COMMUNICATION COMPLEXITY: A SURVEY

Laszlo Lovasz

CS-TR-204-89

February 1989

COMMUNICATION COMPLEXITY: A SURVEY

László Lovász

Department of Computer Science,
Eötvös Loránd University,
Budapest, Hungary H-1088

and

Princeton University,
Princeton, NJ 08544, USA

* Research supported by the joint research project of the Hungarian Academy of Sciences and the Deutsche Forschungsgemeinschaft, and by the Hungarian National Research Fund grant No. 1812.

0. Introduction

Complexity is one of the crucial scientific phenomena of our times. We are far from understanding all its aspects; but the first elements of a comprehensive theory do begin to emerge. To define a measure of complexity, we consider a certain (computing) task, and ask: what is the minimum amount of certain resources that is needed to carry out his task? The resources one considers depend on the device we are using and on other circumstances. The most common complexity measures are time and space, but many others can be considered.

The increasing importance of distributed computing, networking, VLSI and the use of computers in telecommunication have pointed out the significance of **communication** as a resource. In many devices, communication is significantly slower and costlier than local computation, and it is the real bottleneck in solving certain problems. One may mention the obvious example of a rocket approaching Jupiter or Halley's comet: local computation on the Earth can use the most powerful devices, and even the rocket can be equipped with very reasonable computers; but the communication is extremely slow and unreliable (which means that to make it more reliable we have to introduce more redundancy and thereby sacrifice even more time). If we go to more everyday examples it becomes less and less justified to concentrate solely on the problems of communication, but (as we shall see) even within a single chip, communication between various parts may be an important task which has to be solved efficiently in non-trivial ways.

Communication complexity also plays an important role in theoretical studies: many of the known "lower-bound" results in complexity theory are obtained by analyzing the communication between various parts of the input and output. This is perhaps only a temporary phenomenon, since such techniques do not seem to be powerful enough to lead to much-hoped-for results like P$\neq$NP; but it does indicate that separating just this one factor contributing to the complexity of various tasks can lead important information about the whole task.

In the following sections we give some examples from the theory of VLSI that lead to problems concering communication; then give a survey of the theory of communication complexity. I hope also to "communicate" my feeling that this lovely theory brings together a surprising number of ideas from classical and modern mathematics, and it also illustrates in a rather clear way basic notions from complexity theory like non-determinism and randomization.

1. Time–area tradeoff in VLSI: a simple example

Suppose that we want to design a "chip" that checks whether two strings of n bits, arriving simultaneously on $2n$ wires, are the same. More exactly, by a "chip" we mean a rectangular grid graph, with processors sitting in some of the nodes, and edge-disjoint paths (called "wires") connecting some of these processors. On the upper edge of the array, we have $2n$ specified nodes called "input ports", and somewhere else on the boundary we have one more specified node called "output port". We feed n bits $x_1,\dots,x_n,y_1,\dots,y_n$ into the input ports, and we want the gadget to produce a single bit at the output port, which is 1 iff $x_i = y_i$ for all i. We want the "chip" to be fast and small.

We make the following assumptions. The "chip" is *systolic*, i.e., it works in distinct steps. At each step, each processor reads the bits sent to it, computes a bit for some of the wires starting from it, and sends it to the processor at the other end of the wire. It will be convenient to assume that it has a memory of a constant number of bits, but it would not make any essential difference to exclude this.

Our simple task can be solved by a chip with the simplest possible topology: a single path of length $2n$ (Figure 1). Let $p_1,\dots,p_{2n}$ be the nodes of this path, each containing a processor. These receive, in order, bits $x_1,\dots,x_n,y_1,\dots,y_n$. Let each node except p_{n+1} "sleep" at the beginning. In the first step, p_{n+1} sends bit y_1 to the left and an "alarm" bit to the right. This wakes up p_{n+2}, who sends y_2 to the left and an "alarm" bit to the right. Each processor receiving a bit from the right will transmit it to the left in the next step. So the bits $y_1,\dots,y_n$ will march to the left with one space left between any two.

Now when p_1 receives y_1, it compares it with x_1. He sends 1 to the right if they are equal and a 0 otherwise. Processor p_2 receives this bit simultaneously with y_2; he compares y_2 with x_2 and sends a 1 to the right if he received a 1 and also $x_2 = y_2$; else, he sends a 0. This goes on like this until p_n receives a bit from the right (which is 1 if and only if $x_1 = y_1,\dots,x_{n-1} = y_{n-1}$) and simultaneously the bit y_n from the left. He compares x_n and y_n, and produces the output bit.

It is clear that for the simplicity of the topology we had to pay with time: this shifting back and forth took $2n$ steps. It is perhaps more natural to allow more space and solve the problem in about $O(\log n)$ steps, using the chip in Figure 2. In the first step, processors $q_1,\dots,q_n$ make the comparisons $x_1 = y_1,\dots,x_n = y_n$;

their findings are collected by a binary tree in $\log n$ further steps (all log's are base 2).

But we had to make the chip much bigger: it has $2n^2$ nodes; or, if we define the "area" of the chip as the total length of wires, it has area larger than n^2. We could do a little better: combining the ideas from the two designs shown above one obtains a chip working in $O(\log n)$ time, with area $O(n^2/\log n)$. This construction is left to the reader as an exercise.

Now this factor of $\log n$ is not too much gain; could we do better? Let us prove that we cannot. More exactly, we prove the following "area–time tradeoff" theorem, which says that to reduce the area, we have to sacrifice time. This theorem is a special case of the results of Thompson (1979):

1.1 Theorem. *If a chip decides the equality of two 0-1 sequences of length n (in the sense described above), has area A, and works in time T, then*

$$AT \geq n^2.$$

(Note that equality can hold, up to a constant, with $A = O(n)$ and $T = O(n)$, and also with $A = O(n^2/\log n)$ and $T = O(\log n)$, and in fact for every pair of values inbetween satisfying the inequality.)

Proof. Let us draw a vertical line cutting the chip into two, so that one half contains the ports $x_1, \ldots, x_n$ and the other, the ports $y_1, \ldots, y_n$. Let us forget about what happens inside the two parts, and concentrate on the communication across this line. "Obviously" (we'll come back to this!) at least n bits of information must cross this line. Since we only have T steps, there must be a step when simultaneously at least n/T bits cross the line. But one wire can transmit in one step only one bit of information, so there must be at least n/T wires that cross the line.

Now if we shift the line to the right by one position, x_1 and y_1 may be on the same side, so the amount of information sent across the line may be less. But we can argue that the computation still has to decide whether $x_1 \ldots x_{n-1} = y_1 \ldots y_{n-1}$ and so by the same argument, the shifted line must cross at least $(n-1)/T$ wires. Similarly, the line shifted by 2 positions must cross at least $(n-2)/T$ wires etc.

If we add up these numbers, we get exactly the total length of horizontal pieces of the wires, which, in turn, is a trivial lower bound on the area. Hence

$$A \geq \frac{n}{T} + 2\frac{n-1}{T} + 2\frac{n-2}{T} + \ldots = \frac{N^2}{T}.$$

∎

We have used in the proof that at least n bits have to cross a line separating the x-ports from the y-ports. Note that this assertion is now independent of VLSI: the two parts of the chip may be viewed as two processors, each getting a 0-1 sequence of length n, and they have to decide whether these are equal. We want to show that they have to communicate at least n bits.

This seems natural, and it is even true, but an exact proof is not quite obvious. To warn the reader, let us mention that if they are allowed to use a random number generator, and they want the answer only with a certainty of 0.9999999999, then it suffices to communicate $O(\log n)$ bits (see section 6).

In the following sections we develop the theory of communication complexity, which will allow us to prove these kinds of lower bounds. This example has served to expose the problem and we are not going to treat the interesting and important topic of area–time tradeoffs in VLSI in detail. However, we shall return briefly to VLSI problems in section 8, to see which related communication problems are raised by them.

2. The notion of communication complexity, deterministic and non-deterministic

We consider a rather simple model of communication. We have two processors and a direct link between them. Each processor knows some information, called the *input* of that processor; and they want to compute a value which is a function of both inputs. For simplicity, we shall assume (except for one case is section 8) that this value to be computed is a single bit; we call this the *output bit* of the communication problem.

We assume that both processors have unlimited computing power and that local computation is free. However, we are charged for every single bit transmitted from one processor to the other. Our aim is to find methods to solve such problems with a minimum number of bits transmitted.

When talking about this model, it is nicer to imagine that the two processors are two teenagers, Alice and Bob. Alice lives in Budapest, Bob lives in New Zealand. To appreciate the importance of minimizing the length of communication between them, imagine that we (the parents) have to pay the telephone bill for their communication.

The communication analogue to the fundamental notion of an *algorithm* in "ordinary" complexity theory is the notion of a *protocol.* Informally, a *communication protocol* is a set of rules specifying the order and "meaning" of messages sent. So the protocol prescibes who is to send the first bit; depending on the input of that processor, what this bit should be; depending on this bit, who is to send the second bit, and depending on the input of that processor and on the first bit sent, what this second bit should be etc. The protocol terminates when one processor knows the output bit and the other one knows this about the first one. This definition seems strange, but it turns out more convenient technically than the obvious rule that the protocol ends when both know the result. There is no real difference between the two: if Alice knows the result, and Bob knows this about her, then she only needs to send one more bit. Conversely, in every protocol which ends with both of them knowing the answer, the situation before the last bit (which is sent, say, by Alice) is that Alice must know the answer and the Bob must know that she knows it.

The *complexity* of a protocol is the number of bits communicated in the worst case.

There is always a ***trivial protocol***: Alice can send her input to Bob. Then Bob has all the information he needs to compute the answer, and Alice knows this about him. Of course, they can switch roles if this is better. (We shall see that sometimes there is no better protocol than this trivial one).

Let us formalize this notion (following YAO 1979). Let $a_1, \dots, a_n$ be the possible inputs of Alice and $b_1, \dots, b_m$, the possible inputs of Bob. (Note that since local computation is free, we don't have to worry about how these are encoded. Two inputs for Alice are clearly equivalent if they give the same results with any input for Bob. So two such inputs can be identified from our point of view.) Let c_{ij} be the value they want to determine for inputs a_i and b_j. The 0-1 matrix $C = (c_{ij})_{i=1}^{n}{}_{j=1}^{m}$ determines the communication problem; we call this the *communication matrix* associated with the problem. Both players know the matrix C; Alice knows the index i of a row, Bob know the index j of a column, and they want to determine the entry c_ij. The trivial protocol (in which, say, Alice sends the index of her row to Bob) takes $\lceil \log_2 n \rceil$ bits (of course if $m < n$ then the reverse trivial protocol is better).

A protocol has a very simple interpretation in terms of this matrix. First, it determines who sends the first bit; say, Alice does. This bit is determined by the input of Alice; in other words, the protocol partitions the rows of the matrix C into two classes, and the first bit of Alice tells Bob which of the two classes contains her row. From now on, the game is limited to the submatrix C_1 formed by the rows in this class. Next, the protocol describes a partition of the rows or columns of C_1 into two classes (depending on who is to send the second bit), and the second bit itself specifies which of these two classes contains the line (row or column) of the sender. This limits the game to a submatrix C_2 etc.

If the game ends after k bits then the remaining submatrix C_k is the union of an all-1 submatrix and an all-0 submatrix. (We shall call this an almost homogeneous matrix.) In fact, if (say) Alice knows the answer then her row in C_k must be all-0 or all-1, and since Bob knows for sure that Alice knows the answer, this must be true for every row of C_k.

So the determination of the communication complexity is reduced to the following combinatorial problem: given a 0-1 matrix C, in how many rounds can we partition it into almost-homogeneous submatrices, if in each row we can split each of the current submatrices into two (either horizontally or vertically). We shall denote this number by $\kappa(C)$.

Note that in each step, the maximum rank of the submatrices is decreased by a factor of 2 or less, and hence we obtain the following inequality due to MEHLHORN and SCHMIDT (1982):

2.1 Lemma *If C has rank r then $\kappa(C) \geq \log r$.* ∎

In particular, we obtain

2.2 Corollary *If C has full row rank then the trivial protocol is optimal.* ∎

Here is a simple communication problem to which the previous corollary applies directly. Suppose that Alice and Bob want to solve the following important problem: is the recent edition of the Encyclopædia Britannica that Alice has identical — letter for letter — with that of Bob? An obvious solution to this problem is that one of them reads reads the Britannica to the other over the phone. While this may be a pleasure under the circumstances, clearly the parents object. But clearly they cannot stop an important project like this; can they come up with a protocol that is more efficient, i.e., that reduces the telephone bill*?

To formalize this problem, let us assume that we know the size of both editions — say, n bits. Then there are 2^n possible inputs for Alice and the same number of possible inputs for Bob, and so C is a $2^n \times 2^n$ identity matrix. This matrix has rank 2^n, so it follows from Corollary 2.2 that for this problem there is no protocol better than the trivial one: transmitting the whole input of one of the players. This also completes the proof of Theorem 1.1. (We shall return to this question in the next sections; among others, we shall see that randomization is extremely helpful.)

Let us also point out that this example shows that the matrix C can be of enormous size, and typically only implicitly given — the problem specification tells us how to compute any given element. In particular, the rank of C is not always an easily computable parameter (even though it was in this last example).

A slightly more complicated communication problem is ***Disjointness:*** each player has a subset of a specified set S, and they have to decide if their inputs are disjoint. In other words, they both know a 0-1 vector of length n, and they want to know whether the inner product is 0 or not. Various versions of this problem will play an important role later in this paper, but now we can settle the communication complexity of the basic version easily. In fact, if we arrange the subsets appropriately, the matrix associated with this problem will be lower triangular with 1's in its main diagonal, and so non-singular. It follows that *the trivial protocol is optimal for the Disjointness problem.*

Let us mention one more useful example, where the computation of the rank is not so immediate. Let S be a set with n elements, and assume that Alice has a subset $X \subseteq S$ and Bob has a subset $Y \subseteq S$. They want to know the parity of

* This problem is not quite as artificial as it sounds. In the example of communication with a spacecraft, it may be important to check (say, before a new operation is started) whether the program of the computer on board is still correct. This may require the comparison of strings longer than the Britannica — and with much more costly communication.

$|X \cap Y|$. In other words, they both know a 0-1 vector of length n, and they want to compute the inner product modulo 2.

If we write up the corresponding matrix C, it will be symmetric and will have a row (and column) of 0-s: the row corresponding to $\emptyset$. Let us drop this row and column; we claim that the remaining matrix C' is non-singular. To see this, compute $(C')^2$: it turns out to have 2^{n-1} in tis main diagonal and 2^{n-2} elsewhere. It is elementary linear algebra that this matrix is non-singular.

Hence $\mathrm{rk}(C) = \mathrm{rk}(C') = 2^n - 1$, and so (at least if $n \geq 2$)

$$\kappa(C) \geq \lceil \log_2 \mathrm{rk}(C) \rceil = n.$$

So we obtain that ***for computing the inner product modulo 2, the trivial protocol is optimal.***

It is interesting to remark that the rank over GF(2), which would perhaps seem more natural to use in this problem, would give a very poor bound here: the GF(2) rank of C is only n.

The rank of C (not its logarithm!) bounds $\kappa(C)$ from above. It is easy to see that a 0-1 matrix with rank r has at most 2^r distinct rows. Since repeated rows of C can clearly be suppressed without changing the communication complexity of the problem, we obtain

2.4 Proposition. *If C has rank r then $\kappa(C) \leq r$.*

In this Proposition, one could take the GF(2) rank as well, and our previous remark shows that then the bound is sometimes sharp. For the real rank, however, I don't know of any communication problem for which $\kappa(C)$ would be anywhere near the bound r. In particular it appears to be unsettled whether $\kappa(C)$ can be bounded by any polynomial of $\log r$.

In the complexity theory of algorithms, non-deterministic algorithms play a crucial role; they specify important complexity classes like NP. Analogously, non-deterministic protocols play an important role in the complexity theory of communication. They were introduced by Lipton and Sedgewick (1981).

Assume that there is an extra-terrestrial super-being, called E.T., who can monitor the inputs and communication of Alice and Bob, and of course knows the answer to the problem right away. He can also make any announcement on the phoneline. Can he speed up communication by giving the right announcements (hints)? For example, if he gets bored with Bob's voice reading the Encyclopædia to Alice, can he shortcut the communication with an appropriate short announcement from which both Alice and Bob can be certain what the answer is?

In this case, the answer depends on whether the outcome is that the two editions are equal or that they are different. If they are different, E.T. can simply announce 'the 37th character in the 14th line of page 568 of volume 24 is "," in Alice's edition but ";" in Bob's.' At the cost of these few bits, the problem is settled. On the other hand, if the two editions are identical, then even the E.T. cannot save a single bit of communication! This will become clear as soon as we formalize the notion of a non-deterministic protocol.

Given a communication problem (in the form of a matrix C), there will be two non-deterministic protocols, one for "1" and one for "0". A non-deterministic protocol for (say) "1" has to specify a set P of possible ***proofs***. Further, it has to include a rule for Alice that tells her which proofs she should accept depending on her input, and a similar rule for Bob. For this non-deterministic protocol to be correct, we require

$(*)$ for a given pair of inputs the answer to the communication problem is "1" if and only if there exists a proof $p \in P$ which is accepted by both Alice and Bob.

Without loss of generality we may assume that P consists of 0-1 strings, and then the maximum length of these strings is the ***complexity*** of the protocol. Of course, with appropriate encoding, this will be $\lceil \log |P| \rceil$. We are interested in finding the protocol for which this complexity is minimum; this value is denoted by $\kappa_1(C)$. A non-deterministic protocol for "0" and $\kappa_0(C)$ are defined analogously.

Consider a non-deterministic protocol for "1", a particular proof $p \in P$, and the entries of C (i.e. inputs for Alice and Bob) for which this proof "works". By $(*)$, these must be all 1's; also, from the fact that Alice and Bob have to verify the proof independently, we see that these 1's must form a submatrix. So a non-deterministic protocol corresponds to a covering of C by all-1 submatrices. Conversely, every such covering gives a non-deterministic protocol: E.T. can simply announce the name of the submatrix containing the given entry. So we obtain:

2.5 Lemma: *The non-deterministic communication complexity $\kappa_1(C)$ of a matrix C is the least natural number t such that the 1's in C can be covered by at most 2^t all-1 submatrices.*

Note that the all-1 submatrices in the covering do not have to be disjoint. Therefore, there is no immediate relation between the non-deterministic communication and the rank of C.

Yannakakis (1988) introduces two further concepts that further illuminate the connection between rank and communication complexity (deterministic and non-deterministic). He defines the ***unambigous communication complexity*** $\overline{\kappa}_1(C)$ of C as the least natural number a for which the 1's in C can be covered by 2^a disjoint all-1 submatrices; equivalently, it is the minimum complexity of a non- deterministic protocol in which the "proof" in $(*)$ is unique. Obviously,

$$\kappa_1(C) \le \overline{\kappa}_1(C) \le \kappa(C),$$

and

$$\log \mathrm{rk}(C) \le \overline{\kappa}_1(C).$$

It is easy to derive analogous results for the complexity of the answer "0".

There is also a way to formulate, in some sense, an upper bound on the complexity in terms of a certain "rank". Let C be an $n \times m$ 0-1 matrix. Yannakakis

(1988) defines the ***positive rank*** $\mathrm{rk}_+(C)$ of C as the least p for which there exists a non-negative $n \times p$ matrix A and a non-negative $p \times m$ matrix B such that $C = AB$. (Note that without the non-negativity restriction, this would define just the ordinary rank.) Trivially

$$\mathrm{rk}(C) \le \mathrm{rk}_+(C).$$

It is also easy to show that

$$\kappa_1(C) \le \log \mathrm{rk}_+(C).$$

In fact, If $C = AB$ where A and B are non-negative matrices with p columns and rows, respectively, then let a_i denote the ith column of A and b_i, the ith row of B. Then we have

$$C = \sum_{i=1}^{p} a_i b_i^T.$$

Let Z_i be the support of the matrix $a_i b_i^T$, then clearly Z_i is an all-1 submatrix of C and the Z_i's must cover all the 1's in C. Hence $\kappa_1(C) \le \log p$ as claimed. Finally, we also have

$$\log \mathrm{rk}_+(C) \le \kappa(C),$$

which follows by a similar argument as for the rank. Unfortunately, the positive rank of the matrix does not seem to be easy to handle (in particular, no polynomial-time algorithm is known to compute it).

Let us mention one further, rather trivial bound on the non-deterministic complexity. Assume that C has a 1's but every all-1 submatrix of C has at most b entries. The trivially $\kappa_1(C) \ge \log a - \log b$. While I do not know any good applications of this observation, a version of it plays an important role in obtaining lower bounds for randomized protocols (see section 6).

Returning to the Encyclopædia Britannica example, it is obvious that the 1's in the the $N \times N$ identity matrix I_N cannot be covered by fewer that N all-1 submatrices. Hence

$$\kappa_1(I_N) = \lceil \log N \rceil.$$

For our Encyclopædia Britannica example this says that even the E.T. cannot prove that the two editions are equal, any cheaper than reading the whole text!

3. A general upper bound on communication complexity

We have seen three lower bounds on the communication complexity of a matrix:

$$\log \mathrm{rk}(C) \le \kappa(C), \tag{3.1}$$

(3.2) $$\kappa_0(C) \le \kappa(C),$$

and

(3.3) $$\kappa_1(C) \le \kappa(C).$$

The identity matrix shows that it can happen that the (3.3) is very week: the right hand side can be exponentially large compared with the left. Interchanging the roles of 1 and 0, we obtain that the bound in (3.2) can also be very far from $\kappa(C)$. We do not know such a bad example for (3.1), but it is likely that the situation is similar. Note that (3.1) is also essentially symmetric in 0's and 1's: if we interchange the 0's and 1's in a C, the rank changes by at most 1.

However, it is a surprising fact that the product of any two of these three lower bounds is an upper bound on the communication complexity. The first part of the following theorem is due to Aho, Ullman and Yannakakis (1983), the second (and third) to Lovász and Saks (1988b):

3.4 Theorem. *For every matrix C,*

(a) $\kappa(C) \le (\kappa_0(C)+1)(\kappa_1(C)+1)$*;*

(b) $\kappa(C) \le \log \mathrm{rk}(C)(\kappa_0(C)+1)$*;*

(c) $\kappa(C) \le \log \mathrm{rk}(\overline{C})(\kappa_1(C)+1)$.

It is clear that part (b) of this theorem implies part (c). We shall give the proof of a result that is stronger than either one of (a) or (b). Let $\rho_1(C)$ denote the size of the largest square submatrix of C such that (ordering the rows and columns appropriately) every diagonal entry is 1 but every entry above the diagonal is 0. We can define $\rho_0(C)$ analogously.

It is clear that
$$\rho_1(C) \le \mathrm{rk}(C)$$
and
$$\rho_0(C) \le \mathrm{rk}(\bar{C}) \le \mathrm{rk}(C)+1.$$
Furthermore,
$$\rho_1(C) \le 2^{\kappa_1(C)},$$
since in every covering of C with all-1 blocks, the diagonal entries in the submatrix in the definition of $\rho_1(C)$ must belong to different blocks. Hence the following theorem implies all three parts of Theorem 3.4:

3.5 Theorem. *For every matrix C,*

$$\kappa(C) \le (\log \rho_1(C))(\kappa_0(C)+1).$$

Proof. We use induction on $\rho_1(C)$. Let $k = \kappa_0(C)$; then the 0-s of C can be covered by all-0 submatrices $C_1, \ldots, C_N$ where $N \leq 2^k$. Let A_i [B_i] denote the submatrix formed by those rows [columns] of C that meet C_i. Then observe that

$$\rho_1(A_i) + \rho_1(B_i) \leq \rho_1(C).$$

We may assume that for $i = 1, \ldots, M$, we have $\rho_1(A_i) \leq \rho_1(C)/2$ but for $i = M+1, \ldots, N$, we have $\rho_1(B_i) \leq \rho_1(C)/2$. We may also assume that $M \geq N/2$.

Now we can describe the following protocol.

First, Alice looks at her row to see if it intersects any of the submatrices $C_1, \ldots, C_M$. If so, she sends a "1" and the name of such a submatrix. If not, she sends a "0".

If Bob recieves a "0", he looks at his column to see if it intersects any of the submatrices $C_{m+1}, \ldots, C_N$. If so, he send a "1" and the name of such a submatrix. If not, he send a "0".

This describes one round of the protocol. This round may end in three ways:

Case 1. Alice found an appropriate submatrix. Then they both know that Alice's row belongs to A_i $(1 \leq i \leq M)$. Since $\rho_1(A_i) \leq \rho_1(C)/2$, by recurrence they can find the answer by communicating at most

$$\log(\rho_1(A_i))(\kappa_0(A_i)+1) \leq (\log(\rho_1(C)) - 1)(\kappa_0(C)+1)$$

bits. Since Alice's message took $1 + \lceil \log M \rceil \leq 1 + \kappa_0(C)$ bits, this is altogether at most $(\log \rho_1(C))(\kappa_0(C)+1)$ bits.

Case 2. Bob found an appropriate submatrix. This is similar to Case 1. Then they both know that Bob's column belongs to B_i $(M = 1 \leq i \leq N)$, and so by recurrence they can find the answer by communicating at most $(\log(\rho_1(C)) - 1)(\kappa_0(C)+1)$ bits. The count of bits for the round itself is slightly different: 1 for Alice's message and $1 + \lceil \log(M-N) \rceil \leq \kappa_0(C)$ for Bob's message. The final count is the same.

Case 3. Both Alice and Bob failed to find an appropriate submatrix. Then the intersection of their lines cannot belong to any C_i, and so it must be a "1". So they have found the answer. ∎

As a further corollary of this theorem we mention the following result of Yannakakis (1988), bounding the deterministic complexity in terms of the unambigous (but non-deterministic) complexity:

3.5 Corollary: $\kappa(C) \leq \overline{\kappa}_1(C)^2$.

Similarly, we obtain a relation between the rank, positive rank and communication complexity of the matrix:

3.6 Corollary: $\kappa(C) \leq (\log \mathrm{rk}(C))(\log \mathrm{rk}_+(C))$.

Theorem 3.4(a) has an interesting interpretation. Define a *communication problem* as a class $\mathcal{A}$ of 0-1 matrices; for simplicity, assume that they are square matrices. The communication complexity of any $N \times N$ matrix is at most $\log N$. We say that $\mathcal{H}$ is in $\mathrm{P_{comm}}$ if it can be solved substantially better: if there exists a constant $c > 0$ such that $\kappa(C) \leq (\log\log N)^c$ for each matrix $C \in \mathcal{H}$, where N is the dimension of C. Similarly, we say that $\mathcal{H}$ is in $\mathrm{NP_{comm}}$, if there exists a constant $c > 0$ such that for each $C \in \mathcal{H}$, $\kappa_1(C) \leq (\log\log N)^c$. We can define co-$\mathrm{NP_{comm}}$ analogously. Just as for the analogous computational complexity classes, we have the trivial containment

$$\mathrm{P_{comm}} \subseteq \mathrm{NP_{comm}} \cap \text{co-}\mathrm{NP_{comm}}.$$

However, for the communication complexity classes we also have the following, rather interesting facts:

$$\mathrm{P_{comm}} \neq \mathrm{NP_{comm}},$$

$$\mathrm{NP_{comm}} \neq \text{co-}\mathrm{NP_{comm}}$$

(both follow from the example of checking identity), but

$$\mathrm{P_{comm}} = \mathrm{NP_{comm}} \cap \text{co-}\mathrm{NP_{comm}}$$

(by the theorem of Aho, Ullman and Yannakakis 4.3(a)). This idea was developed by Babai, Frankl and Simon (1986), who defined and studied communication analogues of many other well-known complexity classes like #P, PSPACE, BPP etc.

4. Some protocols

We have seen that the Disjointness problem, i.e., the problem of deciding whether two given sets are disjoint, is trivial from the communication point of view: the trivial protocol (Alice sends her input to Bob) is optimal. Often, however, the sets the players have are restricted in one way or the other. We shall discuss some general types of restrictions in the next section; here we analyse two special examples. In both examples the rank lower bound is very far from the complexity of the trivial protocol; and in both cases it will turn out that this rank bound is quite close to the truth. In particular, we shall describe some examples of non-trivial communication protocols.

4.1 Problem. Let T be a tree and let Alice be given a subtree T_A and Bob, a subtree T_B of T. Their task is to decide whether T_A and T_B have a node in common.

Suppose that T has n nodes. The number of subtrees of T can be between $\binom{n}{2} + n + 1$ (if T is a path) and $2^{n-1} + n$ (if T is a star). Typically, however, it is exponential in n; for example, if T has no nodes of degree 2 then it has more than $2^{n/2}$ subtrees.

Let us write up the matrix C_T corresponding to this communication problem (rows and columns indexed by subtrees, with a 1 in a position if and only if the corresponding subtrees are disjoint). The size of C_T can vary, but it is typically exponential. It is perhaps more surprising that the rank of C_T is determined by n, and it is in fact very low:

$$\mathrm{rk}(C_T) = 2n.$$

In the next section, we shall see the general reason behind this lemma; at this moment, we leave it to the reader as an exercise.

The non-deterministic complexity of our problem is easy to find. To exhibit that the two subtrees intersect, E.T. can announce a common node. This takes $\lceil \log n \rceil$ bits. To exhibit that they do not intersect, E.T. can either announce that one or the other is empty, or he can announce an edge uv such that if we delete uv from T, then in the remaining forest, T_A is in the component containing u but T_B is in the component containing v. So there are $2n$ possible announcements of E.T., and — in appropriate encoding — this takes $\lceil \log n \rceil + 1$ bits. It is not difficult to see that these are optimal non-deterministic protocols, i.e.

$$\kappa_1(C_T) = \lceil \log n \rceil + 1, \qquad \kappa_0(C_T) = \lceil \log n \rceil.$$

From this, we obtain by the general results in the previous section the following bounds on the communication complexity of Problem 4.1:

$$\lceil \log n \rceil + 1 \le \kappa(C_T) \le (\lceil \log n \rceil + 1)^2.$$

We show that the lower bound is near the truth (Lovász and Saks 1988a).

4.2 Theorem. $\kappa(C_T) \le \log n + \log\log n + 1$.

Proof. First, we describe a protocol that gives a somewhat worse result: about $2\log n$ as an upper bound. The protocol consists of two parts:

1°: Alice selects any node $x \in V(T_A)$, and sends its name to Bob (we have to reserve one message to indicate if her subtree is empty; but in this case they are done).

2°: Bob determines the (unique) node $y \in V(T_B)$ nearest to x, and sends this to Alice (we have to reserve one message to indicate if his tree is empty).

Now it is easy to argue that the two subtrees T_A and T_B are node-disjoint if and only if $y \notin V(T_A)$. So at this point Alice knows the answer (and Bob knows that she knows).

As it stands, this protocol involves the communication of $2\log(n+1)$ bits (the names of two nodes). To improve it, we use the following lemma:

4.3 Lemma. *The nodes of T can be labelled by $1, 2, \ldots, n$ so that if we delete the nodes labelled $1, \ldots, k-1$, every connected component of the remaining forest* *has no more than $2n/k$ nodes.* ∎

The players agree on such a labelling in advence (it is part of the protocol). Now we replace the first step by

1°°: Alice selects the node $x \in V(T_A)$ with the least label, and sends its label to Bob (she sends 0 is her subtree is empty; in this case, they are done).

Suppose the label of x is k. Receiving this, Bob will know that Alice's tree does not contain the nodes labelled $1, \ldots, k-1$, so he can delete these from the tree. Each of the components of the remaining forest has at most $2n/k$ nodes, and Bob also knows which of these contains T_A, since he knows a node of this tree. So from now on they are restricted to a tree T_0 with at most $2n/k$ nodes, and he can carry out the second step more economically as follows:

2°°: If $V(T_B) \cap V(T_0) \neq \emptyset$, then Bob determines the (unique) node $y \in V(T_B) \cap V(T_0)$ nearest to x, he counts the number l of nodes in T_0 with label not larger than the label of y. He sends to Alice the number l. If $V(T_B) \cap V(T_0) = \emptyset$, he sends 0.

Since this protocol differs from the previous one only in the encoding, its correctness is clear. The number k has $\log k$ bits; the number l is at most $2n/k$, so it has at most $\log(2n/k) = \log n + 1 - \log k$ bits. This is altogether $\log n + 1$ bits.

There is one catch: while sending the bits of k, Alice has to indicate that she is finished. (In other word, she has to use a prefix-free encoding of the integers up to n.) One solution is to start with $\log\log n$ further bits, announcing the length of k. This gives the bound in the theorem. (It can be shown that no encoding trick can get rid of these extra bits.) ∎

Our second example is again a special disjointness problem. It was formulated by Yannakakis (1988) in connection with a complexity problem concerning the vertex packing polytope (see section 7).

4.4 Problem. Let G be a graph and let Alice be given a set A of independent nodes, and let Bob be given a clique B. Their task is to decide whether A and B have a node in common.

Let G have n nodes. The number of independent sets/cliques in G can be exponentially large. The matrix C_G associated with the problem has a row for each independent set, a column for each clique, and (say) a "1" at a position where the corresponding clique and independent set intersect (this is now conversely to the convention in the previous problem, but here this will be the more convenient). The rank of this matrix is very low:

$$\mathrm{rk}(C_G) = n.$$

(This is now quite easy to see: this matrix is the product of the independent set–node incidence matrix and the node–clique incidence matrix.)

The non-deterministic communication complexity for non-disjointness is again $\lceil \log n \rceil$. So we have

$$\kappa_1(C_G) = \log n$$

and hence Theorem 3.4 implies the following bound, which was first proved by more direct means by Yannakakis (1988):

$$\lceil \log n \rceil \leq \kappa(C_G) \leq (\lceil \log n \rceil + 1)^2.$$

We show a protocol due to A. Hajnal that reduces the upper bound by a factor of 2:

4.5 Theorem. $\kappa(C_G) \leq \frac{1}{2}(\log n + 1)^2$.

Proof. We describe the protocol. First, Alice checks if her independent set A contains a node that has degree at least $n/2$. If it does, then she sends the name of such a node v to Bob. Now they both know that Alice's set is contained among the nodes non-adjacent to v (inculding v), and so the problem is reduced to one on a graph with at most $n/2$ nodes.

If Alice does not find such a node, then Bob looks for a node in his clique B to see if B contains a node with degree less than $n/2$. If it does, he sends the name of such a node u to Alice. Then they both know that Bob's set is contained among the nodes u and those adjacent to u, and so the problem is again reduced to one on a graph with at most $n/2$ nodes.

If neither one is succesful, they know that every node of Alice's set has degree less than $n/2$ while every node of Bob's set has degree larger than $n/2$, and hence they know that the two sets are disjoint.

It is easy to estimate the complexity of this protocol and get the bound as given. ∎

Note that in this case the nondeterministic communication complexity for disjointness is not easy to find. A non-deterministic protocol for disjointness corresponds to a family of subsets $\mathcal{H}$ of subsets of V(G) such that for each pair (A, B) where A is an independent set, B is a clique, and $A \cap B = \emptyset$, there exists an $H \in \mathcal{H}$ such that $A \subseteq H \subseteq V(G) - B$. We call such a system $\mathcal{H}$ a *separating family.* The size of the smallest family is not known; in particular, we don't know if a separating family of polynomial size exists for each graph. As a corollary to Theorem 4.5, the non-deterministic communication complexity of disjointness is at most $\frac{1}{2}(\log n)^2$. Hence we obtain the following, purely graph-theoretic result:

4.6 Corollary. *Every graph on n vertices contains has a separating family of size* $n^{(\log n)/2}$. ∎

5. Möbius functions and the rank of the communication matrix

Our previous discussions have shown that the rank of the communications matrix is very intimately related to the communications complexity. They also show, however, that it is in general not an easy task to compute this rank. In this section we are going to study a rather general situation in which this rank can be computed, or at least, reduced to the study of a well-known function in combinatorics.

To motivate our abstractions, let us discuss one further communication problem in graph theory.

5.1 Problem. Let V be a finite set and assume that both Alice and Bob are given a graph with node set V. Their task is to decide whether the union of these graphs is connected.

Hajnal, Maass and Turán (1988) proved that for this problem, the trivial protocol is optimal. One should notice that the trivial protocol is not that Alice transmits the whole graph; this is redundant information for Bob. All he needs to know is which pairs of nodes can connected in Alice's graph. In other words, he needs the partition of V defined by the connected components of Alice's graph. Since the total number of partitions is the Bell number B_n, this trivial protocol takes $\lceil \log_2 B_n \rceil \approx n \log_2 n$ bits.

To prove that this trivial protocol is optimal, Hajnal, Maass and Turan use the rank bound, but to compute the rank of the corresponding communications matrix is by no means obvious. Their method involves the use of the "Möbius function" of the partition lattice. Lovász and Saks (1988a) showed that this method of computing the rank extends to a large class of problems, which can be formulated as follows. Let S be a finite set. A *filter* on S is a non-empty family of subsets of S such that $U \in \mathcal{F}$, $U \subseteq V \subseteq S$ implies $V \in \mathcal{F}$.

5.2 Problem. Let S be a finite set and $\mathcal{F}$, any filter in 2^S. Let Alice be given a set $X \subseteq \mathcal{F}$ and Bob, a set $Y \in \mathcal{F}$. Their task is to decide whether $X \cup Y \in \mathcal{F}$.

Of course, one can formulate the "dual" problem, in which $\mathcal{F}$ is an ideal and Alice and Bob have to determine whether $X \cap Y \in \mathcal{F}$. This is trivially equivalent to 4.2 under complementation. To get Problem 4.1, we take all the edges of the complete graph on V as elements of S, and let $\mathcal{F}$ consist of all connected graphs with this set of vertices.

Let us recall a problem from the previous section.

5.3 Problem. Let T be a tree and let both Alice and Bob be given a subtree of T. Their task is to decide whether these subtrees have a node in common.

We have seen that in this case the trivial protocol is by far not optimal. Let us formulate a generalization of this problem. Let S be a finite set. An *alignment* on S is a collection $\mathcal{A}$ of subsets of S closed under intersection.

5.4 Problem. Let S be a finite set and $\mathcal{A}$, an alignment on S. Let Alice be given a set $X \in \mathcal{A}$ and Bob, a set $Y \in \mathcal{A}$. Their task is to decide whether $X \cap Y = \emptyset$.

Both of these general problems, and several other natural problems of this kind, are equivalent to the following, which sounds perhaps more special but will be easier to handle.

5.5 The Meet Problem. Let $\mathcal{L}$ be a finite lattice. Let Alice be given an element $x \in \mathcal{L}$ and Bob, an element $y \in \mathcal{L}$. Their task is to decide if $x \wedge y = 0$ (here 0 is the zero element of the lattice).

Before showing how the other two problems can be reduced to this, we have to define what we mean by reduction. Fortunately, this is much easier here than in "machine-based" complexity theory. Given an instance of any communication problem, i.e., a 0-1 matrix C, check if there are two equal rows or columns in C. Clearly, one of such a pair of rows can be deleted without changing the complexity (deterministic or non-deterministic) of the problem. Carry out this until every pair of rows and every pair of columns will be different. The remaining matrix will be called the *core* of C.

Now we say that a class $\mathcal{A}$ of 0-1 matrices *can be reduced* to a class $\mathcal{B}$ of 0-1 matrices, if for every $A \in \mathcal{A}$ there exists a $B \in \mathcal{B}$ such that the core of A is the same (up to permutation of rows and columns) as the core of B. Two classes of matrices are *equivalent* if they can be reduced to each other. Lovász and Saks (1988a) proved:

5.6 Lemma. *Problems 5.2, 5.4 and 5.5 are equivalent.*

Proof. Most of the reductions needed to show this are straightforward. We sketch the (perhaps) least trivial fact that 5.2 can be reduced to 5.5. Let S be a finite set and $\mathcal{F}$, a filter on S. Define a relation $\sim$ on 2^S by saying that $X \sim Y$ iff for every $Z \subseteq S$, we have $X \cup Z \in \mathcal{F}$ if and only if $Y \cup Z \in \mathcal{F}$ (i.e., if the rows of the communication matrix corresponding to X and Y are identical). Trivially, this is an equivalence relation. It is equally trivial, but somewhat unexpected, that the following holds:

Claim 1. If $X \sim Y$ then $X \sim X \cup Y$.

This claim implies that the union $\overline{X} = \cup\{Y : Y \sim X\}$ also satisfies $\overline{X} \sim X$. Now it is not difficult to check that

Claim 2. The operation $X \rightarrow \overline{X}$ is a closure operator on 2^S. Furthermore, $X \in \mathcal{F}$ if and only if $\overline{X} = S$.

As usual, call a set X *closed* if $\overline{X} = X$. Then it follows easily that

Claim 3. The closed sets form a co-atomic lattice $\mathcal{L}$.

(A lattice is co-atomic if every element of it is the meet if co-atoms, i.e., elements covered by the top element. This property will, however, play no role in the sequel.)

Now the core of the communication matrix is just the submatrix formed by rows and columns corresponding to closed sets. The problem is equivalent to the modified problem where both players are given a *closed* set and they have to decide if the closure of the union (i.e., the join in the lattice $\mathcal{L}$) is the "top" element S. This is just the same problem as 5.5, but "upside down". ∎

So from now on we shall restrict our attention to the Meet Problem. The corresponding matrix $C = (c_{ij})$ has its rows and columns indexed by the elements of the lattice $\mathcal{L}$, and for $x, y \in \mathcal{L}$ we have

$$c_{xy} = \begin{cases} 1, & \text{if } x \wedge y = 0, \\ 0, & \text{otherwise. /cr} \end{cases}$$

Now this matrix is well-known in algebraic combinatorics! To formulate its main property important for us, we have to introduce a few further matrices associated with the lattice. Let

$$\zeta_{xy} = \begin{cases} 1, & \text{if } x \leq y, \\ 0, & \text{otherwise. /cr} \end{cases}$$

and $Z = (\zeta_{xy})$. This matrix Z is sometimes called the *zeta-matrix* of the lattice. If we order the rows and columns of Z compatibly with the partial ordering defined by the lattice, it will be upper triangular with 1's in its main diagonal. Hence it is invertible, and its inverse $M = Z^{-1}$ is and integral matrix of the same shape. This inverse is a very important matrix, called the *Möbius matrix* of the lattice. Let

$$M = (\mu(x, y))_{x,y\in\mathcal{L}}.$$

The function μ is called the *Möbius function* of the lattice. From the discussion above we see that $\mu(x, x) = 1$ for all $x \in \mathcal{L}$, and $\mu(x, y) = 0$ for all $x, y \in \mathcal{L}$ such that $x \not\leq y$. Moreover, the definition of M implies that for every pair of elements $a \leq b$ of the lattice,

$$\sum_{a\leq x\leq b} \mu(a, x) = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{otherwise;} \end{cases}$$

and

$$\sum_{a\leq x\leq b} \mu(x, b) = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{otherwise.} \end{cases}$$

These identities provide a recursive procedure to compute the Möbius function. It is easy to see from this procedure that the value of the Möbius function $\mu(x, y)$, where $x \leq y$, depends only on the internal structure of the interval $[x, y]$. Also note the symmetry in these two identities. This implies that if μ^* denotes the Möbius function of the lattice turned upside down, then

$$\mu^*(x,y) = \mu(y,x).$$

We cannot survey here the many properties and applications of the Möbius function. We shall restrict ourselves to those issues relevant for determining the rank of C. For more see Rota (1964), Lovász (1979), Chapter 2, or Stanley (1986), Chapter 3.

Let us introduce one further matrix: D is a diagonal matrix defined by $(D)_{xx} = \mu(0,x)$. Now it is not difficult to verify the following identity (Wilf 1968):

5.7 Lemma. $C = Z^T DZ$. ∎

Since Z is invertible, and the rank of the diagonal matrix D is the number of non-zeros in its diagonal, we obtain

5.8 Corollary. *The rank of C is the number of elements $x \in \mathcal{L}$ such that $\mu(0,x) \neq 0$. In particular, if $\mu(0,x) \neq 0$ for all $x \in \mathcal{L}$ then the trivial protocol is optimal for the Meet Problem.* ∎

We shall call an element x of the lattice $\mathcal{L}$ satisfying $\mu(0,x) \neq 0$ a *Möbius element.* To be able to use this corollary, we have to find the Möbius elements. Unfortunately, the Möbius function can be quite complicated, and there is no easy characterization of the Moebius elements. Let us collect a few facts.

It is easy to see that if x is an atom in $\mathcal{L}$ then $\mu(0,x) = 1$, and so every atom is a Möbius element. On the other hand, we have (Ph. Hall 1936):

5.9 Theorem. *Every Möbius element is a join of atoms.* ∎

Unfortunately, the converse is not true, an element that is a join of atoms is not necessarily Möbius. One important condition on the non-vanishing of the Möbius function is the following (Rota 1964). Recall that a lattice is *geometric* if it is semimodular and atomic. Geometric lattices are just lattices of flats of matroids.

5.10 Theorem. *If $\mathcal{L}$ is geometric and $x, y \in \mathcal{L}$, $x \leq y$, then $\mu(x,y) \neq 0$. In particular, every element is Möbius.*

From the theorem, we obtain the result of Hajnal, Maass and Turán immediately.

5.11 Corollary. *The communication complexity of problem 5.1 is at least $n \log n$.*

Proof. The lattice associated with Problem 5.1 by the general construction is just the partition lattice turned upside down. Now the partition lattice is geometric, and so the Möbius function does not vanish on any interval of it. But then it does not vanish on any interval of the "upside down" lattice either, and hence the trivial protocal for the communication problem is optimal. ∎

There are other classes of lattices on which the Möbius function does not vanish. For such a lattice, the trivial protocol is optimal for the disjointness problem. One interesting class to mention here are face lattices of convex polytopes.

In our other starting example, we consider the lattice $\mathcal{L}_T$ of subtrees of a tree T. Using Theorem 5.8 (upside down!), it is easy to compute the Möbius function of this lattice:

5.12 Lemma. *For the Moebius function of $\mathcal{L}_T$,*

$$\mu(0,x) = \begin{cases} 1, & \textit{if } x = \emptyset \textit{ or } x \textit{ is a subtree with 2 nodes,} \\ -1, & \textit{if } x \textit{ is a subtree with a single node,} \\ 0, & \textit{otherwise.} \end{cases}$$

∎

(This way of determining the Möbius function extends to a large class of alignments called *antimatroids*. See Lovász and Saks 1988a.)

From this lemma we see that the rank of C is $2n$. We have seen in the previous section that the bound for this problem is very close to being optimal.

Now let us return to the general Meet Problem, and discuss the non-deterministic complexity of the problem. If E.T. wants to exhibit that $x \wedge y \neq 0$, it suffices to exhibit an atom of $\mathcal{L}$ that lies below both x and y. Hence

$$\kappa_1(C) \leq \log_2(\text{number of atoms}).$$

(It is easy to see that in fact equality holds here.) Hence by Lemma 2.1 and Theorem 3.5, we have the following bounds on the communication complexity of the Meet Problem:

5.13 Theorem. *Let C be the communication matrix associated with the Meet Problem for a lattice $\mathcal{L}$. Let $\mathcal{L}$ have a atoms and b Möbius elements. Then*

$$\log_2 b \leq \kappa(C) \leq (\log_2 a)(\log_2 b).$$

∎

Since every atom is a Möbius element, and $b = \text{rk}(C)$, we obtain:

5.14 Corollary: *For the Meet Problem,*

$$\log_2 \operatorname{rk}(C) \leq \kappa(C) \leq (\log_2 \operatorname{rk}(C))^2.$$

∎

6. Randomized protocols

So far, our protocols have not used randomization. If we do, many of the previously discussed problems become substantially easier. In a *randomized protocol* we allow the players to throw dice, and we only want to result be correct with probability at least 2/3. The smallest k for which such a protocol exists involving the transmission of at most k bits will be denoted by $\kappa^{\text{rand}}(C)$.

We could require in place of 2/3 any fixed number greater than 1/2: repeating the protocol a constant number of times, the probability of getting the correct answer could be pushed arbitrarily close to 1. (If we want the probability of the correct answer only to be greater than 1/2, but allow it to tend to 1/2, we get a different kind of problem. We do not discuss this here, but refer to Alon, Frankl and Rödl (1985) and Babai, Frankl and Simon 1986. For a thorough study of randomized protocols, see also Yao 1983.)

Let us discuss the Encyclopædia Britannica example. We have seen that the best protocol to decide whether two strings of length n are equal is the trivial protocol and requires the communication of n bits. In contrast, we have the following theorem (Yao 1983):

6.1 Theorem. *There exists a randomized protocol to decide the equality of two strings of length n, using $O(\log n)$ bits.*

Proof: The protocol can be viewed as an extension of "binary check". Consider the inputs as two natural numbers x and y, $0 \leq x, y \leq 2^{n-1}$. Alice selects a prime $p \leq n^2$, computes the remainder x' of x modulo p, and then sends the pair (x', p) to Bob. Now Bob computes the remainder y' of y modulo p, and compares it with x'. If they are distinct, he concludes that $x \neq y$. If they are the same, he concludes that $x = y$.

If the two numbers x and y are equal then, of course, so are x' and y' and so the protocol reaches the right conclusion. If x and y are different then, however, it could happen that x' and y' are the same and the protocol reaches the wrong conclusion. This happens if $p | x - y$. Now $|x - y| < 2^n$ and so $x - y$ has fewer than n different prime divisors. On the other hand, Alice had about $n/\log n$ primes to choose from, and so the probability that she chose one of the divisors of $x - y$ tends to 0.

Clearly, this protocol uses at most $4\log n$ bits. ∎

In our treatment of deterministic protocols, the general Disjointness problem and the Inner product modulo 2 behaved very similarly to the Encyclopædia problem; we just had to replace the identity matrix by some other matrix with obviously high rank. But for randomized protocols, they are substantially more difficult. Before stating this exactly, let us formulate some combinatorial conditions and a general linear algebraic lower bound on the randomized complexity. For simplicity of presentation, we restrict ourselves to $n \times n$ matrices.

Consider a matrix C and any randomized protocol to solve the associated communication problem using at most k bits. For the purposes of lower bounds, we assume that the random number generator is public, i.e., the random bits are available to both players (this only makes the life of Alice and Bob easier, and so our task to give a lower bound more difficult). We may also assume that these random bits are available right at the beginning. Now the players look at this sequence and depending on it, they select a (deterministic) protocol that they follow. This protocol ends up with a partition of the matrix into *terminal submatrices.* This time it will be more convenient to assume that the protocol stops when both players know the answer (both have to have the same answer, which, of course, may be wrong). This means that some of the terminal submatrices are labelled "1" and others are labelled "0", meaning that if for a particular input the protocol ends up with this submatrix, then this is concluding value. Since in this case the conclusion is not necessarily correct, the submatrices labelled "1" may contain 0's and vice versa. (We shall see though that in some average sense, terminal submatrices labelled with "1" must have more 1's than 0's.)

For each submatrix T, let $p(T)$ denote the probability of the event that in the randomly chosen protocol, T is a terminal submatrix labelled "1". Our assumption on the probability of error implies that

$$\sum_{\substack{T \\ ij\in T}} p(T) \begin{cases} \ge 2/3, & \text{if } c_{ij} = 1, \\ \le 1/3, & \text{if } c_{ij} = 0. \end{cases} \tag{6.1}$$

Moreover, every protocol produces at most 2^k terminal submatrices, and hence

$$\sum_T p(T) \le 2^k.$$

We may consider the linear program

$$\text{minimize} \sum_T p(T)$$

subject to (6.1) and the obvious constraints

$$p(T) \ge 0. \tag{6.2}$$

If $\mu(C)$ denotes the optimum value of this program, then

$$k \geq \log \mu(C).$$

We call this the ***linear programming bound*** on the randomized communication complexity of the matrix. We can apply linear programming duality and obtain $\mu(C)$ as the maximum of the dual program. This program has variables ϕ_{ij} associated with the entries of the matrix, constraints

$$\phi_{ij} \geq 0, \quad \text{if } c_{ij} = 1, \tag{6.3}$$

$$\phi_{ij} \leq 0, \quad \text{if } c_{ij} = 0, \tag{6.4}$$

$$\sum_{ij \in T} \phi_{ij} \leq 1 \quad \text{for each submatrix } T, \tag{6.5}$$

and objective function

$$\frac{2}{3} \sum_{c_{ij}=1} \phi_{ij} + \frac{1}{3} \sum_{c_{ij}=0} \phi_{ij}. \tag{6.6}$$

So $\mu(C)$ could also be defined as the minimum of (6.6) subject to (6.3), (6.4) and (6.5). In particular, any feasible solution of (6.3)–(6.5) provides a lower bound on the randomized communication complexity. Of course, if we want the probability of error be at most p, then we can replace (6.6) by the objective function

$$(1-p) \sum_{c_{ij}=1} \phi_{ij} + p \sum_{c_{ij}=0} \phi_{ij}.$$

Note that a feasible solution of (6.3)–(6.5) can be viewed as a matrix Φ. Condition (6.5), which is the most awkward, and contains exponentially many contraints, can be replaced by a somewhat stronger condition using some linear algebra. Let T be any submatrix. Let $a \in \mathbb{R}^n$ be the incidence vector of the set of rows in T and $b \in \mathbb{R}^n$, the incidence vector of the set of columns of T. Then the sum of entries in T is just $a^T \Phi b$, and so (6.5) can be rephrased as

$$a^T \Phi b \leq 1$$

whenever a and b are 0-1 vectors. By elementary linear algebra, the left hand side is at most $\|a\| \cdot \|b\| \cdot \|\Phi\|$, where $\|\Phi\|$ is the spectral norm of the matrix Φ (recall that this can be expressed as $\Lambda(\Phi^T \Phi)^{1/2}$, where $\Lambda(M)$ denotes the largest eigenvalue of the matrix M). So if we require that

$$\|\Phi\| \leq \frac{1}{n}, \tag{6.7}$$

then (6.5) is automatically satisfied. Moreover, we can express the objective function as

$$\frac{1}{2}\sum_{i,j}\phi_{ij} - \frac{1}{6}\sum_{i,j}|\phi_{ij}|.$$

Since the first term is at most 1/2 by (6.7), we can disregard it. The second term involves another norm of the matrix Φ:

$$|\Phi| = \sum_{i,j}|\phi_{ij}|.$$

We can formulate our result as follows:

6.2 Lemma. *For every non-zero matrix Φ satisfying (6.3) and (6.4), we have*

$$\kappa^{\mathrm{rand}}(C) \geq \log\left(\frac{|\Phi|}{n\|\Phi\|}\right) - 3.$$

∎

In particular, we can use here the matrix $2C - J$. Clearly $|2C - J| = n^2$, and hence we obtain:

6.3 Corollary. $\kappa^{\mathrm{rand}}(C) \geq \log\left(\frac{n}{\|2C-J\|}\right) - 3.$ ∎

As an application, consider the Inner product modulo 2 problem. For this, $2C - J$ is an Hadamard matrix (a ± 1 matrix with any two columns orthogonal) and hence its spectral norm is $\sqrt{n}$. This implies the following result of Chor and Goldreich (1985): *the randomized communication complexity of computing the inner product of two vectors of length m modulo 2 is $\Omega(m)$.*

Note that if Corollary 6.3 gives any valuable result then the matrix C has to be very homogeneous: $\|2C - J\|$ must be $o(n)$ and hence it follows that each submatrix has to contain almost exactly as many 1's as 0's. But a somewhat weaker condition also works, as shown by Yao (1983):

6.4 Lemma. *Assume that for some $a, b, c > 0$,*

(6.8)

for every submatrix T with $t > an^2$ entries, at least bt entries in T are 0;

(6.9)

at least cn^2 entries of C are 1.

Then the randomized communication complexity of the problem is $\Omega\left(\frac{\log(c/a)}{|\log(bc)|}\right)$.

Proof. If there is a randomized protocol using k bits with error probability $1/3$, then repeting this $O(|\log(bc)|)$ times, we get one with error probability less than $bc/3$. To estimate the complexity of this from below, we construct a feasible solution of (6.3)–(6.5) with objective value $\log(c/a)$; this will prove the lemma. Let

$$\Phi_{ij} = \begin{cases} \frac{1}{an^2}, & \text{if } c_{ij} = 1, \\ \frac{1-b}{abn^2}, & \text{if } c_{ij} = 1. \end{cases}$$

Then a simple computation shows that we have the feasible solution as desired. ∎

As an application of this lemma, Babai, Frankl and Simon (1986) show that *the randomized communication complexity of the Disjointness problem for the subsets of an n-element set is $\Omega(\sqrt{n})$*. They restrict the problem to subsets of cardinality $\sqrt{n}$. Then the matrix associated with this restricted problem satisfies hypotheses (i) and (ii) with $a = \text{const}/\sqrt{n}, b, c = \text{const}$. The proof is combinatorial and not discussed here.

7. VLSI again: How to split information?

Consider the "chip" in Figure 3, which solves the Equality Problem in a very simple way: It compares x_1 with y_1, etc. and then uses a binary tree to collect this information. It works in $O(\log n)$ time and has area $O(n \log n)$. Doesn't this contradict our Theorem 1.1?

The trick is of course that the input is coming now in a different arrangement. In general, if we want a chip to compute a Boolean function $f(x_1, \ldots, x_{2n})$ then we can either prescribe where the input bits come in or (depending on the situation) we may choose this ourselves. In this second case (as the above example shows) we can have quite different results. The communication complexity bound works, but we have to use a more involved measure. We can split the variables of f into two sets of n elements, and consider the ordinary communication complexity of the problem of evaluating f if Alice gets one half of the variables and Bob, the other half. This complexity will depend on the partition. We define the *worst-partition communication complexity* of f as the maximum of this complexity. The *best-partition communication complexity* is defined as the minimum over all partitions of the variables into two sets of equal size (we have to add this restriction to avoid putting all variables into one class).

Now the best-partition communication complexity can replace the ordinary communication complexity in lower bounds if we are free to choose the positions of the ports (even inside the chip). (The worst-partition complexity has other uses.) Unfortunately, it is in general more difficult to compute the best-partition complexity. We shall restrict ourselves to an example.

Hajnal, Maass and Turán (1988) settle the following question. Alice and Bob want to decide if a given graph G is connected. Half of the pairs of nodes are supervised by Alice; she knows if these pairs are adjacent in G or not. The other half is similarly supervised by Bob. For a given partition of the edges, this is an ordinary communication problem. We want to find a lower bound valid for all partitions.

Another way to look at this problem is that Alice knows a graph G_A on a given set V of nodes, Bob knows another graph G_B, and they want to decide whether the union is connected. This is now an ordinary communication complexity problem, similar to problem 5.2. What makes different (easier for Alice and Bob, but more difficult for us) is that they both know that Alice's graph must consist of red edges and Bob's graph must consist of blue edges. Now Hajnal, Maass and Turán show that still essentially the same lower bound holds as for a pair of unrestricted graphs:

7.1 Theorem. *The best-partition communication complexity of determining if a graph is connected is $\Omega(n \log n)$.*

Proof. The proof goes by reduction to Corollary 5.11, but this reduction is quite involved, so we only sketch it. Using the powerful Regularity Lemma of Szemerédi (1976) one can prove the following:

7.2 Lemma. *Let $G_1 \cup G_2$ be a partition of the complete graph on the n-element node set V into two subgraphs with $\frac{1}{2}\binom{n}{2}$ edges. Then there exists a set $W \subseteq V$ with $|W| = \Omega(n)$ with the following property: We can associate with each partition P of V two subgraphs $G_1^P \subseteq G_1$ and $G_2^P \subseteq G_2$ such that for every two partitions P and Q of V, the union $G_1^P \cup G_2^Q$ is connected if and only if $P \cup Q$ is the trivial partition of V.* ∎

Consider the matrix C of the communication problem. This lemma says that this contains the matrix C' of the meet problem for the "upside down" partition lattice of W as a submatrix. So $\kappa(C) \geq \kappa(C') \geq \Omega(n \log n)$. ∎

8. Communication complexity and computational complexity

We discuss some examples where communication complexity seems to have close relationship with certain other complexity issues. The first of these is the work of Yannakakis (1988) about expressing combinatorial optimization problems as linear programs.

a. Most combinatorial optimization problems (Matching, Travelling Salesman, various scheduling problems etc.) can be viewed as the task to find a member of a given set-system $\mathcal{F}$ with maximum value. Furthermore, most often the value is given by associating weights with the elements of the underlying set S, and letting the value of a member of $\mathcal{F}$ be the sum of weights of its elements.

One of the most succesfull approaches to such combinatorial optimization problems is to represent each set $A \in \mathcal{F}$ by its incidence vector $\chi^A \in \mathbb{R}^S$, and consider the convex hull P of these. Our task is then to optimize a linear objective function over P. If we are able to find the system of linear inequalities describing P, then we can apply powerful techniques of linear programming.

There is a basic difficulty with this method: it can easily happen that both the number of vertices and the number of facets of P is exponentially large (in the natural size of the problem, which is a power of the dimension). Hence the system of linear inequalities describing P will be exponentially large. There are ways out of this difficulty; essentially, in some cases one can generate the program as one proceeds with the algorithm (see Grötschel, Lovász and Schrijver 1988, Grötschel and Padberg 1985).

Another possibility is to reduce the size of the program by introducing new variables. Geometrically, this means to represent P as the projection of another polyhedron Q in higher dimension, but with fewer facets (hopefully only a polynomial number of them). Since projecting a polytope may increase the number of facets, this approach is promising. In fact, if we want to translate into a geometric language simple algorithms consisting of case distinctions and dynamic programming methods, we often end up with introducing new variables.

This leads us to the following, purely geometric question: given a polytope P in $\mathbb{R}^n$, can it be represented as the projection of a polytope Q with "few" (say, $O(n^{\mathbf{const}})$) facets? (Note that the dimension in which Q lives is less than the number of its facets.) If this is the case, and if in addition one can efficiently generate a linear description of Q, then we can optimize any linear objective function over P by lifting it to Q and using any polynomial time linear programming algorithm. This approach was tried (unsuccesfully) to solve the travelling salesman problem in polynomial time (Swart 1986). It could be expected that no polytope coming from an NP-hard optimization problem can be lifted like this. But even some "nice" polytopes (coming from polynomially solvable problems, like matching or vertex packing in perfect graphs) seem to resist attempts to obtain them as projections of polytopes with a polynomial number of facets.

Yannakakis made an important step in this question by showing that the matching and travelling salesman polytopes of complete graphs cannot be obtained as such projections, if we assume that the natural symmetries of these polytopes can be lifted to Q. He also sketched a possible route to prove such results without this symmetry assumption. This approach uses the following observation. Given a polytope $P \subset \mathbb{R}^n$, consider the following communication complexity problem: Alice is given a vertex v of P, Bob is given a facet F of P, and they have to decide whether $v \in F$. Let us call this the Vertex-Facet problem, and let $C_{VF}(P)$ denote the corresponding matrix (we write a 1 if the corresponding facet contains the corresponding vertex). Then

8.1 Lemma. *If P is the projection of a polytope Q with t facets then $\kappa_0(C_{VF(P)}) \leq \lceil \log t \rceil$.*

Proof. The vertex v is the projection of a vertex w of Q. The facet F is the projection of a face G (usually not to a facet) of Q. We can obtain G as the intersection of facets; at least one of these, say G', does not contain w. Now E.T. can announce the name of G' as a proof that $v \notin F$; this takes only $\lceil \log t \rceil$ bits.∎

As a special case, consider the ***vertex packing polytope***, i.e., the convex hull of incidence vectors of independent sets of nodes of a graph G. Every (maximal) clique of a graph G defines a facet of the vertex packing polytope. A vertex is not on a face if and only if the corresponding independent set is disjoint from the corresponding clique. Hence by our discussion in section 4 we obtain:

8.2 Corollary. *If a graph has no polynomial size independent set–clique separating family then its vertex packing polytope cannot be obtained as the projection of a polytope with a polynomial number of facets.*

Unfortunately, we do not know if there is any graph to which this corollary applies. On the other hand, as far as I know there could even be perfect graphs with this property.

b. The second application we discuss is a result of Karchmer and Wigderson (1988). Consider a Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$, and a Boolean circuit evaluating this function. This consists of a directed graph that has $2n$ sources and a single sink. The sources are labelled with the variables $x_1, \ldots, x_n$, and their negations. Moreover, every non-source node has indegree 2, and is labelled either by "AND" or by "OR". To operate the circuit, we feed in the actual values of the variables at the sources, and let each other node compute the conjunction or the disjuncton of the two logical values at the tails of the two incoming edges. The value produced at the sink is the value of the function. The *depth* of the circuit is the longest path from the source to the sink. To prove lower bounds on the depth of a Boolean circuit evaluating various given Boolean functions is one of the hottest topics in theoretical computer science.

With a Boolean function f, we can associate the following communication problem. Alice gets a 0-1 sequence $(a_1, \ldots, a_n)$ and Bob gets a 0-1 sequence $(b_1, \ldots, b_n)$, such that $f(a_1, \ldots, a_n) = 0$ and $f(b_1, \ldots, b_n) = 1$. Their task is to find an index i such that $a_i \neq b_i$. We call this the Difference Problem for f. (Note that this is not a decision problem.)

Karchmer and Wigderson show the following:

8.3 Lemma. *A Boolean function can be evaluated by a circuit of depth t if and only if the communication complexity of the Difference Problem for f is at most t.*

Proof. We show how to translate a Boolean circuit into a communication protocol; the reverse construction is similar. We suppose that both Alice and Bob know the Boolean circuit (this is part of the protocol). Then Alice can follow the evaluation of $f(a_1,\dots,a_n)$. This will associate a logical value $\alpha(v)$ with each node v. Similarly, Bob can evaluate $f(b_1,\dots,b_n)$ and thereby associate a logical value $\beta(v)$ with each node v.

During the protocol, Alice and Bob construct a path "backwards", starting from the sink and ending at one of the sources. The source where the path ends will give the index i that they are looking for. More generally, at each step, they will have a node w such that $\alpha(w) = 0$ and $\beta(w) = 1$. Let uw and $u'w$ be the two edges entering w. If this current node w is labelled "AND" then it is Alice's turn. At least one of $\alpha(u)$ and $\alpha(u')$ is 0; Alice sends a bit to Bob to indicate which one, and they move to that node. (Observe that since $\beta(w) = 1$, it follows that $\beta(u) = \beta(u') = 1$.) If w is labelled "OR" then it is Bob's turn and his message is determined analogously. ∎

There is an analogous result if we restrict ourselves to monotone Boolean nctions and monotone circuits computing them (i.e., circuits where only the unnegated variables occur at the sources). The corresponding communication problem is only slightly more difficult: Alice and Bob have to find an index i such that $a_i = 0$ and $b_i = 1$ (since the function is monotone, such an index must exist).

Using this transformation of the problem, Karchmer and Wigderson were able to show that every Boolean circuit determining whether a two points in a given graph G can be connected by a path must have depth at least $(\log n)^2$. More recently Karchmer, Newman and Wigderson (unpublished) gave interesting applications of this transformation in the reverse direction: they showed how to design Boolaen circuits with small depth for various problems using general results similar to those in section 3.

c. Hajnal, Maass and Turán (1988) find another connection of this kind. Fix n Boolean variables $x_1,\dots,x_n$. By a *test tree* we mean a binary tree in which every internal node v has an associated "test function", a Boolean function $g_v : \{0,1\}^n \to \{0,1\}$, and each leaf has value 0 or 1. Such a test tree can be used to compute a Boolean function in n variables: we start from the root and move down to a leave. At each internal node v, compute the test value $g_v(x_1,\dots,x_n)$. If this is 0, we move right, else we move left. When we arrive at a leaf, we read off the associated logical value. The *depth* of the tree is the longest path from the root to a leaf.

We are interested in finding the minimum depth of a test tree computing a given Boolean function f, where the test functions are restricted to some simple class (if they are restricted to single variables, we get the more usual *decision trees*. The following observation of Hajnal, Maass and Turán gives a lower bound on this depth:

8.4 Lemma. *If a test tree computes a Boolean function f with worst-partition communication complexity k, and every test-function has worst-partition communi-*

cation complexity l, then its depth is at least k/l.

One nice set of test functions are disjunctions of variables (i.e., we can test whether a given set of the variables contains a "1"). These have worst-partition communication complexity 2. The lemma implies, in conjunction with Theorem 7.1, that *with such tests one cannot determine the connectivity of a graph in depth less than $\Omega(n \log n)$.*

d. Finally, we sketch an interesting application of communication complexity to random number generation due to Babai, Nisan and Szegedy (1988). Assume that we have a random 0-1 sequence α of length m. We call this the "seed" We want to generate from it a 0-1 sequence β of length $n > m$, which is still "essentially random". By this we mean that it cannot be distinguished from any truly random 0-1 sequence of length n by any machine of some prescribed class; in this case, we consider on-line RAM machines with a sublinear ($O(n^{.99})$ bound on the memory. (For this machine, we may even allow to have a true random number generator; note that this machine does not want to generate random numbers; rather, it wants to show that our way of generating them is bad.) The machine is getting the bits of β one by one, and at any time it can occupy only $O(\log n)$ space. Having seen some bits from β, the machine has to compute a "guess" for the next bit. We say that the pseudorandom sequence β fails the test if the probability that this guess is correct is larger than 51%. Babai, Nisan and Szegedy give a construction of a pseudorandom sequence β with $n = m^{c \log m}$ that passes all logspace on-line tests.

Let p, q, r be positive integers with $pqr \leq m$, and consider a $p \times q$ array (v_{ij}), where each $v_{ij} \in \{0,1\}^r$. The bits in all the v_{ij} are truly random bits of the seed. Let $f(u_1, \ldots, u_q) : \{0,1\}^{r \times q} \to \{0,1\}$ be an appropriate function. For every choice of $1 \leq i_\nu \leq p$ ($\nu = 1, \ldots, q$), compute the bit $x[i_1, \ldots, i_q] = f(v_{i_1,1}, \ldots, v_{i_q,q})$. Order these bits lexicographically according to the index sequences, to get the sequence β of length $n = p^q$.

Let us show how the pseudorandomness of this sequence depends on communication complexity. Consider the moment when the machine has to guess the bit $x[i_1, \ldots, i_q]$. At this moment, it has already seen all bits coming from lexicographically smaller index sequence. We may split this sequence into q segments as follows: let β_1 be the beginning segment of β consisting of those bits $x[j_1, \ldots, j_q]$ with $j_1 < i_1$; generally, let let β_t $(1 \leq t \leq q)$ consist of those bits $x[j_1, \ldots, j_q]$ with $j_1 = i_1$, $j_2 = i_2$, $\ldots$, $j_{t-1} = i_{t-1}$ but $j_t < i_t$. Clearly, each β_t is a segment of β and together they make up the portion of β before $x[i_1, \ldots, i_q]$.

Now comes the key observation: the bits in β_t are independent of $v_{i_t,t}$. Let us invite q players and assign the tth player to "supervise" the segment β_t. This means that he can see all bits in β_t. If the machine can make a good guess of $x[i_1, \ldots, i_q]$ then these players can also compute this guess: the first player computes whatever the machine has in its memory when having processed β_1, and sends this to the second player; now he computes from this whatever the machine has in its memory after having processed β_2, etc. The last player will be able to compute the guess of the machine. If the machine uses at most M bits of memory then this protocol involves the communication of not more than Mq bits.

To obtain a lower bound, we make the life of the players even easier: we allow them to "broadcast" all messeges (so that all the other players also can learn them), and we tell the tth player the whole seed with the exception of the bits in $v_{i_t,t}$. So we have the following communication complexity problem. We have q players, and also q vectors $w_1, \ldots, w_q \in \{0,1\}^r$. The tth player knows all vectors except w_t. Their task is to "guess" $f(w_1, \ldots, w_q)$ at the expense of a minimum number of "broadcasted" bits, so that the probability of guessing correctly is at least 51%.

Babai, Nisan and Szegedy consider the function f that gives the parity of the number of positions where each w_t has a 1 (a generalization of inner product) and show by a rather involved extension of the methods in section 6 that this takes $\Omega(r2^{-q})$ bits. For the choice of $r = m^{.992}$, $q = .001 \log m$ and $p = m^{.007}$, this gives a pseudorandom sequence β of length $m^{.001 \log m}$, which passes all tests using at most $m^{.99}$ space. It is interesting to remark that this function is logspace computable, so the random number generator works very fast.

References:

A. V. Aho, J. D. Ullman and M. Yannakakis (1983), On notions of informations transfer in VLSI circuits, *Proc. 15th ACM STOC*, 133-139.

L. Babai, P. Frankl and J. Simon (1986), Complexity classes in communication complexity theory, *Proc. 27th IEEE FOCS*, 337-347.

L. Babai, N. Nisan and M. Szegedy (1988), Multiparty protocols and logspace-hard pseudorandom sequences, manuscript.

B. Chor and O. Goldreich, (1985), Unbiased bits from sources of weak randomness and probabilistic communication complexity, *Proc. 26th IEEE FOCS*, 429-442.

M. Grötschel, L. Lovász and A. Schrijver (1988), *Geometric Algorithms and Combinatorial Optimization*, Springer.

M Grötschel and M. W. Padberg (1985), Polyhedral computations, in: *The Travelling Salesman Problem* (ed. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan and D. Schmoys), Wiley, 307-360.

A. Hajnal, W. Maass and G. Turán (1988), On the communication complexity of graph properties, *Proc. 20th ACM STOC*, 186-191.

P. Hall (1936), The Eulerian functions of a group, *Quart. J. Math. Oxford* 134-151.

M. Karchmer and A. Wigderson (1988), Monotone circuits for connectivity require super-logarithmic depth, *Proc. 21th ACM STOC*, 539-550.

R. Lipton and R. Sedgewick (1981), Lower bounds for VLSI, *Proc. 13th ACM*

STOC 300-307.

L. Lovász (1979), ***Combinatorial Problems and Exercises***, Akad. Kiadó – North Holland.

L. Lovász and M. Saks (1988a), Lattices, Möbius functions and communication complexity, *Proc. 29th IEEE FOCS*, 81-90.

L. Lovász and M. Saks (1988b), unpublished

K. Mehlhorn and E. M. Schmidt (1982), Las Vegas is better than determinism in VLSI and distributed computing, *Proc. 14th STOC* 330-337.

C. H. Papadimitriou and M. Sipser (1983), Communication complexity, *Proc. 14th ACM STOC*, 196-200.

G.-C. Rota (1964), On the foundations of combinatorial theory I. Theory of Möbius functions, *Z. Wahrscheinlichkeitstheorie* **2**, 340-368.

R. P. Stanley (1986), *Enumerative Combinatorics*, Vol. 1, Wadsworth, Monterey, California.

E. R. Swart (1986), P=NP, *Tech. Report Univ. Guelph.*

E. Szemerédi (1976), Regular partitions of graphs, in: *Problèmes Combinatoire et Théorie des Graphes* (ed. J.-C. Bermond, J.-C. Fournier, M. Las Vergnas and D. Sotteau), CNRS, 399-401.

C. D. Thompson (1979), Area-time complexity for VLSI, *Proc. 11th ACM STOC*, 81-88.

H. S. Wilf (1968), Hadamard determinants, Möbius functions and the chromatic number of a graph, *Bull. Amer. Math. Soc.* **74**, 960-964.

M. Yannakakis (1988), Expressing combinatorial optimization problems by linear programs, preprint.

A. C.-C. Yao (1979), Some complexity questions related to distributive computing, *Proc. 11th ACM STOC*, 209-213.

A. C. Yao (1981), The entropic limitations of VLSI computations, *Proc. 13th ACM STOC*, 209-213.

A. C.-C. Yao (1983), Lower bounds by probabilistic arguments, *Proc. 24th IEEE FOCS*, 420-428.

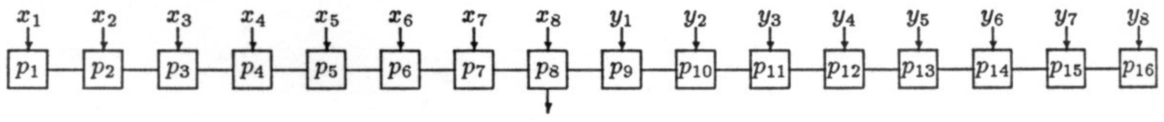
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13} p_{14} p_{15} p_{16}

Figure 1

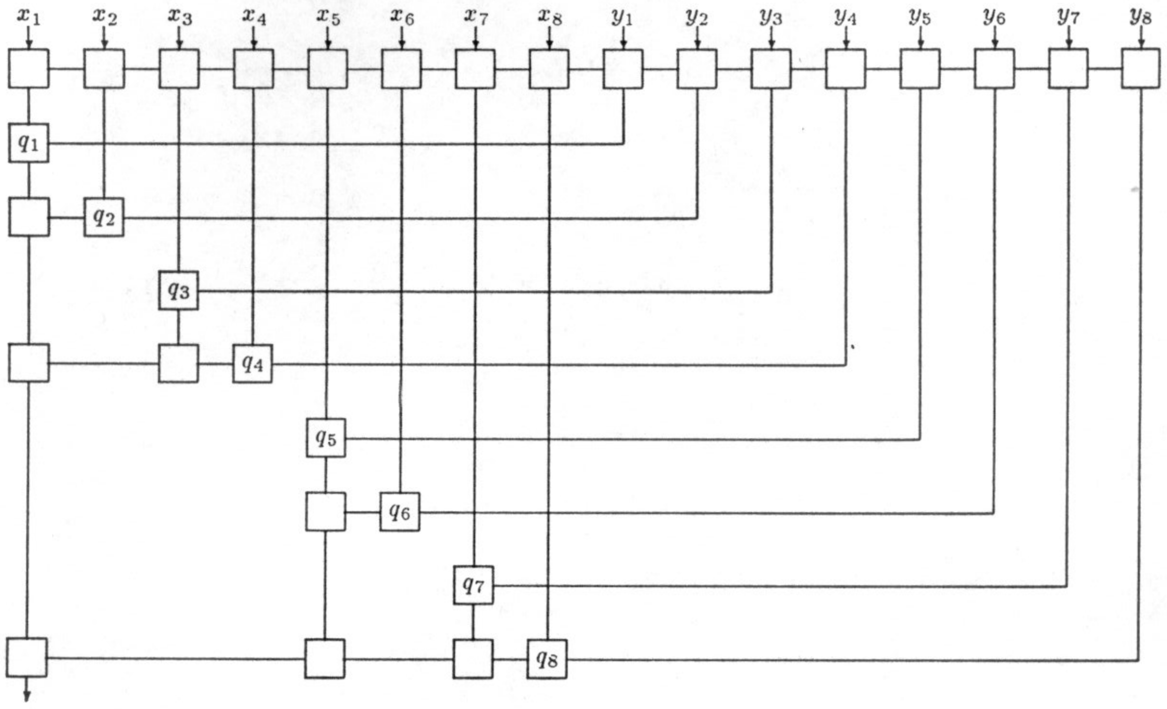
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8

Figure 2

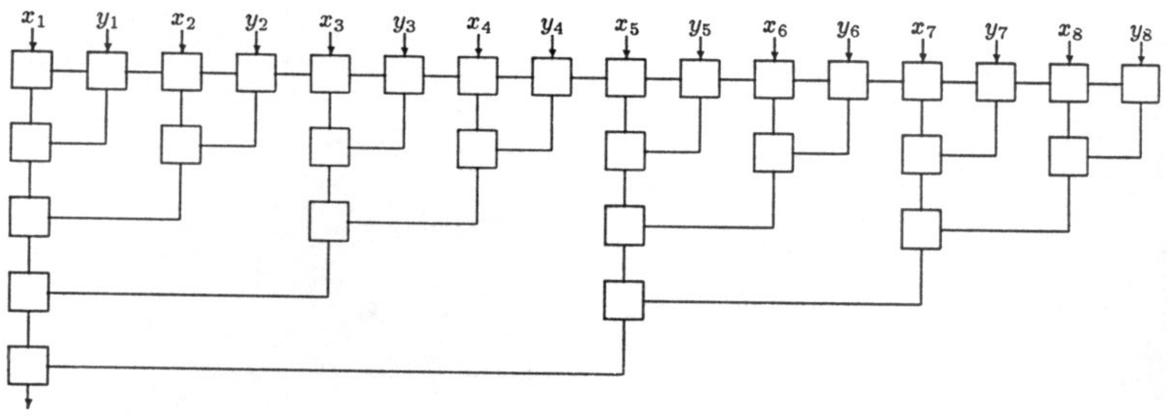

Figure 3