 2012, pp. 676–687.

[12] G. Ballard, J. Demmel, A. Gearhart, B. Lipshitz, Y. Oltchik, O. Schwartz, and S. Toledo, "Network topologies and inevitable contention," in *Proceedings of the First Workshop on Optimization of Communication in HPC*, ser. COM-HPC '16. IEEE Press, 2016, pp. 39–52.

[13] G. Bilardi, M. Scquizzato, and F. Silvestri, "A lower bound technique for communication in bsp," *ACM Transactions on Parallel Computing*, vol. 4, no. 14, 2018.

[14] A. Aggarwal and J. S. Vitter, "The input/output complexity of sorting and related problems," *Communications of the ACM*, vol. 31, no. 9, pp. 1116–1127, 1988.

[15] M. Driscoll, E. Georgamas, P. Koanantakool, and K. Yelick, "A communication-optimal n-body algorithm for direct interactions," in *Proceedings of the 27th International Symposium on Parallel and Distributed Processing (IPDPS)*. IEEE, 2013, pp. 1075–1084.

[16] J. E. Savage, "Extending the hong-kung model to memory hierarchies," in *In Computing and Combinatorics. Lecture Notes in Computer Science*, vol. 959. Springer, Berlin, Heidelberg, 1995, pp. 270–281.

[17] G. Bilardi and E. Peserico, "A characterization of temporal locality and its portability across memory hierarchies," *Automata, Languages and Programming*, pp. 128–139, 2001.

[18] V. Strassen, "Gaussian elimination is not optimal," *Theoretical Computer Science*, vol. 13, no. 4, pp. 354–356, 1969.

[19] S. Winograd, "On multiplication of 2x2 matrices," *Linear algebra and its applications*, vol. 4, no. 4, pp. 381–388, 1971.

[20] E. Karstadt and O. Schwartz, "Matrix multiplication, a little faster," in *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '17. New York, NY, USA: ACM, 2017, pp. 101–110.

[21] J. E. Hopcroft and L. R. Kerr, "On minimizing the number of multiplications necessary for matrix multiplication," *SIAM Journal on Applied Mathematics*, vol. 20, no. 1, pp. 30–36, 1971.

[22] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz, "Graph expansion analysis for communication costs of fast rectangular matrix multiplication," in *Design and Analysis of Algorithms*, vol. 7659. Springer, Berlin, Heidelberg, 2012, pp. 13–36.

[23] P. Hall, "On representatives of subsets," *Journal of the London Mathematical Society*, vol. s1-10, no. 1, pp. 1–80, 1935.

[24] E. Carson, J. Demmel, L. Grigori, N. Knight, P. Koanantakool, O. Schwartz, and H. V. Simhadri, "Write-avoiding algorithms," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2016, pp. 648–658.

[25] N. Ben-David, G. E. Blelloch, J. T. Fineman, P. B. Gibbons, Y. Gu, C. McGuffey, and J. Shun, "Parallel algorithms for asymmetric read-write costs," in *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 2016, pp. 145–156.

[26] G. E. Blelloch, J. T. Fineman, P. B. Gibbons, Y. Gu, and J. Shun, "Efficient algorithms with asymmetric read and write costs," in *Proceedings of the 24th Annual European Symposium on Algorithms (ESA 2016)*, vol. 57, 2016, pp. 14:1–14:18.

[27] N. Ben-David, G. E. Blelloch, J. T. Fineman, P. B. Gibbons, Y. Gu, C. McGuffey, and J. Shun, "Implicit decomposition for write-efficient connectivity algorithms," *2018 IEEE International Parallel and Distributed Processing Symposuim (IPDPS)*, 2018.

[28] G. E. Blelloch, Y. Gu, Y. Sun, and J. Shun, "Parallel write-efficient algorithms and data structures for computational geometry," *CoRR*, vol. abs/1805.05592, 2018.