An Upper Bound for a Communication Game Related to Time-Space Tradeoffs

Pavel Pudlák Jiří Sgall

June 4, 1997

Abstract

We prove an unexpected upper bound on a communication game proposed by Jeff Edmonds and Russell Impagliazzo [2, 3] as an approach for proving lower bounds for time-space tradeoffs for branching programs. Our result is based on a generalization of a construction of Erdős, Frankl and Rödl [5] of a large 3-hypergraph with no 3 distinct edges whose union has at most 6 vertices.

1 Introduction

Suppose that we have two vectors u and v of length k. We want to decide whether $u = v$, but our access to the bits is very limited—at any moment we can see at most one bit of each pair of the bits u_i and v_i. You can imagine the corresponding bits to be written on two sides of a card, so that we can see all the cards but only one side of each card. After every flip we can write down some information, but the memory is not reusable—after the next flip we have to use new memory. We are charged for every bit of memory that we use and for every time we flip one or more cards.

It seems natural to suppose that if we flip the cards only a few times, we need a lot of memory. We prove an unexpected upper bound on the amount of memory needed; our bound is asymptotically tight if the number of card flips is constant. Our result is based on a construction of Erdős, Frankl and Rödl [5], whose special case is a large (in the number of edges) 3-hypergraph (a system of 3-element sets) with no 3 distinct edges whose union has at most 6 vertices, and on a previous result of Rusza and Szemerédi [8]. This special case of their construction corresponds to the case when only two flips of the cards are allowed. Our main idea is a geometric interpretation of the construction of a large set with no three element arithmetic progression due to Behrend [1] and its generalization to higher dimensions. We discuss this connection in Section 4.

In fact, Jeff Edmonds and Russell Impagliazzo proposed this game as a tool for proving lower bounds in complexity of boolean functions and proved that a reasonable lower bound on the sum of the number of flips and the number of bits of memory used during the game would imply a new lower bound for branching programs [2, 3]. We give a simple protocol with $O(\log n)$ probes which uses only $O((\log n)^2)$ bits of memory. We discuss this connection and protocol in Section 5.

First we describe the game more precisely and state some easy facts in Section 2, and prove our upper bound in the communication game setting in Section 3.

2 The game

Formally we describe the game as follows. Each input x is divided into some pieces (substrings) $x_1, \ldots, x_r$ in a fixed way (i.e., it is given which coordinate belongs to which piece). Each piece corresponds to a single card. Let $\Pi_1, \ldots, \Pi_T$ be a sequence of subsets of $[1, r]$ and let the inputs u and v be given. Then $\Pi_t(u, v)$ denotes the input consisting of the pieces x_i defined as $x_i = v_i$ if $i \in \Pi_t$ and $x_i = u_i$ if $i \notin \Pi_t$. These input vectors are called *probes*, and each of them corresponds to a choice of a visible side for each card. The protocol is described by the T probes $\Pi_1, \ldots, \Pi_T$ and a function $F(u)$ on the input vectors. If $F(u) = F(\Pi_1(u, v)) = \ldots = F(\Pi_T(u, v))$, the protocol answers "$u = v$", if not, the answer is "$u \neq v$". Let B be the number of bits of memory that we need, i.e., the maximal length of $F(u)$ over all inputs u. We are interested in the dependency of B on T and k. In the context of time-space tradeoff, the most important quantity is the minimal possible $B + T$, which corresponds to the total communication.

This corresponds to a protocol in which we first write down some information about u, and then after each of T flips we just check whether the current probe is consistent with that information. It is obvious that if we discover inconsistency, the vectors are different. Thus a protocol is correct if no two distinct vectors pass the test. In fact, here the use of memory is even more restricted than in the version described in the introduction—effectively the first time the protocol writes down arbitrary information $F(u)$ but then after each flip we write down only a single bit indicating consistency of the current probe. It is easy to show that this restriction increases the amount of memory by at most the maximum of B and T, see [2].

We can describe the set of probes in another equivalent way, more convenient for our proof. For each $i \leq r$ let $V_i \subseteq [1, T]$ denote the set of indices of the probes such that $t \in V_i$ iff the ith piece of $\Pi_t(u, v)$ is v_i (as opposed to u_i in other probes). We can assume that all sets V_i are distinct, because the pieces which appear in identical sets of probes can be joined (we can use one card instead of two cards that are always flipped together). Also, for every i, V_i is nonempty, as at least one probe has to look at v_i in a correct protocol. This means that $r \leq 2^T - 1$.

In case of $T = 1$, the game is just the usual communication game with one player having access to u and the other player to v; it is easy to see that k bits of memory are needed to test equality in that case. Similarly, it is easy to see that B must be at least the length of any piece u_i. If we fix all other pieces to be all zeros, then every probe is either u_i or v_i, and a correct protocol needs enough memory to distinguish any two distinct inputs u_i. This shows that $B \geq k/r \geq k/(2^T - 1)$.

We describe the best previously known protocol; it uses $B = \lceil k/T \rceil$ bits of memory [2]. Set $r = T$ and divide the input into r pieces of the same length (we assume w.l.o.g. that k is divisible by r). Set $\Pi_t = \{t\}$, or equivalently $V_i = \{i\}$ (i.e., each probe looks at just one piece of v and the rest comes from u). Set $F(u) = u_1 \oplus u_2 \oplus \cdots \oplus u_r$ to be the bitwise parity of the pieces. If the protocol answers "$u = v$", we know that $F(u) = F(\Pi_1(u, v)) = v_1 \oplus u_2 \oplus \cdots \oplus u_r$ and hence $u_1 = v_1$. The same argument is valid for other pieces, hence $u = v$ and the protocol is correct.

It is easy to prove that no protocol in which the function F is linear (over $GF(2)$) can be better. The equation $F(u) = F(\Pi_1(u, v)) = \ldots = F(\Pi_T(u, v))$ translates into a system of BT linear equations with $2k$ unknowns. If the protocol is correct, then the points in the k-dimensional subspace defined by $u = v$ are the only solutions of the system of equations, and hence there have to be at least k equations. This gives $B \geq k/T$.

Jeff Edmonds conjectured that this is in fact optimal even for non-linear protocols, i.e., $B = \Omega(k/T)$ for every protocol [2]. We disprove this conjecture. In fact, we prove that for constant T the easy lower bound is much closer to the truth as our protocol needs only $k/(2^T - 1) + O(\sqrt{k})$ bits of memory. If the number of probes is not bounded it is possible to achieve $B + T = O((\log k)^2)$ using a very simple protocol. We discuss this protocol and its consequences in Section 5.

3 The upper bound

Theorem 1 *For each parameter d and for each T there exists a protocol with T probes such that the number of bits of memory B is at most*

$$\left(1 + \frac{T}{d}\right) \left\lceil \frac{k}{2^T - 1} \right\rceil + (T + 2d + 1 + \log k)2^{2T-1}$$

Corollary 2 *For T constant and $d = \sqrt{k}$ the bound is $k/(2^T - 1) + O(\sqrt{k})$.*

Proof of Theorem 1. First we present the **protocol**.

Each input vector is divided into $r = 2^T - 1$ pieces $u_1, \dots, u_r$ of the same length $l = \lceil k/r \rceil$. We define the probes by taking the sets V_i, $i = 1, \dots, r$ to be all nonempty subsets of $[1, T]$. (In other words, the probes intersect as much as possible and the pieces are all of the same size—we have seen that this is necessary in a good protocol.)

We represent each u_i as a real vector of dimension $\lceil l/d \rceil$, where d is the parameter from the statement of the theorem, as follows. We partition u_i, a string of l bits, into $\lceil l/d \rceil$ substrings of d bits. Each coordinate is one of these substrings (in a given order) interpreted as an integer from $[0, 2^d - 1]$. From now on we abuse the notation and by u_i and v_i we mean the vectors described above, interpreted as points in the Euclidean space $\mathbf{R}^{\lceil l/d \rceil}$.

Now we are ready to describe the function $F(u)$. Let $\|x\|$ denote the Euclidean norm of a vector. Let u_0 denote the center of gravity of $u_1, \dots, u_r$, i.e., $u_0 = (\sum_{i=1}^r u_i)/r$. The function $F(u)$ consists of the concatenation of u_0 and all the distances $\|u_i - u_j\|$, $0 \le i < j \le r$.

As always, we first examine u and write down $F(u)$ and then for each probe we check if the value of F is equal to $F(u)$. This finishes the description of the protocol.

Let us compute the **number of bits** of $F(u)$. Instead of communicating u_0, we can communicate the vector ru_0, as r is a scalar constant. Its coordinates are integers from $[0, r(2^d-1)]$, hence $d+\log r$ bits are sufficient for each coordinate, a total of $(d+\log r)\lceil l/d\rceil \leq (1+T/d)\lceil k/r\rceil+T+d$ bits for all coordinates. Instead of each distance we communicate its square multiplied by r^2. This is a non-negative integer bounded by $r^2(2^{2d}-1)\lceil l/d\rceil < r^2 2^{2d} l < 2r2^{2d}k$, hence it can be represented by at most $T+2d+1+\log k$ bits, a total of $(T+2d+1+\log k)\binom{r+1}{2}$ for all distances. This gives the bound in the theorem.

To prove the **correctness** of the protocol, we need to prove that if two inputs u and v have the same value of F for u and all probes $\Pi_1, \ldots, \Pi_T$, then $u = v$. The intuitive idea is that for a given piece v_i we have sufficient information about its distances from u_j, $j \neq i$, to conclude that $v_i = u_i$.

We need a simple geometric lemma, which we prove later. Recall that a point $x \in \mathbf{R}^n$ is affinely dependent on points $x_1, \ldots, x_r \in \mathbf{R}^n$ if it can be written as their linear combination $\sum_{i=1}^r \alpha_i x_i$ such that $\sum_{i=1}^r \alpha_i = 1$.

Lemma 3 *Let x, $x_1, \ldots, x_r$ be points in Euclidean space $\mathbf{R}^n$ such that x is affinely dependent on $x_1, \ldots, x_r$. Let $y \in \mathbf{R}^n$ be a point satisfying $\|x - x_i\| = \|y - y_i\|$ for all $i = 1, \ldots, r$. Then $x = y$.*

Now we finish the proof of the theorem using this lemma. We can assume that the pieces of input are indexed in such a way that $V_i \supseteq V_j$ implies $i \leq j$, i.e., in the reverse topological order w.r.t. inclusion of the sets V_i. (This means for example that the piece v_1 appears in all probes.)

We prove by induction on $i = 1, \ldots, r$ that $u_i = v_i$. Suppose that we are proving the induction step for i, i.e., we have to prove that $u_i = v_i$. We want to use the lemma with $x = u_i$, $\{x_1, \ldots, x_r\} = \{u_0, \ldots, u_r\} - \{u_i\}$, and $y = v_i$. From the construction we know that x is affinely dependent on the remaining points. We need to prove that $\|u_i - u_j\| = \|v_i - u_j\|$ for all $j = 0, \ldots, r$, $j \neq i$. We distinguish three cases.

First, let $j > i$. Take any $t \in V_i - V_j$. By the assumption about the indexing of the pieces we know that such t exists. This means that $(\Pi_t)_i = v_i$ and $(\Pi_t)_j = u_j$ (we use $(\Pi_t)_i$ as a shorthand for $(\Pi_t(u,v))_i$, i.e., the point that represents the ith piece of the vector $\Pi_t(u,v)$). Now using the fact that $F(u) = F(\Pi_t(u,v))$ we get $\|u_i - u_j\| = \|(\Pi_t)_i - (\Pi_t)_j\| = \|v_i - u_j\|$.

The second case is $0 < j < i$. Now take any $t \in V_i$. By induction assumption we already know that $u_j = v_j$, hence $(\Pi_t)_j = u_j$. As in the previous case, we get $\|u_i - u_j\| = \|(\Pi_t)_i - (\Pi_t)_j\| = \|v_i - v_j\| = \|v_i - u_j\|$.

In the last case, $j = 0$, take again any $t \in V_i$. We know that $u_0 = (\Pi_t)_0$ because u_0 is a part of $F(u)$. The rest is again the same.

We have established the assumptions of the lemma, and its application finishes the induction step. We can conclude that $u = v$, hence the theorem holds. $\square$

Proof of Lemma 3. First we prove that the vector $y - x$ is orthogonal to $x_i - x_j$ for every i and j. Using the assumption of the lemma we have

$$\begin{aligned} 0 &= \frac{1}{2}(\|y - x_i\|^2 - \|x - x_i\|^2 + \|x - x_j\|^2 - \|y - x_j\|^2) \\ &= -y^T x_i + x^T x_i - x^T x_j + y^T x_j = (x - y)^T (x_i - x_j). \end{aligned}$$

This means that x is the projection of y on the affine subspace generated by $\{x_1, \ldots, x_r\}$. Using Pythagoras theorem, the projection of a point outside the subspace is always closer to any point in that subspace, hence y has to be identical with x. $\square$

Let us point out that our protocol works for any r, as long as every two pieces are distinguished by some probe. The choice of $r = 2^T - 1$ is done to optimize the bound.

We could have saved some communication in the protocol. Instead of taking the center of $u_1, \ldots, u_r$ it is possible to communicate the shift from u_r to the center of $u_1, \ldots, u_{r-1}$ and then to use this center instead of u_0 for the distances. If we do this, it is not necessary to communicate the distances from u_r at all. It is also not necessary to communicate the distances to the center, as they can be computed from the other distances. But all these savings still leave us with $\Theta(r^2)$ distances to communicate.

4 The connection with extremal problems

In this section we describe how the communication game can be translated into an extremal problem about hypergraphs and how that problem is related to extremal problems about arithmetic progressions.

Suppose that the input is divided into pieces $u_1, \ldots, u_r$ as before, and let $\mathcal{U}_1, \ldots, \mathcal{U}_r$ be the sets of all possible values for each piece. We assume for simplicity that all pieces have the same size, and denote $m = |\mathcal{U}_1| = \ldots = |\mathcal{U}_r| = 2^{k/r}$. Let G be the complete r-partite r-hypergraph on these sets of vertices. This means that each edge is a set of r points, exactly one from each $\mathcal{U}_i$. Each edge of G naturally corresponds to some input vector $(u_1, \ldots, u_r)$.

The function $F(u)$ from the protocol corresponds to a coloring of the hypergraph by 2^B colors. The condition that the protocol is correct means that for no $u = (u_1, \ldots, u_r)$, $v = (v_1, \ldots, v_r)$, all the edges corresponding to u, $\Pi_1(u, v)$, $\ldots$, $\Pi_T(u, v)$ have the same color. In other words, some specific patterns (or subhypergraphs) are not allowed to be monochromatic. (We get more patterns, because some pieces of u and v may be equal and we get degenerate versions of the original pattern.)

Note that we always need at least m colors, as the edges $(u_1, \ldots, u_{T-1}, u_T)$ and $(u_1, \ldots, u_{T-1}, u'_T)$ must always have different color, since this is a degenerate version of the prohibited pattern. (This is really just a translation of the trivial lower bound into the new language, because this degenerate pattern just corresponds to the case in which we change just one component of u.) This also means that every hypergraph without a prohibited pattern has at most m^{r-1} edges (out of m^r possible).

To prove a lower bound for the communication game it would be sufficient to prove that any hypergraph with too many edges necessarily contains a prohibited pattern. For the upper bound we not only need to find a large set with no prohibited pattern, but to decompose the complete hypergraph into a small number of such sets.

This kind of problems—to find a maximal size of a structure without a given pattern—is well-studied in extremal combinatorics, so it is not surprising that at least the simplest cases of our problem have been studied. Most of the information about it that we present now is from the survey Graham and Rödl [6] and the paper by Erdős, Frankl and Rödl [5]. These papers also contain simple proofs for some of the results that we mention below.

Let us look at the case $T = 2$ and $r = 3$. Now the prohibited pattern are the 3 edges (u_1, u_2, u_3), (u_1, v_2, v_3) and (v_1, v_2, u_3). The degenerate version of this pattern are any two edges that differ in a single point. We are now interested in the maximal number of edges of a hypergraph that does not contain any of these patterns.

For this case Rusza and Szemerédi [8] proved a slightly better bound than the trivial one $O(m^2)$ mentioned above, namely they proved that the number of edges is $o(m^2)$. This is proved using Szemerédi's regularity lemma [9], and unfortunately does not give a good bound for "o".

This problem is actually related to a problem of finding a large set of numbers which contains no arithmetic progression of length 3, as was noticed first in [8]. Suppose that we have a set $A \subseteq [0, (m-1)/2]$ with no arithmetic progression of length 3. Then we construct a hypergraph without a prohibited pattern by taking $\mathcal{U}_1 = \mathcal{U}_2 = \mathcal{U}_3 = [0, (m-1)]$ and putting in all edges of the form $(u, u+a, u+2a)$ for $a \in A$ and u arbitrary (the addition is taken modulo m), i.e., all arithmetic progressions with one element from each set and modulus from A.

Obviously no degenerate prohibited pattern can appear, because if two arithmetic progressions have two points identical, the third is identical as well. Little checking shows that the non-degenerate prohibited pattern corresponds exactly to the situation where the moduli a, a', and a'' are an arithmetic progression of length 3. So, if we have a large set A, we have a large hypergraph. How large can A be? The best known bounds on the size of such A are

$$\frac{m}{2^{-O(\sqrt{\log m})}} < |A| < \frac{m}{(\log m)^{\Omega(1)}}$$

The lower bound is a classical result from Behrend [1], the upper bound is due to Heath-Brown [7]. Improving these bounds is considered to be a very hard problem.

To have a small coloring, we need to decompose the interval $[0, m]$ into a small number of such sets, but that turns out to be easy. We also need to color the edges that are not arithmetic progressions, but that is trivial by using a new set of $m/|A|$ colors for edges of the form $(u, u+a, u+2a+c)$, for every constant c, a total of $m^2/|A|$ colors. Thus the construction based on the largest known set $|A|$ gives us a coloring by $m2^{O(\sqrt{\log m})}$ colors, which corresponds to communicating $k/3 + O(\sqrt{k})$ bits in our game.

Our upper bound for the communication game is based on this construction, translated into the geometric language, so that it can be generalized into a higher dimension. In our construction, the arithmetic progression of three points is replaced by our two points and their geometric center. In particular our protocol gives a construction of large r-hypergraphs without certain prohibited patterns which generalizes the well-known case of 3-hypergraphs with no 3 distinct edges whose union has at most 6 vertices.

5 Connection to time-space tradeoffs

This communication game was proposed by J. Edmonds and R. Impagliazzo [2] as a tool for proving lower bounds in complexity of boolean functions. We shall briefly describe the kind of results that one could possibly obtain without going into details in order to motivate our combinatorial result, for more information about this connection see [3].

A *branching program* is an oriented acyclic graph with one source, two sinks and each vertex, which is not a sink, having outdegree 2; the edges are labelled by variables and negated variables so that for each vertex we have a variable and its negation at the two outgoing edges; the sinks are labelled by *accept* and *reject*. An input vector determines a unique path from the source to a sink, the label at the sink determines, if the vector is accepted or not. The reason for introducing this special kind of a circuit is that the logarithm of the minimal number of vertices of a branching program is a natural measure of *space* (also called capacity) needed for computing a boolean function. This is because we can think of a vertex in the branching program as a configuration of the memory of a computational device. Similarly, the maximal length of a path from the source to a sink corresponds to *time*, however this measure of complexity is interesting only if combined with some restriction on the size of the branching program.

Since proving nontrivial lower bounds to a single measure, such as space, which we have described above, seems to be a very hard task, it is natural to try to prove lower bounds for combined measures. The branching program model of computation is an ideal combinatorial setting for proving a lower bound for the combined measure *time*$\times$*space*. Nevertheless, so far we have only the trivial lower bound $n \log n$ (for an explicitly defined n-variable boolean function).

Edmonds and Impagliazzo showed [3, 4] that if we could prove a lower bound $f(k)$ on the total number of bits of memory plus the total number of flips, we could prove a lower bound of $n\sqrt{f(n)}/\log n$ for time-space product for oblivious branching programs for the function of *element distinctness*.

However, the following simple protocol discovered by Russell Impagliazzo and the authors shows that it is possible to test the equality using only $O(\log k)$ probes and communicating $O(\log k)$ bits about each probe. This protocol can be converted into a protocol of the form used in Sections 2 and 3 that uses $O((\log k)^2)$ bits of memory.

We think of u and v as 0-1 vectors in real vector space $\mathbf{R}^k$. We compute the Euclidean distance of u and v and check if it is 0. To compute the distance, $u \cdot u + v \cdot v - 2(u \cdot v)$, we compute $u \cdot u$ and $v \cdot v$ each using one probe and $\log k$ bits of communication. Then we compute the product $(\sum_{i=1}^{k} u_i)(\sum_{i=1}^{k} v_i)$ using the same probes and additional $2 \log k$ bits of communication. To compute the desired inner product $u \cdot v$ we need to subtract the sum of terms $u_i v_j$ for $i \neq j$. This is easily done using $2 \log k$ probes—choose them so that each of the crossterms can be computed by one of them, and for each probe sum all of these terms assigned to it.

This protocol shows that a lower bound for element distinctness cannot be proved using this communication game. A more general game for which the above protocol cannot be used and thus seems as a feasible approach to time-space lower bounds was proposed by Edmonds and Impagliazzo in [4].

Acknowledgements

We would like to thank Russell Impagliazzo and Jeff Edmonds for many fruitful discussions. We are grateful to Vojta Rödl for pointing to us the literature on the related extremal problems. This work was partly done while visiting UC San Diego and supported by the US–Czechoslovak Science and Technology Fund No. 93025. Part of the work of the second author done at Carnegie-Mellon University.

References

[1] F.A. Behrend, *On sets of integers which contain no three in arithmetic progression*, Proc. Nat. Acad. Sci. 23 (1946), 331-332.

[2] J. Edmonds and R. Impagliazzo *Towards time-space lower bounds on branching programs*, manuscript.

[3] J. Edmonds and R. Impagliazzo *About time-space bounds for st-connectivity on branching programs*, manuscript.

[4] J. Edmonds and R. Impagliazzo *A more general communication game related to time-space tradeoffs*, manuscript.

[5] P. Erdős, P. Frankl, and V. Rödl, *The asymptotic number of graphs not containing a fixed subgraph*, Graphs and Combinatorics 2, 113-121 (1986)

[6] R.L. Graham and V. Rödl, *Numbers in Ramsey Theory*, In: Surveys in combinatorics (ed. I. Anderson), London Mathematical Society lecture note series 103, pp. 111-153, 1985.

[7] D.R. Heath-Brown, *Integer sets containing no arithmetic progressions*, preprint, 1986.

[8] I.Z. Ruzsa and E. Szemerédi, *Triple systems with no six points carrying three triangles*, Coll. Math. Soc. Janos Bolyai 18 (1978), pp. 939-945.

[9] E. Szemerédi, *Regular partitions of graphs*, In: Proc. Coloq. Int. CNRS, Paris, CNRS, 1976, 399-401.

[10] E. Szemerédi, *On sets of integers containing no k elements in arithmetic progression*, Acta Arith. 27 (1975), 199-245.