Communication–Space Tradeoffs for Unrestricted Protocols *

Paul Beame † Martin Tompa † ‡ Peiyuan Yan §

Abstract

This paper introduces communicating branching programs, and develops a general technique for demonstrating communication-space tradeoffs for pairs of communicating branching programs. This technique is then used to prove communication-space tradeoffs for any pair of communicating branching programs that hashes according to a universal family of hash functions. Other tradeoffs follow from this result. As an example, any pair of communicating Boolean branching programs that computes matrix-vector products over GF(2) requires communication-space product $\Omega(n^2)$. These are the first examples of communication-space tradeoffs on a completely general model of communicating processes.

1 Communication and Space

The amount of communication required among processors cooperatively performing a computation is often the dominant factor in determining the efficiency of parallel or distributed systems, in both practical and theoretical terms. In addition, communication complexity has found surprising applications in the complexity of Boolean circuits (Karchmer and Wigderson [14], Raz and Wigderson [19]), Boolean decision trees (Hajnal, Maass, and Turán [13]), combinatorial optimization (Yannakakis [23]), VLSI (Aho, Ullman and Yannakakis [3], Lipton and Sedgewick [16], Mehlhorn and Schmidt [18], Yao [25]), and pseudorandom number generators (Babai, Nisan, and Szegedy [5]).

Nearly all previous work on the communication complexity of various problems has focused on their communication requirements alone, in the absence of any limitations on the individual processors. Lam, Tiwari, and Tompa [15] initiated the study of communication complexity when the processors have limited work space. As is customary, the systems studied consist of two communicating processors that are given private inputs x and y, respectively, and are to output some function $f(x, y)$. With no restriction on the workspace it is impossible to prove superlinear lower bounds on the amount of communication, since one processor can send its entire input to the other, which then computes and outputs $f(x, y)$. In contrast, Lam, Tiwari, and Tompa proved several nonlinear lower bounds on communication in the straight-line model, when space is limited. For example, one of their results of particular relevance to what follows is that multiplication of an $n \times n$ matrix by an n-vector in the Boolean straight-line model with one-way communication requires communication $C = \Theta(n^2/S)$ when the processors' workspace is restricted to S.

In this paper we remove the restrictions of straight-line computation and one-way communication, proving for the first time communication-space tradeoffs on a completely general model of communicating processes. This result is analogous to Borodin and Cook's time-space tradeoff for sorting on a general sequential model [7].

More specifically, we introduce the notion of communicating branching programs. We use these to demonstrate that if one of the branching programs is given a member h of a universal family of hash functions (Carter and Wegman [9, 10]) and

*This material is based upon work supported in part by the National Science Foundation, under grants CCR-8858799 and CCR-8907960, and by IBM, under Research Contract 16980043.

†Department of Computer Science and Engineering, FR-35, University of Washington, Seattle, Washington 98195

‡Thomas J. Watson Research Center, IBM Research Division, P. O. Box 218, Yorktown Heights, New York 10598

§Mathematics Department, Lycoming College, Williamsport, Pennsylvania 17701

CH2925-6/90/0000/0420$01.00 © 1990 IEEE

the other is given x, and their goal is to compute $h(x)$ cooperatively, then their communication C and space S must satisfy the tradeoff $CS = \Omega(nm)$, where h maps n-bit inputs to m-bit outputs. As an example, any pair of communicating Boolean branching programs that multiplies an $n \times n$ matrix by an n-vector over GF(2) satisfies $CS = \Omega(n^2)$. Similar applications hold over more general finite fields, and for other hash functions such as arithmetic over large finite fields, convolution, and matrix multiplication.

If a single processor can compute $f(x, y)$ in time C and space S, then a system of two processors can compute $f(x, y)$ in communication $O(C)$ and space $O(S)$, simply by communicating every intermediate value computed by either. Thus, the lower bounds outlined above imply the corresponding time-space tradeoffs of Grigoryev [12] for straight-line programs and Abrahamson [1] for branching programs. The converse, however, is false. Whereas the time T and space S must satisfy $TS = \Omega(n^2)$ when computing the discrete Fourier transform [21, 27, 1] or sorting [21, 8, 6], Lam, Tiwari, and Tompa [15] demonstrated that both of these functions can be computed in linear communication steps and $O(\log n)$ space simultaneously. Thus, these results strictly generalize previous time-space tradeoffs.

2 Communicating Branching Programs

The general framework for dealing with problems of two party communication requires an accurate notion of both the computational power of the two parties involved and their method of communicating with each other. A restricted model in which each party executes a straight-line program was defined by Lam, Tiwari, and Tompa [15]. In their model each straight-line program is augmented with *send* and *receive* instructions. They leave open the question of defining an appropriate nonoblivious model.

Since branching programs have proved to be useful sequential models for the simultaneous measure of time and space it is natural to use them to model the communicating parties. Making the analogous changes to branching programs that Lam, Tiwari, and Tompa made to straight-line programs leads to the following model.

A *communicating pair of (Boolean) branching programs* consists of two branching programs, known as the X-program and the Y-program, that have input vectors $x \in X = \{0,1\}^{n_X}$ and $y \in Y = \{0,1\}^{n_Y}$, respectively. The X-program is a labelled directed acyclic graph with a designated start node, and each of whose nodes has outdegree 0 or 2. Each node of outdegree 2 is labelled either by an index in $\{1, \ldots, n_X\}$ or by **receive**, and its two emanating edges are labelled 0 and 1, respectively. In addition to its 0 or 1 label, an edge may be labelled by a set of output statements of the form $z_j = 0$ or $z_j = 1$. Also, an edge may have a sequence of labels of the form **send**(0) or **send**(1). The Y-program is defined analogously.

The pair of branching programs computes a function $f : X \times Y \to Z \subseteq \{0,1\}^{\leq n_Z}$ in the following natural way. Each program accesses its portion of the input and, starting at its start node, operates like a conventional branching program by following the edge labelled x_i (respectively, y_i) when encountering a node labelled i. Outputs are produced according to the output label on this edge, if any. When a program encounters a **receive** node it waits until the other program traverses an edge labelled **send**(0) or **send**(1), and then the receiving program branches based on the value of this bit. Similarly a program executing a **send** is blocked until the other program reaches a **receive** node. The function f is computed correctly on inputs x and y if the union of the outputs produced by the two programs is consistent and comprises the bits of $f(x, y)$.

The *space* of each branching program is the base 2 logarithm of the number of its nodes. (This is the standard definition for branching programs, motivated by the fact that each node represents a different configuration of the program.) The *space* of the pair of programs is the maximum of the space of the two branching programs, and the *communication* is the length of the longest sequence of **send-receive** pairs executed on any input (x, y). The definitions can be generalized to communicating R-way branching programs for any R.

This model is a very natural one and a very general one as well. It can simulate, for example, two

communicating space-bounded random access machines with a common write-only area for their output values.

One aspect of communicating branching programs that is somewhat subtle is the way in which output values are produced. Since all branchings of one of the programs that do not affect its communication with the other program are hidden from that other program, output values may be produced by one branching program without the explicit knowledge of the other branching program. In fact all the bits communicated by the pair of branching programs may not be sufficient to determine the value of the function. However, the model in which all output values are communicated is a useful special case. We say that a pair of communicating branching programs is *open* if and only if encodings of all the output statements produced by either processor are communicated bit by bit to the other processor.

The following lemma translates lower bounds on open pairs of communicating branching programs to unrestricted ones. The loss in the translation is not severe especially in the case of pairs of branching programs that are *output-oblivious*, that is, the order in which they produce their output values is fixed in advance.

Lemma 1: If a function $f : X \times Y \to \{0,1\}^{\leq n_z}$ is computed by a pair $\mathcal{P}$ of communicating branching programs using communication C and space S, then f can be computed by an open pair of communicating branching programs using communication $O(C + n_z \log n_z)$ and space $O(S + \log n_z)$. Furthermore, if $\mathcal{P}$ is output-oblivious then this can be done using communication $O(C + n_z)$ and space $O(S + \log n_z)$.

Proof: Each output value can be communicated by encoding its index and its value in binary and this takes $O(\log n_z)$ bits. To distinguish output values from communication values an additional bit for each output or communication bit in the original programs will be sufficient. The additional space is required for the **receive** nodes.

If the original pair is output-oblivious then the $O(\log n_z)$ bits per output value are not required, since both programs know in advance which output value is being produced and only its value must be communicated. □

3 The General Lower Bound

The technique we develop here is an extension of the technique of Borodin *et al.* [7, 8] for time-space tradeoffs on sequential branching programs. To prove time-space tradeoffs for a function one must find two things (ignoring most of the quantitative aspects):

1. a probability distribution on the set of inputs such that, with high probability, a large number of output values are produced on an input, and

2. a proof that, given the distribution in (1), for any way of fixing a limited number of input variables, the probability that an input whose variables are so fixed produces a fixed set of k output values is exponentially small in k.

We develop a similar pair of conditions that allow proofs of communication-space tradeoffs. Our general technique is directly applicable to open pairs of communicating branching programs. Results for arbitrary communicating branching programs follow by the reduction given in Lemma 1.

In order to motivate the properties that are appropriate for showing lower bounds for pairs of communicating branching programs, we first develop some facts about their operation.

Fix any pair (u, v) of nodes in the pair of communicating branching programs, u in the X-program and v in the Y-program, and consider the action of the branching programs on input pairs (x, y) starting at (u, v). For each input pair (x, y) we can follow the paths that the computation would take starting at (u, v), and stop when either a total of c bits of communication have been sent in both directions or the programs halt. (It is possible that there is no consistent computation on input (x, y) starting at (u, v), but any input (x, y) that reaches (u, v) will have such a computation. We consider an input pair (x, y) for which there is a consistent computation.) This produces a string $\gamma^c_{(u,v)}(x, y)$ of communication bits on (x, y) such

that $|\gamma^c_{(u,v)}(x,y)| \leq c$, with inequality only if the programs halt before c bits of communication have been sent. For each string $\alpha \in \{0,1\}^{\leq c}$ we can define a set

$$R^\alpha_{(u,v)} = \{(x,y) \in X \times Y \mid \gamma^c_{(u,v)}(x,y) = \alpha\}.$$

A set $R \subseteq X \times Y$ is a *rectangle* if and only if there are sets $A \subseteq X$ and $B \subseteq Y$ such that $R = A \times B$.

Lemma 2: Let (u,v) be a pair of nodes in a pair of communicating branching programs, u in the X-program and v in the Y-program. The sets $R^\alpha_{(u,v)}$ for $\alpha \in \{0,1\}^{\leq c}$ are disjoint rectangles in $X \times Y$ whose union contains all input pairs (x,y) that reach (u,v).

Proof: The fact that the sets $R^\alpha_{(u,v)}$ are disjoint is immediate from their definition. It is also clear that if (x,y) reaches (u,v) then $\gamma^c_{(u,v)}(x,y) = \alpha$ is defined and so $(x,y) \in R^\alpha_{(u,v)}$. The fact that each $R^\alpha_{(u,v)}$ is a rectangle follows by standard arguments in communication complexity (Yao [24]). It is proved inductively on the prefixes of α. □

We are now ready to state properties of a function that make it possible to prove communication-space tradeoffs:

If $f(x,y) = z$ we will call the bits of x and y *input values* and the bits of z *output values*. For a function $f : X \times Y \to Z$ and a distribution $\mathcal{D}$ on $X \times Y$, the two properties are as follows:

Property A: There are $p \leq 1$ and a positive integer m such that

$$\Pr_{\mathcal{D}}[f(x,y) \text{ has at least } m \text{ output values }] \geq p.$$

Property B: There are $\beta < 1$, $q < 1$, $a \geq 0$, and a positive integer K such that, for all positive integers $k \leq K$, the following holds: Let $R \subseteq X \times Y$ be any rectangle such that $\Pr_{\mathcal{D}}[(x,y) \in R] \geq q$. Then, for any set $V = \{z_{i_1} = b_1, \ldots, z_{i_k} = b_k\}$ of k output values,

$$\Pr_{\mathcal{D}}[f(x,y) \text{ is consistent with } V \mid (x,y) \in R] \leq \beta^{k-a}.$$

Theorem 3: Suppose that for $f : X \times Y \to Z$ there is a distribution $\mathcal{D}$ on $X \times Y$ such that Properties A and B hold with $p > \max(2^{-S+1}\beta^{-a}, 2\beta^{m-a})$. Then any open pair of communicating branching programs $\mathcal{P}$ computing f using space S and communication C must satisfy $C \cdot S = \Omega(m \log(1/\beta) \min(K, \log(p/q)))$.

Note that although the hypothesis $p > 2^{-S+1}\beta^{-a}$ refers to the space bound it is even weaker than $p > 2\beta^{-a}/n$, where n is the number of input bits, since reading this many bits requires $S \geq \log_2 n$. Note also that, since there are 2^k choices for k output values and Property B must hold for all choices of output values, β must be at least $1/2$. Thus the $\log(1/\beta)$ term is at most a constant. However, β may be close to 1, so that the $\log(1/\beta)$ term in the lower bound may be less than 1.

Proof: Let $\mathcal{P}$ be an open pair of communicating branching programs computing f and let C and S be the communication and space, respectively, used by $\mathcal{P}$. Let $c = \min(C, \lfloor \log_2(p/q) - 2S - 2 \rfloor)$.

Fix any pair (u,v) of nodes, u in the X-program and v in the Y-program of $\mathcal{P}$. Fix $\alpha \in \{0,1\}^{\leq c}$ such that all input pairs (x,y) that reach (u,v) and are in $R^\alpha_{(u,v)}$ have some set V of $k \leq K$ output values in common; that is, α determines k output values. Let (x,y) be chosen at random according to $\mathcal{D}$. Suppose that $\Pr_{\mathcal{D}}[(x,y) \in R^\alpha_{(u,v)}] \geq q$. By Property B, Lemma 2, and the fact that $\mathcal{P}$ correctly computes f,

$$\Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \mid (x,y) \in R^\alpha_{(u,v)} \wedge \alpha \text{ determines } k \text{ output values}] \leq \beta^{k-a}.$$

Call $R^\alpha_{(u,v)}$ *tiny* if and only if $\Pr_{\mathcal{D}}[(x,y) \in R^\alpha_{(u,v)}] < q$. Let T be the set of inputs pairs that are in *any* tiny $R^\alpha_{(u,v)}$. We will discard the input pairs in T. Since there are at most 2^{c+1} choices of α and at most 2^{2S} choices of the pair (u,v),

$$\Pr_{\mathcal{D}}[(x,y) \in T] < 2^{c+1}2^{2S}q \leq p/2.$$

Using the definition of T, for all $k \leq K$,

$$\Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \mid (x,y) \in R^\alpha_{(u,v)} \wedge \alpha \text{ determines } k \text{ output values} \wedge (x,y) \notin T] \leq \beta^{k-a}.$$

Thus

$$\Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \wedge (x,y) \in R^{\alpha}_{(u,v)} \wedge \alpha \text{ determines } k \text{ output values} \wedge (x,y) \notin T] \leq \beta^{k-a}.$$

For fixed (u,v), Lemma 2 says that the sets $R^{\alpha}_{(u,v)}$ are disjoint and their union contains all points (x,y) that reach (u,v), so for all $k \leq K$,

$$\Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \wedge \gamma^{c}_{(u,v)}(x,y) \text{ determines } k \text{ output values} \wedge (x,y) \notin T] \leq \max_{\alpha \in \{0,1\}^{\leq c}} \Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \wedge (x,y) \in R^{\alpha}_{(u,v)} \wedge \alpha \text{ determines } k \text{ output values} \wedge (x,y) \notin T] \leq \beta^{k-a}. \quad (1)$$

Now by Property A, $\mathcal{P}$ produces at least m output values with probability at least p. Thus

$$\Pr_{\mathcal{D}}[f(x,y) \text{ has} \geq m \text{ output values} \wedge (x,y) \notin T] \geq p/2. \quad (2)$$

Inequalities (1) and (2) will be combined in two different ways. For the first, it follows that

$$C > \min(K, \lfloor \log_2(p/q) - 2S - 2 \rfloor). \quad (3)$$

For assume to the contrary that $C \leq K$ and $C \leq \lfloor \log_2(p/q) - 2S - 2 \rfloor$. Then $c = C$. In Inequality (1), choose u and v to be the start nodes of their respective branching programs, and choose $k = m$. Since $\mathcal{P}$ is open, $k = m \leq C \leq K$, so that Inequality (1) holds. Combining this with Inequality (2) yields $p/2 \leq \beta^{k-a} = \beta^{m-a}$, contradicting the hypothesis $p > 2\beta^{m-a}$.

Now let $k = \min(K, m/\lceil C/c \rceil)$ in Inequality (1). For every input (x,y) on which $f(x,y)$ has at least m output values, (x,y) reaches some pair (u,v) of nodes such that $\gamma^{c}_{(u,v)}(x,y)$ determines at least $m/\lceil C/c \rceil \geq k$ output values, for otherwise fewer than m output values are produced by $\mathcal{P}$ on input (x,y). Therefore, since $k \leq K$ and there are at most 2^{2S} node pairs (u,v),

$$p/2 \leq \Pr_{\mathcal{D}}[(\exists u,v)(x,y) \text{ reaches } (u,v) \wedge \gamma^{c}_{(u,v)}(x,y) \text{ determines } k \text{ output values} \wedge (x,y) \notin T] \leq 2^{2S}\beta^{k-a}.$$

Solving yields $2S + \log_2(2\beta^{-a}/p) \geq k \log_2(1/\beta)$. Since $p \geq 2^{-S+1}\beta^{-a}$,

$$3S \geq k \log_2(1/\beta).$$

CASE 1 ($K < m/\lceil C/c \rceil$): Then $k = K$. Since $\mathcal{P}$ is open, $C \geq m$, so $3CS \geq mK \log_2(1/\beta)$.

CASE 2 ($S+2 \geq \log_2(p/q)/4$): Since $\log_2(1/\beta) \leq 1$ and $C \geq m$ because $\mathcal{P}$ is open, $C(S+2) \geq m \log_2(1/\beta) \log_2(p/q)/4$.

CASE 3 ($K \geq m/\lceil C/c \rceil$ AND $S + 2 < \log_2(p/q)/4$): Then $k = m/\lceil C/c \rceil \geq mc/(C+c)$, so

$$\begin{aligned} 6CS &\geq 3(C+c)S \\ &\geq (C+c)k \log_2(1/\beta) \\ &\geq mc \log_2(1/\beta) \\ &\geq m \log_2(1/\beta) \min(C, \lfloor \log_2(p/q) - 2S - 2 \rfloor) \\ &\geq m \log_2(1/\beta) \min(K, \lfloor \log_2(p/q) - 2S - 2 \rfloor) \\ &\geq m \log_2(1/\beta) \min(K, \log_2(p/q)/2). \end{aligned}$$

The penultimate inequality follows from Inequality (3). □

It is not too hard to see how the argument and Properties A and B can be modified to deal with R-way branching programs (Borodin and Cook [7]) or when the output values described in Properties A and B are of a restricted type (as in, for example, Abrahamson [2]).

4 Hash Functions

We now apply the lower bound technique of the previous section to universal families of hash functions (Carter and Wegman [9, 10]). This will allow us to obtain lower bounds for a variety of interesting computational problems. We make use of a beautiful analog due to Mansour, Nisan, and Tiwari [17] of a lemma of Lindsey [4, 11] concerning Hadamard matrices.

Note that our results (and those in [17]) use the more restrictive definition of a universal family of hash functions given by Carter and Wegman in [10] (which they called 'strongly universal' in [10]) rather than the somewhat broader definition given in [9]. To emphasize the nature of this stronger requirement we will call such families *pairwise universal.*

A pairwise universal family H of hash functions from a set X to a set Z satisfies the following two properties for h chosen uniformly at random from H:

1. For any $x \in X$, $h(x)$ is uniformly distributed in Z.
2. The events $h(x) = z$ are pairwise independent for $x \in X$ and $z \in Z$.

We say that a pair of communicating branching programs computes the universal family of hash functions H if and only if it computes the function $f : X \times H \to Z$ given by $f(x,h) = h(x)$.

Of the two properties of a function required to apply our lower bound technique, Property B is the more difficult to prove. The following lemma on pairwise universal hash functions is critical in proving Property B for families of hash functions.

Lemma 4: [Mansour, Nisan, and Tiwari [17]] Let H be a pairwise universal family of hash functions from X to Z. Let $A \subseteq X$, $B \subseteq H$, and $C \subseteq Z$. Then

$$\left| \Pr_{x \in A, h \in B}[h(x) \in C] - \frac{|C|}{|Z|} \right| \leq \sqrt{\frac{|H| \cdot |C|}{|A| \cdot |B| \cdot |Z|}}.$$

This lemma is used by Mansour, Nisan, and Tiwari [17] to prove time-space tradeoffs for computing hash functions. A somewhat weaker form of this lemma was proved independently by Yan [22] for the special case when the family of hash functions is given by matrix-vector product over $GF(2)$.

Theorem 5: Any open pair of communicating branching programs computing a pairwise universal family of hash functions from X to Z requires communication C and space $S \geq \log_2 n$ such that $C \cdot S = \Omega(nm)$, where $n = \lfloor \log_2 |X| \rfloor$, $m = \lfloor \log_2 |Z| \rfloor - 1$, and $Z \subseteq \{0,1\}^{\leq m+l}$ for $l < \min(\log_2 n, m) - 3$.

Proof: Let $\mathcal{D}$ be the uniform distribution on pairs (x,h). Since $h(x)$ is uniformly distributed in Z, Property A is satisfied with $p = 1/2$. Let $R = A \times B$ satisfy $|R| \geq 2^{K-n}|X \times H|$, where $K = n/2$. For any set $V = \{z_{i_1} = b_1, \ldots, z_{i_k} = b_k\}$ of $k \leq K$ output values, let $C \subseteq Z$ be the set of vectors consistent with V. At most $2^{m+l-k+1}$ vectors are in C so that $|C|/|Z| \leq 2^{-(k-l)}$. Then Lemma 4 states that

$$\begin{aligned} &\Pr_{\mathcal{D}}[h(x) \text{ is consistent with } V \mid (x,h) \in R] \\ &\leq \frac{|C|}{|Z|} + \sqrt{\frac{|H|}{|R|} \cdot \frac{|C|}{|Z|}} \\ &\leq 1/2^{k-l} + 1/\sqrt{2^{K-n}|X|2^{k-l}} \\ &\leq 1/2^{k-l-1}. \end{aligned}$$

Thus Property B is satisfied with $q = 2^{-n/2}$, $\beta = 1/2$, $a = l + 1$, and $K = n/2$. Since $l < \min(\log_2 n, m) - 3$, $p = 1/2 > 2\beta^{-a}/n$ and $p > 2\beta^{m-a}$ so Theorem 3 implies that $C \cdot S = \Omega(nm)$. □

Corollary 6:
Any pair of communicating branching programs computing a pairwise universal family of hash functions from X to Z with communication C and space $S = o(n/\log m)$ satisfies $C \cdot S = \Omega(nm)$, where $n = \lfloor \log_2 |X| \rfloor$, $m = \lfloor \log_2 |Z| \rfloor - 1$, and $Z \subseteq \{0,1\}^{\leq m+l}$ for $l < \min(\log_2 n, m) - 3$.

Proof: From Lemma 1 and Theorem 5,

$$(C + m \log m)(S + \log m) = \Omega(nm).$$

It will be shown that $S = \Omega(\log m)$, that is,

$$(C + m \log m)S = \Omega(nm).$$

Since, by hypothesis, $Sm \log m = o(nm)$, the conclusion $CS = \Omega(nm)$ will follow.

Since the outputs $h(x)$ must be uniformly distributed in Z, the number of pairs of paths, one from the X-program and one from the H-program, must be at least $|Z|$. The number of such pairs of paths is at most $(2^{2^S})^2$, so that $2^{2^{S+1}} \geq |Z| = 2^m$, that is, $S + 1 \geq \log_2 m$. □

Similar statements to Corollary 6 can be made for each of the following corollaries. We simply state our results for open pairs of branching programs for convenience.

Corollary 7: Any open pair of communicating branching programs computing the product of an $n \times n$ matrix and an n-vector over $GF(2)$ requires communication C and space S such that $C \cdot S = \Omega(n^2)$.

Corollary 8: If $r \geq 2^n$ then any open pair of communicating branching programs computing $f : GF(r) \times GF(r)^2 \rightarrow GF(r)$ given by $f(x,(a,b)) = a \cdot x + b$ (in $GF(r)$) requires communication C and space S such that $C \cdot S = \Omega(n^2)$.

The next two corollaries follow from Theorem 5 exactly as shown by Mansour, Nisan, and Tiwari [17] for time-space tradeoffs.

Corollary 9: Any open pair of communicating branching programs computing the m bit convolution of an n bit string with an $(n+m-1)$ bit string requires communication C and space S such that $C \cdot S = \Omega(nm)$.

Corollary 10: Any open pair of communicating branching programs computing the product of two $n \times n$ matrices over $GF(2)$ requires communication C and space S such that $C \cdot S = \Omega(n^3)$.

Corollaries 8, 9, and 10 are interesting in their own right and because they demonstrate tradeoffs in cases where the lower bound is greater than the total number of inputs that the two programs receive.

Using the natural generalization of communicating branching programs to pairs of r-way branching programs that are allowed to send and receive values in $GF(r)$ one can prove, either by direct simulation or an analog of Theorem 3, the following analog of Theorem 5 for hash functions whose domain and range are vectors over $GF(r)$.

Theorem 11: Any open pair of communicating r-way branching programs computing a pairwise universal family of hash functions from X to Z requires communication C and space S such that $C \cdot S = \Omega(nm \log r)$, where $n = \lfloor \log_r |X| \rfloor$ and $m = \lfloor \log_r |Z| \rfloor - 1$.

This theorem has corollaries analogous to those of Theorem 5 such as the following.

Corollary 12: Any open pair of communicating r-way branching programs computing the product of an $n \times n$ matrix and an n-vector over $GF(r)$ requires communication C and space S such that $C \cdot S = \Omega(n^2 \log r)$.

5 Open Questions

It is an interesting question whether or not similar bounds hold for $\wedge$-$\vee$ matrix-vector product. The results of Lam, Tiwari, and Tompa [15] show that such results do hold in a more restricted model in which the programs are restricted to being oblivious, i.e. straight-line, and the communication is one-way.

A natural approach to proving such a bound would be to try to prove properties A and B for this problem using the distribution $\mathcal{D}$ employed by Babai, Frankl, and Simon [4] for proving a distributional communication complexity lower bound of $\Omega(\sqrt{n})$ for $\wedge$-$\vee$ dot product (i.e. set disjointness) and by Abrahamson [2] for proving a time-space tradeoff of $TS = \Omega(n^{1.5})$ on matrix-vector product. However, this approach cannot yield any interesting communication-space tradeoff since under this distribution, which chooses each input bit independently to be 1 with probability $1/\sqrt{n}$ and 0 with probability $(1 - 1/\sqrt{n})$, the program with the vector can simply communicate its value in expected $O(\sqrt{n} \log n)$ bits to the matrix program which can store this value and perform the rest of the computation on its own.

An alternative approach would be to try to generalize the distribution on inputs that Razborov [20] used to prove that the distributional communication complexity of the set disjointness problem is $\Omega(n)$. Unfortunately, the fact that this distribution does not set the values of the inputs to the two players independently creates serious problems when trying to generalize from a problem whose input consists of two vectors to a problem with a matrix and a vector as input. It seems unlikely that one can maintain sufficient independence between the inputs to the two players while maintaining sufficient information content in the two inputs. It may be that the oblivious one-way result is leading us astray, but it seems more likely that we are unable to generalize it because our technique is fundamentally distributional in nature.

The question of the communication-space tradeoff for $\wedge$-$\vee$ matrix product and $GF(2)$ matrix-vector product raises another interesting question. Suppose that function f on $X \times Y$ has ϵ-error distributional communication complexity (Yao [26]) at

least D_ϵ. Under what circumstances does the function F on $X^n \times Y$ given by $F((x_1, \ldots, x_n), y) = (f(x_1, y), \ldots, f(x_n, y))$ have communication-space tradeoff $\Omega(nD_\epsilon)$? As defined by Yao [26], a lower bound $D_\epsilon \geq k$ can be obtained by showing that, for an appropriate distribution on $X \times Y$, any rectangle R, in which the probability of $f(x, y)$ taking on a particular value is less than ϵ, must have total probability at most $1/2^k$. This condition is very similar to our Property B except that we require that this be true for ϵ much smaller than a constant, that is, for $\epsilon = \beta^k$ for $\beta < 1$. If the only rectangles $A \times B$ in $X^n \times Y$ to handle had A of the form $A_1 \times \cdots \times A_n$ then there would be a direct translation of distributional communication complexity lower bounds for f to those for F. It is not clear what conditions on f will allow the handling of general A as well. The technique of Mansour, Nisan, and Tiwari [17] and Yan [22] implies that it is sufficient to have not only a small probability of a value in such a rectangle R but also a small variance in the probability of the value occurring in the rows (or columns) of R.

Acknowledgements

We thank Johan Håstad, Noam Nisan, Larry Ruzzo, and Prasoon Tiwari for helpful comments.

References

[1] K. Abrahamson. Time-space tradeoffs for branching programs contrasted with those for straight-line programs. In *27th Annual Symposium on Foundations of Computer Science*, pages 402–409, Toronto, Ontario, Oct. 1986. IEEE.

[2] K. Abrahamson. A time-space tradeoff for boolean matrix multiplication. In *31st Annual Symposium on Foundations of Computer Science*, St. Louis, MO, Oct. 1990. IEEE.

[3] A. V. Aho, J. D. Ullman, and M. Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 133–139, Boston, MA, Apr. 1983.

[4] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *27th Annual Symposium on Foundations of Computer Science*, pages 337–347, Toronto, Ontario, Oct. 1986. IEEE.

[5] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols and logspace-hard pseudorandom sequences. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 1–11, Seattle, WA, May 1989.

[6] P. Beame. A general sequential time-space tradeoff for finding unique elements. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 197–203, Seattle, WA, May 1989.

[7] A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM Journal on Computing*, 11(2):287–297, May 1982.

[8] A. Borodin, M. J. Fischer, D. G. Kirkpatrick, N. A. Lynch, and M. Tompa. A time-space tradeoff for sorting on non-oblivious machines. *Journal of Computer and System Sciences*, 22(3):351–364, June 1981.

[9] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[10] J. L. Carter and M. N. Wegman. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–277, 1981.

[11] P. Erdös and J. Spencer. *Probabilistic Methods in Combinatorics.* Academic Press, 1974.

[12] D. Y. Grigoriev. An application of separability and independence notions for proving lower bounds of circuit complexity. In *Notes of Scientific Seminars*, volume 60, pages 38–48. Steklov Mathematical Institute, Leningrad Department, 1976. In Russian.

[13] A. Hajnal, W. Maass, and G. Turán. On the communication complexity of graph properties. In *Proceedings of the Twentieth Annual*

ACM Symposium on Theory of Computing, pages 186–191, Chicago, IL, May 1988.

[14] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require superlogarithmic depth. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 539–550, Chicago, IL, May 1988.

[15] T. Lam, P. Tiwari, and M. Tompa. Tradeoffs between communication and space. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 217–226, Seattle, WA, May 1989.

[16] R. J. Lipton and R. Sedgewick. Lower bounds for VLSI. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 300–307, Milwaukee, WI, May 1981.

[17] Y. Mansour, N. Nisan, and P. Tiwari. The computational complexity of universal hashing. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, Baltimore, MD, May 1990.

[18] K. Mehlhorn and E. M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 330–337, San Francisco, CA, May 1982.

[19] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 287–292, Baltimore, MD, May 1990.

[20] A. A. Razborov. On the distributional complexity of disjointness. In *Automata, Languages, and Programming: 17th International Colloquium*, volume 443 of *Lecture Notes in Computer Science*, pages 249–253, Warwick University, England, July 1990. Springer-Verlag.

[21] M. Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. *Journal of Computer and System Sciences*, 20:118–132, Apr. 1980.

[22] P. Yan. A tradeoff between communication and space. Manuscript, 1989.

[23] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 223–228, Chicago, IL, May 1988.

[24] A. C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 209–213, Atlanta, GA, Apr.-May 1979.

[25] A. C. Yao. The entropic limitations of VLSI computations. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 308–311, Milwaukee, WI, May 1981.

[26] A. C. Yao. Lower bounds by probabilistic arguments. In *24th Annual Symposium on Foundations of Computer Science*, pages 420–428, Tucson, AZ, Nov. 1983. IEEE.

[27] Y. Yesha. Time-space tradeoffs for matrix multiplication and the discrete Fourier transform on any general sequential random-access computer. *Journal of Computer and System Sciences*, 29:183–197, 1984.