The I/O Complexity of Sparse Matrix Dense Matrix Multiplication

Gero Greiner   Riko Jacob

April 22, 2010

Abstract

We consider the multiplication of a sparse $N \times N$ matrix $\mathbf{A}$ with a dense $N \times N$ matrix $\mathbf{B}$ in the I/O model. We determine the worst-case non-uniform complexity of this task up to a constant factor for all meaningful choices of the parameters N (dimension of the matrices), k (average number of non-zero entries per column or row in $\mathbf{A}$, i.e., there are in total kN non-zero entries), M (main memory size), and B (block size), as long as $M \geq B^2$ (tall cache assumption).

For large and small k, the structure of the algorithm does not need to depend on the structure of the sparse matrix $\mathbf{A}$, whereas for intermediate densities it is possible and necessary to find submatrices that fit in memory and are slightly denser than on average.

The focus of this work is asymptotic worst-case complexity, i.e., the existence of matrices that require a certain number of I/Os and the existence of algorithms (sometimes depending on the shape of the sparse matrix) that use only a constant factor more I/Os.

1 Introduction

Traditionally, the aim of a good algorithmic design is to achieve a task with as few CPU-operations as possible. In a setting where the main calculation is phrased as matrix and vector operations, this usually means that the matrices are kept sparse and it is exploited that many entries are zero. This reduction of CPU-operation sometimes comes at the price of irregular access patterns induced by the sparse matrix operations, which can lead to a situation where memory access is the real bottleneck of the computation.

One successful way of modeling this bottleneck is the so called I/O-model (also known as Disk-Access-Model DAM) introduced by Aggarwal and

Vitter [1]. It assumes that the CPU can only operate on a main memory of size M whereas further intermediate results (just like input and final result) must be stored on an infinite disk, that is organized in blocks of size B. The resulting performance measure counts the number of read/write operations of the disk, the so called I/O-operations (or I/Os for short). By now this model is accepted and the I/O-complexity of many tasks is well understood.

We consider the multiplication of a sparse $N \times N$ matrix $\mathbf{A}$ containing kN non-zero entries ($\mathbf{A}$ is called k-sparse) with a dense $N \times N$ matrix $\mathbf{B}$, computing $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$. Throughout the paper, we abbreviate this task as SDM_k. We study the worst case complexity of this task in the I/O-model, i.e., we determine up to some constant factor the number of I/Os that are necessary and sufficient to compute $\mathbf{C}$. Here, we use the so-called semiring I/O-machine, where algorithms can only use addition and multiplication, but cannot rely upon subtraction or division (as detailed in Section 2 "Model of Computation"). We consider worst-case complexities, where the worst case is taken over the shape (or conformation) of the matrix $\mathbf{A}$ as given by the positions of the non-zero entries. Our notion of an algorithm is non-uniform in the sense that we ask for a program that, depending on the conformation of $\mathbf{A}$, computes $\mathbf{C}$ with few I/Os irrespective of the complexity to create this program.

This model of computation and notion of sparseness has been successfully used in [3] to study multiplying a sparse matrix with a dense vector. It also coincides with the notion of "independent evaluation" of Hong and Kung [5] to study the multiplication of two dense matrices. In this case (in our notation $k = N$), we do know that this is a restriction as it disallows algorithms like Strassen's. However, the algebraic complexity of dense matrix multiplication is still unknown [7]. In this computational model, our notion of k-sparseness of $\mathbf{A}$ has the effect that the matrix multiplication requires precisely kN^2 multiplications of numbers.

In this paper, we determine the complexity to multiply two $N \times N$ matrices, one of which is k-sparse in the semiring I/O-model with tall cache $M \geq B^2$ to be

$$\Theta\left(\max\left\{\frac{kN^2}{B\Delta}, \frac{kN^2}{B\sqrt{M}}, \frac{N^2}{B}, 1\right\}\right)$$

where

$$\Delta = \max\left\{\frac{\ln \frac{N}{M}}{\overline{\ln}\, \frac{N \ln^2 \frac{N}{M}}{Mk}}, \sqrt{\frac{kM}{N}}\right\},$$

and $\overline{\ln}$ is defined by $\overline{\ln}\, x = \max\{1, \ln x\}$.

This expression yields three interesting ranges of the parameter k, but for all k the I/O-complexity boils down to the question how many of the kN^2

elementary products can be performed on M elements that are simultaneously in memory (i.e. in one so called Hong-Kung round). For large k the situation is basically that of two dense matrices, in particular, for $k = N$ it coincides with the classical result of Hong and Kung [5] that multiplying two dense square matrices has complexity $\Theta\left(\frac{N^3}{B\sqrt{M}}\right)$, i.e, that at most $M^{\frac{3}{2}}$ multiplications per round are possible and can be achieved by using $\sqrt{M} \times \sqrt{M}$ tiles. For small k it resembles that of $\mathbf{A}$ being a permutation matrix where M multiplications per round are best possible (i.e. loaded elements cannot be reused).

Additionally, there is a density from which point on the complexity (reuse of loaded operands) can be described by above average dense submatrices consisting of M entries and having on average $\min\{\Delta, \sqrt{M}\}$ entries per row and column. Our complexity analysis proceeds by showing that there exist matrices that have essentially no denser submatrices. We get a matching upper bound by showing that every matrix that has sufficiently many entries must have such dense submatrices. The resulting algorithm hence depends upon the conformation (shape) of the sparse input matrix in a complicated manner (which does not influence the theoretical statement). How difficult it is to actually compute a good program is not completely understood. We only have preliminary results [8] showing that finding such a program is NP-complete, and that determining the maximum possible density cannot be approximated within an arbitrarily small constant factor. This limited structural insight is already more than what is known for the multiplication of a sparse matrix with a dense vector [3], where it is only clear that difficult matrices exist (by a counting argument), but there is no characterization of which (permutation) matrices are difficult to multiply with. One key difference in the consideration of this paper and [3] is that here the block size B is basically irrelevant (it is only the scaling factor to translate I/O-volume to number of I/Os), whereas matrix vector multiplication becomes trivial if $B = 1$.

For the sake of clarity, throughout the paper we stick to the case where all matrices are square. The results naturally generalize to non-square situations as long as the smallest dimension (side-length) is at least $\sqrt{M} \geq B$.

Clearly, the results presented here are theoretical in nature, and the presented algorithms are stated more to complement the lower bounds than to be implemented. The real goal is to devise and evaluate practical algorithms, and the results presented here give important limits on what to expect from them. Further, such practical algorithms exist and are implemented in many sparse matrix libraries, which indicates that the considered problem is of a certain practical relevance. Also there it has been recognized that memory access patterns are an important factor in the overall execution time, and memory aware algorithms have been proposed [2, 9, 4].

Here, it is worth noting that our notion of k-sparseness fits to the established experimental performance measure "number of floating point operations performed per second". Because the number of floating point operations is precisely kN^2 this immediately translates to running time, and hence if memory is the bottleneck, to number of I/Os.

In contrast to the mentioned practical considerations, our work is theoretical in nature, with all the well known consequences: The presented results are mathematical theorems, stating all assumptions and having a clear conclusion. This generality of course comes at a price, in our context mainly the abstraction of the model of computation (neglecting everything but the memory access patterns of a program, and disallowing Strassen-like algorithms) and the focus on the worst-case input (where practitioners can and have to exploit the special structure in the input at hand). Nevertheless, we believe that our theoretical findings and understanding give an important reference point also for the practical work: What are the difficult instances? In what respect are practical inputs easy? What is a good worst-case behavior of an algorithm?

Outline of the paper. We introduce the precise model of computation in Section 2. This is followed by Section 3 where some easy to derive inequalities, so called Observations are given. In Section 4 we describe the different algorithms, depending on the density parameter k, whereas Section 5 shows that there exist matrices that require the stated number of I/Os.

2 Model of Computation

We consider the number of I/Os induced by a program as a measurement of costs. Therefore, we use the model described in [3] which consists of two memory layers, as is standard. We assume a single processing unit with a fast memory of limited capacity M assigned to it. Calculations can only be performed on the elements residing in this *internal memory*, whereas the programs input and any (intermediate) results are stored on an *external memory* of infinite size which is organized in blocks of size B. Elements are moved between memory layers in blocks, where the movement of a block incurs costs 1.

Memory elements are to belong to a *commutative semiring* $\mathbb{S} = (\mathbb{R}, +, \cdot)$, i.e., a set $\mathbb{R}$ with operations addition ($+$) and multiplication ($\cdot$) that are associative, distributive and commutative. Further, there is a neutral element 0 for addition, 1 for multiplication and 0 is annihilating with respect to multiplication. In contrast to rings and fields, inverse elements are neither guaranteed for addition nor for multiplication, i.e., the program is not allowed to use subtraction and division.

Definition [3] The *semiring I/O machine* consists of an internal memory which can hold up to M elements of a commutative semiring $\mathbb{S}$, and an external memory of infinite size which is organized in blocks of B consecutive elements. The current *configuration* of a machine is described by the content $\mathcal{M} = (m_1, \ldots, m_M)$, $m_i \in \mathbb{R}$ of internal memory, and an infinite sequence of blocks $t_i \in \mathbb{R}^B$, $i \in \mathbb{N}$ of external memory. An *operation* is a transformation of one configuration into another, which can be one of the following types

- *Computation*, performs any operation of $\mathbb{S}$ on elements in $\mathcal{M}$.
- *Input*, replaces some chosen elements $m_{i_1}, \ldots, m_{i_B}$ in $\mathcal{M}$ by a block t_i.
- *Output*, replaces a block t_i of external memory by some chosen elements $m_{i_1}, \ldots, m_{i_B}$ of $\mathcal{M}$.

Using this, we define a *program* P as a finite sequence of operations. The number of input and output operations describes the I/O costs of P. An *algorithm* is a family of programs where the program can be chosen according to the parameters N, k, and the *conformation* of $\mathbf{A}$, i.e., the position of the non-zero entries in $\mathbf{A}$. By $L(k, N)$, we denote the required number of I/Os induced by an algorithm for SDM_k with $N \times N$ matrices, and kN non-zero entries in $\mathbf{A}$ for all kinds of conformations.

The value of an element c_{ij} of the result matrix $\mathbf{C} := \mathbf{A} \cdot \mathbf{B}$ is given by $\sum_{l \in A_i} a_{il} \cdot b_{lj}$ where $A_i \subseteq \{1, \ldots, N\}$ describes the positions of non-zero elements in the i-th row of $\mathbf{A}$. Since our model is based on a semiring, the computation of $\mathbf{C}$ includes the calculation of exactly kN^2 *elementary products* $a_{il} \cdot b_{lj}$. Further, any intermediate result can be seen as a sum $\sum_{l \in S} a_{il} \cdot b_{lj}$ for a subset $S \subseteq A_i$. If $|S| < |A_i|$, we refer to this as a *partial result* of c_{ij}.

Since we can assume that every program requires at least one I/O, when writing complexity using $\mathcal{O}$, Θ, or Ω at least 1 is meant.

3 Math

For the proofs provided in Section 4 and 5, the following Observations are necessary.

Observation 1. *For* $0 \leq x \leq 1/2$*, it holds that* $\ln(1 - x) \geq -2x$.

Observation 2. *For* $0 < a \leq e$*, for any* $x > 0$ *it holds* $x \geq a \ln x$.

Observation 3. *For $x \geq y \geq 1$ it holds*

$$\left(\frac{x}{y}\right)^y \overset{(a)}{\leq} \binom{x}{y} \overset{(b)}{\leq} \left(\frac{ex}{y}\right)^y .$$

Observation 4. *For $n \geq k \geq a \geq 1$ it holds*

$$\binom{n}{k} \geq \left(\frac{n-k}{k}\right)^a \binom{n}{k-a}.$$

Proof. By definition of binomial coefficients

$$\binom{n}{k} \cdot \binom{n}{k-a}^{-1} = \frac{n!(n-k+a)!(k-a)!}{(n-k)!k!n!} = \prod_{i=1}^{a} \frac{n-k+i}{k-a+i}.$$

By showing that for each $1 \leq i \leq a$, $\frac{n-k+i}{k-a+i} \geq \frac{n-k}{k}$ the statement follows. This is equivalent to $(a-2i)k \leq (a-i)n$ which holds for any $1 \leq k \leq n$ and $1 \leq i \leq a$. □ □

Observation 5. *For $D, f, x > 0$, the inequality $D \ln fD \leq x$ is satisfied for $D \leq \frac{x}{\overline{\ln}\, xf}$ where $\overline{\ln}\, x = \max\{1, \ln x\}$.*

Proof. By substitution, we obtain

$$D \ln fD \leq \frac{x}{\overline{\ln}\, xf} \ln\left(f \frac{x}{\overline{\ln}\, xf}\right) \leq \frac{x}{\overline{\ln}\, xf} \ln\left(f \frac{x}{1}\right) \leq x.$$

□ □

Observation 6. *For $D, f, x \geq 0$, the inequality $D \ln fD > x$ is fulfilled if $D > \frac{2x}{\ln 2xf}$.*

Proof. Substituting D yields

$$D \ln fD > \frac{2x}{\ln 2xf} \ln\left(f \frac{2x}{\ln 2xf}\right) \geq \frac{2x}{\ln 2xf} \ln \sqrt{2xf} = x$$

where we use $\sqrt{2xf} \geq 2 \ln \sqrt{2xf}$ given by Observation 2. □ □

4 Algorithms

Theorem 1. *For $1 \leq k \leq N$, SDM_k is possible with*

$$\mathcal{O}\left(\max\left\{\frac{kN^2}{B\Delta}, \frac{kN^2}{B\sqrt{M}}, \frac{N^2}{B}\right\}\right)$$

I/Os for

$$\Delta = \max\left\{\frac{\ln\frac{N}{M}}{\ln\frac{N\ln^2\frac{N}{M}}{Mk}}, \sqrt{\frac{kM}{N}}\right\}$$

if $M = \Omega\left(B^2\right)$. Note that Δ is lower bounded by a constant.

For the sake of clarity, we omit the use of ceiling functions from now on. Since all the fractions are greater or equal 1, this only increases the bounds by constant factors.

4.1 Layouts

For the algorithms presented, we either need the dense matrices $\mathbf{B}$ and $\mathbf{C}$ in column major layout where elements are saved column wise, or row major layout. As we prove by lower bounds, a different layout chosen by the algorithm does not lead to an asymptotic speed up. For the first and the third algorithm presented in this section, a row major layout is required, i.e., elements are saved row wise in external memory. If $\mathbf{B}$ is saved in column major layout, the matrix has to be transposed first. In [1], Aggarwal and Vitter showed that this is possible with $\mathcal{O}\left(N^2/B\right)$ I/Os assuming a tall cache, i.e., $M \geq B^2$. Similarly, if required, $\mathbf{C}$ has to be transposed in the end. For the tile-based approach, the matrices are read in tiles. However, assuming a tall cache, the algorithm is applicable if $\mathbf{B}$ is in column major layout. This also applies for small instances where $M \geq kN + N$.

For all algorithms presented, we assume that $\mathbf{A}$ is a list of the non-zero entries in arbitrary order. For the direct algorithm, the ordering is indeed unimportant, whereas the desired layout for the other algorithms can be obtained by sorting. Sorting the elements of $\mathbf{A}$ is possible with $\mathcal{O}\left(\frac{kN}{B}\log_{M/B}\frac{kN}{M}\right)$ I/Os [1]. Since $k \leq N$, this is at most $2c\frac{kN}{B}\log_{M/B}\frac{N}{\sqrt{M}} \leq c\frac{kN^2}{B\sqrt{M}}$ since $M \geq 2B$. Note that this is less than the number of I/Os required for the computation itself.

4.2 Direct algorithm

The direct algorithm for permuting can simply be extended to any $1 \leq k \leq N$. Let b_i be the i-th row of $\mathbf{B}$, and c_i the i-th row of $\mathbf{C}$. By one scan of $\mathbf{A}$ while adding for each non-zero entry a_{ij} the product $a_{ij} \cdot b_j$ to c_i, the result matrix $\mathbf{C}$ is computed with $\mathcal{O}\left(\frac{kN^2}{B}\right)$ I/Os. As we will see, for $k \leq (N/M)^{1-\epsilon}$ and any constant $\epsilon > 0$ this algorithm is asymptotically optimal since Δ in Theorem 1 becomes a constant.

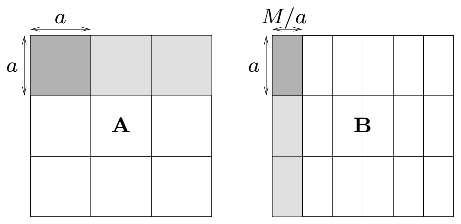

Figure 1: An illustration of the tiles in **A** and **B**. **C** is partitioned similarly to **B**.

4.3 Tile-based algorithm

For denser cases of **A**, there is a modification of the tile-based algorithm in [6] that clearly outperforms the direct algorithm. More precisely, this algorithm works for any $k \geq \frac{N}{M}$ and $M \leq kN$. For the ease of notation, let $3M$ be the size of internal memory. For this approach, matrices **B** and **C** have to be partitioned into tiles of size $a \times M/a$ for $a = \sqrt{MN/k}$, while **A** is partitioned into tiles of size $a \times a$ (cf. Fig. 1). Let $\mathbf{A}_{ij}$,$\mathbf{B}_{ij}$, and $\mathbf{C}_{ij}$ denote the j-th tile within the i-th tile row of the corresponding matrix. Clearly, it holds $\mathbf{C}_{ij} = \sum_{l=1}^{n} \mathbf{A}_{il}\mathbf{B}_{lj}$ for $n = N/a$. Throughout the calculation of a certain $\mathbf{C}_{ij}$, partial results can be kept in internal memory while $\mathbf{A}_{il}$ and $\mathbf{B}_{lj}$ are loaded consecutively for each l. Since each **B**-tile contains exactly M elements, each such tile can be loaded in a whole and only once per calculation of a $\mathbf{C}_{ij}$. The same holds for tiles of **A** containing at most M entries. For tiles with more non-zero entries, elements are loaded in bunches of M elements while the corresponding **B**-tile is kept in memory.

Hence, for saving all **C**-tiles no more than $\frac{N^2}{B}$ I/Os are required. Loading **B**-tiles costs at most $\frac{N^2}{M} \cdot n\frac{M}{B}$ I/Os. Each tile, and thus, each element in **A** has to be loaded for the calculation of no more than $N/\frac{M}{a}$ tiles of **C**. Therefore, loading the non-zero elements of **A** requires at most $\frac{kN}{B}\frac{N\sqrt{N}}{\sqrt{kM}}$ I/Os. Altogether, this sums up to

$$\mathcal{O}\left(\sqrt{\frac{kN}{M}}\frac{N^2}{B}\right)$$

I/Os for the computation of **C**.

4.4 Using dense parts of A

In this section, we show that by loading M elements from each matrix, even for $k < \frac{N}{M}$ a number of $\omega\left(M\right)$ elementary products can be obtained, i.e., more than the direct algorithm achieves. This is done by loading dense parts from $\mathbf{A}$.

For the sake of illustration, we consider the matrix $\mathbf{A}$ as an adjacency matrix of a bipartite graph $G = (U \cup V, E)$, where $a_{ij} \neq 0$ constitutes a connection between the i-th node of U and the j-th node of V. If there are sufficiently many subgraphs containing $\mathcal{O}(M)$ edges, with average degree $\Omega(D)$, SDM$_k$ is possible in time $\mathcal{O}\left(\frac{kN^2}{BD}\right)$.

Lemma 1. *Given a bipartite graph $G = (U \cup V, E)$, $|U| = |V| = N$, $|E| = kN$, i.e., k is the average degree.*

Then, for $2 \leq k \leq \frac{N}{32M} \ln^2 \frac{N}{M}$ and $M \leq N/4$ there exist two subsets $X \subseteq U$, $Y \subseteq V$, such that the subgraph induced by X and Y has average degree at least

$$D = \min \left\{ \frac{\ln \frac{N}{M}}{2 \ln \frac{N \ln^2 \frac{N}{M}}{4Mk}}, \sqrt{\frac{M}{2}} \right\}$$

and it holds that $|X|, |Y| \leq M/D$. This D satisfies $D \leq k/2$.

Before showing this, we need the following preliminary lemma.

Lemma 2. *For $k \leq \frac{N}{32M} \ln^2 \frac{N}{M}$, $M \leq N$, and D according to Lemma 1 the inequality*

$$k \leq \frac{ND}{4M} \ln \frac{N}{M} \tag{1}$$

is satisfied.

Proof. For $D = \frac{\ln \frac{N}{M}}{2 \ln \frac{N}{4Mk} \ln^2 \frac{N}{M}}$, (1) holds as follows. Substituting D yields

$$k \ln \frac{N \ln^2 \frac{N}{M}}{4Mk} \leq \frac{N}{8M} \ln^2 \frac{N}{M}.$$

Observe, that for $\frac{x}{k} \geq e$, the term $k \ln \frac{x}{k}$ for $x > 0$ is monotonously increasing in k. Its derivative is $\ln \frac{x}{k} - 1$, and hence, positive for $\frac{x}{k} \geq e$. Since by assumption $\frac{N \ln^2 \frac{N}{M}}{4Mk} \geq e$, we can substitute k resulting in

$$\frac{N \ln 8}{32M} \ln^2 \frac{N}{M} \leq \frac{N}{6M} \ln^2 \frac{N}{M}$$

which is obviously true.

If the second term of the minimum in D applies, i.e. $D = \sqrt{M/2}$, we have to distinguish the cases $\frac{N}{32M} \ln^2 \frac{N}{M} \leq N$, i.e., $M \geq \frac{\ln^2 \frac{N}{M}}{32}$ and vice versa, i.e. only $k \leq N$ has to hold. By substituting D and k in (1) both cases hold within the desired range. □ □

Lemma 1. In order to make a statement about the minimal degree of a node, we transform G such that the maximal degree in V is restricted to at most k. Therefore, split each node $v_i \in V$ with degree $d_i > k$ into $v_{i,1}, \dots, v_{i,\lceil d_i/k \rceil}$ such that each node has degree no more than k. Let V' denote this transformation of V, E' the transformed set of edges and $G' = (U \cup V', E')$ the created graph. By construction, the size of V will increase by no more than N, i.e., $|V'| \leq 2N$. From this, we can conclude that there are at least $N/2$ nodes with degree no less than $k/2$: Suppose that c nodes in the original set V have degree less than $k/2$. Hence, the degrees of the remaining $N - c$ nodes sum up to at least $(N - c/2)k$. By construction of V', for each node with degree $d_i > k$, there will be at most one new node with degree less than $k/2$. This leads to no less than $(N - c/2)k - (N - c)k/2 = kN/2$ edges that have to belong to nodes with degree at least $k/2$. Since all nodes have degree at most k, there have to be at least $N/2$ nodes of degree at least $k/2$. We call the subset of these nodes $V'_{k/2}$.

Observe that any subgraph G'_S in G' with average degree D consisting of nodes $X \subseteq U$ and $Y \subseteq V'$ can be transformed into a subgraph G_S of G with average degree at least D by simply replacing any $v_{i,j} \in Y$ by the corresponding node v_i of the original graph. The subgraph G_S contains at least the amount of edges of G'_S, but no more nodes than G'_S. Hence, it suffices to show the existence of the desired X and Y for G'. To this end, we will prove that in G' for a random $X \subseteq U$, $|X| = M/D$, the expected number of nodes in V' that have degree $\geq D$ into X is at least M/D.

Now, choose $X \subseteq U$ uniformly at random and consider a vertex $v_i \in V'_{k/2}$. The number of vertices chosen for X in the neighborhood of v_i is given by a hypergeometric distribution, resembling drawing at least $k/2$ times without replacement from an urn with N marbles, M/D of which are black. The event we are interested in is that at least D of the drawn marbles are black.

We lower bound this probability by considering only the case of drawing precisely D black marbles. The probability can be expressed in the following way: Consider drawing the $k/2$ marbles one after another, and choose precisely D positions where black marbles are drawn. The probability of such a drawing can then be calculated as the product of the fractions of black (white) marbles that are left in the urn before each drawing. For black marbles the fraction is at least $p = (\frac{M}{D} - D)/N$, for white it is at least $q = 1 - \frac{M}{D}/(N - \frac{k}{2})$. In the following, we use $D \leq \sqrt{M/2}$, i.e., $D \leq \frac{M}{2D}$, and $k \leq N$ to simplify these expressions.

The overall probability of drawing D black marbles can then be bounded by summing the probabilities of all possible choices to position the D black marbles in the consecutive drawing. Let X_i be the number of black marbles drawn, i.e. the number of edges from v_i into X. Thus, we can lower bound the probability similar to a binomial distribution:

$$\mathrm{P}\left(X_i \geq D\right) \geq \binom{k/2}{D} p^D q^{k/2-D} \geq \left(\frac{k}{2D}\frac{M}{2DN}\right)^D \left(1-\frac{2M}{DN}\right)^{k/2}.$$

Taking logarithm we get

$$\ln \mathrm{P}\left(X_i \geq D\right) \geq D \ln \frac{Mk}{4ND^2} + \frac{k}{2}\ln\left(1-\frac{2M}{ND}\right) \geq D\ln\frac{Mk}{4ND^2} - k\frac{2M}{ND}$$

where the last inequality is justified by Observation 1 and $4M \leq N$.

Since we consider at least $N/2$ nodes, the goal is now to choose the biggest D satisfying

$$D \ln \frac{4ND^2}{Mk} + k\frac{2M}{ND} \leq \ln \frac{ND}{2M}.$$

By Lemma 2, $k \leq \frac{ND}{4M}\ln\frac{N}{M}$, i.e., $k\frac{2M}{ND} \leq \frac{1}{2}\ln\frac{N}{M}$, holds. Hence, we are interested in

$$D\ln\frac{4ND^2}{Mk} \leq \frac{1}{2}\ln\frac{N}{M} + \ln\frac{D}{2}$$

which is implied by

$$D \ln \frac{2\sqrt{N}D}{\sqrt{Mk}} \leq \frac{1}{4}\ln\frac{N}{M} \tag{2}$$

for $D \geq 2$. Otherwise, since $k \geq 2$ one can obtain the desired subgraph by choosing M adjacent edges in G. This yields a subgraph consisting of at most $M+1$ vertices, i.e. with average degree at least $2/(1+1/M)$.

Now, we can use Observation 5 with $f = \sqrt{\frac{4N}{Mk}}$ and $x = \frac{1}{4}\ln\frac{N}{M}$, and get the approximation

$$D \leq \frac{x}{\overline{\ln}\, xf} = \frac{\ln\frac{N}{M}}{2\,\overline{\ln}\,\frac{N\ln^2\frac{N}{M}}{4Mk}}$$

for which inequality (2) holds.

Finally, we check $D \leq k/2$. To this end, we plug $k/2$ as D into (2) and get

$$\frac{k}{2}\ln\frac{4N\cdot k^2}{Mk\cdot 4} \leq \frac{1}{2}\ln\frac{N}{M}$$

$\frac{k}{2}\ln\frac{Nk}{M} \leq \frac{1}{2}\ln\frac{N}{M}$, assuming $k \leq 1$, contradicting one of the assumptions. □

□

In the following, we assume an internal memory of size $2M$ to ease notation. Now consider a subgraph $G_S = (U_S \cup V_S, E_S)$ with M edges and average degree D. By construction of $G = (U \cup V, E)$, we considered a non-zero entry a_{ij} as an edge between u_i and v_j. Let I_U, I_V be the set of indices of vertices in U_S, V_S respectively. In order to create elementary products corresponding to E_S, the M corresponding non-zero entries a_{ij} with $i \in I_U$, $j \in I_V$ have to be loaded. Then, for each column $1 \leq k \leq N$, by loading all elements b_{jk} with row indices $j \in I_V$ together, M elementary products for rows with indices I_U in $\mathbf{C}$ can be obtained.

To efficiently load certain elements of a column in $\mathbf{B}$, we extract these rows into a separate $|I_V| \times N$ matrix and transpose it to column major layout. This is possible with $2\frac{NM}{DB}$ I/Os, since we assume $\mathbf{B}$ to be in row major layout. Then, elements corresponding to a certain column can be loaded with at most $\frac{M}{DB}$ I/Os. Similarly, partial products can be stored into a $|I_U| \times N$ matrix in column major layout. Transposing this, and adding the rows to the corresponding rows in $\mathbf{C}$ requires no more than $3\frac{NM}{DB}$ I/Os. Hence, given a subgraph with M edges and average degree D, NM elementary products can be created with at most $6\frac{NM}{DB} + \frac{M}{B} = \mathcal{O}\left(\frac{NM}{DB}\right)$ I/Os.

Lemma 1 only states the existence of at least one dense subgraph. However, after creating all the elementary products corresponding to the edges of a dense subgraph, one can think of removing these edges. This will decrease the number of edges by no more than M, and we can use Lemma 1 for graphs with $kN - M$ edges again. Clearly, half of the elementary products can be obtained by subgraphs with average degree $D(k/2)$. This is possible with $\mathcal{O}\left(\frac{kN^2}{2D(k/2)B}\right)$ I/Os.

Let $D_1(k) = \ln \frac{N}{M} / (2 \ln \frac{N \ln^2 \frac{N}{M}}{4Mk})$, i.e., the first argument of the minimum of D in Lemma 1. Altogether, the number of I/Os necessary to create all elementary products for $\mathbf{C}$ is bounded from above by

$$\begin{aligned} L(k,N) &\leq \frac{kN}{B} + \sum_{i=1}^{\infty} 6 \max\left\{ \frac{2kN^2}{2^i B\sqrt{M}}, \frac{kN^2}{2^i D_1(k/2^i) B} \right\} \\ &\leq \frac{kN}{B} + \frac{12kN^2}{B\sqrt{M}} + 6kN^2 \sum_{i=0}^{\infty} \frac{\ln \frac{N \ln^2 \frac{N}{M}}{4Mk} + \ln 2^i}{2^i B \ln \frac{N}{M}} = \mathcal{O}\left(\frac{kN^2}{DB}\right). \end{aligned}$$

Observe that for $k \geq \frac{N}{32M} \ln^2 \frac{N}{M}$, $\sqrt{\frac{kN}{M}} = \Omega\left(\ln \frac{N}{M}\right)$ and thus, the tile-based algorithm is asymptotically better.

4.5 Small instances

For smaller instances where $M \geq kN$, neither the tile-based algorithm is applicable, nor does the proof of dense subgraphs hold. Once $M \geq kN + N$, a degenerated version of the tile-based approach becomes the one of choice: Initially, all non-zero entries of $\mathbf{A}$ are loaded into internal memory. Afterwards, $\mathbf{B}$ is loaded in whole columns. For each column, the corresponding column in $\mathbf{C}$ can be calculated directly and written to external memory. This requires only $\mathcal{O}\left(\frac{N^2}{B}\right)$ I/Os.

5 Lower bounds

Theorem 2. *For $1 \leq k \leq N$ any program for SDM$_k$ needs*

$$\Omega\left(\max\left\{\frac{kN^2}{B\Delta}, \frac{kN^2}{B\sqrt{M}}, \frac{N^2}{B}\right\}\right)$$

I/Os, with Δ according to Theorem 1.

Theorem 2 will be proven throughout this section. Therefore, we make use of the following technique introduced by Hong and Kung in [5].

Lemma 3. *A* round-based program *consists of q rounds where each round consists of M/B input operations, followed by M/B output operations such that after the round internal memory is empty. A lower bound on the number of rounds q_{min} of any round-based program with internal memory of size $2M$ can be transformed into a lower bound on the number of I/Os l of any (normal) program with internal memory of size M by $l \geq \frac{M}{B} \cdot (q_{min} - 1)$.*

Recall that the overall number of elementary products that have to be produced for SDM$_k$ is kN^2. Thus, given an upper bound on the number of elementary products that can be made during one round, a lower bound on the number of necessary rounds is obtained. We will do this by showing that there are matrices with only few dense parts. In the following, we consider the matrix $\mathbf{A}$ again as an adjacency matrix.

Lemma 4. *Let $\mathcal{G}$ be the family of bipartite graphs $G = (U \cup V, E)$ with $|U| = |V| = N$ and $|E| = kN$ for $k \leq N/2$.*

For any $M \leq kN$ there is a graph $G \in \mathcal{G}$ such that G contains no subgraph $G_S = (U_S \cup V_S, E_S)$ with $|E_S| = M$ and average degree

$$D'_M > \max\left\{\frac{8\ln\frac{N}{M}}{\ln\frac{16N\ln^2\frac{N}{M}}{Mk}}, e^4 \cdot \sqrt{\frac{kM}{N}}\right\}. \quad (3)$$

Proof. We will show this by upper bounding the number of graphs containing at least one such dense subgraph and compare this to the cardinality of $\mathcal{G}$. The upper bound is given by the number of possibilities to choose $2M/D'_M$ vertices from $U \cup V$ and the number of possibilities to insert M edges between the selected vertices. Furthermore, the remaining $kN - M$ edges are chosen uniformly within the graph. The former presumes $M/D'_M \leq N$. However, since $M \leq kN$ and $D'_M > \sqrt{\frac{kM}{N}}$ this is implied. Further, we can assume $D'_M \leq \sqrt{M}$ since this is the maximum average degree of a subgraph consisting of M edges. Hence, if the inequality

$$\binom{2N}{2M/D'_M}\binom{(M/D'_M)^2}{M}\binom{N^2}{kN-M} < \binom{N^2}{kN}$$

holds for the parameters given, Lemma 4 is proven.Observation 4 yields

$$\binom{2N}{2M/D'_M}\binom{(M/D'_M)^2}{M} < \left(\frac{N^2 - kN}{kN}\right)^M .$$

Estimating binomial coefficients according to Observation 3, taking logarithms and multiplying by D'_M/M, we obtain

$$2\ln\frac{eD'_M N}{M} + D'_M \ln\frac{eM}{{D'_M}^2} < D'_M \ln\frac{N^2 - kN}{kN} = D'_M \ln\frac{N}{k} + D'_M \ln\left(1 - \frac{k}{N}\right).$$

The last term can be estimated for $k \leq N/2$ by Observation 1 resulting in

$$2\ln\frac{eD'_M N}{M} + D'_M \ln\frac{eM}{{D'_M}^2} < D'_M \ln\frac{N}{k} - D'_M \frac{2k}{N}.$$

And by simple transformations, we obtain

$$D'_M \ln\frac{{D'_M}^2 N}{kM} > \underbrace{2\ln\frac{N}{M}}_{\text{Term 1}} + \underbrace{2\ln eD'_M + D'_M\left(1 + 2\frac{k}{N}\right)}_{\text{Term 2}}. \tag{4}$$

Equation 4 is implied if Terms 1 and 2 are both bounded by $\frac{1}{2}D'_M \ln\frac{{D'_M}^2 N}{kM}$. We first check this for Term 2 only. By Observe 2 $\ln eD'_M \leq D'_M$. Thus,

$$\frac{1}{2}D'_M \ln\frac{{D'_M}^2 N}{kM} > 2\ln eD'_M + 2D'_M$$

is implied by $D'_M > e^4 \cdot \sqrt{\frac{kM}{N}}$. For any such D'_M Inequality 4 holds if

$$D'_M \ln\frac{D'_M\sqrt{N}}{\sqrt{kM}} > 2\ln\frac{N}{M}. \tag{5}$$

By substitution of D'_M by $e^4 \cdot \sqrt{\frac{kM}{N}}$, (5) already holds for $\sqrt{k} > \frac{1}{2e^4}\sqrt{\frac{N}{M}} \ln \frac{N}{M}$, i.e. especially for $M \geq N$. For $\sqrt{k} \leq \frac{1}{2e^4}\sqrt{\frac{N}{M}} \ln \frac{N}{M}$, we use Observation 6. Altogether, for

$$D'_M > \max \left\{ \frac{8 \ln \frac{N}{M}}{\overline{\ln} \frac{16N \ln^2 \frac{N}{M}}{Mk}}, e^4 \cdot \sqrt{\frac{kM}{N}} \right\}$$

not all possible graphs in $\mathcal{G}$ are covered and therefore, Lemma 4 holds. Since the second term is a sufficient bound for any $\sqrt{k} > \frac{1}{2e^4}\sqrt{\frac{N}{M}} \ln \frac{N}{M}$, we use $\overline{\ln}$ instead of ln to derive a closed formula by bounding the first term. Finally, note that $D'_M > 4$ holds for $k \geq 1$. □ □

Lemma 5. *Let $\mathcal{G}$ be the family of bipartite graphs $G = (U \cup V, E)$ with $|U| = |V| = N$ and $|E| = kN$ for $k \leq N/2$.*

For any $M \leq kN$, there is a graph $G \in \mathcal{G}$ such that G contains at most $M - 1$ edges in subgraphs $G_S = (U_S \cup V_S, E_S)$ with $|E_S| \leq M$ and average degree $D' \geq 2e^4\Delta$ where Δ is defined according to Theorem 2.

Proof. By Lemma 4, this holds already for subgraphs consisting of exactly M edges. For smaller subgraphs, we prove the statement by contradiction.

Suppose that there are at least M edges in subgraphs with average degree at least D' consisting of less than M edges. Let $\mathcal{S}$ be the set of such subgraphs. Since each subgraph in $\mathcal{S}$ has less than M edges, there exists a subset S' of subgraphs in $\mathcal{S}$ with a total number of cM edges for $1 \leq c < 2$. The subgraph $G_{S'} = (U_{S'} \cup V_{S'}, E_{S'})$ induced by S' has obviously still average degree at least D'.

Wlog let $|U_{S'}| \geq |V_{S'}|$ and consider the vertices $U_{S'}$ in $G_{S'}$. Now choose the $\lceil \frac{M}{D'} \rceil$ vertices in $U_{S'}$ with highest degree, and let $U'_{S'}$ denote the set of these. Since the vertices $U_{S'}$ have average degree D' in $G_{S'}$, the subset $U'_{S'}$ cannot have a lower average degree. Hence, the subgraph $G'_{S'}$ induced by $U'_{S'}$ and $V_{S'}$ contains at least M edges, but consists of no more than $\frac{M}{D'} + \frac{cM}{D'} + 1$ vertices. Therefore, any subgraph induced by exactly M edges of $G'_{S'}$ has average degree at least $\frac{2MD'}{M+cM+D'}$. Since $D' \leq \sqrt{M}$, the average degree is at least $\frac{2D'}{2+c} \geq \frac{1}{2}D'$. This contradicts Lemma 4 for any $D' \geq 2D'_M$. □ □

Using this, we can finally prove Theorem 2. Recall that Lemma 4, and thus, 5 fails for $D'_M > \sqrt{M}$. However, the maximum degree of a subgraph with M edges is $\sqrt{M}$. The total number of elementary products, necessary for SDM$_k$ is kN^2. By Lemma 5, there are at most $N(M-1)$ elementary products which might be calculated faster than the rest. For the remaining

$kN^2 - NM + N$ elementary products, the following holds. Consider any round-based program for SDM_k. Within each round, there are at most $2M$ elements of $\mathbf{B}$ and $\mathbf{C}$ loaded. Let s_{ij}, t_{ij} be the number of elements from the j-th column of $\mathbf{B}$, $\mathbf{C}$ respectively, loaded in round i. By Lemma 5 and the observation that any subgraph has degree at most $\sqrt{M}$, there can be made no more than $\sum_{j=1}^{N} \min\{D', \sqrt{M}\} \cdot s_{ij}t_{ij} = 2M \cdot \min\{D', \sqrt{M}\}$ elementary products during each round. Hence, there have to be at least

$$\frac{kN^2 - MN + N}{2M \cdot \min\{D', \sqrt{M}\}}$$

rounds. This yields a lower bound of

$$\frac{M}{B}\left(\frac{kN^2 - MN + N}{2M \cdot \min\{D', \sqrt{M}\}} - 1\right) = \Omega\left(max\left\{\frac{kN^2}{B\Delta}, \frac{kN^2}{B\sqrt{M}}\right\}\right)$$

I/Os for SDM_k.

5.1 Closing the parameter range

Recall that Lemma 5 only holds for $k \leq N/2$. However, $\Omega\left(max\left\{\frac{kN^2}{B\Delta}, \frac{kN^2}{B\sqrt{M}}\right\}\right)$ is a lower bound for $N/2 \leq k \leq N$ as well since increasing the number of non-zero entries in A cannot decrease the number of I/Os. For $M \geq kN$ a scanning bound of $\Omega\left(\frac{N^2}{B}\right)$ holds for the output of $\mathbf{C}$.

References

[1] Alok Aggarwal and Jeffrey S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.

[2] Michael Bader and Alexander Heinecke. Cache oblivious dense and sparse matrix multiplication based on peano curves. In *Proceedings of the PARA 08*, Lecture Notes in Computer Science. Springer, December 2009. accepted for publication.

[3] Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Riko Jacob, and Elias Vicari. Optimal sparse matrix dense vector multiplication in the I/O-model. In *Proceedings of SPAA '07*, pages 61–70, New York, NY, USA, 2007. ACM.

[4] Aydın Bulucc and John R. Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In *Proceedings of ICPP'08*, pages 503–510, Portland, Oregon, USA, September 2008.

[5] Hong, Jia-Wei and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proceedings of STOC '81*, pages 326–333, New York, NY, USA, 1981. ACM.

[6] Markus Kowarschik and Christian Weiß. An overview of cache optimization techniques and cache-aware numerical algorithms. *Algorithms for Memory Hierarchies*, pages 213–232, 2003.

[7] Joseph M. Landsberg. Geometry and the complexity of matrix multiplication. *Bulletin of the American Mathematical Society*, 45:247–284, 2008.

[8] Tobias Lieber. Combinatorial approaches to optimizing sparse matrix dense vector multiplication in the I/O-model. Master's thesis, Informatik Technische Universität München, 2009.

[9] Juan J. Navarro, Elena García-Diego, Josep-L. Larriba-Pey, and Toni Juan. Block algorithms for sparse matrix computations on high performance workstations. In *Proceedings of ICS '96*, pages 301–308, New York, NY, USA, 1996. ACM.