COMMUNICATION–SPACE TRADEOFFS FOR UNRESTRICTED PROTOCOLS*

PAUL BEAME†, MARTIN TOMPA†, AND 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 $\Omega(n^2)$, provided the space used is $o(n/\log n)$. These are the first examples of communication-space tradeoffs on a completely general model of communicating processes.

Key words. Communication complexity, lower bound, tradeoff, branching program, universal family of hash functions

AMS subject classifications. 68Q05, 68Q10, 68Q22, 68Q25

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].

* *SIAM Journal on Computing*, vol. 23, no. 3, June 1994, 652–661. 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

‡ Mathematics Department, Lycoming College, Williamsport, Pennsylvania 17701

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 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, provided $S = o(n/\log m)$. 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 $CS = \Omega(n^2)$, provided $S = o(n/\log n)$. 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 the result of every instruction executed by either. Thus, the lower bounds outlined above imply the corresponding time-space tradeoffs of Grigoryev [12] for straight-line programs and Abrahamson [2] 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 [2, 21, 27] or sorting [6, 8, 21], 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 [1, 2, 6, 7, 8, 27] 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 either by an output statement of the form $z_j = 0$ or $z_j = 1$ or a communication statement of the form **send**(0) or **send**(1). The Y-program is defined analogously.

The pair of branching programs computes a function $f : X \times Y \rightarrow Z \subseteq \{0,1\}^{\leq n_Z}$ 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**(b), and then the receiving program follows the edge labelled b. Similarly a program executing a **send** is blocked until the other program reaches a **receive** node.

When a program reaches a node of outdegree 0, it halts. We require that each output bit may only be produced once on any given input pair. The function f is computed correctly on inputs x and y if the union of the outputs produced by the two programs 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 [8], 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 [7].

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 explicitly is a useful special case. We say that a pair of communicating branching programs is *open* if and only if the natural encoding of each output statement produced by either processor is communicated bit by bit to the other processor.

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. Being purposely vague for the moment, their technique requires the following:

(1) a probability distribution on the set of inputs such that, with high probability, a large number of output bits are produced on any given 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 given set of k output bits whose values are fixed in any given way is exponentially small in k.

We develop a similar pair of conditions that allow proofs of communication-space tradeoffs. We first state our general technique for open pairs of communicating branching programs since this is the most natural argument. Then we outline how it can be extended to arbitrary pairs of communicating branching programs.

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 some $c > 0$ and 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 pair (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. (A third possibility is 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 a consistent computation. In the following definitions, we consider such an input pair

(x, y) for which there is a consistent computation.) This produces a string of up to c communication bits, with fewer than c bits only if the programs halt before c bits of communication have been sent. Let $\gamma^c_{(u,v)}(x, y)$ be the following representation of this sequence of up to c bits communicated on input (x, y) starting at (u, v): represent each communicated bit b by two bits, using the extra bit to indicate which program sent b. For each string $\alpha \in \{0, 1\}^{\leq 2c}$ 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 3.1. *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 elements of $\{R^\alpha_{(u,v)} \mid \alpha \in \{0, 1\}^{\leq 2c}\}$ 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. These properties will depend on certain parameters p, m, β, q, a, and K that will be set in later applications.

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 $0 < p \leq 1$ and a positive integer m such that

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

(Recall that, as (x, y) varies, $f(x, y)$ may have varying lengths.)

Property B: There are $0 < \beta < 1$, $0 < q < 1$, $a \geq 2$, 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,

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

THEOREM 3.2. *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 $p > \max(a2^{-S+1}, 2a\beta^m)$. Then any open pair $\mathcal{P}$ of communicating branching programs computing f using space S and communication C must satisfy*

$$C \cdot S = \Omega(m \log_a(1/\beta) \min(K, \log(p/q))).$$

(Note that, although the hypothesis $p > a2^{-S+1}$ refers to the space bound, it is even weaker than the relatively innocuous assumption $p > 2a/n$, 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, $2^k a\beta^k \geq 1$. Thus, $\log_a(1/\beta) \leq \log_a 2 + 1/k \leq 2$, so $\log_a(1/\beta)$

contributes at most a constant factor to the lower bound. However, β may be close to 1, so that the $\log_a(1/\beta)$ factor in the lower bound may be close to 0.)

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 $\mathcal{P}$.

CASE 1 ($S+2 \geq \log_2(p/q)/4$): As explained above, $\log_a(1/\beta) \leq 2$. By Property A and the fact that $\mathcal{P}$ is open, $C \geq m$. Thus,

$$6CS \geq 2C(S+2) \geq m \log_a(1/\beta) \log_2(p/q)/4.$$

CASE 2 ($S + 2 < \log_2(p/q)/4$): Let $c = \min(C, \lfloor \frac{1}{2}\log_2(p/q) - S - 1 \rfloor) \geq \min(C, \lfloor 2S + 4 - S - 1 \rfloor) > 0$. Fix any pair (u, v) of nodes, u in the X-program and v in the Y-program of $\mathcal{P}$. Fix $\alpha \in \{0,1\}^{\leq 2c}$ such that α determines $k \leq K$ output values, that is, α contains the encoding of 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 3.1, and the fact that $\mathcal{P}$ correctly computes f,

$$\Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \mid (x,y) \in R^{\alpha}_{(u,v)}] \leq a\beta^k. \tag{1}$$

Call $R^{\alpha}_{(u,v)}$ *tiny* if and only if $\Pr_{\mathcal{D}}[(x,y) \in R^{\alpha}_{(u,v)}] < q$. Let $T_{(u,v)}$ be the set of α such that $R^{\alpha}_{(u,v)}$ is tiny. For fixed (u, v) and varying α, Lemma 3.1 says that the sets $R^{\alpha}_{(u,v)}$ are disjoint and their union contains all pairs (x, y) that reach (u, v). Let

$$W = \{\alpha \in \{0,1\}^{\leq 2c} \mid \alpha \text{ determines } k \text{ output values}\}.$$

For all (u, v) and all $k \leq K$,

$$\begin{aligned}
&\Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \ \wedge \ \gamma^c_{(u,v)}(x,y) \text{ determines } k \text{ output values }] \\
&= \sum_{\alpha \in W} \Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \ \wedge \ (x,y) \in R^{\alpha}_{(u,v)}] \\
&= \sum_{\alpha \in W \setminus T_{(u,v)}} \Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \ \wedge \ (x,y) \in R^{\alpha}_{(u,v)}] \\
&\quad + \sum_{\alpha \in T_{(u,v)}} \Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \ \wedge \ (x,y) \in R^{\alpha}_{(u,v)}] \\
&< \sum_{\alpha \in W \setminus T_{(u,v)}} \Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \ \wedge \ (x,y) \in R^{\alpha}_{(u,v)}] + |T_{(u,v)}|q \\
&\leq 2^{2c+1}q + \sum_{\alpha \in W \setminus T_{(u,v)}} \Pr_{\mathcal{D}}[(x,y) \text{ reaches } (u,v) \mid (x,y) \in R^{\alpha}_{(u,v)}] \\
&\qquad \times \Pr_{\mathcal{D}}[(x,y) \in R^{\alpha}_{(u,v)}] \\
&\leq 2^{2c+1}q + a\beta^k \sum_{\alpha \in W \setminus T_{(u,v)}} \Pr_{\mathcal{D}}[(x,y) \in R^{\alpha}_{(u,v)}] \\
&\leq a\beta^k + 2^{2c+1}q, \qquad (2)
\end{aligned}$$

the last two lines following from Inequality (1) and the fact that the sets $R^{\alpha}_{(u,v)}$ are disjoint, respectively.

Inequality (2) is used in two different ways. For the first, by applying it to the start nodes u and v of their respective branching programs, we have

$$C > \min(K, \left\lfloor \frac{1}{2}\log_2(p/q) - S - 1 \right\rfloor). \tag{3}$$

For assume to the contrary that $C \leq K$ and $C \leq \left\lfloor \frac{1}{2}\log_2(p/q) - S - 1 \right\rfloor$. Then by the definition of c, $c = C$. Now choose $k = m$. By Property A and the fact that $\mathcal{P}$ is open, $k = m \leq C \leq K$, so that Inequality (2) holds. By the definition of c, $2^{2c+1}q \leq 2^{\log_2(p/q)-2+1}q = p/2$, and so Inequality (2) and Property A together yield $p \leq a\beta^m + 2^{2c+1}q \leq a\beta^m + p/2$. That is, $p/2 \leq a\beta^m$ which contradicts the hypothesis $p > 2a\beta^m$.

Now let $k = \min(K, m/\lceil C/c \rceil - 1)$ in Inequality (2). Suppose $\mathcal{P}$ uses exactly C' bits of communication on input (x, y). Divide this sequence of C' bits into $\lceil C'/c \rceil$ segments, each consisting of c consecutive bits of communication (except possibly the last segment). If $f(x, y)$ has at least m output values, then (x, y) reaches some pair (u, v) of nodes such that $\gamma^c_{(u,v)}(x, y)$ determines at least $m/\lceil C'/c \rceil - 1 \geq m/\lceil C/c \rceil - 1 \geq k$ output values, for otherwise fewer than m output values are produced by $\mathcal{P}$ on input (x, y). (The -1 term accounts for the possibility that an output crosses a segment boundary.) Therefore, since $k \leq K$ and there are at most 2^{2S} node pairs (u, v),

$$\begin{aligned} p &\leq \Pr_{\mathcal{D}}[(\exists u, v)((x, y) \text{ reaches } (u, v) \wedge \\ &\qquad \gamma^c_{(u,v)}(x, y) \text{ determines } k \text{ output values})] \\ &\leq 2^{2S}(a\beta^k + 2^{2c+1}q) = 2^{2S}a\beta^k + 2^{2S}2^{2c+1}q \leq 2^{2S}a\beta^k + p/2, \end{aligned}$$

by the definition of c. Solving this yields $2S + \log_2(2a/p) \geq k\log_2(1/\beta)$. Since $p \geq a2^{-S+1}$,

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

Case 2.1 ($K < m/\lceil C/c \rceil - 1$): Then $k = K$. By Property A and the fact that $\mathcal{P}$ is open, $C \geq m$, so $3CS \geq mK\log_2(1/\beta) \geq mK\log_a(1/\beta)$.

Case 2.2 ($K \geq m/\lceil C/c \rceil - 1$): Then $k + 1 = m/\lceil C/c \rceil \geq mc/(C + c)$, so

$$\begin{aligned} 12CS &\geq 6(C + c)S \\ &\geq 2(C + c)k\log_2(1/\beta) \\ &\geq (C + c)(k + 1)\log_2(1/\beta) \\ &\geq mc\log_2(1/\beta) \\ &\geq m\log_2(1/\beta)\min(C, \left\lfloor \frac{1}{2}\log_2(p/q) - S - 1 \right\rfloor) \\ &\geq m\log_2(1/\beta)\min(K, \left\lfloor \frac{1}{2}\log_2(p/q) - S - 1 \right\rfloor) \quad (4) \\ &\geq m\log_2(1/\beta)\min(K, \log_2(p/q)/4) \quad (5) \\ &\geq m\log_a(1/\beta)\min(K, \log_2(p/q)/4). \end{aligned}$$

Inequality (4) follows from Inequality (3), and Inequality (5) from the condition of Case 2. □

Theorem 3.3 extends Theorem 3.2 to the case when the pair of communicating branching programs is not necessarily open.

THEOREM 3.3. *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 $p > \max(a2^{-S+1}, 2a\beta^m)$. Then any pair $\mathcal{P}$ of communicating branching programs computing f using space S and communication C must satisfy*

$$(C + m \log m) \cdot S = \Omega(m \log_a(1/\beta) \min(K, \log(p/q))).$$

Proof. For any pair $\mathcal{P}$ of communicating branching programs, let the *observable behavior* of $\mathcal{P}$ on input (x, y) be the sequence of communicated bits and up to the first m output values produced on (x, y) by $\mathcal{P}$. Because of the structure of branching program pairs there is no ambiguity about the order in which communication steps occur. However, the interleaving of the two programs' output values between communication steps is not determined, so for definiteness we assume that, between communication steps, outputs in the observable behavior produced by the X-program precede those produced by the Y-program. Since each output value is produced only once by the pair of branching programs, it is easy to see that, for any input pair, the observable behavior may be encoded using $O(C + m \log m)$ bits, where each output value is encoded using $O(\log m)$ bits that specify (1) that it is an output rather than a communication, (2) which output bit is being produced, (3) its value, and (4) which program produced it. Let $2C^*$ be the maximum, over all (x, y), of the length of the observable behavior on input (x, y). (We use $2C^*$ as the analogue of the $2C$ bits used in Theorem 3.2 to encode the communication in the strings $\gamma^c_{(u,v)}(x, y)$.)

The proof of Theorem 3.3 is analogous to that of Theorem 3.2, except that the communication C is replaced by the number of bits needed to describe the observable behavior, C^*. The main technical difference is in the definition of $\gamma^c_{(u,v)}(x, y)$, which, instead of being the string of length $2c$ describing the next c bits of communication on (x, y) starting at (u, v), is now the string of the next $2c$ bits of the observable behavior of $\mathcal{P}$ on (x, y) starting at (u, v). Since outputs are included in the observable behavior, the string $\gamma^c_{(u,v)}(x, y)$ determines output values just as the communication did in the case of an open pair of branching programs. The crucial fact, which is easily verified, is that the new $R^\alpha_{(u,v)}$ based on this definition of $\gamma^c_{(u,v)}(x, y)$ still are disjoint rectangles that cover the set of input pairs that arrive at (u, v).

The remainder of the proof is identical to that of Theorem 3.2 except for a couple of points. The part of the proof of Theorem 3.2 where the communication is divided into segments of length c is slightly different. When c bits of communication is replaced by $2c$ bits of observable behavior, it is no longer obvious that every boundary between segments of the computation on (x, y) can be chosen to be at a pair of nodes (u, v) since an output value may be produced along an edge and this is $O(\log m)$ bits of observable behavior as opposed to the single bit of communication that may occur on an edge. However, because the argument ignores any output whose production overlaps the boundary, if a boundary would naturally fall in the middle of an edge, then the boundary may be shifted past the edge to the subsequent node with no loss in the argument. The remaining difference is that, in the places where Property A and openness were used to show that $m \leq C$, in the modified proof $m \leq C^*$ follows directly from Property A and the definition of C^*. The conclusion of the argument is exactly the same with C^* replacing C as required. □

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 [1]).

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.

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. For any $x, x' \in X$ with $x \neq x'$, and any $z, z' \in Z$, the events $h(x) = z$ and $h(x') = z'$ are independent.

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 $f : X \times H \rightarrow Z$ 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 4.1 (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 $E \subseteq Z$. Then*

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

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 4.2. *Let $\mathcal{P}$ be an open pair of communicating branching programs computing a pairwise universal family of hash functions from X to Z using communication C and space S. Let $n = \lfloor \log_2 |X| \rfloor$, $m = \lfloor \log_2 |Z| \rfloor - 1$, and $Z \subseteq \{0,1\}^{\leq m+l}$, where $l < \min(\log_2 n, m) - 3$ is included to allow some slack in the output encoding. Then $C \cdot S = \Omega(nm/l)$.*

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 $E \subseteq Z$ be the set of vectors consistent with V. At most $2^{m+l-k+1}$ vectors are in E so that $|E|/|Z| \leq 2^{-(k-l)}$. Then Lemma 4.1 states that

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

Thus Property B is satisfied with $q = 2^{-n/2}$, $\beta = 1/2$, $a = 2^{l+1}$, and $K = n/2$. Since $l < \min(\log_2 n, m) - 3$, $p = 1/2 > 2a/n$ and $p > 2a\beta^m$ so Theorem 3.2 implies that $C \cdot S = \Omega(nm/l)$. □

Theorem 4.3 extends Theorem 4.2 to the case in which the pair of communicating branching programs is not necessarily open.

THEOREM 4.3. *Any pair of communicating branching programs computing a pairwise universal family of hash functions from X to Z with communication C and space $S = o(n/(l \log m))$ satisfies $C \cdot S = \Omega(nm/l)$, 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. Using Theorem 3.3 in place of Theorem 3.2 in the proof of Theorem 4.2,

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

Since, by hypothesis, $Sm \log m = o(nm/l)$, the conclusion $CS = \Omega(nm/l)$ follows. □

Similar statements to Theorem 4.3 can be made for each of the following corollaries. We simply state our results for open pairs of branching programs for convenience.

In the following corollaries, we can always choose l to be a constant.

COROLLARY 4.4. *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 4.5. *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 4.2 exactly as shown by Mansour, Nisan, and Tiwari [17] for time-space tradeoffs.

COROLLARY 4.6. *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 4.7. *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 4.5, 4.6, and 4.7 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 3.2, the following analog of Theorem 4.2 for hash functions whose domain and range are vectors over $GF(r)$.

THEOREM 4.8. *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 $C \cdot S = \Omega(nm(\log r)/l)$, where $n = \lfloor \log_r |X| \rfloor$, $m = \lfloor \log_r |Z| \rfloor - 1$, and $Z \subseteq (GF(r))^{\leq m+l}$, for $l < \min(\log_r n, m) - 3$.*

This theorem has corollaries analogous to those of Theorem 4.2 such as the following.

COROLLARY 4.9. *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 [1] 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 programs 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 programs 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 [24, 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 shown by Yao [24, 26] and extended by Babai, Frankl, and Simon [4], a lower bound $D_\epsilon \geq k$ can be obtained by showing that, for an appropriate distribution on $X \times Y$ under which f takes on each value at least a constant fraction of the time, 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, the important difference being 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. We are particularly grateful to the anonymous referees for their very conscientious and constructive reviews.

REFERENCES

[1] 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, pp. 412–419.

[2] K. Abrahamson, *Time–space tradeoffs for algebraic problems on general sequential models*, Journal of Computer and System Sciences, 43 (1991), pp. 269–289.

[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, Boston, MA, Apr. 1983, pp. 133–139.

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

[5] L. Babai, N. Nisan, and M. Szegedy, *Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs*, Journal of Computer and System Sciences, 45 (1992), pp. 204–232.

[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, Seattle, WA, May 1989, pp. 197–203.

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

[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 (1981), pp. 351–364.

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

[10] ———, *New hash functions and their use in authentication and set equality*, Journal of Computer and System Sciences, 22 (1981), pp. 265–277.

[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, vol. 60, Steklov Mathematical Institute, Leningrad Department, 1976, pp. 38–48. 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, Chicago, IL, May 1988, pp. 186–191.

[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, Chicago, IL, May 1988, pp. 539–550.

[15] T. W. Lam, P. Tiwari, and M. Tompa, *Trade-offs between communication and space*, Journal of Computer and System Sciences, 45 (1992), pp. 296–315.

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

[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, San Francisco, CA, May 1982, pp. 330–337.

[19] R. Raz and A. Wigderson, *Monotone circuits for matching require linear depth*, Journal of the ACM, 39 (1992), pp. 736–744.

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

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

[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, Chicago, IL, May 1988, pp. 223–228.

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

1979, pp. 209–213.

[25] ———, *The entropic limitations of VLSI computations*, in Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, Milwaukee, WI, May 1981, pp. 308–311.

[26] ———, *Lower bounds by probabilistic arguments*, in 24th Annual Symposium on Foundations of Computer Science, Tucson, AZ, Nov. 1983, IEEE, pp. 420–428.

[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 (1984), pp. 183–197.