Monotone Circuit Depth Lower Bounds

Prashant Vasudevan

April 10, 2012

Table of Contents

Communication Complexity

Yao's model:

- Two players, Alice and Bob, given inputs $a \in A$ and $b \in B$, respectively. (Typically, $A = B = \{0,1\}^n$.)

Communication Complexity

Yao's model:

- Two players, Alice and Bob, given inputs $a \in A$ and $b \in B$, respectively. (Typically, $A = B = \{0,1\}^n$.)
- They wish to compute function $f : (A, B) \to Z$ by communicating with each other while minimising number of bits of communication. ($Z = \{0,1\}$ for decision problems.)

Communication Complexity

Yao's model:

- Two players, Alice and Bob, given inputs $a \in A$ and $b \in B$, respectively. (Typically, $A = B = \{0,1\}^n$.)
- They wish to compute function $f : (A, B) \to Z$ by communicating with each other while minimising number of bits of communication. ($Z = \{0,1\}$ for decision problems.)
- No bounds on computational power of players.

Communication Complexity

Yao's model:

- Two players, Alice and Bob, given inputs $a \in A$ and $b \in B$, respectively. (Typically, $A = B = \{0,1\}^n$.)
- They wish to compute function $f : (A, B) \to Z$ by communicating with each other while minimising number of bits of communication. ($Z = \{0,1\}$ for decision problems.)
- No bounds on computational power of players.
- For each function, the players establish a protocol beforehand.

Communication Complexity

Yao's model:

- Two players, Alice and Bob, given inputs $a \in A$ and $b \in B$, respectively. (Typically, $A = B = \{0,1\}^n$.)
- They wish to compute function $f : (A, B) \to Z$ by communicating with each other while minimising number of bits of communication. ($Z = \{0,1\}$ for decision problems.)
- No bounds on computational power of players.
- For each function, the players establish a protocol beforehand.
- Communication complexity of f is defined as the number of bits communicated in the protocol involving the least communication.

Communication Protocol

A protocol dictates the sequence of sending messages on any input and when to stop. The message sent by a player at any instant is a function of the input to the player and all the communication that has already happened.

Communication Protocol

A protocol dictates the sequence of sending messages on any input and when to stop. The message sent by a player at any instant is a function of the input to the player and all the communication that has already happened.

A useful representation is as the *communication tree* which is a binary tree where each inner node represents a decision made by some player and each edge represents a bit of communication.

Communication Matrix

The *communication matrix* is a $|A| \times |B|$ matrix M where $M_{ab} = f(a, b)$.

Communication Matrix

The *communication matrix* is a $|A| \times |B|$ matrix M where $M_{ab} = f(a, b)$.

A set of positions R in a matrix is said to be a rectangle if whenever $(x_1, y_1) \in R$ and $(x_2, y_2) \in R$, then $(x_1, y_2) \in R$ and $(x_2, y_1) \in R$.

Communication Matrix

The *communication matrix* is a $|A| \times |B|$ matrix M where $M_{ab} = f(a, b)$.

A set of positions R in a matrix is said to be a rectangle if whenever $(x_1, y_1) \in R$ and $(x_2, y_2) \in R$, then $(x_1, y_2) \in R$ and $(x_2, y_1) \in R$.

A monochromatic rectangle is one where the value of f at all positions in it is the same.

Lower Bounds

It is important to note that the set of pairs (a, b) which lead the players to any particular node in the communication tree form a rectangle.

Lower Bounds

It is important to note that the set of pairs (a, b) which lead the players to any particular node in the communication tree form a rectangle.

This gives us lower bounds on the number of leaves in the communication tree, which are at least as many in number as the number of disjoint monochromatic rectangles needed to tile the communication matrix.

Lower Bounds

It is important to note that the set of pairs (a, b) which lead the players to any particular node in the communication tree form a rectangle.

This gives us lower bounds on the number of leaves in the communication tree, which are at least as many in number as the number of disjoint monochromatic rectangles needed to tile the communication matrix.

Which in turn gives a lower bound on the depth of the communication tree and hence on the communication complexity of the function itself.

The Karchmer-Wigderson Game

A and B are disjoint subsets of $\{0,1\}^n$, and the objective is to find an index at which the strings a and b differ, i.e., to compute $f(a,b) = i : a_i \neq b_i$.

The Karchmer-Wigderson Game

A and B are disjoint subsets of $\{0,1\}^n$, and the objective is to find an index at which the strings a and b differ, i.e., to compute $f(a,b) = i : a_i \neq b_i$.

The minimum depth of any communication tree is again the *communication complexity* $C(A,B)$ of the pair A, B.

The Karchmer-Wigderson Game

A and B are disjoint subsets of $\{0,1\}^n$, and the objective is to find an index at which the strings a and b differ, i.e., to compute $f(a,b) = i : a_i \neq b_i$.

The minimum depth of any communication tree is again the *communication complexity* $C(A,B)$ of the pair A, B.

The *communication complexity* of a boolean function f is $C(A,B)$ with $A = f^{-1}(0)$ and $B = f^{-1}(1)$.

Table of Contents

Circuit Depth

Let $D(f)$ be the minimum depth of a formula with 2-input AND, OR and NOT gates computing f. We have the following intriguing connection between circuit depth and communication complexity.

Circuit Depth

Let $D(f)$ be the minimum depth of a formula with 2-input AND, OR and NOT gates computing f. We have the following intriguing connection between circuit depth and communication complexity.

Theorem (Karchmer-Wigderson, 1988)

For every boolean function f, $D(f) = C(f)$.

Monotone functions

A *monotone boolean function* is one in which switching any variable from false to true can never change the value of the function from true to false. These are precisely those functions that can be computed using only AND and OR gates.

Monotone functions

A *monotone boolean function* is one in which switching any variable from false to true can never change the value of the function from true to false. These are precisely those functions that can be computed using only AND and OR gates.

We define a *monotone version* of the Karchmer-Wigderson game in which the players are requied to find an i such that $a_i = 0$ and $b_i = 1$. Such an i may not always exist, but if $A = f^{-1}(0)$ and $B = f^{-1}(1)$ and f is a monotone boolean function, then it surely does.

Monotone functions

A *monotone boolean function* is one in which switching any variable from false to true can never change the value of the function from true to false. These are precisely those functions that can be computed using only AND and OR gates.

We define a *monotone version* of the Karchmer-Wigderson game in which the players are requied to find an i such that $a_i = 0$ and $b_i = 1$. Such an i may not always exist, but if $A = f^{-1}(0)$ and $B = f^{-1}(1)$ and f is a monotone boolean function, then it surely does.

Note that the depth of circuits with only AND and OR gates and communication complexity as per the monotone KW game for monotone boolean functions also satisfy the theorem stated earlier.

STCON

The *st-connectivity problem* $STCON_m$ is, given a directed graph on $m+2$ vertices (with special vertices s and t), to determine whether it has a path from s to t.

STCON

The *st-connectivity problem* $STCON_m$ is, given a directed graph on $m+2$ vertices (with special vertices s and t), to determine whether it has a path from s to t.

The graph is specified in the input by the characteristic string of its edges, e, such that $e_{(ij)} = 1$ iff there is an edge from i to j.

STCON

The *st-connectivity problem* $STCON_m$ is, given a directed graph on $m+2$ vertices (with special vertices s and t), to determine whether it has a path from s to t.

The graph is specified in the input by the characteristic string of its edges, e, such that $e_{(ij)} = 1$ iff there is an edge from i to j.

Notice that $STCON$ is a monotone function, as adding more edges cannot remove connectivity.

STCON

The *st-connectivity problem* $STCON_m$ is, given a directed graph on $m+2$ vertices (with special vertices s and t), to determine whether it has a path from s to t.

The graph is specified in the input by the characteristic string of its edges, e, such that $e_{(ij)} = 1$ iff there is an edge from i to j.

Notice that $STCON$ is a monotone function, as adding more edges cannot remove connectivity.

We give Alice a graph G such that $STCON(G) = 1$ and Bob a graph H (on the same vertex set) such that $STCON(H) = 0$. (We exchange $f^{-1}(0)$ and $f^{-1}(1)$ between the players, but this hardly matters.)

STCON

The monotone KW game now translates into finding an edge in G that is not present in H.

STCON

The monotone KW game now translates into finding an edge in G that is not present in H.

As we shall be looking into lower bounds, we may concern ourselves with special cases as we please. Here, let G be a simple path from s to t. Colour H with a coluring c such that $c(v) = 0$ if v is reachable from s and 0 otherwise. The game is now to find an edge (u, v) in G such that $c(u) = 0$ and $c(v) = 1$. We shall henceforth refer to this restricted game instead as $STCON_m$.

STCON

The monotone KW game now translates into finding an edge in G that is not present in H.

As we shall be looking into lower bounds, we may concern ourselves with special cases as we please. Here, let G be a simple path from s to t. Colour H with a coluring c such that $c(v) = 0$ if v is reachable from s and 0 otherwise. The game is now to find an edge (u, v) in G such that $c(u) = 0$ and $c(v) = 1$. We shall henceforth refer to this restricted game instead as $STCON_m$.

By binary searching on edges in the path in G, we can obtain $C(STCON_m) = O((\log m)^2)$.

FORK

We define one last game. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

In the fork game on a subset $S \subseteq [n]^l$, Alice and Bob are given strings $a, b \in S$ respectively.

FORK

We define one last game. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

In the fork game on a subset $S \subseteq [n]^l$, Alice and Bob are given strings $a, b \in S$ respectively.

The objective is to find a position i (1-indexed) such that $a_i \neq b_i$ and, if $i > 1$, $a_{i-1} = b_{i-1}$. Further, if $a_l = b_l$, l is also a valid answer.

FORK

We define one last game. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

In the fork game on a subset $S \subseteq [n]^l$, Alice and Bob are given strings $a, b \in S$ respectively.

The objective is to find a position i (1-indexed) such that $a_i \neq b_i$ and, if $i > 1$, $a_{i-1} = b_{i-1}$. Further, if $a_l = b_l$, l is also a valid answer.

Denote by $C(FORK_{n,l})$ the communication complexity of the game when played with $S = [n]^l$.

Reducion to STCON

The reduction comes about by considering the string in $[n]^l$ as a path in a graph on an $n \times l$ grid, where in each row i the path contains the vertex corresponding to a_i.

Reducion to STCON

The reduction comes about by considering the string in $[n]^l$ as a path in a graph on an $n \times l$ grid, where in each row i the path contains the vertex corresponding to a_i.

Add start and end vertices s and t, and connect these to the terminal vertices in the above path.

Reducion to STCON

The reduction comes about by considering the string in $[n]^l$ as a path in a graph on an $n \times l$ grid, where in each row i the path contains the vertex corresponding to a_i.

Add start and end vertices s and t, and connect these to the terminal vertices in the above path.

Computing $FORK_{n,l}$ on strings a and b now reduces to solving $STCON_{nl}$ on the two graphs, one corresponding to the path for a, and the other in which s and vertices in the path for b are coloured 0 and the rest 1.

Reducion to STCON

The reduction comes about by considering the string in $[n]^l$ as a path in a graph on an $n \times l$ grid, where in each row i the path contains the vertex corresponding to a_i.

Add start and end vertices s and t, and connect these to the terminal vertices in the above path.

Computing $FORK_{n,l}$ on strings a and b now reduces to solving $STCON_{nl}$ on the two graphs, one corresponding to the path for a, and the other in which s and vertices in the path for b are coloured 0 and the rest 1.

This gives us $C(FORK_{n,l}) \leq C(STCON_{nl}) = O((\log nl)^2)$.

Table of Contents

The Lower Bound

We shall now embark upon a most perilous journey in order to show that the above bound is almost optimal for $C(FORK_{n,n})$.

The Lower Bound

We shall now embark upon a most perilous journey in order to show that the above bound is almost optimal for $C(FORK_{n,n})$.

Call any protocol that for some subset $S \subseteq [n]^l$ with $|S| \geq \alpha n^l$ plays the fork game an (α, l)-protocol. Let $C(\alpha, l)$ be the minimum communication complexity of any (α, l)-protocol.

The Lower Bound

We shall now embark upon a most perilous journey in order to show that the above bound is almost optimal for $C(FORK_{n,n})$.

Call any protocol that for some subset $S \subseteq [n]^l$ with $|S| \geq \alpha n^l$ plays the fork game an (α, l)-protocol. Let $C(\alpha, l)$ be the minimum communication complexity of any (α, l)-protocol.

$C(FORK_{n,n}) = C(1, n)$.

Some Claims

Claim

For $l \geq 1$ *and* $\alpha \geq 1/n$, $C(\alpha, l) > 0$.

Some Claims

Claim

For $l \geq 1$ *and* $\alpha \geq 1/n$, $C(\alpha, l) > 0$.

Claim

For $l \geq 1$ *and any* α, *if* $C(\alpha, l) > 0$, *then* $C(\alpha, l) \geq C(\alpha/2, l) + 1$.

Some Claims

Claim

For $l \geq 1$ and $\alpha \geq 1/n$, $C(\alpha, l) > 0$.

Claim

For $l \geq 1$ and any α, if $C(\alpha, l) > 0$, then $C(\alpha, l) \geq C(\alpha/2, l) + 1$.

Starting at $C(1, n)$ and applying the second result above $\log n$ times, we get $C(1, n) \geq C(1/n, n) + \log n \geq \log n$. This gives $C(FORK_{n,n}) = \Omega(\log n)$, but we need a better lower bound.

Some Claims

Claim

For $l \geq 1$ and $\alpha \geq 1/n$, $C(\alpha, l) > 0$.

Claim

For $l \geq 1$ and any α, if $C(\alpha, l) > 0$, then $C(\alpha, l) \geq C(\alpha/2, l) + 1$.

Starting at $C(1, n)$ and applying the second result above $\log n$ times, we get $C(1, n) \geq C(1/n, n) + \log n \geq \log n$. This gives $C(FORK_{n,n}) = \Omega(\log n)$, but we need a better lower bound.

Notice that here we have left the n in $C(1, n)$ unchanged. We shall now harvest this.

Amplification

Lemma (Amplification Lemma)

For every $l \geq 2$ *and* $\alpha \geq 1/\sqrt{n}$, $C(\alpha, l) \geq C(\frac{\sqrt{\alpha}}{2}, \frac{l}{2})$.

Amplification

Lemma (Amplification Lemma)

For every $l \geq 2$ and $\alpha \geq 1/\sqrt{n}$, $C(\alpha, l) \geq C(\frac{\sqrt{\alpha}}{2}, \frac{l}{2})$.

In order to prove this, we shall need the following lemma.

Lemma

In a bipartite graph $G(U, V, E)$ with $|U| = |V|$ and $|E| \geq \alpha |V|^2$, at least one of the following holds:

a) *Some $u \in U$ is adjacent to at least $\sqrt{\frac{\alpha}{2}}|V|$ edges.*

b) *There is an $U' \subseteq U$ such that $|U'| \geq \sqrt{\frac{\alpha}{2}}|U|$ and each $u \in U'$ is adjacent to at least $\frac{\alpha}{2}|V|$ edges.*

Proving the Amplification Lemma

Assume the existence of an (α, l)-protocol Π for a set S.

Proving the Amplification Lemma

Assume the existence of an (α, l)-protocol Π for a set S.

Construct a bipartite graph with the vertex set on each side being the set of strings $[n]^{l/2}$. Edges are present between vertices u on the left and v on the right if the string uv is in the set S.

Proving the Amplification Lemma

Assume the existence of an (α, l)-protocol Π for a set S.

Construct a bipartite graph with the vertex set on each side being the set of strings $[n]^{l/2}$. Edges are present between vertices u on the left and v on the right if the string uv is in the set S.

If, on this graph, condition (a) of the above lemma should hold, then there is some $u \in [n]^{l/2}$ that has $\sqrt{\frac{\alpha}{2}}|V|$ neighbours. We may now obtain a $(\sqrt{\frac{\alpha}{2}}, \frac{l}{2})$-protocol for this set of neighbours from Π by placing u in front of any string in this set and running Π.

Proving the Amplification Lemma

If condition (b) holds, consider the $n \times l/2$ block out of which strings in $[n]^{l/2}$ on the right partition in the graph are formed by picking one symbol from each layer. At each layer, pick $n/2$ symbols at random and form thus two $n/2 \times l/2$ blocks called X and Y.

Proving the Amplification Lemma

If condition (b) holds, consider the $n \times l/2$ block out of which strings in $[n]^{l/2}$ on the right partition in the graph are formed by picking one symbol from each layer. At each layer, pick $n/2$ symbols at random and form thus two $n/2 \times l/2$ blocks called X and Y.

For any $u \in U'$, with probability at least $1 - 2e^{-\alpha n/4}$ there is an extension $v_X(u)$ of u that is entirely in X and another, $v_Y(u)$, entirely in Y.

Proving the Amplification Lemma

We can then use Markov's inequality to show that if we keep $\alpha \geq n^{-1/2}$ ($\alpha >> 1/n$), then there exists some choice of X and Y such that at least $1/\sqrt{2}$ of all strings in U', that is, at least $\sqrt{\alpha}/2$ of all strings in $[n]^{l/2}$ on the left have extensions in both X and Y.

Proving the Amplification Lemma

We can then use Markov's inequality to show that if we keep $\alpha \geq n^{-1/2}$ ($\alpha >> 1/n$), then there exists some choice of X and Y such that at least $1/\sqrt{2}$ of all strings in U', that is, at least $\sqrt{\alpha}/2$ of all strings in $[n]^{l/2}$ on the left have extensions in both X and Y.

This set that has these extensions is our new set S' on which we have a $(\frac{\sqrt{\alpha}}{2}, \frac{l}{2})$-protocol - given $u_a, u_b \in S'$, run Π on $u_a v_X(u_a)$ and $u_b v_Y(u_b)$. As these strings have no common symbols in the right half, the answer that Π gives is that for u_a and u_b.

Proving the Amplification Lemma

We can then use Markov's inequality to show that if we keep $\alpha \geq n^{-1/2}$ ($\alpha >> 1/n$), then there exists some choice of X and Y such that at least $1/\sqrt{2}$ of all strings in U', that is, at least $\sqrt{\alpha}/2$ of all strings in $[n]^{l/2}$ on the left have extensions in both X and Y.

This set that has these extensions is our new set S' on which we have a $(\frac{\sqrt{\alpha}}{2}, \frac{l}{2})$-protocol - given $u_a, u_b \in S'$, run Π on $u_a v_X(u_a)$ and $u_b v_Y(u_b)$. As these strings have no common symbols in the right half, the answer that Π gives is that for u_a and u_b.

This proves the lemma.

Using the Amplification

Again start with $C(1, n)$ and apply the result like we did far above to obtain:

$$C(1,n) \geq C(\frac{2}{\sqrt{n}}, n) \geq C(\frac{16}{n}, n) + \Omega(\log n)$$
$$\geq C(\frac{2}{\sqrt{n}}, \frac{n}{2}) + \Omega(\log n)$$

Using the Amplification

Again start with $C(1, n)$ and apply the result like we did far above to obtain:

$$C(1,n) \geq C(\frac{2}{\sqrt{n}}, n) \geq C(\frac{16}{n}, n) + \Omega(\log n)$$
$$\geq C(\frac{2}{\sqrt{n}}, \frac{n}{2}) + \Omega(\log n)$$

We may apply this $\Theta(\log n)$ times to finally get:
$C(1,n) \geq C(\frac{2}{\sqrt{n}}, 1) + \Omega((\log n)^2) = \Omega((\log n)^2)$

Using the Amplification

Again start with $C(1, n)$ and apply the result like we did far above to obtain:

$$C(1,n) \geq C(\frac{2}{\sqrt{n}}, n) \geq C(\frac{16}{n}, n) + \Omega(\log n)$$
$$\geq C(\frac{2}{\sqrt{n}}, \frac{n}{2}) + \Omega(\log n)$$

We may apply this $\Theta(\log n)$ times to finally get:
$C(1,n) \geq C(\frac{2}{\sqrt{n}}, 1) + \Omega((\log n)^2) = \Omega((\log n)^2)$

This gives us $C(FORK_{n,n}) = \Omega((\log n)^2)$.

Finally

Going back, we see that since $C(STCON_m) \geq C(FORK_{\sqrt{m},\sqrt{m}})$, $C(STCON_m) = \Omega((\log m)^2)$.

Finally

Going back, we see that since $C(STCON_m) \geq C(FORK_{\sqrt{m},\sqrt{m}})$, $C(STCON_m) = \Omega((\log m)^2)$.

This tells us that any monotone circuit that computes *st-connectivity* in directed graphs on n vertices necessarily has depth $\Omega((\log n)^2)$.