1 The Overall View

In a collection of seminal works by Jehoshua Bruck and others, the circuit complexity of computing various arithmetic functions is studied using a linear threshold element (LTE) as a basic gate. Interestingly, it turns out what many arithmetic functions needed for Shor's factoring algorithm can be done in constant-depth in a threshold circuit.

These include

1. Powering: the raising of an n-bit constant b to an n-bit power $|x\rangle$, yielding an n^2-bit final product z as the first stage in modular exponentiation. This takes depth 4 in LTE's.

2. Division: finding the quotient of an n^2-bit integer z with an n-bit integer m, as the first step in modular reduction, which itself is the second step in modular exponentiation.

3. Multiplication: finding the product of the quotient q in the previous step and the divisor b also from the previous step, to get y, the largest multiple of m less than z, as the second step of modular reduction.

4. Subtraction: subtracting y from z, to get the modular residue $w = b^{|x\rangle} \bmod m$.

Due to Hoyer and Spalek, it is known how to perform a quantum threshold gate in constant-depth using constant-depth teleportation and fanout. In particular, this was calculated by Andrea McCool to take the following resources:

Fill this in here from Andrea's report, I think it was for AC.

Therefore, the main task of mapping these results from classical circuit complexity to find a nearest-neighbor quantum architecture for factoring is as follows:

1. Mapping the quantum threshold gate from AC to CCNTC, using the Clifford group and Toffoli and finding the overhead incurred in resources. Find a way to use Cody Jone's Fourier state distillation.

2. Finding the circuit size, in LTE's, for the classical circuits above (powering, division, subtraction, multiplication), using constant-depth fanout and teleportation.

3. Finding the circuit width, in LTE's, for the classical circuits above, (powering, division, subtraction, multiplication) using constant-depth fanout and teleportation.

4. Calculate the lower bounds for the circuit size and width, using the lemma mentioned in the next section.

2 Linear Threshold Elements

The linear threshold element is a primitive which we assume can implement any linear threshold function. This is a reasonable assumption given our construction of a quantum threshold gate which can perform something something.

Also fact check the resolution needed for phase rotations, in rotation by Hamming value and by Hamming weight, needed for the quantum threshold gate.

3 Summary of Bounded-Depth Quantum Circuit Complexity

Here we can summarize the results from Takahashi and Tani, namely something about AC^0 being properly contained in TC^0, when using only quantum AND, quantum OR gates, but with the containment known to collapse when we are allowed to use unbounded quantum-fanout.

4 Motvations

We have several motivations for studying the complexity of threshold circuits. The first is that some basic functions such as PARITY and MAJORITY are known to be impossible in constant-depth circuits consisting only of AND, OR, XOR, and NOT. However, when considering threshold gates, or a special case MAJORITY, as a basic gate, then complicated arithmetic functions such as PARITY, ADDITION, EXACT, COUNT, and so forth, become possible in constant-depth.

The second motivation is the use of beautiful new tools that were seemingly unrelated previous to circuit complexity.

1. The harmonic analysis of boolean functions, via the spectral norms of its coefficients, which provides a beautiful connection between the idea of a function's spectral norm to bound the number of terms in its polynomial representation. We can characterize the terms in this representation, and therefore lower bound the size of the circuit needed to implement the function in LTE's, without knowing the explicit function itself. This uses the probabilistic method and the creation of random polynomials.

2. The polynomial representation of a Boolean function and its relationship to the Sylvester-form of the Hadamard matrix.

3. Something else? I feel like there was another connection.

5 Spectral Norms

Every Boolean function in n variables can be represented as a polynomial threshold function with an exponential number of terms and with exponential weights. However, these can be difficult to implement. We are interested in functions which can be implemented with polynomially-bounded terms (so-called *sparse polynomials*) and polynomially-bounded weights.

That is

$$f : \{+1, -1\} \rightarrow \{+1, -1\} \tag{1}$$

where we assume without loss of generality that $F(X) \neq 0$:

$$f(X) = sgn(F(X)) = +1 \text{ if } F(X) > 0, -1 \text{ if } F(X) < 0 \tag{2}$$

and

$$F(X) = \sum_{\alpha \in \{0,1\}^n} w_\alpha X^\alpha \qquad X^\alpha = \prod_{i=1}^{n} x_i^{\alpha_i} \tag{3}$$

We call the weights w_α the spectral coefficients of f. Although they can be real-valued, it is known that restricting them to the rational numbers does not decrease the power of the resulting class of boolean functions (citation needed).

In fact, there is a unique representation of every boolean function f in terms of these coefficients, which we call the *polynomial representation* of f in the monomials which consist of all possible multilinear combinations of the input variables $\{x_1, \ldots, x_n\}$.

It is now possible to define a norm on these spectral coefficients for f, the so-called *spectral norms* from which our lower-bounding method applies. FACT-CHECK: Do these polynomials provide an upper bound or a lower-bound? I think it is a lower-bound. The explicit constructions are the ones that provide an upper bound.

The L_1 norm is defined as follows:

$$L_1(f) = \sum_{\alpha \in \{0,1\}^n} |w_\alpha| \tag{4}$$

The L_2 norm is defined as follows:

$$L_2(f) = \sum_{\alpha \in \{0,1\}^n |w_\alpha|^2} \tag{5}$$

The L_∞ norm is defined as follows:

$$L_\infty(f) = \max_{\alpha \in \{0,1\}^n} |w_\alpha| \tag{6}$$

We can state the following facts

6 Uniqueness of Polynomial Representation

We can in fact determine the spectral coefficients in the polynomial representation directly (and uniquely) from the boolean function on n variables using the $2^n \times 2^n$ Hadamard matrix:

$$A_{2^n} = H_{2^n} p_{2^n} \tag{7}$$

7 Lower Bounding Terms in the Polynomial Representation