Communication Lower Bounds for Distributed-Memory Computations*

Michele Scquizzato[†1] **and Francesco Silvestri**[2]

1 **Department of Computer Science, University of Pittsburgh, Pittsburgh, US**
scquizza@pitt.edu
2 **Department of Information Engineering, University of Padova, Padova, Italy**
silvest1@dei.unipd.it

Abstract

In this paper we propose a new approach to the study of the communication requirements of distributed computations, which advocates for the removal of the restrictive assumptions under which earlier results were derived. We illustrate our approach by giving tight lower bounds on the communication complexity required to solve several computational problems in a distributed-memory parallel machine, namely standard matrix multiplication, stencil computations, comparison sorting, and the Fast Fourier Transform. Our bounds rely only on a mild assumption on work distribution, and significantly strengthen previous results which require either the computation to be balanced among the processors, or specific initial distributions of the input data, or an upper bound on the size of processors' local memories.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Communication, lower bounds, distributed memory, parallel algorithms, BSP

Digital Object Identifier 10.4230/LIPIcs.STACS.2014.627

1 Introduction

Communication is a major factor determining the performance of algorithms on current computing systems, as the time and energy needed to transfer data between processing and storage elements is often significantly higher than that of performing arithmetic operations. The gap between computation and communication costs, which is ultimately due to basic physical principles, is expected to become wider and wider as architectural advances allow to build systems of increasing size and complexity. Hence, the cost of data movement will play an even greater role in future years.

As in all endeavors where performance is systematically pursued, it is important to evaluate the distance from optimality of a proposed algorithmic solution, by establishing appropriate lower bounds. Given the well-known difficulty of establishing lower bounds, results are often obtained under restrictive assumptions that may severely limit their applicability. It is therefore important to progressively reduce or fully eliminate such restrictions.

In this spirit, we consider lower bounds on the amount of communication that is required to solve some classical computational problems on a distributed-memory parallel system.

* This work was supported, in part, by the University of Padova Project *CPDA121378* and by MIUR of Italy under project *AMANDA*.
† Supported, in part, by a fellowship of "Fondazione Ing. Aldo Gini", University of Padova, Italy. Most of this work was done while this author was a Ph.D. student at the University of Padova.

31st Symposium on Theoretical Aspects of Computer Science (STACS'14).
Editors: Ernst W. Mayr and Natacha Portier; pp. 627–638

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Specifically, we revisit the assumptions and constraints under which preceding results were derived, and prove new lower bounds which use much weaker hypotheses and thus have wider applicability. Even when the functional form of the bounds remains the same, our results do yield new insights to algorithm developers since they might reveal if some settings are needed, or not, in order to obtain better performance.

We model the machine using the standard *Bulk Synchronous Parallel* (BSP) model of computation [32], which consists of a collection of p processors, each equipped with an unbounded private memory and communicating with each other through a communication network. The distribution of inputs and outputs effectively forms a part of the problem specification, thus restricting the applicability of upper and lower bounds. Much of previous work on BSP algorithms considers a version of the BSP model equipped with an additional *external memory*, which serves as the source of the input and the destination for the output (see, e.g., [29]). This modification significantly alters the spirit of the BSP of serving as a model for distributed-memory machines, making it very similar to shared-memory models like the LPRAM [1]. In fact, in a distributed-memory machine, the inputs might already be distributed in some manner prior to the invocation of the algorithm, and the outputs are usually left distributed in the processors' local memories at the end of the execution, especially if the computation is a subroutine of a larger computation. Thus, lower bounds that use this assumption, which essentially exploit this "hack" to guarantee that acquiring the n input elements contributes to the communication cost of algorithms (as some processor must read at least $\lceil n/p \rceil$ input values), are not directly applicable to distributed-memory architectures.

Other authors, within the original BSP model, assume specific distributions of the input data. As we shall see later, it is usually assumed that the input is initially *evenly* distributed among the p processors, that is, each processor is assigned either $\lceil n/p \rceil$ or $\lfloor n/p \rfloor$ pieces of the input. However, this apparently reasonable hypothesis is somewhat restrictive, and actually not part of the logic of the BSP model. In fact, the physical distribution of input data across the processors may depend on several factors, ranging from how the input data set gets acquired, to how the output of the preceding computation is distributed in the case of algorithms being cascaded (that is, when the output of one is the input for the next), to file system policies. Moreover, a uniform partition of the inputs postulates, but does not prove, that unbalanced distributions may cause severe communication bottlenecks.

One possibility to circumvent both the issues discussed above is to require, in place of the even distribution of the inputs and of the presence of an external memory, that algorithms exhibit some level of *load balancing* of the computation. Typically, if $\mathcal{W}$ denotes the total work required by any algorithm to solve the given problem, it is required that each processor performs $O(\mathcal{W}/p)$ elementary computations. However, this way it is implicitly assumed, but (as above) not proved, that optimal solutions balance computation. In fact, in general there is a tradeoff between computation costs and communication costs. Some papers (see, e.g., [22, 35]) quantify such tradeoffs by establishing lower bounds on the communication cost of any algorithm as a function of its computation time. Nevertheless, results of this kind usually indicate that the higher lower bounds on communication correspond only to perfectly (to within constant factors) work-balanced computations, and such bounds are tight since achieved by balanced algorithms. This leaves open the possibility that a substantial saving on communication costs could actually be achieved at a price of a small unbalance of the computation loads.

Another common assumption is putting an upper bound on the *size* of processors' local memories. However, current technological advances allow to build cheap memory and storage

devices that, for many applications, allow a single machine to store the whole input data set and the intermediate data. Moreover, results derived under this assumption are less general than results that put no limits on the amount of storage available to processors; indeed, lower bounds are relatively easier to establish, as the model essentially becomes a parallel version of the standard external memory model for sequential computations, for which much more results and techniques are known (see, e.g., [16, 2]).

In contrast, lower bounds presented in this paper do not hinge on any of the above assumptions. We develop new lower bounds for a number of key computational problems, namely standard matrix multiplication, stencil computations, comparison sorting, and the Fast Fourier Transform, using the weak assumption that no processor performs more than a constant fraction of the total required work. This requires more involved arguments, and substantially strengthens previous work on communication lower bounds for distributed-memory computations.

The model. The *Bulk Synchronous Parallel* (BSP) model of computation was introduced by Valiant [32] as a bridging model for general-purpose parallel computing. The architectural component of the model consists of p processing elements $P_0, P_1, \ldots, P_{p-1}$, each equipped with an unbounded local memory, interconnected by a communication medium. The execution of a BSP algorithm consists of a sequence of *supersteps*, where each processor can perform operations on data in its local memory, send/receive messages (each occupying a constant number of words) and, at the end, execute a global synchronization. The running time of the i-th superstep is expressed in terms of two parameters, g and ℓ, as $T_i = w_i + h_i g + \ell$, where w_i is the maximum number of local operations performed by any processor, and h_i is the maximum number of messages sent or received by any processor. The *running time* $T_{\mathcal{A}}$ of a BSP algorithm $\mathcal{A}$ is the sum of the times of its supersteps and can be expressed as $W_{\mathcal{A}} + H_{\mathcal{A}} g + S_{\mathcal{A}} \ell$, where $S_{\mathcal{A}}$ is the number of supersteps, $W_{\mathcal{A}} = \sum_{i=1}^{S_{\mathcal{A}}} w_i$ is the *local computation complexity*, and $H_{\mathcal{A}} = \sum_{i=1}^{S_{\mathcal{A}}} h_i$ is the *communication complexity*.

Previous work. The complexity of communication on various models of computation has received considerable attention. Lower bounds are often established through adaptations of the techniques of Hong and Kung [16] for hierarchical memory, or by critical path arguments, such as those in [1]. For applications of these and other techniques see [22, 2, 25, 15, 8, 6, 17, 24, 4, 9, 5] as well as [26] and references therein. In the following, we discuss previous work on lower bounds for the communication complexity of the problems studied in this paper.

A standard computational problem is the multiplication of two $n \times n$ matrices. For the classical $\Theta\left(n^3\right)$ algorithm, an $\Omega\left(n^2/p^{2/3}\right)$ lower bound has been previously derived for the BSP [31] and the LPRAM [1]. However, both results hinge on the hypothesis that the input initially resides outside the processors' local memories and thus must be read, contributing to the communication complexity of the algorithms. As such, these results are an immediate consequence of a result of [16] (then restated in [17]) which, loosely speaking, bounds from above the amount of computation that can be performed with a given quantity of data. When input is assumed to be initially evenly distributed across the p processors' local memories, the same lower bound is claimed in [11]. Recently, Ballard et al. [3] obtained a result of the same form by assuming perfectly balanced (to within constant factors) computations, and disallowing any initial replication of inputs. The very same bound was found also by Irony et al. [17], who restrict their attention to computations that take place on machines where processors' local memory size is assumed to be $M = O\left(n^2/p^{2/3}\right)$ (see also [4]). Finally, Solomonik and Demmel [28] investigate tradeoffs between input replication and communication complexity.

A class of computations ubiquitous in scientific computing is that of stencil computations, where each computing node in a multi-dimensional grid is updated with weighted values contributed by neighboring nodes. These computations include the *diamond* DAG in the two-dimensional case and the *cube* DAG in three dimensions. For the former, Papadimitriou and Ullman [22] present a communication-time tradeoff which yields a tight $\Omega(n)$ lower bound on the communication complexity only for the case of balanced computations. Aggarwal et al. [1] extend this result to all algorithms whose computational complexity is within a constant factor of the number of nodes of the DAG. To the best of our knowledge, this is the sole example of a tight lower bound that holds under the same hypothesis used in this paper. By generalizing the technique in [22], Tiskin [31] establishes a tight bound for the cube DAG, and claims its extension to higher dimensions. However, this results only hold when the computational load is balanced among the p processors.

Another key problem is sorting. Many papers assume that the n inputs initially reside outside processors' local memories, thus obtaining an $\Omega(n/p)$ lower bound which turns out to be tight when it is additionally assumed that problem instances have sufficient *slackness*, that is, $n >> p$ (e.g., $p^2 \leq n$ is a common assumption). Under some technical assumptions, a bound of the form $\Omega(n \log n/(p \log(n/p)))$, which is tight for all values of $p \leq n$, was first given within the LPRAM model [1]. This bound, however, includes the cost to read the input from the shared memory. A similar lower bound was derived later by Goodrich [15] within the BSP model, but the result holds only for the subclass of algorithms performing supersteps of degree $h = \Theta(n/p)$, and when the inputs are evenly distributed among the processors.

Previous work on the communication required to compute an FFT DAG of size n is similar to previous work for sorting. By exploiting the property that, as shown in [34], the cascade of three FFT networks has the topology of a full sorting network, the aforementioned lower bounds for sorting also hold for the FFT DAG. In a recent paper [9], we obtain the same result assuming that the maximum number of outputs held by any processor at the end of the algorithm is at most $n/2$, and without assumptions on the distribution of the input and of the computational loads; while these hypotheses are not equivalent to the one we are using in this paper, the result in [9] is the closest to the one that we will develop in Section 5.

Our contribution. In this paper we present lower bounds on the communication complexity required by key computational problems such as standard matrix multiplication, stencil computations, comparison sorting, and the Fast Fourier Transform, when solved by parallel algorithms on the BSP model. These results, which are all tight for the whole range of model parameters, rely on the hypothesis that no processor performs more than a *constant* fraction of the total required work. More formally, let $\mathcal{W}$ be the total work required by any algorithm to solve the given problem (if the problem is represented by a directed acyclic graph, then $\mathcal{W}$ is the number of nodes of the DAG, otherwise $\mathcal{W}$ is a lower bound on the computation time required by any sequential algorithm), and let W be the maximum amount of work performed by any BSP processor; then, W is assumed to satisfy the bound $W \leq \epsilon\mathcal{W}$, for some constant $\epsilon \in (0, 1)$. The rationale behind this approach is that communication is the major bottleneck of a distributed-memory computation unless the latter is sequential or "nearly sequential", in which case the main contribution to the running time T of an algorithm comes from computation. Since it is directly linked to the running time metric, and it does not allow for any other restrictive assumptions suggested by orthogonal constraints, we believe that this is the right approach to perform a systematic analysis of the communication requirements of distributed-memory computations.

We emphasize that, in contrast to previous work, our lower bounds do not count the communication required to acquire the input, allow for any initial distribution of the input among the processors' local memories, assume no upper bound on the sizes of the latter, and do not require computations to be balanced. On the other hand, some of our results make use of additional technical assumptions, such as the non-recomputation of intermediate results in the course of the computation, or some restrictions on the replication of input data. Such restrictions, however, were already in place in almost all of the corresponding state-of-the-art lower bounds.

A full version of the paper can be found in [27].

2 Matrix Multiplication

In this section we consider the problem of multiplying two $n \times n$ matrices, A and B, using only semiring operations, that is, addition and multiplication. Hence, each element $c_{i,j}$ of the output matrix C is an explicit sum of products $a_{i,k} \cdot b_{k,j}$, which are called *multiplicative terms*. This rules out, e.g., Strassen's algorithm and the Boolean matrix multiplication algorithm of Tiskin [30]. As shown in [19], any algorithm using only semiring operations must compute at least n^3 distinct multiplicative terms.

In this section we establish a lower bound on the communication complexity of any parallel algorithm for matrix multiplication on a BSP with p processors. This result is derived assuming that no processor performs more than a constant fraction of the n^3 total work required by any algorithm, measured as the number of scalar multiplications, and that each input element is initially stored in the local memory of exactly one processor. The bound has the form of $\Omega\left(W^{2/3}\right)$, where W is the maximum number of multiplicative terms evaluated by a processor, and is tight for all values of p between two and n^2. The argument through which we establish such a result is a repeated application of a "bandwidth" argument which, loosely speaking, is as follows. Consider a processor which performs the maximum amount of work. If this processor initially holds "few" input values, then, since it computes at least n^3/p multiplicative terms, it must receive "many" inputs from the submachine including the other processors; otherwise, if it initially holds "many" inputs, then it has to send many of them to the other processors, because it cannot perform too much work on its own, and thus the other processors have to perform at least a constant fraction of the total work. The lower bound applies to any distribution of input and output matrices, and only requires that the input matrices are not initially replicated.

Towards this end, we first establish a lower bound of $\Omega\left(n^2\right)$ under the same hypotheses outlined above for two processors. This result is derived using a bandwidth argument that bounds from below the amount of data that must travel across the communication network of a two-processor machine. A bound of the same form can be found in [17, Section 6], which holds only when the elements of the input matrices A and B are evenly, or almost evenly, distributed among the two processors. Our result, which instead allows any initial distribution of the input matrices (without replication), establishes the same bound by using a mild hypothesis on the maximum computation load faced by the processors. Due to space constraints, the proof of the result is deferred to the full version of the paper.

▶ **Lemma 1.** *Let $\mathcal{A}$ be any algorithm for computing the matrix product $C = AB$, using only semiring operations, on a BSP with two processors. If each processor computes at most ϵn^3 multiplicative terms, where ϵ is an arbitrary constant in $[1/2, 1)$, and the input matrices are not initially replicated, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n, p) = \Omega\left(n^2\right).$$

Now we can prove the main result of this section. The following theorem establishes an $\Omega\left(W^{2/3}\right)$ lower bound to the communication complexity of any standard algorithm, where W denotes the maximum number of multiplicative terms evaluated by a processor. By the result of [19] and by the pigeonhole principle, there exists a processor that computes at least n^3/p multiplicative terms, from which the standard $\Omega\left(n^2/p^{2/3}\right)$ lower bound follows.

▶ **Theorem 2.** *Let $\mathcal{A}$ be any algorithm for computing the matrix product $C = AB$, using only semiring operations, on a BSP with p processors, where $1 < p \leq n^2$, and let W be the maximum number of multiplicative terms evaluated by a processor. If $W \leq \max\{n^3/p, n^3/11^3\}$, and the input matrices are not initially replicated, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n,p) = \Omega\left(W^{2/3}\right).$$

Proof. Without loss of generality, we assume that any multiplicative term computed by the processors is actually used towards the computation of some entry of the output matrix C (that is, processors do not perform "useless" computations). Consider one of the processors that compute W multiplicative terms, and without loss of generality let P_0 denote such a processor. Let I be the number of input elements initially held by this processor in its local memory.

Consider first the case $I \leq W^{2/3}/5$. By [16, Lemma 6.1], a processor that computes W multiplicative terms either accesses, during the whole execution of algorithm $\mathcal{A}$, at least $(W/2)^{2/3}$ input elements, or computes multiplicative terms relative to at least $(W/2)^{2/3}$ elements of the output matrix. In the first case, since P_0 initially holds $I \leq W^{2/3}/5$ input elements, it must receive at least $(W/2)^{2/3} - I = \Omega\left(W^{2/3}\right)$ data words from other processors, and the theorem follows. On the other hand, suppose P_0 computes multiplicative terms relative to $(W/2)^{2/3}$ entries of the output matrix, and partition such entries into three groups: G_1, the set of entries whose multiplicative terms have all been computed by the processor; G_2, the set of entries produced by the processor but for which some multiplicative term or partial sum has been communicated by some other processor; G_3, the set of entries not produced by the processor. Clearly, at least one of these three groups must have size at least $(W/2)^{2/3}/3$. If $|G_1| \geq (W/2)^{2/3}/3$, then P_0 must have computed at least $n(W/2)^{2/3}/3$ multiplicative terms, and since any entry of the input matrices occurs in only n of such terms, the processor must have received $(W/2)^{2/3}/3 - I = \Omega\left(W^{2/3}\right)$ elements from other processors. A similar argument applies to both G_2 and G_3.

Now suppose $I > W^{2/3}/5$ and $p \geq 11^3$. Note that in this case, since $p \geq 11^3$, it holds that $W \leq n^3/11^3$. Assume, without loss of generality, that P_0 initially holds at least $I/2$ elements of matrix A. Since any entry of the input matrices occurs in n multiplicative terms, there are at least $In/2$ multiplicative terms that depend on the entries of A initially held by the processor. Since W multiplicative terms are computed by the processor, the remaining $In/2 - W \geq W^{2/3}n/10 - W^{2/3}n/11 = W^{2/3}n/110$ ones are computed by other processors. Since, by hypothesis, each entry of A is initially non-replicated and a processor can compute at most n multiplicative terms using a single entry of A, we have that $(In/2 - W)/n \geq W^{2/3}/110$ messages are required for sending the appropriate entries of A to the processors that will compute the remaining entries. Hence, $H_{\mathcal{A}}(n,p) \geq W^{2/3}/110$.

Finally, when $I > W^{2/3}/5$ and $p < 11^3$, the sought lower bound follows by Lemma 1. Indeed, the p processors can be virtually partitioned into two subsets, each consisting of exactly $p/2$ processors; in particular, processor P_0^* will be identified with the submachine including the first half of the p processors, and P_1^* with the submachine including the second half. Since $p < 11^3$, by hypothesis each BSP processor computes at most n^3/p multiplicative

terms, and thus both P_0^* and P_1^* compute at most $(n^3/p)(p/2) = n^3/2$ multiplicative terms overall. Hence we can apply Lemma 1 to processors P_0^* and P_1^*, obtaining the desired result. ◀

The proposed bound is tight and is matched by the algorithm that decomposes the problem into $n^3/W \leq p$ subproblems of size $W^{1/3} \times W^{1/3}$, and then solves each subproblem sequentially in each round. Since $W \geq n^3/p$, the minimum communication complexity is $\Omega\left(n^2/p^{2/3}\right)$, which is achieved by the standard 3D algorithm (see, e.g., [17]).

3 Stencil Computations

A *stencil* defines the computation of an element in a $(d-1)$-dimensional spatial grid at time t as a function of neighboring grid elements at time $t-1, \ldots, t-\tau$, for some value $\tau \geq 1$ and constant $d > 1$ (see, e.g., [14]). We provide an $\Omega\left(n^{d-1}/p^{(d-2)/(d-1)}\right)$ lower bound to the communication complexity of any algorithm evaluating n time steps of a $(d-1)$-dimensional stencil. For simplicity we assume $\tau = 1$, however our bounds still apply in the general case. The bound follows by investigating the (n, d)-*stencil problem*, which consists in evaluating all nodes of a d-dimensional array DAG of size n. A *d-dimensional array DAG* has n^d nodes $\langle i_0, \ldots, i_{d-1}\rangle$, for each $0 \leq i_0, \ldots, i_{d-1} < n$, and there is an arc from $\langle i_0, \ldots, i_k, \ldots, i_{d-1}\rangle$ to $\langle i_0, \ldots, i_k + 1, \ldots, i_{d-1}\rangle$, for each $0 \leq k < d$ and $0 \leq i_0, \ldots, i_{d-1} < n-1$. Observe that $\langle 0, \ldots, 0\rangle$ and $\langle n-1, \ldots, n-1\rangle$ are the single input and output nodes, respectively. A lower bound to the (n, d)-stencil problem applies to the computation of n steps of a stencil: indeed, the DAG given by the $(d-1)$-dimensional grid plus the time dimension spans a d-dimensional spacetime containing an $(n/2, d)$-array as a subgraph.

Our result hinges on the restriction on the nature of the computation whereby each vertex of the DAG is computed exactly once. In this setting, the crucial property is that for each arc (u, v) such that u is computed by processor P and v is computed by processor P', $P \neq P'$, there corresponds a message from P to P' (which may also cross other processors). Such arcs are referred to as *communication arcs*.

We now introduce some preliminary definitions, which will be used throughout the section. We envision an (n, d)-stencil DAG as partitioned into $p^{d/(d-1)}$ smaller d-dimensional arrays, called *blocks*, of size $n/p^{1/(d-1)}$, and denote each block with $B_{i_0,\ldots,i_{d-1}}$ for $0 \leq i_0, \ldots, i_{d-1} < p^{1/(d-1)}$. Block $B_{i_0,\ldots,i_{d-1}}$ contains nodes $\langle i'_0, \ldots, i'_{d-1}\rangle$, for each $i_k n/p^{1/(d-1)} \leq i'_k < (i_k + 1)n/p^{1/(d-1)}$. A block has $n^d/p^{d/(d-1)}$ nodes, and is said *ℓ-owned* if more than half of its nodes are evaluated by processor P_ℓ, with $0 \leq \ell < p$. A block is *owned* if there exists some ℓ, with $0 \leq \ell < p$, such that it is ℓ-owned; it is *shared* otherwise. Two blocks $B_{i_0,\ldots,i_{d-1}}$ and $B_{i'_0,\ldots,i'_{d-1}}$ are said to be *adjacent* if their coordinates differ in just one position k and $|i_k - i'_k| = 1$ (i.e., they share a face). For the sake of simplicity, we assume that n and p are powers of 2^{d-1} and thus the previous values (e.g., $n/p^{1/(d-1)}$) are integral: since d is a constant, this assumption is verified by suitably increasing n and decreasing p by a constant factor which does not asymptotically affect our lower bounds.

In order to establish our main lower bound, we need two preliminary lemmas (whose proofs are deferred to the full version). The first one gives a slack lower bound based on the d-dimensional version of the Loomis-Whitney geometric inequality [21], and resembles the result of Theorem 2 for matrix multiplication when $d = 3$.

▶ **Lemma 3.** *Let $\mathcal{A}_d$ be any algorithm solving the (n, d)-stencil problem, without recomputation, on a BSP with p processors, where $1 < p \leq n^{d-1}$, and denote with W the maximum number of nodes evaluated by a processor. If $W \leq \epsilon n^d$, for an arbitrary constant $\epsilon \in (0, 1)$,*

then the communication complexity of the algorithm is

$$H_{\mathcal{A}_d}(n,p) = \Omega\left(W^{(d-1)/d}\right).$$

Now we need a second lemma that bounds from below the number of messages exchanged by a processor P_ℓ while evaluating nodes in an ℓ-owned block and in an adjacent block which is not ℓ-owned.

▶ **Lemma 4.** *Consider an ℓ-owned block B adjacent to a shared or ℓ'-owned block B', with $\ell \neq \ell'$. Then, the number of messages exchanged by processor P_ℓ for evaluating, without recomputation, nodes in B and B' is $\Omega\left(\frac{n^{d-1}}{p}\right)$.*

The next theorem gives the claimed $\Omega\left(n^{d-1}/(p^{(d-2)/(d-1)})\right)$ lower bound, and its proof is inspired by the argument in [31] for the cube DAG (which however assumes balanced work). The lower bound is matched by the balanced algorithm given in [31], which decomposes the (n,d)-stencil into $p^{d/(d-1)}$ subDAGs of dimension d and size $n/p^{1/(d-1)}$. The proof of the theorem is deferred to the full version.

▶ **Theorem 5.** *Let $\mathcal{A}_d$ be any algorithm for solving the (n,d)-stencil problem, without recomputation, on a BSP with p processors, where $1 < p \leq n^{d-1}$, and let W be the maximum number of nodes evaluated by a processor. If $W \leq \epsilon n^d$, for an arbitrary constant $\epsilon \in (0,1)$, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}_d}(n,p) = \Omega\left(\frac{n^{d-1}}{p^{(d-2)/(d-1)}}\right).$$

4 Sorting

In this section we give a lower bound to the communication complexity of comparison-based sorting algorithms. Comparison sorting is defined as the problem in which a given set X of n input keys from an ordered set has to be sorted, such that the only operations allowed on members of X are pairwise comparisons. Our bound only requires that no processor does more than a constant fraction ϵ of the $\Theta(n \log n)$ comparisons required by any comparison sorting algorithm, for any $\epsilon \in (0,1)$, and does not impose any protocol on the distribution of the inputs and the outputs on the processors, nor upper bounds to the size of their local memories, or specific communication patterns. As for previous work, we still need the technical assumptions that the inputs are not initially replicated, and that the processors store only a constant number of copies of any input key at any moment during the execution of the algorithm.

The main result follows from the application of two lemmas, each of which provides a different and independent lower bound to the communication complexity of sorting. Both rely on non-trivial counting arguments, adapted from [2, 1], that hinge on the fact that any comparison sorting algorithm must be able to distinguish between all the $n!$ permutations of the n inputs. The first lemma provides a lower bound as a function of the maximum number S of input keys initially held by a processor. The second gives a lower bound as a function of the number Π of permutations that can be distinguished before any communications take place. We begin by stating the first lemma.

▶ **Lemma 6.** *Let $\mathcal{A}$ be any algorithm sorting n keys on a BSP with p processors, with $1 < p \leq n$, and let S denote the maximum number of input keys initially held by a processor. If each processor performs at most $\epsilon(n \log n)$ comparisons, with ϵ being an arbitrary constant*

in $(0,1)$*, and the input is not initially replicated, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n,p) = \Omega(S).$$

We now provide a second lemma, which bounds from below the communication complexity of sorting in BSP as a function of the number Π of permutations that can be distinguished before any communications take place, that is, when processors can only compare their local inputs.

▶ **Lemma 7.** *Let* $\mathcal{A}$ *be any algorithm sorting* n *keys on a BSP with* p *processors, where* $1 < p \leq n$*, and let* Π *be the number of distinct permutations that can be distinguished by* $\mathcal{A}$ before *the second superstep, that is, by comparing the inputs that (possibly) reside initially in the processors' local memories. If* $\mathcal{A}$ *stores only a constant number of copies of any key at any time instant, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n,p) = \Omega\left(\frac{n\log n - \log\Pi}{p\log(n/p)}\right).$$

Now we are ready to prove the main result of this section, an $\Omega((n\log n)/(p\log(n/p)))$ lower bound to the communication complexity of any comparison sorting algorithm. The result follows by combining the bounds given by the previous two lemmas. Both bounds are not tight when considered independently, the first (Lemma 6) because it is weak when at the beginning the input keys tend to be distributed evenly among the processors, the second (Lemma 7) because it is weak when the input keys tend to be concentrated on one or few processors. However, the simultaneous application of both provides the sought (tight) lower bound. Once again, the proof of the following theorem is deferred to the full version.

▶ **Theorem 8.** *Let* $\mathcal{A}$ *be any algorithm for sorting* n *keys on a BSP with* p *processors, with* $1 < p \leq n$*. If each processor performs at most* $\epsilon(n\log n)$ *comparisons, with* ϵ *being an arbitrary constant in* $(0,1)$*, the inputs are not initially replicated, and the* p *processors store only a constant number of copies of any key at any time instant, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n,p) = \Omega\left(\frac{n\log n}{p\log(n/p)}\right).$$

5 Fast Fourier Transform

In this section we consider the problem of computing the Discrete Fourier Transform of n values using the n-input FFT DAG. In the FFT DAG, a vertex is a pair $\langle w,l\rangle$, with $0 \leq w < n$ and $0 \leq l \leq \log n$, and there exists an arc between two vertices $\langle w,l\rangle$ and $\langle w',l'\rangle$ if $l' = l+1$, and either w and w' are identical or their binary representations differ exactly in the l'-th bit. We show that, when no processor computes more than a constant fraction of the total number of vertices of the DAG, the communication complexity is $\Omega(n\log n/(p\log(n/p)))$. Our bound does not assume any particular I/O protocol, and only requires that every input resides in the local memory of exactly one processor before the computation begins; as for preceding results, our bound also hinges on the restriction on the nature of the computation whereby each vertex of the FFT DAG is computed exactly once. The bound is tight for any $p \leq n$, and is achieved by the well-known recursive decomposition of the DAG into two sets of smaller $\sqrt{n}$-input FFT DAGs with each set containing $\sqrt{n}$ of such subDAGs (see, e.g., [7]).

We will first establish a lemma which, under the same hypothesis of the main result, provides a lower bound to the communication complexity as a function of the maximum work performed by any processor. The proof of the lemma (which, for space limitations, is deferred to the full version) is based on a bandwidth argument, which exploits the fact that an FFT DAG can perform all cyclic shifts (see, e.g., [20]), and on the following technical result which is implicit in the work of Hong and Kung (a simplified proof is due to Aggarwal and Vitter [2]).

▶ **Lemma 9** ([16]). *Consider the computation of the n-input FFT DAG. During the computation, if a processor accesses at most S nodes of the DAG, then it can evaluate at most $2S \log S$ nodes, for any $S \geq 2$.*

▶ **Lemma 10.** *Let $\mathcal{A}$ be any algorithm computing, without recomputation, an n-input FFT DAG on a BSP with p processors, with $1 < p \leq n$, and let W be the maximum number of nodes of the FFT DAG computed by a processor. If $W \leq \epsilon(n \log n)$, for an arbitrary constant in $(0, 1)$, and the inputs are not initially replicated, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n, p) = \Omega\left(\frac{W}{\log W}\right).$$

The main result of this section follows by a simple application of the preceding lemma and of a result implicit in the proof of the lower bound due to Bilardi et al. [9, Corollary 1]. The proof is deferred to the full version.

▶ **Theorem 11.** *Let $\mathcal{A}$ be any algorithm computing, without recomputation, an n-input FFT DAG on a BSP with p processors, where $1 < p \leq n$. If each processor computes at most $\epsilon(n \log n)$ nodes, for an arbitrary constant in $(0, 1)$, of the FFT DAG and the inputs are not initially replicated, then the communication complexity of the algorithm is*

$$H_{\mathcal{A}}(n, p) = \Omega\left(\frac{n \log n}{p \log(n/p)}\right).$$

6 Conclusions

We have presented new lower bounds on the amount of communication required to solve some key computational problems in distributed-memory parallel architectures. All our bounds have the same functional form of previous results that appear in the literature; however, the latter are built by making a critical use of some assumptions that rule out a large part of possible algorithms. The novelty and the significance of our results stem from the assumptions under which our lower bounds are developed, which are much weaker than those used in previous work.

Our bounds are derived within the BSP model of computation, but can be easily extended to other models for distributed computations based on or similar to the BSP, such as LogP [13] and MapReduce [18, 23]. Moreover, we believe that our results can be also ported to models for multicore computing (see, e.g., [10, 33, 12]), since our proofs are based on some techniques that have already been exploited in this scenario.

There is still much to do towards the establishment of a definitive theory of communication-efficient algorithms. In fact, we were not able to remove all the restrictions there were in place in previous work: in some cases our lower bounds still make use of some technical assumptions, such as the non-recomputation of intermediate results, or restrictions on the replication of

input data. Although it seems that such restrictions can be relaxed to encompass a small amount of recomputation or input replication, it is an open question to assess whether these assumptions are inherent to our proof techniques or can be removed. In particular, it is not clear, in general, when recomputation has the power to reduce communications, since many lower bound techniques do not apply in this more general scenario (see, e.g., [5]). Providing tight lower bounds that hold also when recomputation is allowed is a fascinating and challenging avenue for future research.

Acknowledgements. The authors would like to thank Gianfranco Bilardi and Andrea Pietracaprina for useful comments.

References

1 Alok Aggarwal, Ashok K. Chandra, and Marc Snir. Communication complexity of PRAMs. *Theoretical Computer Science*, 71:3–28, 1990.

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

3 Grey Ballard, James Demmel, Olga Holtz, Benjamin Lipshitz, and Oded Schwartz. Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds. In *Proc. 24th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 77–79, 2012.

4 Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Minimizing communication in numerical linear algebra. *SIAM J. on Matrix Analysis and Applications*, 32(3):866–901, 2011.

5 Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Graph expansion and communication costs of fast matrix multiplication. *J. of the ACM*, 59(6):32:1–32:23, 2012.

6 Gianfranco Bilardi, Andrea Pietracaprina, and Paolo D'Alberto. On the space and access complexity of computation DAGs. In *Proc. 26th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 47–58, 2000.

7 Gianfranco Bilardi, Andrea Pietracaprina, Geppino Pucci, and Francesco Silvestri. Network-oblivious algorithms. In *Proc. 21st IEEE International Parallel and Distributed Processing Symposium*, pages 1–10, 2007.

8 Gianfranco Bilardi and Franco Preparata. Processor-time tradeoffs under bounded-speed message propagation: Part II, lower bounds. *Theory of Computing Systems*, 32(5):531–559, 1999.

9 Gianfranco Bilardi, Michele Scquizzato, and Francesco Silvestri. A lower bound technique for communication on BSP with application to the FFT. In *Proc. 18th International Conference on Parallel Processing*, pages 676–687, 2012.

10 Guy E. Blelloch, Rezaul A. Chowdhury, Phillip B. Gibbons, Vijaya Ramachandran, Shimin Chen, and Michael Kozuch. Provably good multicore cache performance for divide-and-conquer algorithms. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 501–510, 2008.

11 Thomas Cheatham, Amr F. Fahmy, Dan C. Stefanescu, and Leslie G. Valiant. Bulk synchronous parallel computing – a paradigm for transportable software. In *Proc. 28th Hawaii International Conference on System Sciences*, pages 268–275, 1995.

12 Rezaul Alam Chowdhury, Vijaya Ramachandran, Francesco Silvestri, and Brandon Blakeley. Oblivious algorithms for multicores and networks of processors. *Journal of Parallel and Distributed Computing*, 73(7):911–925, 2013.

13 David E. Culler, Richard M. Karp, David A. Patterson, Abhijit Sahay, Eunice E. Santos, Klaus E. Schauser, Ramesh Subramonian, and Thorsten von Eicken. LogP: A practical model of parallel computation. *Communications of the ACM*, 39(11):78–85, 1996.

14 Matteo Frigo and Volker Strumpen. Cache oblivious stencil computations. In *Proc. 19th International Conference on Supercomputing*, pages 361–366, 2005.

15 Michael T. Goodrich. Communication-efficient parallel sorting. *SIAM Journal on Computing*, 29(2):416–432, 1999.

16 Jia-Wei Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proc. 13th ACM Symposium on Theory of Computing*, pages 326–333, 1981.

17 Dror Irony, Sivan Toledo, and Alexandre Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *Journal of Parallel and Distributed Computing*, 64(9):1017–1026, 2004.

18 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 938–948, 2010.

19 Leslie Robert Kerr. *The Effect of Algebraic Structure on the Computational Complexity of Matrix Multiplication*. PhD thesis, Cornell University, 1970.

20 Frank T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers Inc., 1992.

21 L.H. Loomis and H. Whitney. An inequality related to the isoperimetric inequality. *Bulletin of The American Mathematical Society*, 55:961–962, 1949.

22 Christos H. Papadimitriou and Jeffrey D. Ullman. A communication-time tradeoff. *SIAM Journal on Computing*, 16(4):639–646, 1987.

23 Andrea Pietracaprina, Geppino Pucci, Matteo Riondato, Francesco Silvestri, and Eli Upfal. Space-round tradeoffs for MapReduce computations. In *Proc. 26th International Conference on Supercomputing*, pages 235–244, 2012.

24 Desh Ranjan, John Savage, and Mohammad Zubair. Strong I/O lower bounds for binomial and FFT computation graphs. In *Proc. 17th Annual International Conference on Computing and Combinatorics*, pages 134–145, 2011.

25 John E. Savage. Extending the Hong-Kung model to memory hierarchies. In *Proc. First Annual International Conference on Computing and Combinatorics*, pages 270–281, 1995.

26 John E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley Longman Publishing Co., Inc., 1998.

27 Michele Scquizzato and Francesco Silvestri. Communication lower bounds for distributed-memory computations. *CoRR*, abs/1307.1805, 2013.

28 Edgar Solomonik and James Demmel. Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms. In *Proc. 17th International Conference on Parallel Processing*, pages 90–109, 2011.

29 Alexander Tiskin. BSP (bulk synchronous parallelism). In *Encyclopedia of Parallel Computing*, pages 192–199. Springer, 2011.

30 Alexandre Tiskin. Bulk-synchronous parallel multiplication of Boolean matrices. In *Proc. 25th Int'l Colloquium on Automata, Languages and Programming*, pages 494–506, 1998.

31 Alexandre Tiskin. *The Design and Analysis of Bulk-Synchronous Parallel Algorithms*. PhD thesis, University of Oxford, 1998.

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

33 Leslie G. Valiant. A bridging model for multi-core computing. *Journal of Computer and System Sciences*, 77(1):154–166, 2011.

34 Chuan-Lin Wu and Tse-Yun Feng. The universality of the shuffle-exchange network. *IEEE Transactions on Computers*, 30:324–332, 1981.

35 I-Chen Wu and H. T. Kung. Communication complexity for parallel divide-and-conquer. In *Proc. 32nd annual Symposium on Foundations of Computer Science*, pages 151–162, 1991.