Discrete Mathematics and Theoretical Computer Science **2**, 1998, 35–47

Lower bounds for sparse matrix vector multiplication on hypercubic networks

Giovanni Manzini

Dipartimento Scienze e Tecnologie Avanzate, I-15100 Alessandria, Italy and Istituto di Matematica Computazionale, CNR, I-56126 Pisa, Italy.
E-mail: `manzini@mfn.al.unipmn.it`

In this paper we consider the problem of computing on a local memory machine the product $\mathbf{y} = A\mathbf{x}$, where A is a random $n \times n$ sparse matrix with $\Theta(n)$ nonzero elements. To study the average case communication cost of this problem, we introduce four different probability measures on the set of sparse matrices. We prove that on most local memory machines with p processors, this computation requires $\Omega((n/p)\log p)$ time on the average. We prove that the same lower bound also holds, in the worst case, for matrices with only $2n$ or $3n$ nonzero elements.

Keywords: Sparse matrices, pseudo expanders, hypercubic networks, bisection width lower bounds

1 Introduction

In this paper we consider the problem of computing $\mathbf{y} = A\mathbf{x}$, where A is a random $n \times n$ sparse matrix with $\Theta(n)$ nonzero elements. We prove that, on most local memory machines with p processors, this computation requires $\Omega((n/p)\log p)$ time in the average case. We also prove that this computation requires $\Omega((n/p)\log p)$ time in the worst case for matrices with only $3n$ nonzero elements and, under additional hypotheses, also for matrices with $2n$ nonzero elements. We prove these results by considering only the communication cost, i.e. the time required for routing the components of $\mathbf{x}$ to their final destinations.

The average communication cost of computing $\mathbf{y} = A\mathbf{x}$ has also been studied in [10] starting from different assumptions. In [10] we restricted our analysis to the algorithms in which the components of $\mathbf{x}$ and $\mathbf{y}$ are partitioned among the processors according to a fixed scheme not depending on the matrix A. However, it is natural to try to reduce the cost of sparse matrix vector multiplication by finding a 'good' partition of the components of $\mathbf{x}$ and $\mathbf{y}$. Such a partition can be based on the nonzero structure of the matrix A which is assumed to be known in advance. This approach is justified if one has to compute many products involving the same matrix, or involving matrices with the same nonzero structure. The development of efficient partition schemes is a very active area of research; for some recent results see [3, 4, 5, 14, 15].

In this paper we prove lower bounds which also hold for algorithms which partition the components of $\mathbf{x}$ and $\mathbf{y}$ on the basis of the nonzero structure of A. First, we introduce four probability measures on the class of $n \times n$ sparse matrices with $\Theta(n)$ nonzero elements. Then, we show that with high probability the cost of computing $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube is $\Omega((n/p)\log p)$ for *every* distribution of the components of $\mathbf{x}$ and $\mathbf{y}$ among the processors. Since the hypercube can simulate with constant slowdown any $\Theta(p)$-processor butterfly, CCC, mesh of trees and shuffle exchange network, the same bounds hold for these communication networks as well (see [7] for descriptions and properties of the hypercube, and of the other networks mentioned above).

Our results are based on the expansion properties of a bipartite graph associated with the matrix A. Expansion properties, as well as the related concept of separators, have played a major role in the study of the fill-in generated by Gaussian and Cholesky factorization of a random sparse matrix (see [1, 8, 11, 12]), but to our knowledge, this is the first time they have been used for the analysis of parallel matrix vector multiplication algorithms.

We establish lower bounds for the hypercube assuming the following model of computation. Processors have only local memory and communicate with one another by sending packets of information through the hypercube links. At each step, each processor is allowed to perform a bounded amount of local computation, or to send one packet of bounded length to an adjacent processor (we are therefore considering the so-called weak hypercube).

This paper is organized as follows. In section 2 we introduce four different probability measures on the set of sparse matrices, and the notion of the pseudo-expander graph. We also prove that the dependency graph associated with a random matrix is a pseudo-expander with high probability. In section 3 we study the average cost of sparse matrix vector multiplication, and in section 4 we prove lower bounds for matrices with $3n$ and $2n$ nonzero elements. Section 5 contains some concluding remarks.

2 Preliminaries

Let $X = \{x_1, \dots, x_n\}$, and $Y = \{y_1, \dots, y_n\}$. Given an $n \times n$ matrix A, we associate with A a directed bipartite graph $G(A)$, called the *dependency graph*. The vertex set of $G(A)$ is $X \cup Y$, and (x_i, y_j) is an edge of $G(A)$ if and only if $a_{ji} \neq 0$. In other words, the vertices of $G(A)$ represent the components of $\mathbf{x}$ and $\mathbf{y}$, and the edge $(x_i, y_j) \in G(A)$ if and only if the value y_j depends upon x_i.

Definition 2.1 Let $0 < \alpha < 1$, $\beta > 1$ and $\alpha\beta < 1$. We say that the graph $G(A)$ is an (α, β, n)-*pseudo-expander* if for each set $U \subseteq X$ the following property holds:

$$\frac{n\alpha}{2} \leq |U| \leq n\alpha \implies |\mathrm{Adj}\,(U)\,| > \beta |U|$$

Note that we must have $\alpha\beta < 1$, since each set of αn vertices can be connected to at most n vertices. The notion of a pseudo-expander graph is similar to the well known notion of an expander graph [2]. We recall that a graph is an (α, β, n)-expander if $|U| \leq n\alpha$ implies $|\mathrm{Adj}\,(U)\,| > \beta|U|$ (hence, all expanders are pseudo-expanders).

The average case complexity of sparse matrix vector multiplication will be obtained combining two basic results: in this section, we prove that a random sparse matrix is a pseudo-expander with probability very close to one, in the next section we prove that computing $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p)\log p)$ time if $G(A)$ is a pseudo-expander.

To study the average cost of a sparse matrix vector multiplication, we introduce four probability measures on the set of $n \times n$ sparse matrices. Each measure represents a different type of sparse matrix with $\Theta(n)$ nonzero elements. Although the nonzero elements can be arbitrary real numbers, we assume the behaviour of the multiplication algorithms does not depend upon their numerical values. Hence, all measures are defined on the set of 0–1 matrices, where 1 represents a generic nonzero element.

1. Let $0 \leq d_1 \leq n$ such that $d_1 n$ is an integer. We denote by $\mathcal{M}_1$ the probability measure such that $\Pr\{A\} \neq 0$ only if the matrix A has exactly $d_1 n$ nonzero elements; moreover, all $\binom{n^2}{d_1 n}$ matrices with $d_1 n$ nonzero elements are equally likely.
2. Let $0 \leq d_2 \leq n$, $p = d_2/n$. We denote by $\mathcal{M}_2$ the probability measure such that for each entry a_{ij} $\Pr\{a_{ij} \neq 0\} = p$. Hence, we have $\Pr\{A\} = p^k(1-p)^{n^2-k}$, where k is the number of nonzero elements of A.
3. Let d_3 be an integer $0 \leq d_3 \leq n$. We denote by $\mathcal{M}_3$ the probability measure such that $\Pr\{A\} \neq 0$ only if each row of A contains exactly d_3 nonzero elements; moreover, all $\binom{n}{d_3}^n$ matrices with d_3 nonzero elements per row are equally likely.
4. Let d_4 be an integer $0 \leq d_4 \leq n$. We denote by $\mathcal{M}_4$ the probability measure such that $\Pr\{A\} \neq 0$ only if each column of A contains exactly d_4 nonzero elements; moreover, all $\binom{n}{d_4}^n$ matrices with d_4 nonzero elements per column are equally likely.

The above measures are clearly related since they all represent unstructured sparse matrices. We have chosen these measures since they are quite natural, but of course, they do not cover the whole spectrum of 'interesting' sparse matrices. In the following, the expression 'random matrix' will denote a matrix chosen according to one of these probability measures.

Given a random matrix A we want to estimate the probability that the graph $G(A)$ is a pseudo-expander. In this section we prove that for the measures $\mathcal{M}_1$–$\mathcal{M}_4$ previously defined such probability is very close to one (Theorem 2.6). In the following, we make use of some well known inequalities. For $0 \leq k \leq n$,

$$\binom{n}{k} \leq \left(\frac{en}{k}\right)^k \tag{1}$$

and, for any real number $x \geq 1$,

$$\left(1 - \frac{1}{x}\right)^x < e^{-1} \tag{2}$$

In addition, a straightforward computation shows that for any triplet of positive integers m, n, p such that $m + p \leq n$ we have

$$\binom{n-p}{m} \cdot \binom{n}{m}^{-1} \leq \left(1 - \frac{m}{n}\right)^p \tag{3}$$

Lemma 2.2 *Let A be a random matrix chosen according to the probability measure $\mathcal{M}_i$, $i = 1, \ldots, 4$. If $U \subseteq X$, $V \subseteq Y$ we have*

$$\Pr\{\text{Adj}\,(U) \cap V = \phi\} \leq \left(1 - \frac{d_i}{n}\right)^{|U|\,|V|} \tag{4}$$

Proof. We consider only $i = 1, 3$, since the cases $i = 2, 4$ are similar. Let $Q = \{a_{ji} | x_i \in U, y_j \in V\}$. We have $|Q| = |U|\,|V|$ and $\text{Adj}\,(U) \cap V = \phi$ if and only if $a_{ji} = 0$ for all $a_{ji} \in Q$. For the probability measure $\mathcal{M}_1$, using (3) we get

$$\Pr\{\text{Adj}\,(U) \cap V = \phi\} = \binom{n^2 - |Q|}{d_1 n} \binom{n^2}{d_1 n}^{-1} \leq \left(1 - \frac{d_1}{n}\right)^{|U|\,|V|}$$

For the probability measure $\mathcal{M}_3$ we have $\mathrm{Adj}\,(U) \cap V = \phi$ only if the $|V|$ rows corresponding to V do not contain nonzero elements in the $|U|$ columns corresponding to U. By (3) we get

$$\Pr\{\mathrm{Adj}\,(U) \cap V = \phi\} = \left[\binom{n-|U|}{d_3}\binom{n}{d_3}^{-1}\right]^{|V|} \leq \left(1 - \frac{d_3}{n}\right)^{|U|\,|V|}$$

□

Lemma 2.3 *Let $U \subseteq X$, $|U| = k$, and assume* (4) *holds. If*

$$0 < \lambda < 1, \qquad \frac{\alpha n}{2} \leq k \leq \alpha n, \qquad d_i \geq \frac{2}{\alpha\lambda}\left[1 - \log(1-\alpha\beta)\right] \tag{5}$$

then

$$\Pr\{|\mathrm{Adj}\,(U)| \leq \beta k\} < \left(1 - \frac{d_i}{n}\right)^{k(n-\beta k)(1-\lambda)}$$

Proof. First note that

$$\Pr\{|\mathrm{Adj}\,(U)| \leq \beta k\} = \Pr\{|\mathrm{Adj}\,(U)| \leq \lfloor \beta k \rfloor\}$$

We have that $|\mathrm{Adj}\,(U)| \leq \lfloor \beta k \rfloor$ only if there exists a set $\tilde{V} \subseteq Y$, with $|\tilde{V}| = n - \lfloor \beta k \rfloor$, such that $\mathrm{Adj}\,(U) \cap \tilde{V} = \phi$. Since there are $\binom{n}{n-\lfloor \beta k \rfloor}$ sets V of size $n - \lfloor \beta k \rfloor$, if (4) holds, we have

$$\Pr\{|\mathrm{Adj}\,(U)| \leq \beta k\} \leq \binom{n}{n - \lfloor \beta k \rfloor}\left(1 - \frac{d_i}{n}\right)^{k(n-\lfloor \beta k \rfloor)}$$

By (1) we have

$$\begin{aligned}
\Pr\{|\mathrm{Adj}\,(U)| \leq \beta k\} &\leq \left(\frac{en}{n - \lfloor \beta k \rfloor}\right)^{n-\lfloor \beta k \rfloor}\left(1 - \frac{d_i}{n}\right)^{k(n-\lfloor \beta k \rfloor)\lambda}\left(1 - \frac{d_i}{n}\right)^{k(n-\lfloor \beta k \rfloor)(1-\lambda)} \\
&\leq \left(\frac{en}{n - \beta k}\right)^{n-\lfloor \beta k \rfloor}\left(1 - \frac{d_i}{n}\right)^{k(n-\lfloor \beta k \rfloor)\lambda}\left(1 - \frac{d_i}{n}\right)^{k(n-\beta k)(1-\lambda)}
\end{aligned}$$

Hence, we need to show that

$$\frac{en}{n-\beta k}\left(1 - \frac{d_i}{n}\right)^{k\lambda} < 1$$

Since $n\alpha/2 \leq k \leq n\alpha$, using (2) we get

$$\frac{en}{n-\beta k}\left(1 - \frac{d_i}{n}\right)^{k\lambda} \leq \frac{en}{n - n\alpha\beta}\left(1 - \frac{d_i}{n}\right)^{\frac{\alpha n \lambda}{2}} < \frac{e^{1-\frac{d_i\lambda\alpha}{2}}}{1-\alpha\beta}$$

This completes the proof since the hypothesis (5) on d_i implies that

$$e^{1-\frac{d_i\lambda\alpha}{2}} \leq 1 - \alpha\beta$$

□

Lemma 2.4 *Suppose that the hypotheses of Lemma 2.3 hold. If*

$$d_i \geq \left(1 + \log \frac{4}{\alpha}\right) \frac{1}{(1-\lambda)(1-\alpha\beta)} \tag{6}$$

then the probability that there exists a set $U \subseteq X$ with $|U| = k$ such that $|\text{Adj}\,(U)| \leq \beta k$ is less than 2^{-k}.

Proof. The probability that a set of size k is connected to less than βk vertices is given in Lemma 2.3. Since there exist $\binom{n}{k}$ of such sets, we have

$$\begin{aligned} \Pr\{\exists U \text{ such that } |\text{Adj}\,(U)| \leq \beta k\} &< \binom{n}{k}\left(1 - \frac{d_i}{n}\right)^{k(n-\beta k)(1-\lambda)} \\ &< \left(\frac{en}{k}\right)^k \left(1 - \frac{d_i}{n}\right)^{k(n-\beta k)(1-\lambda)} \end{aligned}$$

Therefore, we need to prove that

$$\frac{en}{k}\left(1 - \frac{d_i}{n}\right)^{(n-\beta k)(1-\lambda)} \leq \frac{1}{2} \tag{7}$$

Since $\frac{n\alpha}{2} \leq k \leq n\alpha$, we have

$$\begin{aligned} \frac{en}{k}\left(1 - \frac{d_i}{n}\right)^{(n-\beta k)(1-\lambda)} &\leq \frac{2e}{\alpha}\left(1 - \frac{d_i}{n}\right)^{n(1-\alpha\beta)(1-\lambda)} \\ &\leq \frac{2}{\alpha} e^{1-d_i(1-\alpha\beta)(1-\lambda)} \end{aligned} \tag{8}$$

From (6) we have

$$\log \frac{4}{\alpha} + 1 - d_i(1-\alpha\beta)(1-\lambda) \leq 0$$

hence

$$\frac{4}{\alpha} e^{1-d_i(1-\alpha\beta)(1-\lambda)} \leq 1$$

The Lemma follows by comparing this last inequality with (7) and (8). □

Note that Lemma 2.4 holds only if both (5) and (6) hold. That is, d_i must satisfy

$$d_i \geq \max\left\{\frac{2}{\alpha\lambda}\left[1 - \log\left(1-\alpha\beta\right)\right], \left(1 + \log\frac{4}{\alpha}\right)\frac{1}{(1-\lambda)(1-\alpha\beta)}\right\} \tag{9}$$

for some $0 < \lambda < 1$. It is easy to verify that by taking

$$\bar{\lambda} = \frac{\gamma}{\gamma + \alpha\left(1 + \log\frac{4}{\alpha}\right)}, \qquad \text{with } \gamma = 2(1-\alpha\beta)\left[1 - \log\left(1-\alpha\beta\right)\right]$$

the two arguments of the $\max$ function in (9) are equal. By substituting λ with $\bar{\lambda}$, we obtain that (9) is equivalent to

$$d_i \geq \frac{1 + \log\frac{4}{\alpha}}{1-\alpha\beta} + \frac{2}{\alpha}\left[1 - \log\left(1-\alpha\beta\right)\right]$$

Lemma 2.5 *Let A be an $n \times n$ random matrix for which* (4) *holds. If*

$$d_i \geq \frac{1 + \log \frac{4}{\alpha}}{1 - \alpha\beta} + \frac{2}{\alpha}\left[1 - \log\left(1 - \alpha\beta\right)\right] \tag{10}$$

the graph $G(A)$ is an (α, β, n)-pseudo-expander with probability greater than $1 - 2^{1-\frac{\alpha n}{2}}$

Proof. The graph $G(A)$ is not an (α, β, n)-pseudo-expander only if there exists a set $U \subseteq X$ with $n\alpha/2 \leq |U| \leq n\alpha$ such that $|\mathrm{Adj}\,(U)\,| \leq \beta|U|$. By Lemma 2.4, we know that

$$\Pr\left\{G(A) \text{ is not a pseudo-expander }\right\} \leq \sum_{k=\lceil n\alpha/2\rceil}^{\lfloor n\alpha\rfloor} 2^{-k}$$

Therefore,

$$\Pr\left\{G(A) \text{ is a pseudo-expander}\right\} > 1 - 2^{-\lceil \frac{n\alpha}{2} \rceil}\left(\sum_{i=0}^{\infty} 2^{-i}\right) \geq 1 - 2^{1-\frac{n\alpha}{2}}$$

□

Theorem 2.6 *Let A be a random matrix chosen according to the probability measure $\mathcal{M}_i$ $i = 1, \ldots, 4$. If*

$$d_i \geq \frac{1 + \log \frac{4}{\alpha}}{1 - \alpha\beta} + \frac{2}{\alpha}\left[1 - \log\left(1 - \alpha\beta\right)\right] \tag{11}$$

the graph $G(A)$ is an (α, β, n)-pseudo-expander with probability greater than $1 - 2^{1-\frac{\alpha n}{2}}$

Proof. The proof follows by Lemmas 2.2 and 2.5. □

As an example, for $\alpha = \frac{1}{2}, \beta = \frac{16}{15}$ inequality (11) becomes $d_i \geq 13.65\ldots$. Theorem 2.6 tells us that, under this assumption, $G(A)$ is an $(\frac{1}{2}, \frac{16}{15}, n)$-pseudo-expander with probability $1 - 2^{1-\frac{n}{4}}$. Similarly, if $d_i \geq 15.08\ldots$ $G(A)$ is a $(\frac{2}{5}, \frac{5}{4}, n)$-pseudo-expander with probability greater than $1 - 2^{1-\frac{n}{5}}$.

If $G(A)$ is an (α, β, n)-pseudo-expander, then, given $U \subseteq X$ such that $n\alpha/2 \leq |U| \leq n\alpha$ there are more than $\beta|U|$ values y_j that depend on the values $x_i \in U$. Similarly, if $G(A^T)$ is an (α, β, n)-pseudo-expander, given $V \subseteq Y$, $\alpha/2 \leq |V| \leq \alpha$, the values $y_j \in V$ depend upon more than $\beta|V|$ values x_i. By symmetry considerations we have the following corollary of Theorem 2.6.

Corollary 2.7 *Let A be a random matrix chosen according to the probability measure $\mathcal{M}_i$ $i = 1, \ldots, 4$. If d_i satisfies* (11)*, then the graph $G(A^T)$ is an (α, β, n)-pseudo-expander with probability greater than* $1 - 2^{1-\frac{\alpha n}{2}}$ □

3 Study of the Average Case Communication Complexity

Let A be an $n \times n$ sparse matrix, and let $p \leq n$. In this section we prove a lower bound on the average case complexity of computing the product $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube. Our analysis is based on the cost of routing the components of $\mathbf{x}$ to their final destination. That is, we consider only the cost of routing, to the processor that computes y_j, the values x_i's for all i such that $a_{ji} \neq 0$. A major difficulty for establishing a lower bound is that we can compute partial sums $p_j = a_{ji_1}x_{i_1} + \cdots + a_{ji_k}x_{i_k}$ during the routing process. In this case, by moving a single value p_j we can 'move' several x_i's. However, since the a_{ji}'s can be arbitrary, we assume that the partial sum $p_j = a_{ji_1}x_{i_1} + \cdots + a_{ji_k}x_{i_k}$ can be used only for computing the j-th component y_j.

In the previous section, we have shown that a random matrix is a pseudo-expander with high probability. Therefore, to establish an average case result it suffices to obtain a lower bound for this class of matrices. In the following, we assume that there exist two functions F_x, F_y mapping $\{1, 2, \ldots, n\}$ into $\{0, \ldots, p-1\}$ such that

1. for $i = 1, \ldots, n$, the value x_i is initially contained *only* in processor number $F_x(i)$;
2. for $j = 1, \ldots, n$, at the end of the computation the value y_j is contained in processor number $F_y(j)$.

In the following, $F_x^{-1}(k)$ will denote the set $\{x_i \mid F_x(i) = k\}$, that is, the set of all components of $\mathbf{x}$ initially contained in processor k. Similarly, $F_y^{-1}(k)$ will denote the components of $\mathbf{y}$ contained in processor k.

Theorem 3.1 *Let A be a matrix such that $G(A)$ is an (α, β, n)-pseudo-expander with $\alpha = 1/2$, $\frac{4}{3} < \beta < 2$. If conditions 1–2 hold, and for $0 \le h < p$, $|F_y^{-1}(h)| = \lfloor n/p \rfloor$ or $\lceil n/p \rceil$, any algorithm for computing the product $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p)\log p)$ time.*

Proof. The two functions F_x, F_y define a mapping of the vertices of $G(A)$ into the processors of the hypercube. If the edge (x_i, y_j) belongs to $G(A)$ (that is, $a_{ji} \neq 0$) the value y_j depends upon x_i. Hence, the value x_i has to be moved from processor $F_x(i)$ to processor $F_y(j)$.

For $1 \le k \le \log p$ we consider the set of dimension k links of the hypercube (that is, the links connecting processors whose addresses differ in the k-th bit). If the dimension k links are removed, the hypercube is bisected into two sets H_1, H_2 with $p/2$ processors each. From the hypothesis on F_y it follows that at the end of the computation each set contains at most $(p/2)\lceil n/p \rceil \le 2n/3$ values y_j's.

We can assume that H_1 initially contains at least $n/2$ values x_i's (if not, we exchange the roles of H_1 and H_2). A value $x_i \in H_1$ can reach H_2 either by itself or inside a partial sum $\sum a_{jh}x_h$. Note that a value x_i that reaches H_2 by itself can be utilized for the computation of several y_j's, but we assume that each partial sum can be utilized for computing only one y_j.

Let n_1 denote the number of values $x_i \in H_1$ that traverse the dimension k links by themselves, and let n_2 be the number of partial sums that traverse the dimension k links. We claim that $\max(n_1, n_2) > \varepsilon n$ where $\varepsilon = \frac{3\beta-4}{6(\beta+1)}$.

If $n_1 \le \varepsilon n$, we consider the set $\tilde{X}_1$ of the values $x_i \in H_1$ that *do not* traverse the dimension k links by themselves. We have

$$|\tilde{X}_1| \ge \frac{n}{2} - n_1 \ge \frac{n}{2} - \varepsilon n = \frac{7n}{6(\beta+1)} \tag{12}$$

Since $G(A)$ is a pseudo-expander, and

$$\frac{n}{4} \le \frac{7n}{6(\beta+1)} \le \frac{n}{2}$$

we have that more than $\frac{7\beta n}{6(\beta+1)} - \frac{2n}{3}$ values $y_j \in H_2$ depend on the values $x_i \in \tilde{X}_1$. We have

$$\frac{7\beta n}{6(\beta+1)} - \frac{2n}{3} = n\frac{7\beta - 4(\beta+1)}{6(\beta+1)} = \varepsilon n$$

that is, more than εn values $y_j \in H_2$ depend upon the values $x_i \in \tilde{X}_1$. Moreover, the values $x_i \in \tilde{X}_1$ can reach H_2 only inside the n_2 partial sums that traverse the dimension k links. Since each partial sum can be utilized for the computation of only one y_j, we have that $n_2 > \varepsilon n$ as claimed.

This proves that $\Omega(n)$ items must traverse the dimension k links. Since the same result holds for all dimensions, the sum of the lengths of the paths traversed by the data is $\Omega(n \log p)$. Since at each step at most p links can be traversed, the computation of $\mathbf{y} = A\mathbf{x}$ requires $\Omega((n/p) \log p)$ time. □

Theorem 3.2 *Let A be a matrix such that $G(A^T)$ is an (α, β, n)-pseudo-expander with $\alpha = 1/2$, $\frac{4}{3} < \beta < 2$. If conditions 1–2 hold, and, for $0 \le h < p$, $|F_x^{-1}(h)| = \lfloor n/p \rfloor$ or $\lceil n/p \rceil$, any algorithm for computing the product $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p) \log p)$ time.*

Proof. Let H_1, H_2 denote the sets obtained by removing the dimension k links of the hypercube. From the hypothesis on F_x follows that initially each set contains at most $(p/2) \lceil n/p \rceil \le 2n/3$ values x_i's. We can assume that at the end of the computation H_2 contains at least $n/2$ values y_j's (if not we consider H_1). Let n_1, n_2 be defined as in the proof of Theorem 3.1. Clearly, it suffices to prove that $\max(n_1, n_2) > \varepsilon n$ with $\varepsilon = \frac{3\beta - 4}{6(\beta+1)}$.

If $n_2 \le \varepsilon n$, we consider the set $\tilde{Y}_2$ of the values $y_j \in H_2$ that are computed entirely inside H_2. That is, $y_h \in \tilde{Y}_2$ only if no partial sum $\sum a_{hi} x_i$ traverses the dimension k links. We have

$$|\tilde{Y}_2| \ge \frac{n}{2} - n_2 \ge \frac{n}{2} - \varepsilon n = \frac{7n}{6(\beta+1)}$$

Since $G(A^T)$ is a pseudo-expander, the values $y_j \in \tilde{Y}_2$ depend on more than $\frac{7\beta n}{6(\beta+1)} - \frac{2n}{3}$ values $x_i \in H_1$. By construction, these values x_i's must traverse the dimension k links by themselves. Hence

$$n_1 > \frac{7\beta n}{6(\beta+1)} - \frac{2n}{3} = \varepsilon n$$

□

By combining Theorems 3.1 and 3.2 with a result on the complexity of balancing on the hypercube, it is possible to prove that the computation of $\mathbf{y} = A\mathbf{x}$ requires $\Omega((n/p) \log p)$ time even if there are processors containing $\Omega(n/p)$ components of $\mathbf{x}$ or $\mathbf{y}$.

Lemma 3.3 *Suppose that n items are distributed over the p processors of a hypercube, and let m be the maximum number of items stored in a single processor. There exists an algorithm* BALANCE *that redistributes the items so that each processor contains $\lceil n/p \rceil$ or $\lfloor n/p \rfloor$ items. The running time of* BALANCE *is $O\left(m \log^{1/2} p + \log^2 p\right)$.*

Proof. See [13, Theorem 5]. □

We also need the converse of this lemma. Given a distribution of n items over p processors with no more than m items per processor, there exists an algorithm UNBALANCE that constructs such distribution starting from an initial setting in which each processor contains $\lceil n/p \rceil$ or $\lfloor n/p \rfloor$ items. The algorithm UNBALANCE is obtained by executing the operations of the algorithm BALANCE in the opposite order, hence its running time is again $O\left(m \log^{1/2} p + \log^2 p\right)$.

In the following we use the notation $f(n) = o(g(n))$ to denote that for any $c > 0$, there exists n_0 such that $f(n) < cg(n)$ for all $n \geq n_0$.

Theorem 3.4 *Let A be a matrix such that $G(A)$ is an (α, β, n)-pseudo-expander with $\alpha = 1/2$, $\frac{4}{3} < \beta < 2$. If conditions 1–2 hold, and, for $0 \leq h < p$, $|F_y^{-1}(h)| = O((n/p) \log^\omega p)$ with $0 \leq \omega < 1/2$, $p \log p = o(n)$, any algorithm for computing the product $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p) \log p)$ time.*

Proof. Assume by contradiction that there exists an algorithm SPARSEPROD for computing $\mathbf{y} = A\mathbf{x}$ such that:

(a) the running time of SPARSEPROD is $t(n, p)$ with $t(n, p) = o((n/p) \log p)$,

(b) at the end of the computation each processor contains $O((n/p) \log^\omega p)$ components y_j's.

Clearly, using the procedure BALANCE, we can transform SPARSEPROD into an algorithm in which at the end of the computation each processor contains $\lceil n/p \rceil$ or $\lfloor n/p \rfloor$ components of $\mathbf{y}$. The running time of this new algorithm is

$$O\Big((n/p) \log^\omega p \log^{1/2} p + \log^2 p + t(n, p)\Big)$$

that by hypothesis is $o((n/p) \log p)$. This is impossible by Theorem 3.1. □

Using the procedure UNBALANCE and Theorem 3.2, it is straightforward to prove the following result.

Theorem 3.5 *Let A be a matrix such that $G(A^T)$ is an (α, β, n)-pseudo-expander with $\alpha = 1/2$, $\frac{4}{3} < \beta < 2$. If conditions 1–2 hold, and, for $0 \leq h < p$, $|F_x^{-1}(h)| = O((n/p) \log^\omega p)$ with $0 \leq \omega < 1/2$, $p \log p = o(n)$, any algorithm for computing the product $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p) \log p)$ time.* □

We can summarize the results of this section as follows: if the components of one of the vectors $\mathbf{x}$ or $\mathbf{y}$ are distributed 'evenly' (in the sense of Theorems 3.4 and 3.5) among the processors, the data movement required for computing $\mathbf{y} = A\mathbf{x}$ takes $\Omega((n/p) \log p)$ time with high probability. The result holds for the weak hypercube and, *a fortiori*, for the hypercubic networks that can be emulated with constant slowdown by the hypercube.

Note that no hypothesis has been made on how the nonzero entries of A are stored. That is, all lower bounds hold even if each processor contains in its local memory all nonzero elements of the matrix A. Moreover, since these results hold for any pair of function F_x, F_y, we have that the knowledge of the nonzero structure of A does not help to reduce the average cost of the computation.

4 Matrices with $3n$ and $2n$ Nonzero Elements

In the previous section we have shown that, if A has $\Theta(n)$ nonzero elements, computing $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube takes $\Omega((n/p) \log p)$ time with high probability. It is interesting to investigate what is the minimum number of nonzero elements of A for which this property holds. In this section we give a partial answer to this question.

Definition 4.1 Let $1 < \beta < 2$. Given a matrix A, we say that the graph $G(A)$ is a *β-weak-expander* if for each set $U \subseteq X$ the following property holds:

$$|U| = \lfloor n/2 \rfloor \implies |\mathrm{Adj}\,(U)\,| > \beta |U|$$

Obviously, all (α, β, n)-pseudo-expanders with $\alpha \geq 1/2$ are weak-expanders but the converse is not true. As we will see, weak-expanders can be still used to get lower bounds for matrix vector multiplication. In addition, the next lemma shows that there exist matrices with only $3n$ nonzero elements whose dependency graph is a weak-expander.

Lemma 4.2 *For all $n \geq 5$, there exists an $n \times n$ matrix $\tilde{A}$ such that*

1. *$\tilde{A} = \pi_1 + \pi_2 + \pi_3$, where π_1, π_2, π_3 are permutation matrices,*
2. *both $G(A)$ and $G(A^T)$ are $\frac{9}{8}$-weak-expanders.*

Proof. The existence of matrix $\tilde{A}$ is established in the proof of Theorem 4.3 in [11]. □

Lemma 4.3 *Let A be a matrix with a constant number of nonzero elements per row, and such that $G(A^T)$ is a β-weak-expander. If conditions 1–2 of section 3 hold, $p|n$, and, for $0 \leq h < p$, $|F_y^{-1}(h)| = n/p$, any algorithm for computing the product $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p)\log p)$ time.*

Proof. As in the proof of Theorem 3.1, it suffices to prove that, for $1 \leq k \leq \log p$, there are $\Omega(n)$ items that must traverse the dimension k links of the hypercube.

Let H_1, H_2 denote the sets obtained by removing the dimension k links. As usual, we can assume that H_1 contains at least $n/2$ values x_i's. Clearly, the $n/2$ values $y_j \in H_2$ depend upon at least $(\beta - 1)(n/2)$ values $x_i \in H_1$. These values can traverse the dimension k links either by themselves or inside a partial sum. However, each partial sum can contain only a constant number of values x_i, hence, $\Omega(n)$ items must traverse the dimension k links. □

An analogous proof yields the following result.

Lemma 4.4 *Let A be a matrix with a constant number of nonzero elements per column, and such that $G(A)$ is a β-weak-expander. If conditions 1–2 of Section 3 hold, $p|n$, and, for $0 \leq h < p$, $|F_x^{-1}(h)| = n/p$, any algorithm for computing $\mathbf{y} = A\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p)\log p)$ time.* □

Using Lemmas 4.2-4.4, we can easily prove that, in the worst case, the computation of $\mathbf{y} = A\mathbf{x}$ requires $\Omega((n/p)\log p)$ time also for matrices with only $3n$ nonzero elements.

Theorem 4.5 *For all $n \geq 5$, there exists an $n \times n$ matrix $\tilde{A}$ with at most $3n$ nonzero elements, such that, if the components of one of the vectors $\mathbf{x}$ or $\mathbf{y}$ are evenly distributed among the processors, any algorithm for computing $\mathbf{y} = \tilde{A}\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p)\log p)$ time.* □

The case of matrices with three nonzero elements per row appears to be the boundary between easy and difficult problems. In fact, if a matrix B contains at most two nonzero elements in each row and in each column, we can arrange the components of $\mathbf{x}$ and $\mathbf{y}$ so that the product $\mathbf{y} = A\mathbf{x}$ can be computed on the hypercube in $O(n/p)$ time. To see this, it suffices to notice that each vertex of $G(B)$ has degree at most two. Hence, the connected components of $G(B)$ can only be chains or rings and can be embedded in the hypercube with constant dilation and congestion.

We conclude this section by studying the complexity of sparse matrix vector multiplication when we require that, for $i = 1, \ldots, n$, the value y_i is stored in the same processor containing x_i. A typical situation in which this requirement must be met, is when the multiplication algorithm is utilized for computing the sequence $\mathbf{x}^{(k+1)} = A\mathbf{x}^{(k)}$ generated by an iterative method. Note that, using the notation of section 3, this requirement is equivalent to assuming that $F_x \equiv F_y$.

Theorem 4.6 *Let $\mathcal{H}$ be the class of matrix vector multiplication algorithms such that, for $i = 1, \ldots, n$, the value y_i is stored in the same processor containing x_i. For all $n \geq 5$, there exists an $n \times n$ matrix $\tilde{B}$ such that*

1. *each row and each column of $\tilde{B}$ contains at most two nonzero elements,*
2. *if the component of $\mathbf{x}$ are evenly distributed, any algorithm in $\mathcal{H}$ for computing $\mathbf{y} = \tilde{B}\mathbf{x}$ on a p-processor hypercube requires $\Omega((n/p)\log p)$ time.*

Proof. By Lemmas 4.2 and 4.4, we know that there exists a matrix $\tilde{A}$ such that the communication cost of computing $\mathbf{y} = \tilde{A}\mathbf{x}$ is $\Omega((n/p)\log p)$ time (note that this bound holds for the algorithms not in $\mathcal{H}$). Moreover, we know that $\tilde{A} = \pi_1 + \pi_2 + \pi_3$, where π_1, π_2, π_3 are permutation matrices. Now consider the matrix $A' = \pi_1^{-1}\tilde{A}$. Since the multiplication by a permutation matrix does not require any communication (for the algorithms not in $\mathcal{H}$!), the communication cost of computing $\mathbf{y} = A'\mathbf{x}$ is $\Omega((n/p)\log p)$. Since $A'\mathbf{x} = \mathbf{x} + B\mathbf{x}$, any algorithm in $\mathcal{H}$ for computing $B\mathbf{x}$ can be used for computing $A'\mathbf{x}$ with no extra communication cost. It follows that any algorithm in $\mathcal{H}$ for computing $B\mathbf{x}$ requires $\Omega((n/p)\log p)$ time. This completes the proof, since B is equal to the sum of two permutation matrices. □

5 Concluding Remarks

One of the most challenging problems in the field of distributed computing is to find good data distributions for irregular problems. In this paper we have analysed the issue of data distribution for the problem of sparse matrix vector multiplication. We have performed an average case analysis by introducing four different probability measures on the set of $n \times n$ matrices with $\Theta(n)$ nonzero elements. We have shown that, on average, computing $\mathbf{y} = A\mathbf{x}$ on many $\Theta(p)$-processor networks requires $\Omega((n/p)\log p)$ time. The result holds for any balanced distribution of the components of $\mathbf{x}$ and $\mathbf{y}$ among the processors.

A parallel algorithm for computing the product $\mathbf{y} = A\mathbf{x}$, where A is a $n \times n$ matrix with $O(n)$ nonzero elements, has been given in [9]. The algorithm runs in $O((n/p)\log p)$ time on a p-processor hypercubic network. The results of this paper establish the 'average case' optimality of the algorithm in [9] for the class of unstructured matrices. However, our results do not rule out the possibility that the product $\mathbf{y} = A\mathbf{x}$ can be computed in $o((n/p)\log p)$ time for important classes of matrices which are not pseudo-expanders. Typical examples are the matrices arising in finite elements and finite difference discretization of partial differential equations for which $O(n/p)$ multiplication algorithms exist (see for example [6, Ch. 11]).

There are several possible extensions of our results which deserve further investigation. In our analysis we have considered only the cost of routing each value x_i to the processors in charge of computing the values y_j's which depend upon x_i. It is natural to ask if one could get more general results by taking into account also the computation cost, which, as the number of nonzero elements grows, may well exceed the cost of communication. Another interesting open problem is whether we can break the $\Omega((n/p)\log p)$

lower bound by allowing processors to contain multiple copies of the components of $\mathbf{x}$. To be fair, our algorithm should produce multiple copies of the components of $\mathbf{y}$ so that it can be used to compute sequences of the kind $\mathbf{x}^{(k+1)} = A\mathbf{x}^{(k)}$.

References

[1] A. Agrawal, P. Klein and R. Ravi. Cutting down on fill using nested dissection: Provably good elimination orderings. In A. George, R. Gilbert and J. Liu, editors, *Graph Theory and Sparse Matrix Computation*, 31–55. Springer-Verlag, 1993.

[2] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[3] U. Catalyuerek and C. Aykanat. Decomposing irregularly sparse matrices for parallel matrix-vector multiplication. *Parallel Algorithms for Irregulary Structured Problems (IRREGULAR '96)*, LNCS 1117, 75–86. Springer-Verlag, 1996.

[4] T. Dehn, M. Eiermann, K. Giebermann and V. Sperling. Structured sparse matrix-vector multiplication on massively parallel SIMD architectures. *Parallel Computing* **21**:1867–1894, 1995.

[5] P. Fernandes and P. Girdinio. A new storage scheme for an efficient implementation of the sparse matrix-vector product. *Parallel Computing* **12**:327–333, 1989.

[6] V. Kumar, A. Grama, A. Gupta and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings, Redwood City, CA, 1994.

[7] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1991.

[8] R. Lipton, D. Rose and R. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.* **16**:346–358, 1979.

[9] G. Manzini. Sparse matrix computations on the hypercube and related networks. *Journal of Parallel and Distributed Computing* **21**:169–183, 1994.

[10] G. Manzini. Sparse matrix vector multiplication on distributed architectures: Lower bound and average complexity results. *Information Processing Letters* **50**:231–238, 1994.

[11] G. Manzini. On the ordering of sparse linear systems. *Theoretical Computer Science* **156**(1–2):301–313, 1996.

[12] V. Pan. Parallel solution of sparse linear and path systems. In J. H. Reif, editor, *Synthesis of Parallel Algorithms*, 621–678. Morgan Kaufmann, 1993.

[13] C. G. Plaxton. Load balancing, selection and sorting on the hypercube. *Proc. of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures*, 64–73, 1989.

[14] L. Romero and E. Zapata. Data distributions for sparse matrix vector multiplication. *Parallel Computing* **21**:583–605, 1995.

[15] L. Ziantz, C. Oezturan and B. Szymanski. Run-time optimization of sparse matrix-vector multiplication on SIMD machines. In *Parallel Architectures and Languages Europe (PARLE '94)*, LNCS 817, 313–322. Springer-Verlag, 1994.