On Multiparty Games for Pointer Jumping

Carsten Damm       Stasys Jukna

Electronic copies of technical reports are available:

Via FTP: URL ftp://ftp.informatik.uni-trier.de/pub/Users-Root/reports
Via WWW: URL http://www.informatik.uni-trier.de/Reports/Current.html
Via email: Send a mail to ftpmail@ftp.informatik.uni-trier.de, subject
            'HELP', for detailed instructions

Printed copies:

Trierer Forschungsberichte
Fachbereich IV -
Mathematik / Informatik
Universität Trier
D-54286 Trier

ISSN 0944-0488
On Multiparty Games for Pointer Jumping

Carsten Damm Stasys Jukna†

Department of Theoretical Computer Science
University of Trier
D-54286 Trier, Germany
E-mail: {damm,jukna}@uni-trier.de

Abstract

We consider the multiparty communication complexity of the pointer jumping function $\text{Jump}_k^n$. Our main results are matching upper and lower bounds of order $\Theta \left( n \log^{(k-1)} n \right)$ for the strong one-way $k$-party communication complexity of $\text{Jump}_k^n$. The function itself is important since it simulates many natural occurring functions (e.g. shifting, multiplication of binary numbers, convolution, etc.) but no non-trivial lower bounds on its one-way communication complexity are known even for the case of $k = 4$ players. Our lower bound establishes an exponential gap between the strong one-way and general multiparty communication complexity. Such a gap was previously known between oblivious and one-way models. Best previous upper bound for $\text{Jump}_k^n$ was $O(n \log \log n)$. The work is motivated by an approach suggested by the results of Yao (1990) and Håstad & Goldmann (1991) to prove lower bounds for ACC circuits, as well as by the results of Valiant (1977) and Nisan & Wigderson (1993) to prove super-linear lower bounds for logarithmic depth circuits.

†On leave from Institute of Mathematics, Vilnius, Lithuania. Research supported by DFG grant Me 1077/5-2
1 Introduction

Multiparty games were introduced by Chandra, Furst and Lipton [5] and have found curious applications ([2], [3], [4], [8] [9], [11]) as a means of proving lower bounds on the computational complexity of explicit Boolean functions. In the most common model of multiparty communication, \( k \) players wish to compute a function \( f(v_1, \ldots, v_k) \) for which the \( i \)-th player sees all inputs except \( v_i \). The players have unlimited computational power. They share a blackboard, viewed by all players, where they can exchange messages according to a protocol. A message is a 0–1 string. The objective is to minimize the amount of communication. The cost of a protocol is the number of bits written on the blackboard for the worst case input. The multiparty communication complexity of \( f \) is the minimum cost of a protocol for \( f \).

In the general model the number of rounds and the order in which players speak is not limited. In most applications, however, we can find some restrictions, which make, at least potentially, the task of proving lower bounds easier. The weakest concept is the oblivious communication complexity. In this model each of the \( k \) players sends (independently from the others) one message to a referee, who sees none of the inputs. The referee then announces the result. Thus, in the oblivious model no communication between the players is allowed, they act independently; the twist is that they share some inputs (if \( k \geq 3 \)). In some applications ([3], [4]) we need a more general – one-way communication – model where communication between players is allowed but the order in which players can speak is prescribed: first speaks Player 1, then Player 2, etc.

The model of multiparty communication turns out to capture diverse computational models. In [5] it was used to prove that majority requires superlinear length constant width branching programs. Babai, Nisan and Szegedy [3] demonstrate that bounds for one-way communication complexity can be applied to Turing machine, branching program and formulae lower bounds. Results of Håstad and Goldmann [8] and Yao [13] show that a super-polynomial lower bound for the oblivious communication complexity of \( f \) with super-polynomial number of players implies that \( f \notin ACC \), i.e. that \( f \) has no polynomial size, bounded depth and unbounded fanin circuit with AND, OR, NOT and MOD\( m \) gates. Nisan and Wigderson [9] have shown, using a result of Valiant [12], that a lower bound \( \Omega(n / \log \log n) \) on a function \( f \) in a 3-party oblivious communication model would imply that \( f \) cannot be computed by a circuit of size \( O(n) \) and depth \( O(\log n) \).

The best lower bound for the general model of multiparty complexity is an \( \Omega(n / \log^k) \) lower bound of Babai, Nisan and Szegedy [3]. This means that we have no lower bounds at all for \( k = \Omega(\log n) \). Recently, there was a remarkable progress in understanding the oblivious case. Nisan and Wigderson [9] gave an \( \Omega(\sqrt{n}) \) lower bound for \( k = 3 \). Pudlák, Rödl and Sgall [11] and Babai, Kimmel and Lokam [2] proved an \( \Omega(n^{1/(k-1)}) \) lower bound for the oblivious communication complexity of, respectively, the iterated shift function

\[
\text{shift}_k^n(s_1, \ldots, s_{k-1}, \overline{x}) = x_{(s_1 + \cdots + s_{k-1}) \mod n}
\]

and the generalized addressing function

\[
GAF_n^k(s_1, \ldots, s_{k-1}, \overline{x}) = x_{s_1 \oplus \cdots \oplus s_{k-1}}
\]
where $\oplus$ is the bitwise sum mod 2. In both these functions, $s_i$ are log $n$-bit numbers and $\mathbf{x} \in \{0, 1\}^n$. Surprising oblivious protocols for $\text{shift}^n_k$ were found in [11] which got an $O\left( n(\log \log n / \log n)^k \right)$ upper bound for any constant $k$, and an $O(n^{6/7})$ bound if $k = \Omega(\log n)$. For polylogarithmic number of players the bound was recently improved by Ambainis [1] to $O(n^\epsilon)$ for any constant $\epsilon > 0$. Interesting upper bounds were proved also for $\text{GAF}_{n, k}$ in [2]: if $k = \Theta(\log n)$ then there is an oblivious protocol for $\text{GAF}_{n, k}$ in which players send at most $O\left( (\log n)^2 \right)$ bits to the referee. These bounds show that an impression about the simplicity of this extremely restricted model of communication may be quite wrong, and that it may be hard to get lower bounds larger than $\Omega(\sqrt{n^{1/(k-1)}})$ even for constant $k$.

One function often considered in the context of communication complexity is the pointer jumping function $\text{Jump}^n_k$ (see, e.g. [10, 6, 7, 9, 11]). There are $k + 1$ copies $A_1, \ldots, A_{k+1}$ of the set $[n] = \{1, 2, \ldots, n\}$, and $k$ sets of functions (“pointers”) $F_i = \{ f : A_i \rightarrow A_{i+1} \}, i = 1, \ldots, k$. There is a fixed point $a_1 \in A_1$. Input to $\text{Jump}^n_k$ is a string of functions $(f_1, \ldots, f_k)$ with $f_i \in F_i$. The output is

$$\text{Jump}^n_k(f_1, \ldots, f_k) = f_k(\ldots f_1(a_1) \ldots).$$

This is an important function, since it simulates many natural occurring functions, e.g. shifting, addressing, multiplication of binary numbers, convolution, etc. In particular, the functions $\text{shift}^n_k$ and $\text{GAF}_{n, k}$ correspond to $f_d(x) = x + s_i \mod n$ and $f_i(x) = x \oplus s_i$, respectively.

We consider so-called strong one-way $k$-party games for pointer jumping: There are $k$ players: Player 1, Player 2, ..., Player $k$. Player $i$ can see the functions $f_{i+1}, \ldots, f_k$. Additionally he can see the point $a_i = f_{i-1}(\ldots f_1(a_1) \ldots)$ — the point reached so far1. The players cooperate to determine the last point $a_{k+1} = f_k(\ldots f_1(a_1) \ldots)$. The twist is that Player $i$ does not see “his” function $f_i$. Communication is allowed only in one-way order: Player 1 begins; looking at the string $(a_1, f_2, \ldots, f_k)$ he writes some message $w_1$ on the blackboard. Then Player 2 looking at the string $(a_2; f_3, \ldots, f_k)$ and the message $w_1$ writes some message $w_2$, etc. The last player, Player $k$, looking at the string $(a_k)$ and the messages $w_1, w_2 \ldots w_{k-1}$, writes the answer $a_{k+1}$. The cost of the game is the number of bits exchanged between the players for the worst case input.

In this paper we strengthen our knowledge about multiparty communication in the following ways.

1. We prove that, for any constant $k$, the function $\text{Jump}^n_k$ requires $\Omega(n \log^{(k-1)} n)$ bits of communication in the strong one-way model. Similar bounds hold also for $k = k(n)$ such that $\log^{(k-1)} n \rightarrow \infty$ as $n \rightarrow \infty$. Here and throughout $\log^{(i)} n$ is the usual iterated logarithm: $\log^{(1)} n = \log n$ and $\log^{(i+1)} n = \log \log^{(i)} n$.

2. We show how the possibility to communicate the information from one player to others (even in restricted one-way manner) can be used to get an almost matching upper bound. We give an explicit strong one-way protocol for $\text{Jump}^n_k$ which uses only $O(n \log^{(k-1)} n)$ bits of communication, for any constant $k$. The bound holds also for any $k = k(n)$ such that $n \rightarrow \infty$.

1 Intuitively this seems to be the only useful information Player $i$ could draw from $(f_1, f_2, \ldots, f_{i-1}, s)$ — however it is a restriction of the general one-way $k$-party communication game, where Player $i$ has access to all functions except $f_i$. 

2
log\((k-1)\) \( n \to \infty \) as \( n \to \infty \). We also show that \( k = \Omega(\log \log n) \) players can compute \textit{Jump}_k^n using only \( O(n) \) bits. The best known upper bound for this function was an \( O(n \log \log n) \) bound of [11] in an \textit{almost oblivious} model (the difference from the oblivious model is that the last player takes the role of the referee; so he can see all but the last coordinate). This bound does not decrease with increasing \( k \); the protocols in [11] use colorings of random graphs and hence are nonconstructive. On the other hand, our protocols are non-oblivious since every player uses the information conducted so far.

2 The Upper Bound

Let \( \text{Pointer}_k [m, \mathcal{F}] \) denote a strong one-way \( k \)-party pointer jumping game among \( k \) players on the input space \( A \times \mathcal{F} \) where \( A \subseteq A_1, |A| = m \) and \( \mathcal{F} \subseteq F_1 \times \cdots \times F_k \), and there is a subset \( B \subseteq A_2 \) (known to all players) of size at most \( m \) such that for each \( (f_1, \ldots, f_k) \in \mathcal{F} \) holds \( f_i(A) \subseteq B \). The original game is a \( \text{Pointer}_k [n, F_1 \times \cdots \times F_k] \) game.

Suppose the players have to solve a \( \text{Pointer}_k [m, \mathcal{F}] \) game. We first demonstrate how the first two players can communicate to “shrink the input space by \( t \),” for an arbitrary \( t \), \( 1 < t \leq m \).

All players know an \( m \)-sized set \( A \subseteq A_1 \) containing \( a_1 \) and a set \( B \subseteq A_2 \) of size at most \( m \) that contains \( a_2 = f_1(a_1) \). Player 1 partitions \( B \) into \( t \) blocks \( B_1, \ldots, B_t \) of size at most \( m/t \) and sends the vector \( v(1) = (f_2(B_1), \ldots, f_t(B_t)) \) to the others. Only one block of these is relevant for the outcome of the game — the one containing \( a_2 \). Player 2 in the first part of his message announces the index \( r(1) = j \) of the relevant block \( B_j \) containing \( a_2 \). The remaining players have enough information to recover the set \( C = f_2(B_j) \). This set contains at most \( m/t \) points.

So Players 2 through \( k \) now need only to solve a \( \text{Pointer}_{k-1} [m/t, \mathcal{F}_1] \) game, where \( B_j \) and \( C \) play the role of \( A \) and \( B \) above, \( \mathcal{F}_1 \subseteq F'_2 \times \cdots \times F'_k \) and \( F'_2 \) consists of functions \( f \in F_2 \) with \( f(A_2) \subseteq C \).

How many bits of communication are needed for this step? For each of the sets \( f_1(B_1), \ldots, f_1(B_t) \) there are \( \sum_{i=1}^{m/t} \binom{n}{s} < 2^{nH\left(\frac{n}{m}\right)} \) possibilities, where \( H(p) = -\log p - (1 - p) \log (1 - p) \) is the binary entropy function. So the first player needs at most \( t n H\left(\frac{n}{m}\right) \) bits to send his message. Observe that \( \lim_{s \to \infty} H(1/s) \cdot \frac{s}{\log s} = 1 \) which means that for large \( s \) we have \( s \cdot H(1/s) \approx \log s \). Hence if \( \frac{nt}{m} \to \infty \) we can substitute \( s = \frac{nt}{m} \) and obtain that the player communicates less than

\[
ntH\left(\frac{m}{nt}\right) = m \cdot \frac{nt}{m} \cdot H\left(\frac{m}{nt}\right) < 2m \cdot \log \frac{nt}{m}
\]

bits to send \( v(1) \) if \( n \) is large enough. Additionally Player 2 uses \( \log t \) bits to announce the index \( r(1) = j \) of the relevant block.

We iterate this idea and follow the above scheme for \( k - 2 \) steps: in Step \( i \) Player \( i \) sends the vector \( v(i) \) of images of blocks partitioning the image of the recent relevant block contained in \( A_i \). The next in turn — Player \( i + 1 \) — additionally sends in the first part of his message the index \( r(i) \) of the relevant block in this stage. After these \( k - 2 \) steps Players \( k - 1 \) and \( k \) are left with a \( \text{Pointer}_2 [m, \mathcal{F}] \) game. This game can be solved with \( m \cdot \log n + \log n \) bits of communication: Suppose \( A \subseteq A_{k-1} \) is the set of possible start points and \( f_{k-1}(A) \subseteq B \) for any
input \( \langle f_k, f_h \rangle \in F \). Then Player \( k - 1 \) simply sends the whole table \( \langle f_k(b) \mid b \in B \rangle \). Player \( k \), knowing \( a_{k-1} \), can announce the result.

**Theorem 1** Let \( k = k(n) \) be such that \( \log^{(k-1)} n \to \infty \) as \( n \to \infty \). Then \( O(n \cdot \log^{(k-1)} n) \) bits of communication are enough for the strong one-way \( k \)-party pointer jumping game.

**Proof.** Recall that the original game is a Pointer\(_k\)[\( n,F_1 \times \cdots \times F_k \)] game. For \( i = 1, \ldots, k-2 \), set \( t_i = \log^{(k-i-1)} n \). In the first \( k-2 \) steps the input space will be shrunk by \( t_1, t_2, \ldots, t_{k-2} \), respectively. This means that for \( i \leq k-2 \), after \( i \)th message has been sent, we have a Pointer\(_{k-i}[m_i,F_i]\) game with \( F_0 = F_1 \times \cdots \times F_k, F_i \subseteq F_{i+1} \times \cdots \times F_k \), \( m_0 = n \) and \( m_i = m_{i-1}/t_i \).

For messages \( \tau(1), \ldots, \tau(k-2) \) only \( \sum_{i=1}^{k-2} \log t_i = O(k \log \log n) \) bits are needed.

Message \( \nu(i) \) consists of at most \( n t_i \cdot H \left( \frac{m_{i-1}}{m_{i}} \right) \) bits where \( \frac{n t_i}{m_{i-1}} = t_i \to \infty \) since \( t_i \to \infty \).

So by (1) message \( i \) consists of less than

\[
2m_{i-1} \cdot \log \left( \frac{n t_i}{m_{i-1}} \right) = 2 \cdot \frac{n}{t_0 \cdots t_i} \cdot \log (t_1 \cdots t_i) = 2n \cdot \gamma_i
\]

bits, where \( t_0 = 1 \) and \( \gamma_i = \frac{\log (t_1 \cdots t_i)}{t_2 \cdots t_i} \).

It holds \( \gamma_1 = \log t_1 = \log^{(k-1)} n \). Observe that \( t_{i-1} = \log t_i \) for \( i > 1 \). Hence the factor \( \gamma_2 = \frac{\log t_2 + t_4}{t_2} \) tends to 1 and for \( 2 < i \leq k - 2 \) we have \( \gamma_i = \frac{\log t_i + t_{i+2} + \cdots t_{i-1}}{t_2 \cdots t_i} \) which tends to 0. This means messages \( \nu(1), \tau(1), \ldots, \nu(k-2), \tau(k-2) \) require less than \( 2n \log^{(k-1)} n \) bits if \( n \) is large enough.

To solve the remaining Pointer\(_2[m_{k-2},F_{k-2}] \) game as described above \( m_{k-2} \log n + \log n = \frac{n}{t_2 \cdots t_{k-2}} + \log n \) bits are needed. Altogether \( 3n \log^{(k-1)} n \) bits of communication are sufficient.

Theorem 1 is valid for every fixed number of players \( k \). It is also valid for growing \( k \) if \( \log^{(k-1)} n \to \infty \). If we allow more players, we can get better protocols, as the following theorem shows.

**Theorem 2** If \( k(n) \geq \log \log n + 3 \) then \( k(n) \) players can solve the strong one-way \( k \)-party pointer jumping game with \( O(n) \) bits of communication.

**Proof.** Consider a one-way pointer jumping game with \( k \) players. We show, how \( l < k \) players can shrink the number of start points for the \( (l+1) \)th pointer to \( n/2^{l-1} \) while sending only few bits. Without loss of generality let \( n \) be a power of 2.

The first player knows \( a_1 \) and the mapping \( f_2 \). He sends the vector of the last bits of all values \( f_2(1), f_2(2), \ldots, f_2(n) \). This is Message \( \omega_1 \). Player 2 knows \( a_2 = f_1(a_1) \) and can therefore compute the last bit of \( a_3 = f_2(a_2) \) and the set \( B_3 \subseteq A_3 \) of all points in \( A_3 \) that agree in the last bit with \( a_3 \). This is the set of possible start points for the third pointer. Player 2 sends the
last bit of \( a_3 \) and the last two bits of \( f_3(a) \) for all \( a \in B_3 \). This is Message \( w_2 \). The next player
knows \( a_3 \) and can infer from \( w_1 \) and \( w_2 \) the last two bits of \( a_4 = f_3(a_3) \) and the set \( B_4 \subseteq A_4 \) of
all points in \( A_4 \) that agree with \( a_4 \) on the last two bits. Message \( w_3 \) consists of the last two bits
of \( a_4 \) and the last three bits of \( f_4(a) \) for all \( a \in B_4 \). We continue this way until the \( l \)th player
has spoken. Observe that \( |B_3| = n/2 \), \(|B_4| = n/4 \), \(|B_5| = n/8 \) etc. Hence the \( l \)th player can
compute a subset of \( A_{l+1} \) of size \( n/2^{l-1} \) that contains \( a_{l+1} \). How many bits are needed for these
\( l \) rounds? We have \(|w_1| = n \), \(|w_2| = 1 + 2 \cdot n/2 \), \(|w_3| = 2 + 3 \cdot n/4 \), \ldots \, \, |w_l| = l - 1 + l \cdot n/2^{l-1} \).
Altogether no more than \( \sum_{i=1}^{l-1} i + \sum_{i=1}^{\infty} i \cdot n \left( \frac{1}{2} \right)^{i-1} \leq 4n + O(\ell^2) \) bits are communicated.

Now suppose \( k(n) \geq \log \log n + 3 \). Let \( l = \log \log n + 1 \). The first \( l \) players follow the above
protocol. The \( l \), \((l+1)\)-th, player computes the set \( B_{l+2} \) and sends the whole table of values
\( f_{l+2}(a) \) for all \( a \in B_{l+2} \). The last player can compute \( a_{l+3} \) from this message and announce the
final result \( a_{l+1} \). The last two messages consist of \( \frac{1}{2} \cdot \log n + \log n < n \) bits.

\[ \]

Remark 3 Communication complexity of pointer jumping functions has been probably first introduced in [10]. In particular they consider the following 2-party \( k \)-round version: we have 2
players, pointers from \( A \) to \( B \) (known to Player 2) and from \( B \) to \( A \) (known to Player 1). There
is a fixed start point \( a_1 \in A \). Player 1 - the “wrong” player - starts, and the number of rounds
is bounded by \( k \). This \( 2 \)-party \( k \)-round game can be regarded as a restricted version of the above
\( k \)-party game, where the input space is \( \{(f_1, \ldots, f_k) : f_{i+2} = f_i \text{ for } i = 1, \ldots, k - 2\} \) and Player
\( P_i \) sees \( \{*, f_2, *, f_2, *, f_2, \ldots\} \) if \( i \) is odd and \( \{f_1, *, f_1, *, f_1, *, \ldots\} \) if \( i \) is even. The first general
lower bound for this game \( \Omega(n/k^2) \) was proved in [6]. In [9] this was essentially improved to a
lower bound of order \( \Omega(n) \) for deterministic protocols. For \( \epsilon \)-error protocols they prove a lower
bound of order \( \Omega(n/k^2) \) and an upper bound of order \( O((n/k) \log n) \). Using the same ideas as in
Theorem 1 and Theorem 2 one can improve these bounds to \( O(n \log (k-1)/n) \) for constant \( k \geq 2, \)
and to \( O(n) \) for \( k = \Omega(\log \log n) \). in round \( i > 1 \) the player in turn sends a \( \log n \) code for the
point \( a_i \) additional to the message according to the above protocols; the next player then can
compute the next point.

3 The Lower Bound Strategy

Given a finite set \( \Omega \) (universe), the density of a subset \( X \in \Omega \) is \( \mu_X(X) = \frac{|X|}{|\Omega|} \). If \( \Omega \) is clear
from the context, we will drop the subindex and write \( \mu(X) \). A subset \( X \subseteq \Omega \) is said to be an
\( \alpha \)-fraction of \( \Omega \) if \( \mu(X) \geq \alpha \).

We again want to consider appropriate “sub-games” of a pointer game. By a Pointer\(_k(B, \mathcal{F}) \)
game we will mean a strong one-way \( k \)-party game with input space \( B \times \mathcal{F} \) where \( B \subseteq A_1 \)
and \( \mathcal{F} \subseteq F_1 \times \cdots \times F_k \). A protocol solves this game if it outputs the correct value \( \tilde{f}(b) = f_k(\ldots f_1(b) \ldots) \) for every pair \((b, \mathcal{F}) \in B \times \mathcal{F} \). A Pointer\(_k(m, a) \) game is a Pointer\(_k(B, \mathcal{F}) \) game
with \(|B| \geq m \), and \( \mu(\mathcal{F}) \geq \alpha \). Observe that Pointer\(_k(1, 1) \) is exactly the original pointer jumping
game; the input space of this game is \( \{a_1\} \times \mathcal{F} \) with \( \mathcal{F} = F_1 \times \cdots \times F_k \). Since any protocol
for the Pointer\(_k(1, 1) \) game solves also the Pointer\(_k(n, 1) \) game\(^2\), it is enough to prove the lower

\(^2\)All players can see the start point.
bound for the Pointer$_k(n,1)$ game.

Given a protocol $\Phi$ for Pointer$_k(n,1)$ we will traverse it by keeping the set of inputs, consistent with what the first several players have said so far, as large as possible. Our main tool is a “reduction lemma” (Lemma 6 below). Using this lemma we define a sequence of subgames Pointer$_{k-1}(m_1, \alpha_1), \text{Pointer}_{k-2}(m_2, \alpha_2), \ldots, \text{Pointer}_1(m_{k-1}, \alpha_{k-1})$ such that, for each $i = 1, \ldots, k$, the protocol $\Phi$, when started with the $i$-th player, correctly solves the Pointer$_{k-i+1}(m_i, \alpha_i)$ game. We then show that, if the cost $L$ of the original protocol $\Phi$ is too small, then at least two of the remaining $\alpha_{k-1} \cdot n^\alpha$ functions from $A_k$ to $A_{k+1}$ must differ on at least one point in $A_k$, meaning that the last player is forced to make an error.

We will use the following two simple combinatorial facts.

Let $\Gamma_\alpha(m, n)$ denote the maximal number $r$ such that for any $\alpha$-fraction $F$ of all $n^\alpha$ functions $f : [n] \to [n]$ and for any subset $B \subseteq [n]$ with $|B| = m$, there is a $b \in B$ for which $|\{f(b) : f \in F\}| \geq r$.

Lemma 4

$$\Gamma_\alpha(m, n) \geq \alpha \frac{1}{m} n.$$  

Proof. Let $r = \Gamma_\alpha(m, n)$ and assume that there is a set of functions $F \subseteq \{f : [n] \to [n]\}$ with $|F| \geq \alpha n^\alpha$ and a subset $B \subseteq [n]$ such that $|B| = m$ and $|\{f(b) : f \in F\}| < r$ for all $b \in B$. Then $\alpha n^\alpha \leq |F| < r^m n^{\alpha - m}$, which gives the desired bound on $r$. \qed

Let $U$ and $V$ be finite sets, and let $G \subseteq U \times V$ be a bipartite graph of edges between $U$ and $V$. Say that a vertex $v \in V$ has degree-density $\beta$ if the degree of $v$ is $\beta |U|$. Let $V_\beta$ denote the set of all $v \in V$ with degree-density at least $\beta$.

Lemma 5

$$\mu(V_\beta) > \frac{\mu(G) - \beta}{1 - \beta}.$$  

Proof. Let $\kappa = |V_\beta|$ and $\gamma = \mu(G)$. There are at most $\kappa \cdot |U|$ edges incident to $V_\beta$, and less that $(|V| - \kappa) \cdot \beta |U|$ edges incident to $V \setminus V_\beta$. Thus $\kappa \cdot |U| + (|V| - \kappa) \cdot \beta |U| > |G| \geq \gamma |U| \cdot |V|$, which gives the desired bound $\kappa(1 - \beta) \geq (\gamma - \beta)|V|$. \qed
4 The Reduction Step

Our argument is based on the following “reduction lemma”.

**Lemma 6 (Reduction Lemma)** Let $0 \leq \beta \leq \alpha \leq 1$ and $1 \leq m \leq n$. If some Pointer$_k(m, \alpha)$ game has a strong protocol of cost $L$ then there is a Pointer$_{k-1}(m', \alpha')$ game which also has a strong protocol of cost $L$ where $m' \geq \binom{n}{\alpha} \cdot (2L+1)$ and

$$
\alpha' = \frac{\alpha - \beta}{(1 - \beta) \cdot m \cdot \binom{n}{m'}} (2L+1).
$$

**Proof.** Let $\Phi$ be a strong one-way protocol for a Pointer$_k(m, \alpha)$ game of cost $L$. This means we have sets $B \subseteq A_1$ and $\mathcal{F} \subseteq F_1 \times \cdots \times F_k$ such that $|B| = m$, $\mu(\mathcal{F}) = \alpha$ and $\Phi$ is correct on all inputs $(b, \vec{f})$ with $b \in B$ and $\vec{f} \in \mathcal{F}$. Every player $\Pi_i$ is a mapping

$$
\Pi_i : A_i \times F_{i+1} \times \cdots \times F_k \times \left( \{0,1\}^{\leq L} \right)^{i-1} \rightarrow \{0,1\}^{\leq L}.
$$

The protocol is a straight-line program

$$
\Phi = \Phi(b, \vec{f}) = \left\{ \begin{array}{l}
  z_1 \leftarrow \Pi_1(b; f_2, \ldots, f_k) \\
  z_2 \leftarrow \Pi_2(f_1(b); f_3, \ldots, f_k; z_1) \\
  z_3 \leftarrow \Pi_3(f_2(f_1(b)); f_4, \ldots, f_k; z_1, z_2) \\
  \vdots \\
  z_k \leftarrow \Pi_k(f_{k-1}(\ldots f_1(b) \ldots); z_1, z_2, \ldots, z_{k-1}) \\
\end{array} \right.
$$

The correctness of a protocol on a set $B \times \mathcal{F}$ where $B \subseteq A_1$ and $\mathcal{F} \subseteq F_1 \times \cdots \times F_k$ means that $z_k = \vec{f}(b) = f_k(\ldots f_1(b) \ldots)$ for any $b \in B$ and $\vec{f} \in \mathcal{F}$. We will use essentially the fact that the first message $z_1$ does not depend on the first coordinate of $\vec{f}$.

Our goal is to construct two sets $B' \subseteq A_2$ and $\mathcal{F}' \subseteq F_2 \times \cdots \times F_k$ with $|B'| = m'$ and $\mu(\mathcal{F}') \geq \alpha'$, such that for at least one message $w$, the following claim holds.

**Claim 7** There is a mapping $\nu : B' \times \mathcal{F}' \ni (c, \vec{g}) \mapsto (b_{c,\vec{g}}, h_{c,\vec{g}}) \in B \times F_1$ which fulfills, for every pair $(c, \vec{g}) \in B' \times \mathcal{F}'$, the following three conditions: (i) $(b_{c,\vec{g}}, h_{c,\vec{g}}, \vec{g}) \in B \times \mathcal{F}$, (ii) $\Pi_1(b_{c,\vec{g}}, h_{c,\vec{g}}, \vec{g}) = w$, and (iii) $h_{c,\vec{g}}(b_{c,\vec{g}}) = c$.

We first finish the proof of the lemma using this claim, and then prove the claim itself.

For $c \in B'$ and $\vec{g} = (g_1, \ldots, g_{k-1}) \in \mathcal{F}'$ consider the protocol

$$
\Phi' = \Phi'(c, \vec{g}) = \left\{ \begin{array}{l}
  z'_1 \leftarrow \Pi'_1(c; g_2, \ldots, g_{k-1}) \\
  z'_2 \leftarrow \Pi'_2(g_1(c); g_3, \ldots, g_{k-1}; z'_1) \\
  \vdots \\
  z'_{k-1} \leftarrow \Pi'_{k-1}(g_{k-2}(\ldots g_1(c) \ldots); z'_2, \ldots, z'_{k-2}) \\
\end{array} \right.
$$
where
\[
\begin{align*}
\Pi'_1(c; g_2, \ldots, g_{k-1}) &= \Pi_2(c; g_2, \ldots, g_{k-1}; w) \\
\Pi'_1(g_1(c); g_3, \ldots, g_{k-1}; z'_1) &= \Pi_3(g_1(c); g_3, \ldots, g_{k-1}; w, z'_1) \\
& \cdots \\
\Pi'_{k-1}(g_{k-2}(\ldots g_1(c) \ldots); z'_2, \ldots, z'_{k-2}) &= \Pi_k(g_{k-2}(\ldots g_1(c) \ldots); w, z'_2, \ldots, z'_{k-2})
\end{align*}
\]

The cost of this protocol is at most \(L - |w| \leq L\). Protocol \(\Phi'\) is correct on the set \(B' \times F'\). Indeed, for any \((c, \tilde{g}) \in B' \times F'\) by Claim 7 there exist \(b = b_{c, \tilde{g}}\) and \(h = h_{c, \tilde{g}}\) fulfilling (i)-(iii). Since \(\Phi\) is correct on \(B \times F\) it correctly produces the output \(g_{k-1}(\ldots g_1(h(b)) \ldots)\) on \((b, h, \tilde{g})\), which by (i) is a legal input for \(\Phi\). But by (ii) and the definition of the new protocol this is also the output of \(\Phi'\) on \((h(b), \tilde{g})\). Since \(g_{k-1}(\ldots g_1(h(b)) \ldots) = g_{k-1}(\ldots g_1(c) \ldots)\) according to (iii), this is the correct answer. Thus, the protocol \(\Phi'\) solves a Pointer\(_{k-1}\)(m', \alpha') game, as desired.

**Proof of Claim 7.** Say that a pair \((b, C)\) with \(b \in A_1\) and \(C \subseteq A_2\), is good for a string \(\tilde{g} \in F_2 \times \cdots \times F_k\) if \(|C| = \Gamma_\beta(m, n), b \in B\) and \(C \subseteq \{h(b) : h \in H_{\tilde{g}}\}\) where \(H_{\tilde{g}} = \{h \in F_1 : (h, \tilde{g}) \in F\}\). Some strings \(\tilde{g}\) have good pairs, some not. We first show that many strings \(\tilde{g}\) do have such pairs.

Let \(G\) be the set of all strings \(\tilde{g} \in F_2 \times \cdots \times F_k\) for which \(\mu(H_{\tilde{g}}) \geq \beta\). Applying Lemma 5 with \(U = F_1, V = F_2 \times \cdots \times F_k\) and \(G = F\), we get
\[
\mu(G) = \mu(V) \geq \frac{\mu(F) - \beta}{1 - \beta} = \frac{\alpha - \beta}{1 - \beta}.
\]

By Lemma 4, every string \(\tilde{g} \in G\) has at least one good pair \((b_{\tilde{g}}, C_{\tilde{g}})\). Fix a mapping \(G \ni \tilde{g} \mapsto (b_{\tilde{g}}, C_{\tilde{g}}),\) and consider the set of inputs
\[
\Delta = \{(b_{\tilde{g}}, h, \tilde{g}) : h \in F_1, \tilde{g} \in G \text{ and } (h, \tilde{g}) \in F\} \subseteq B \times F.
\]

The first player makes a partition of this set into at most \(\ell = 2^{L+1} - 1\) blocks
\[
\Delta_w = \{(b_{\tilde{g}}, h, \tilde{g}) \in \Delta : \Pi_1(b_{\tilde{g}}, h, \tilde{g}) = w\}
\]

with \(w \in \{0,1\}^{\leq L}\). Fix a message \(w\) for which the set
\[
G_w = \{\tilde{g} \in G : (b_{\tilde{g}}, h, \tilde{g}) \in \Delta_w \text{ for some } h \in F_1\}
\]
is maximal. Then \(\mu(G_w) \geq \mu(G)/\ell\). Since \(G_w \subseteq G\), every string \(\tilde{g} \in G_w\) has at least one good pair. Since there are at most \(N = m \cdot \binom{m}{2}\) good pairs, at least one of them, say \((b, C)\), must be good for at least \(1/N\) fraction \(G'\) of strings in \(G_w\). Since \(|C| = m'\) and \(\mu(G') \geq \mu(G_w)/N \geq \alpha'\), it remains to define a mapping \(\nu : C \times G' \rightarrow B \times F_1\) for which all the three conditions (i), (ii) and (iii) hold.

Consider the set of inputs \(\Delta' = \{(b, h, \tilde{g}) \in \Delta_w : \tilde{g} \in G'\}\). For every \(\tilde{g} \in G'\) there is at least one \(h \in F_1\) for which \((b, h, \tilde{g}) \in \Delta'\) (because \(G'\) is a subset of \(G_w\)). Since the first player cannot see the first coordinate \(h\) of an input \((h, \tilde{g})\), he is forced to send the same message on all the inputs \((b, h, \tilde{g})\) with \((h, \tilde{g}) \in F\). Hence \(\{b\} \times H_{\tilde{g}} \times \{\tilde{g}\} \subseteq \Delta',\) for every \(\tilde{g} \in G'\). On the other hand, the
goodness of the pair \((b, C)\) means that \(C \subseteq \{h(b) : h \in H_{\tilde{C}}\}\), for every \(\tilde{g} \in G'\). Thus, there is a mapping \(C \times G' \ni (c, \tilde{g}) \mapsto h_{c, \tilde{g}} \in H_{\tilde{C}} \subseteq F_1\) such that \((b, h_{c, \tilde{g}}, \tilde{g}) \in \Delta'\) and \(h_{c, \tilde{g}}(b) = c\), for every \(c \in C\) and \(\tilde{g} \in G'\). Extend this mapping to \(\nu : C \times G' \mapsto (b_{c, \tilde{g}}, h_{c, \tilde{g}}) \in B \times F_1\) by setting \(b_{c, \tilde{g}} = b\) for all \((c, \tilde{g})\). This mapping satisfies all three conditions: it satisfies (i) since \(\Delta' \subseteq B \times \mathcal{F}\), (ii) since \(\Delta' \subseteq \Delta_w\), and (iii) since \(h_{c, \tilde{g}}(b_{c, \tilde{g}}) = h_{c, \tilde{g}}(b) = c\).

**Theorem 8** For any constant \(k \geq 2\), every one-way protocol for the \(k\)-party pointer jumping game requires \(\Omega(n \log^{(k-1)} n)\) bits of communication.

**Proof.** Let \(L\) be the cost of an optimal protocol for the one-way \(k\)-party pointer jumping game. We will use Lemma 6 with \(\beta = \frac{\alpha}{2(2^{k+1} - 1)}\), which gives \(\alpha' \geq \alpha \cdot 2^{-\ell}\) where \(\ell = L + n + \log n\). For \(i = 0, 1, \ldots, k - 1\), set \(\alpha_i = 2^{-i\ell}\). By Lemma 6, the sequence of integers \(m_1 \geq m_2 \geq \ldots \geq m_k\) defined by the recurrence \(m_1 = \Gamma_{\alpha_0}(n, n)\) and \(m_{i+1} = \Gamma_{\alpha_i}(m_i, n)\) must end with \(m_k < 2\). We will make a rough estimate using the fact that \(\alpha_0 = 1\) and \(\alpha_i \geq \alpha = 2^{-i\ell}\) for all \(i\).

Let \(\ell = \frac{sn}{k}\), so that \(\alpha = 2^{-s\alpha}\). We want to estimate the function \(s = s(n)\). The sequence \(m_1 \geq m_2 \geq \ldots \geq m_k\) defined by the recurrence \(m_1 = \Gamma_{1}(n, n) = n\) and \(m_{i+1} = \Gamma_{2^{-s\alpha}}(m_i, n)\) ends with \(m_k \geq n/c_k\) where \(c_k = 2^{2^{k+1} - 2^{s+1} - 1}\). Therefore \(m_k < 2\) only if \(s + 2 \log s \geq \log^{(k-1)} n\), which gives the desired lower bound \(L = \frac{2n}{k} - n - \log n = \Omega\left(n \log^{(k-1)} n\right)\).

**Remark 9** For growing \(k = k(n)\) the bound is \(\Omega\left(n \log^{(k-1)} n\right)\) as long as \(\log^{(k-1)} n \geq (1 + \epsilon)k\) for some \(\epsilon > 0\).

## 5 Conclusion and open problems

We have proved non-trivial bounds for the strong one-way communication complexity of the pointer jumping function in the case of more than 3 players. The lower bounds hold for any constant \(k\) as well as for slowly growing \(k\). The main open problem remains to find nontrivial lower bounds for general one-way communication. The pointer jumping function seems to be a good candidate for a function not in \(\text{ACC}\). It would be therefore important to understand if this function can be computed by oblivious or one-way protocols with \(k = \text{polylog}(n)\) players using only \(\text{polylog}(n)\) bits. In order to get more insight, the following two problems should be solved.

We have seen that a constant number of players must use at least \(\Omega(n \log^{(k-1)} n)\) bits (Theorem 8) while \(k = \Omega(\log \log n)\) players can compute \(\text{Jump}_k^n\) with only \(O(n)\) bits of communication (Theorem 2).

**Problem 1:** Can the upper bound \(O(n)\) for \(\text{Jump}_k^n\) be improved to \(n^\epsilon\) for any constant \(\epsilon > 0\), using more than \(\log \log n\) players?

In some applications (e.g. to branching programs [3, 4]) we need good lower bounds on the one-way multiparty communication complexity of a Boolean version \(\text{Jump}_k^n\) of \(\text{Jump}_k^n\) defined
by

\[
\text{jump}^n_k(f_1, \ldots, f_k, \bar{x}) = x_{f(a_1)}
\]

where \( f(a_1) = f_k(\ldots f_1(a_1) \ldots) \) and \( \bar{x} \in \{0,1\}^n \) is part of the input, i.e. vector \( \bar{x} \) is seen by all \( k \) players. This function has an obvious one-way protocol of cost at most \( n \): Player 1 sends the \( n \)-bits string \( \langle x_{f(a_1)}, \ldots f_2(b) \rangle \mid b \in A_2 \rangle \),

and Player 2 announces the result, namely — the \( f_i(a_1) \)-th bit of this string.

**Problem 2:** Can the number of bits be substantially reduced, till say \( n' \), using log \( n \) players?

**Acknowledgement**

We are thankful to Ran Raz for pointing out to a gap in an earlier version.
References


