Recent advances in Complexity CIS 6930/CIS 4930 October 15

Lecture 15

Lecturer: Dr. Meera Sitharam Scribe: Erwin Jansen

1 Introduction

In today's lecture we started on proving circuit depth lower bounds (regardless of size) for *st*-connectivity. We will prove lower bounds for monotone circuit depth. Note that *stcon* is a monotone function (why?) The proof for a lower bound for this function will be done according to the following ideas:

Idea 1: Give a communication complexity lower bound. Recall from previous lectures the following definition and theorem that relates communication complexity to (monotone) circuit depth of functions:

> **Definition 1** *For a boolean function* $f : \{0,1\}^n \rightarrow \{0,1\}$, *let* $X = f^{-1}(1)$ *(i.e. the set of all* x*'s such that* $f(x) = 1$*) and* $Y = f^{-1}(0)$. *Let* $R_f \subseteq X \times Y \times \{1, ..., n\}$ *consist of all triples* (x, y, i) *such that* $x_i \neq y_i$. *Let* $M_f \subseteq X \times Y \times \{1, ..., n\}$ *consist of all triples* (x, y, i) *such that* $x_i = 1$ *and* $y_i = 0$. *I.e,* $x_i > y_i$. *Note that this is analogous to* R_f *(and makes sense) for monotone functions* f.
>
> **Theorem 2** *For every* $f : \{0,1\}^n \rightarrow \{0,1\}$, *we have* $d(f) = D(R_f)$. *And* $d_m(f) = D(M_f)$. *Here* $d(f)$ *is the min depth of a circuit computing* f, *and Here* $d_m(f)$ *is the min depth of a monotone circuit computing* f. *The latter result says that the communication complexity of a relation of the form of* M_f *(as opposed to* R_f*) is only useful for lower bounding bounding the monotone circuit depth (as opposed to general circuit depth) of* f.

Idea 2: We want to prove a lower bound for *st*-connectivity, the function that gives a 1 if there is a path between s and t, by proving a lower bound for R_{fork} and by giving a reduction from the protocol for *st*-connectivity (M_{stcon}) to the protocol for fork (R_{fork}) that does not *increase* the depth. In other words:

$$R_{fork} \leq_{\triangle} M_{stcon}$$

Idea 3: Show the lower bound for R_{fork}.

Due to theorem 2 we don't need to explicitly define the function fork. As long as we are able to define the set R_{fork} we can proceed. We will define the set R_{fork} as follows:

R_{fork}

Definition 3 *Let $j, w, l, i \in \mathbb{N}$ and $x, y \in \{1, ..., w\}^l$. We define the set R_{fork} as follows:*

$$R_{fork} = \{(x, y, i) | x_i = y_i \wedge x_{i+1} \neq y_{i+1}\}$$

There is one drawback with the this definition however. Not every pair x, y is in R_{fork}:

Example 1 *For $w = 3$, let $x = 223$ and $y = 121$ and $i = 2$, then $(x, y, i) \in R_{fork}$. However for given $x = 222$ and $y = 222$ there is no i such that $(x, y, i) \in R_{fork}$*

It is rather inconvenient that not every x, y is in R_{fork}, so we extend the definition a little bit by adding some extra information to x and y. We are going to place a digit in front and to the back of x and y. We obtain the following definition:

R_{fork}

Definition 4 *Let $\cdot$ denote string concatenation. Let $j, w, l, i \in \mathbb{N}$ and $x = 1 \cdot \{1, ..., w\}^l \cdot w$ and $y = 1 \cdot \{1, ..., w\}^l \cdot (w - 1)$. Now we define R_{fork} as follows:*

$$R_{fork} = \{(x, y, i) | x_i = y_i \wedge x_{i+1} \neq y_{i+1}\}$$

This new definition has the advantage that for every pair x and y, there is some i such that (x, y, i) is an element of the set R_{fork}:

Claim 5

$$\forall_{x,y} \exists_i \langle (x, y, i) \in R_{fork} \rangle$$

Proof: Trivial, just find the first place where they differ. Convince yourself that such a place exists. ■

The function st-connectivity gives a one when there exists a path between the source s and the target t. Formally this function is defined as follows:

st-connectivity

Definition 6 (st-connectivity) *Let $G = (V, E)$ be a digraph (directed graph) where $|V| = n$. Let G have two vertices $s, t \in V$ called the source and target. The boolean function stcon is defined as follows:*

$$stcon(G) = 1 \iff \exists_{e \subseteq E} \langle s \overset{e}{\rightsquigarrow} t \rangle$$

Recall from previous lectures that we can represent a graph as a binary string (of length $n^2 - n$) by indicating whether an edge exists (1) or not (0). Since we can always relabel a graph we assume that s and t are fixed vertices in the input. Notice that *stcon* is a monotone function. If s and t are connected, then the addition of another edge will not disconnect s and t.

2 Reduction

We will now examine how we can convert a communication protocol for *stcon* into a communication protocol for *fork* without increasing the depth, the amount of bits communicated, of the protocol. First let us make sure we formally understand the definition of M_{stcon}.

M_{stcon}

Definition 7 (M_{stcon}) *Let stcon be a boolean function for a graph of n vertices. Let $G_1 = stcon^{-1}(1)$ and $G_2 = stcon^{-1}(0)$. Now M_{stcon} is defined as:*

$$M_{stcon} = \{(x, y, i) \mid x \in G_1, y \in G_2, x_i \neq y_i\}$$

Since the input of the function *stcon* is a string of bits indicating whether or not there exists an edge in the graph G, an element $(x, y, i) \in M_{stcon}$ indicates that there does not exists an edge in the graph x where there is an edge in the graph y. A protocol for M_{stcon} is actually giving this edge as an output.

We will look at a subset $M'_{stcon} \subseteq X' \times Y \times \{1, ..., n^2 - n\}$. The reason for looking at a subset is that it is easier for us and it is sufficient due to the following condition:

$$R_{fork} \leq_{\triangle} M'_{stcon} \leq_{\triangle} M_{stcon}$$

We will show the first reduction, whereas the latter one is trivial. For the first reduction we have to show that we can convert the input to from R_{fork} to an input of M'_{stcon} and are able to convert the output of M'_{stcon} back to R_{fork}, as depicted in figure 2.

The conversion of the input will be two graphs. One graph G_1 used by Alice and one graph G_2 used by Bob. The graph that Alice and Bob will use is a layered graph as depicted in figure 1. The graph will be of size n. We have $l + 2 = \sqrt{n}$ layers of $w = \sqrt{n}$ vertices.

Conversion

Definition 8 (Converting the input) *Here is how we convert an input from R_{fork} to an input of M'_{stcon}. Take the string $x \in 1 \cdot \{1, ..., w\}^l \cdot w$, interpreted by Alice as a graph consisting of a single path from 1 to w. The string $y \in 1 \cdot \{1, ..., w\}^l \cdot (w - 1)$ will be interpreted by Bob as a graph in G_2 which contains the path given by y. In addition to this, Bob is going to throw in* all *other edges between adjacent layers, except those who originate in one of the vertices on the path.*

The graphs that Bob and Alice use might remind you of the positive and negative test graphs of some lectures ago. Alice is using a minimal graph for which *stcon* returns true whereas Bob is using a maximal graph for which *stcon* returns false. Notice that in figure 1 we did *not* draw all the edges for Bob.

Claim 9 *Let the output of the protocol M_{stcon} be the edge (u, v) that is in $G1$ and not in $G2$. Let u be in the i^{th} layer of G_1. (By definition of layered graphs, v is in $(i + 1)^{st}$ layer). Now i is a correct output for a protocol for R_fork on input x, y, i.e. $R_{fork}(x, y, i) = 1$*

Proof: Since $(u, v) \in G_1$, it follows that $u = x_i$ and $v = y_i$ by construction of G_1.We now have to show that $(u, v) \notin G_2$. By construction of G_2, the edges in G_2 go from everything other than y_i in i^{th} layer to everything in the $(i+1)^{th}$ layer, and the single edge from y_i to y_{i+1}. So, the edges *not* in G2 are exactly those that go from y_i in i^{th} layer to everything other than y_{i+1} in the $(i + 1)^{th}$ layer. Therefore $u = y_i$ and $v \neq y_{i+1}$. Therefore i is such that $x_i = y_i$, but $x_{i+1} \neq y_{i+1}$. Hence i is a correct output for a protocol for R_{fork} on input x,y. ■

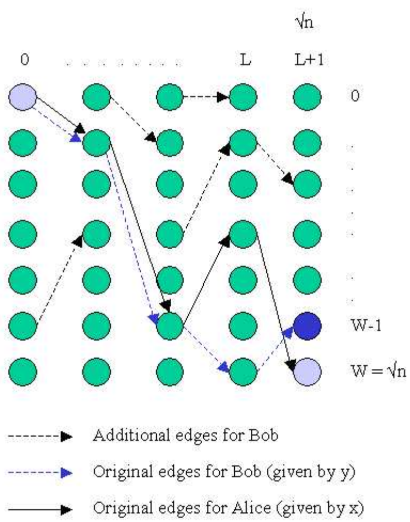

Figure 1: Layered Graph

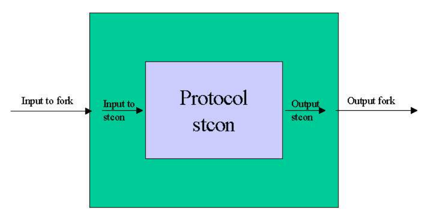

Figure 2: Reduction