Simulating Threshold Circuits by Majority Circuits*

Mikael Goldmann†

Numerical Analysis and Computing Science
Royal Institute of Technology
Stockholm

Marek Karpinski‡

Department of Computer Science, University of Bonn
53117 Bonn
and
International Computer Science Institute
Berkeley, California

Abstract

We prove that a single threshold gate with arbitrary weights can be simulated by an explicit polynomial-size depth 2 majority circuit. In general we show that a polynomial-size, depth-d threshold circuit can be simulated uniformly by a polynomial-size majority circuit of depth $d+1$. Goldmann, Håstad, and Razborov showed in [10] that a non-uniform simulation exists. Our construction answers two open questions posed in [10]: we give an explicit construction whereas [10] uses a randomized existence argument, and we show that such a simulation is possible even if the depth d grows with the number of variables n (the simulation in [10] gives polynomial-size circuits only when d is constant).

Key words. threshold circuits, majority circuits, circuit complexity.

AMS subject classification. 68Q05, 68Q22, 68Q25.

*A preliminary version of this paper appeared in Proc. 25th ACM STOC (1993), pp. 551–560.

†Email: `migo@theory.lcs.mit.edu`. This author's work was done in part while visiting the University of Bonn

‡Email: `marek@cs.bonn.edu`. Supported in part by Leibniz Center for Research in Computer Science, by the DFG Grant KA 673/4–1, and by the ESPRIT BR Grant 7097.

1 Introduction

A threshold gate is a fairly simple device that computes a weighted sum of its inputs and compares it to a threshold value and outputs 1 or 0 depending on the outcome of the comparison. A threshold circuit is an acyclic network of threshold gates. The *size* of a circuit is the number of wires in it.

Small-weight threshold gates are a restricted type of threshold gates. In this case the magnitude of the (integer) weights of the gate is bounded by a polynomial in the number of inputs to the gate. The corresponding circuits are called small-weight threshold circuits. In this case the magnitude of the weights in the circuit is bounded by a polynomial in the total number of inputs to the circuit. It is easy to see that a majority gate can simulate a small weight threshold gate by simply duplicating input wires and adding some constant inputs. This only leads to a polynomial increase in the number of wires. Hence, depth d polynomial-size majority circuits are equivalent to depth d polynomial-size, small weight threshold circuits.

Threshold circuits have been shown to be surprisingly powerful. It is implicit in work by Beame, Cook, and Hoover [4] that integer division can be carried out by polynomial-size threshold circuits of constant depth. Allender [1] (inspired by Toda [29]) shows that any function in AC^0 can be computed by depth 3 majority circuits of quasi-polynomial size. Yao [34] extends this to all of ACC^0 (see also [5]).

There are some strong lower bounds for majority circuits of very small depth. Hajnal et al. [11] prove exponetial lower bounds on the size of depth 2 majority circuits computing "inner product mod 2." These results were extended in [13] to depth 3 majority circuits where the gates on the bottom level have very small fanin, and recently super-polynomial bounds were proved for depth 3 majority circuits where the gates on the bottom level are arbitrary gates of fan-in $n^{1-\epsilon}$ [23]. For depth 3 majority circuits with no extra restrictions, no super-polynomial lower bounds are known. Siu, Roychowdhury, and Kailath proved super-linear lower bounds on the number of wires in constant-depth majority circuits computing parity [28]. Recently Impagliazzo, Paturi, and Saks extended the results of [28] to constant-depth threshold circuits (with large weights) computing parity [15].

If one considers threshold circuits with arbitrary weights, even less is known. There is no super-polynomial lower bound for depth 2 threshold circuits computing some function in NP. As we mentioned above, such bounds exist for majority circuits [11]. It is therefore interesting to explore the power of large weights in threshold circuits.

It has long been known that a single threshold gate is strictly more powerful than a single small-weight threshold gate. For instance, Myhill and Kautz showed in 1961 that there are functions computable by a single threshold gate that require weights of size $\Omega(2^n/n)$, where n is the number of inputs [17]. Also, it was recently shown that depth 2, polynomial-size threshold circuits are more powerful than depth 2, polynomial-size, small-weight threshold circuits [10].

On the other hand, it was proved by Chandra, Stockmeyer, and Vishkin [8] that addition of n binary encoded integers can be performed by constant depth majority circuits (see also [21]). This implies that depth d, polynomial-size threshold circuits can be simulated by depth $O(d)$, polynomial-size, small-weight threshold circuits.

In [25] Siu and Bruck gave a non-constructive proof that polynomial-size, depth-d threshold circuits can be simulated by polynomial-size, depth-$2d+1$ majority circuits. Alon and Bruck gave a uniform construction in [3] achieving this. Goldmann, Håstad, and Razborov showed in [10] that any function computed by a depth d, polynomial-size threshold circuit is computable by a depth $d+1$, polynomial-size majority circuit (note that for $d = 1, 2$ this is optimal). Two open questions were posed in [10]: Can one make an explicit construction, and can one make it

work also for non-constant depth? ([10] uses a probablilistic argument, and the blowup in size is super-polynomial if the depth grows with the number of variables.) We give positive answers to both questions.

For a thourough survey of complexity theoretic results on threshold circuits, see [22].

2 Preliminaries

It will be convenient to work over $\{1,-1\}$ rather than $\{0,1\}$. An n-variable Boolean function thus maps $\{1,-1\}^n$ to $\{1,-1\}$. This is a simple transformation that does not affect the power of threshold gates. A threshold gate computes its output as the sign of a linear form:

$$g(x) = \text{sign}\left(w_0 + \sum_{i=1}^{n} w_i x_i\right),$$

where the w_i are the *weights*, and the sign-function takes a real-valued input and is defined by

$$\text{sign}\,(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Since g is to be ± 1-valued, the argument to the sign function is required to be nonzero for all $x \in \{1,-1\}^n$. The following well-known result by Muroga [16] gives a bound on the magnitude of the weights.

Theorem 1 ([16, Theorem 9.3.2.1]) *Let $f(x)$ be an arbitrary n-variable threshold function. Then $f(x)$ can be written as*

$$f(x) = \text{sign}\left(w_0 + \sum_{1}^{n} w_i x_i\right),$$

where for all $i = 0, \ldots, n$,

$$w_i \in \mathbb{Z} \quad \text{and} \quad |w_i| \leq 2^{-n}(n+1)^{(n+1)/2}.$$

Remark 1 *A recent result by Håstad [12] gives a lower bound on the weights required for a particular threshold function. The lower bound nearly matches the upper bound in Theorem 1.*

A threshold circuit is a directed acyclic graph the nodes of which are either threshold gates or input variables. We allow multiple output nodes, and a circuit is evaluated in topological order. For a given circuit C, f_C is the function computed by C. The size of a circuit is the number of wires in it.

A family of circuits $\{C_n\}$ is said to compute a function $f\colon \{1,-1\}^* \to \{1,-1\}^*$ provided $f\restriction_{\{1,-1\}^n} = f_{C_n}$ for all n.

Definition 1 *We use the following notation from [6].*

- LT_d *is the class of functions computable by depth d polynomial-size threshold circuits.*
- $\widehat{\mathrm{LT}}_d$ *is the class of functions computable by depth d polynomial-size threshold circuits with polynomially bounded integer weights.*

Note that if we allow a gate to have multiple wires from an input (i.e., the underlying structure is a *multi*-graph), then depth d, polynomial-size majority circuits can compute any function in $\widehat{\mathrm{LT}}_d$.

We introduce also uniform classes of threshold circuits. In doing so we can use the threshold circuit descriptions, e.g. similar to the "Direct Connection Language" of Ruzzo [24] (see also [9]).

Let BIN be the set of binary encodings of the positive integers. A threshold circuit C_n with n inputs is described by a labeled multi-graph (V_n, E_n). $V_n = I_n \dot{\cup} G_n$, where $I_n = \{1, x_1, \ldots, x_n\}$ is the set of inputs (note that the constant "1" is included in I_n), and $G_n = \{g_1, \ldots, g_s\}$ is the set of gate-labels. The edges in E_n are tuples $(e, v, g, k, \delta) \in V_n \times BIN \times G_n \times BIN \times \{1, -1\}$ with the following interpretation: the edge with label e goes from v to g (where e is a unique edge-label), and the weight of the edge is $\delta \cdot 2^k$. There is no need to assign a threshold value to the gates in G_n. We give them all threshold 0 and by having an appropriate weight assigned to the constant input 1 any threshold can be achieved. Also, an edge with weight w can be split into about $\log w$ edges the weights of which are powers of 2. We use $\overline{C_n}$ to denote the description of C_n.

We use the same syntax when describing small-weight threshold circuits, but k in an edge-tuple (n, e, v, g, k, δ) is intepreted as the weight having magnitude k rather than 2^k.

A circuit-family has *polynomial-size descriptions* if there is a constant c such that

- For each $n \in \mathbb{N}$ and each $g \in G_n$ it holds that $|g| \leq c(1 + \log(n+1))$.
- For each $n \in \mathbb{N}$ and each edge $(e, v, g, k, \delta) \in E_n$ it holds that $|e| \leq c(1 + \log(n+1))$ and that $|k| \leq c(1 + \log(n+1))$.

Note that given the above constraints both V_n and E_n have size polynomial in n.

To define uniformity, let $\mathcal{C} = \{C_n\}$ be a sequence of circuits, and for each C_n let G_n and E_n be the corresponding gates and edges. $\mathcal{C}$ defines the language $L_{\mathcal{C}} = L_G \cup L_E$, where $L_G = \{\, (n, g) \mid g \in G_n \,\}$ and $L_E = \{\, (n, e, v, g, k, \delta) \mid (e, v, g, k, \delta) \in E_n \,\}$.

Definition 2 *A function is in L–uniform LT_d if there is a sequence $\mathcal{C}$ of depth d threshold circuits with polynomial-size descriptions, such that there is a Turing machine that accepts $L_{\mathcal{C}}$ in linear space.*

A function is in L–uniform $\widehat{LT}_d$ if there is a sequence $\mathcal{C}$ of depth d small-weight threshold circuits with polynomial-size descriptions, such that there Turing machine that accepts $L_{\mathcal{C}}$ in linear space.

The reason for the terminology "L–uniform" is that the description of a circuit of the above type can be generated in space that is logarithmic in the size of the description.

We call a sequence of circuits P–uniform if the description of the nth circuit can be generated in polynomial time on input 1^n (cf., e.g. [9]).

We define the following operator.

Definition 3 *The operator* rem *is defined as follows. For integers a and b,*

$a \text{ rem } b = c,$

where c is the unique integer such that

$a \equiv c \pmod{b}$ and $-b/2 < c \leq b/2.$

3 The Idea Behind the Construction

Let f be an arbitrary threshold gate given by

$$f(x) = \text{sign}\,(F(x))\,,$$

where

$$F(x) = w_0 + \sum_{i=1}^{n} w_i x_i.$$

Since we have integer weights, and we require that the argument of 'sign' is nonzero, we have $|F(x)| \geq 1$ for all x.

In the next section we will build a parametrized *approximator* for f. For a fix input, the approximator is good for randomly chosen parameters. To show the intuition behind the construction, we build an approximator φ for f that computes correctly for a random input.

In the construction it will be convenient to use rational weights. All the weights in the circuits we construct are of the form $w/2$, where w is an integer. By multiplying all weights in the circuit by 2 we get integer weights, and the increase in the magnitude of the weights is just a factor 2.

To describe the construction we need a couple of parameters that we call W and m. Let $w_{max}(f)$ be the largest magnitude of a weight of the gate f. It will be convenient to assume the following:

$$W \geq \max\{w_{max}(f), 2^n\}\,, \tag{1}$$

$$10n \leq m \leq W. \tag{2}$$

The parameter m controls the quality of the approximator. In the circuit that implements the approximator the weights will have magnitude about m^2. To have small-weight circuits, clearly m must be kept small. On the other hand, intutively it seems like allowing larger weights allows a more accurate approximator. The parameter m reflects this trade-off.

The construction uses the following function of one integer variable y.

$$\begin{aligned} M^m(y) \;=\; & \frac{1}{2}\,\text{sign}\left(y - m + \frac{1}{2}\right) \\ & -\frac{1}{2}\,\text{sign}\left(y - 2m + \frac{1}{2}\right) \\ & +\frac{1}{2}\,\text{sign}\left(y + m - \frac{1}{2}\right) \\ & -\frac{1}{2}\,\text{sign}\left(y + 2m - \frac{1}{2}\right) \end{aligned}$$

The function $M^m(y)$ has the following useful property, which is immediate from the definition of M^m.

Lemma 2 *If $m \leq |y| < 2m$ then $M^m(y) = \text{sign}\,(y)$, and otherwise $M^m(y) = 0$.*

Let us look at a fixed input x. Assume that $2^l \leq |F(x)| < 2^{l+1}$. Given this information, we can use less precision in the weights w_i. Set

$$F_{(l)}(x) = \lfloor w_0 m/2^l \rfloor + \sum_{i=1}^{n} \lfloor w_i m/2^l \rfloor x_i.$$

If we disregard the error introduced by the floor operation, we will have

$m \leq |F_{(l)}(x)| < 2m.$

It is plausible that for most inputs x the truncation error will not matter, and then sign $(F(x)) =$ sign $\left(F_{(l)}(x)\right)$. To get the weights small, we just look at $F_{(l)}(x)$ modulo some small prime $p > m^2$.

We are now ready to construct a small gadget $\varphi^{(l)}(x)$ that implements the computation described above.

$$\begin{aligned} w_i^{(l)} &= \lfloor w_i m/2^l \rfloor \text{ rem } p, \\ \Phi_{(l)}(x) &= w_0^{(l)} + \sum_{i=1}^{n} w_i^{(l)} x_i, \\ \varphi_{(l)}(x) &= \sum_{j=-\lfloor (n+1)/2 \rfloor}^{\lceil (n+1)/2 \rceil} M^m \left(\Phi_{(l)}(x) + jp\right). \end{aligned} \tag{3}$$

Let us establish some properties of $\varphi_{(l)}$ as defined by (3).

Proposition 3 *For $p > 4m$ the following holds.*

1. *For any x, $\left|\varphi_{(l)}(x)\right| \in \{0, 1\}$.*
2. *If $\left|F_{(l)}(x) \text{ rem } p\right| \notin [m, 2m)$ then $\varphi_{(l)}(x) = 0$.*
3. *If $m \leq \left|F_{(l)}(x)\right| < 2m$ then $\varphi_{(l)}(x) = \text{sign}\left(F_{(l)}(x)\right)$.*

Proof. All statments follow from Lemma 2.

1. Each term $M^m\left(\Phi_{(l)}(x) + jp\right)$ is either 0, 1, or -1, and since $p > 4m$ there is at most one j for which $\left|\Phi_{(l)}(x) + jp\right| < 2m$, and thus at most one non-zero term $M^m\left(\Phi_{(l)}(x) + jp\right)$.
2. By Lemma 2, $M^m(y) \neq 0$ implies that $|y| < 2m$, and since $p > 4m$ this in turn implies $y = y \text{ rem } p$. Thus, the only term that might be non-zero is th term $M^m\left(\Phi_{(l)}(x) + jp\right)$ for which $\Phi_{(l)}(x) + jp = \Phi_{(l)}(x) \text{ rem } p$. Now just observe that $F_{(l)}(x) \equiv \Phi_{(l)}(x) \pmod p$.
3. For the third statement we have the following. Assume that $m \leq \left|F_{(l)}(x)\right| < 2m$. We then have $\Phi_{(l)}(x) \text{ rem } p = F_{(l)}(x)$.

   Let j_0 be the integer that satisfies $\Phi_{(l)}(x) \text{ rem } p = \Phi_{(l)}(x) + j_0 p$.

   Since $\left|\Phi_{(l)}(x)\right| \leq (n+1)(p-1)/2$, j_0 occurs in the sum in (3). The third statement now follows from Lemma 2. □

It is tempting to set

$$\varphi(x) = \sum_l \varphi_{(l)}(x), \tag{4}$$

and hope that $\varphi(x) = \text{sign}\,(F(x)) = f(x)$. The idea is that there is always an l such that $2^l \leq |F(x)| < 2^{l+1}$. If there was no error introduced by the floor operations, we would have $m \leq \left|F_{(l)}(x)\right| < 2m$ and sign $\left(F_{(l)}(x)\right) =$ sign $(F(x))$. By Proposition 3, $\varphi_{(l)}(x) = \text{sign}\left(F_{(l)}(x)\right)$ and for all other l we would probably have $\varphi_{(l)}(x) = 0$. Then we would have $\varphi(x) = \varphi_{(l)}(x) =$ sign $(F(x)) = f(x)$.

Remark 2 *We know that $|F(x)| \leq (n+1)W$. Thus, it is sufficient to sum over l such that $0 \leq l \leq \log((n+1)W)$ in (4).*

The problems with (4) are of course that the floor operation sometimes introduces *truncation errors* (e.g., when $2^l \leq F(x)$ but $F_{(l)}(x) < m$), and sometimes the "wrong" l gives a nonzero contribution to the sum (a *modular error*). Equation (4) is not such a bad idea though, because it works for most x.

4 The Approximator

To get an approximator based on the ideas of the previous section we want to spread the value $F(x)$ in some random fashion. This was done in [10] by considering sign $(\alpha F(x))$ for a randomly chosen integer $\alpha \in \{1, 2, \ldots, 2^{2n}\}$, and since $\alpha > 0$ one has sign $(F(x)) =$ sign $(\alpha F(x))$. It was shown that for any x, the approximator would compute sign $(\alpha F(x))$ correctly with high probability. By taking many independent α's and taking the sign of the average, one gets an approximator that behaves well on all inputs. If the range of α was smaller, we could get an explicit approximator by taking the average over all α's.

We will modify the construction. Our approximator will depend on two parameters, and for any input, the approximator will be good with high probability if the parameters are chosen at random. In our case the probability space is small enough that we may take the average over all possible choices of parameter values. Just like in [10] we will use a multiplier α, but it will be much smaller. For a random α, the probability of a truncation error is small. To handle the modular errors, we will also choose the prime p, used in the construction, randomly, and the probability that we get a modular error for a random p is small.

Definition 4 *We define the following sets. Let*

$$\begin{aligned} [m] &= \{1, 2, \ldots, m\}, \\ \mathcal{PR}^m &\quad \textit{is the set of the first } m^2 \textit{ primes that are greater than } m^2. \end{aligned}$$

When α and p are chosen randomly, they are picked as a pair $(\alpha, p) \in [m] \times \mathcal{PR}^m$ according to the uniform distribution.

Instead of looking at the weights w_i, we make the approximator for weights αw_i. Thus, instead of looking at $F(x)$ we look at $\alpha F(x)$ but this works since sign $(\alpha F(x)) =$ sign $(F(x)) = f(x)$. We call the corresponding truncated linear form $F^{\alpha}_{(l)}(x)$, that is

$$F^{\alpha}_{(l)}(x) = \lfloor \alpha w_0 m/2^l \rfloor + \sum_{i=1}^{n} \lfloor \alpha w_i m/2^l \rfloor x_i.$$

Now we are ready to define the parametrized approximator $\varphi^{\alpha,p}(x)$.

$$\begin{aligned} w_i^{(l)} &= \left\lfloor \alpha w_i m/2^l \right\rfloor \text{ rem } p \\ \Phi^{\alpha,p}_{(l)}(x) &= w_0^{(l)} + \sum_{i=1}^{n} w_i^{(l)} x_i \\ \varphi^{\alpha,p}_{(l)}(x) &= \sum_{j=-\lfloor (n+1)/2 \rfloor}^{\lceil (n+1)/2 \rceil} M^m(\Phi^{\alpha,p}_{(l)}(x) + jp) \\ \varphi^{\alpha,p}(x) &= \sum_{l=0}^{\lfloor 3 \log W \rfloor + 1} \varphi^{\alpha,p}_{(l)}(x), \end{aligned} \tag{5}$$

where the upper bound of the last summation follows from an argument analogous to Remark 2, the assumptions (1) and (2), and the fact that $\alpha \leq m$.

The following proposition is the (α, p)-version of Proposition 3. The proof is completely analogous.

Proposition 4 *For $p > m^2$ the following holds.*

1. *For any x, $\left|\varphi_{(l)}^{\alpha,p}(x)\right| \in \{0,1\}$.*
2. *If $\left|F_{(l)}^{\alpha}(x) \text{ rem } p\right| \notin [m, 2m)$ then $\varphi_{(l)}^{\alpha,p}(x) = 0$.*
3. *If $m \leq \left|F_{(l)}^{\alpha}(x)\right| < 2m$ then $\varphi_{(l)}^{\alpha,p}(x) = \text{sign}\left(F_{(l)}^{\alpha}(x)\right)$.*

Corollary 5 *If we assume (1) and (2), then for all $x \in \mathbb{Z}^n$ we have $\varphi^{\alpha,p}(x) \in \mathbb{Z}$ and $|\varphi^{\alpha,p}(x)| \leq 2 + 3\log W$.*

Proof. It is not hard to see that statement 1 of Proposition 4 holds even for $x \in \mathbb{Z}^n$. The corollary follows since we sum over at most $2 + 3\log W$ different l in (5). □

For any x there are usually α's and p's such that we get truncation errors or modular errors. We will show that for any fixed x most pairs (α, p) do not give such errors.

Definition 5

1. *We say that α is* bad *for x if there is some l such that $\left||\alpha F(x)| - 2^l\right| \leq 2^{l+1}n/m$. Otherwise α is* good *for x.*
2. *We say that p is* bad *for x and α if there is an l such that $\left|F_{(l)}^{\alpha}(x)\right| \geq 2m$ but $\left|F_{(l)}^{\alpha}(x) \text{ rem } p\right| < 2m$. Otherwise p is* good *for x and α.*
3. *We call a pair (α, p)* bad *for x if α is bad for x, or if p is bad for x and α. Otherwise (α, p) is* good *for x.*

We will show that when a pair (α, p) is good for x, then $\varphi^{\alpha,p}(x) = f(x)$. We will also show that a for any fixed x, a random pair (α, p) is likely to be good.

First we show that $\varphi^{\alpha,p}(x) = f(x)$ when (α, p) is good.

Proposition 6 *Let x be an arbitrary fixed input. If (α, p) is good for x, then $\varphi^{\alpha,p}(x) = f(x)$.*

Proof. Let x be a fixed input and assume that (α, p) is a good pair for x. Without loss of generality we assume that $F(x) > 0$.

First we will show that there are no truncation errors that matter in this case. Let l_0 be the integer such that

$$2^{l_0} \leq \alpha F(x) < 2^{l_0+1}.$$

Since $F(x) \leq (n+1)W$ and $\alpha \leq m$ we have $\alpha F(x) \leq m(n+1)W \leq 2W^3$, and thus $l_0 \leq \lfloor 3\log W \rfloor + 1$.

For any l

$$\left|F_{(l)}^{\alpha}(x) - \frac{\alpha F(x) m}{2^l}\right| < n + 1, \tag{6}$$

since the floor operation makes an error less than one for each weight of the gate.

Equation (6) immediately tells us that

$$m < F^{\alpha}_{(l_0)}(x) < 2m.$$

This implies that the term $\varphi^{\alpha,p}_{(l_0)}(x)$ will be equal to 1.

For $l \neq l_0$ we have the following. Let $l < l_0$ be an arbitrary integer. Since α is good for x we have

$$2^{l+1}\left(1+\frac{2n}{m}\right) < \alpha F(x),$$

and from (6) it follows that

$$2m < F^{\alpha}_{(l)}(x).$$

A completely analogous argument shows that for any $l > l_0$ one has

$$m > F^{\alpha}_{(l)}(x).$$

Thus, no error is introduced due to truncation.

We still need to show that there are no modular errors, that is, that when we do the computation modulo p we do not get a contribution from some $l > l_0$ for which $F^{\alpha}_{(l)}(x) > 2m$. However, this would mean that for som l one has $F^{\alpha}_{(l)}(x) > 2m$ but $\left|F^{\alpha}_{(l)}(x) \text{ rem } p\right| < 2m$, and this implies by definition that p is bad for x and α contrary to our assumption. □

It remains to show that for any fixed x most α's are good, and for any fixed x and α most p's are good.

Lemma 7 *If W and m satisfy (1) and (2), then for any x*

$$\Pr\left[\alpha \textit{ is bad for } x\right] < (16\log^2 W)/m.$$

PROOF. Let $r = |F(x)|$. We want to show that for most α we have $\left|2^l - \alpha r\right| > 2^{l+1}n/m$ for all l.

Look at a fixed l, if there is any α which is bad with respect to l, then r must satisfy the following equations.

$$r \leq 2^l\left(1+\frac{2n}{m}\right) < 2^{l+1}, \tag{7}$$

$$r \geq \frac{2^l}{m}\left(1-\frac{2n}{m}\right) > 2^{l-1}/m. \tag{8}$$

The bad interval for α has length $2^{l+2}n/(rm)$. If r and l do not satisfy (8), then all α's are too small to be bad. On the other hand, if r and l satisfy (8), there are at most $8n$ different α's that fit in the bad interval.

For any r, there are less than $2\log m$ different l that satisfy (7) and (8). Thus, any x has at most $16n\log m$ bad α. The lemma follows by our assumption that (1) and (2) hold. □

Lemma 8 *If W and m satisfy (1) and (2), then for any x and α*

$$\Pr\left[p \textit{ is bad for } x \textit{ and } \alpha\right] < (16\log^2 W)/m.$$

PROOF. Once again consider a fixed l first, and assume that for input x and multiplier α we have $|F^{\alpha}_{(l)}(x)| \geq 2m$ but $\left|F^{\alpha}_{(l)}(x) \text{ rem } p\right| < 2m$ for $4m \log W$ of the primes. By the pigeon hole principle we must have more than $\log W$ primes that give the same remainder k, such that $|k| < 2m$. By the Chinese remainder theorem, $F^{\alpha}_{(l)}(x)$ is uniquely determined modulo the product of those primes. That product is greater than $m^{2\log W}$. Since we assume (1) and (2),

$$\begin{aligned}\left|F^{\alpha}_{(l)}(x)\right| &\leq |m\alpha F(x)| \\ &\leq m^2(n+1)W \\ &\ll m^{2\log W}.\end{aligned}$$

This implies that $k = F^{\alpha}_{(l)}(x)$ which means that $\left|F^{\alpha}_{(l)}(x)\right| < 2m$, and we have a contradiction. Since there are at most $2 + 3\log W < 4\log W$ different l, the lemma follows. □

Lemma 7 and Lemma 8 imply the following lemma.

Lemma 9 *If W and m satisfy (1) and (2), then for an arbitrary fixed input x,*

$\Pr[(\alpha, p)$ *is bad for* $x] \leq (32\log^2 W)/m$.

We conclude this section by showing how the approximator allows us to simulate a single threshold gate (with large weights) by a depth 2, small-weight, threshold circuit. This is a constructive version of Theorem 7.8 from [10].

Theorem 10 $\mathrm{LT}_1 \subsetneq \widehat{\mathrm{LT}}_2$.

PROOF. We will prove the inclusion. That it is strict follows from the fact that parity is in $\widehat{\mathrm{LT}}_2 \setminus \mathrm{LT}_1$. Assume $n \geq 100$, smaller inputs are handled by writing the function in disjuntive normal form.

We are given a threshold gate

$$f(x) = \operatorname{sign}\left(w_0 + \sum_{i=1}^{n} w_i x_i\right).$$

Let $W = \max\{w_{max}(f), 2^n\}$ be the bound used in the construction, and set the parameter $m = 128\log^3 W$ (observe that W and m satisfy (1) and (2)). Consider the following function:

$$g(x) = \operatorname{sign}\left(\sum_{(\alpha,p)} \varphi^{\alpha,p}(x)\right).$$

We claim two things: $f(x) = g(x)$ for all $x \in \{1,-1\}^n$, and $g(x)$ can be computed by a depth 2, polynomial-size, threshold circuit with polynomially bounded weights.

The correctness of the construction follows from

$$\mathrm{E}\left[f(x)\varphi^{\alpha,p}(x)\right] > 0, \tag{9}$$

where x is arbitrary and fixed, and we take expectation over the uniform distribution on pairs (α, p). Let us prove (9).

$$\begin{aligned}&\mathrm{E}\left[f(x)\varphi^{\alpha,p}(x)\right] \\ &\quad\geq \Pr[(\alpha,p) \text{ good}] \\ &\qquad -(2+3\log W)\Pr[(\alpha,p) \text{ bad}] \\ &\quad\geq 1 - \frac{1}{4\log W} - \frac{2+3\log W}{4\log W} \\ &\quad> 0,\end{aligned}$$

where the first inequality follows from Corollary 5 and the second inequality follows from Lemma 9. This shows that $f(x) = g(x)$ for all x.

Now, to implement $g(x)$ as a circuit, let us look at what $g(x)$ does.

$$\begin{aligned} g(x) &= \operatorname{sign}\left(\sum_{\alpha,p,l,j} \frac{1}{2}\operatorname{sign}\left(\Phi_{(l)}^{\alpha,p}(x) + jp - m + \frac{1}{2}\right) - \frac{1}{2}\operatorname{sign}\left(\Phi_{(l)}^{\alpha,p}(x) + jp - 2m + \frac{1}{2}\right)\right. \\ &\quad \left. + \frac{1}{2}\operatorname{sign}\left(\Phi_{(l)}^{\alpha,p}(x) + jp + m - \frac{1}{2}\right) - \frac{1}{2}\operatorname{sign}\left(\Phi_{(l)}^{\alpha,p}(x) + jp + 2m - \frac{1}{2}\right)\right). \end{aligned}$$

$\Phi_{(l)}^{\alpha,p}(x)$ is simply a weighted sum of the inputs. Hence, we only have two levels of sign-functions, and one is the outmost operator, so $g(x)$ can be computed by a depth 2 threshold circuit. The size is the total number of terms in the summations (recall that each $\Phi_{(l)}^{\alpha,p}(x)$ depends on $n \le \log W$ variables). There are $m^3 = O\left(\log^9 W\right)$ pairs (α, p), and $O(\log W)$ different l. This gives a total size of $O(\log^{12} W)$.

As for the weights, they are not necessarily integers, but if we multiply them by two they are. The only weights that are not ± 1 are of the form $2w_j^{(l)}$, $2(w_0^{(l)} + jp \pm m)$, or $2(w_0^{(l)} + jp \pm 2m)$. All the $w_i^{(l)}$ are reduced modulo some prime p that is among the first $2m^2 = O(\log^6 W)$ primes. This implies that $p \le O(\log^7 W)$. Since $-\lfloor (n+1)/2 \rfloor \le j \le \lceil (n+1)/2 \rceil$, the weights are all of magnitude $O(\log^8 W)$.

If we assume Muroga's bound on weights to hold for the original threshold gate f, then $\log W \le O(n \log n)$, and hence we have a polynomial-size, polynomial-weight threshold circuit of depth 2 that computes $f(x)$. Actually, we only need to have $\log W$ polynomial in n. □

5 Extending the Construction to Circuits

In the previous section we showed that a single threshold gate can be simulated by a polynomial-size majority circuit. We will now generalize this to show that a depth d, polynomial-size, threshold circuit can be simulated by a depth $d+1$, polynomial-size, majority circuit.

In [10] Goldmann, Håstad, and Razborov introduced a circuit class that mixes small and large weights.

Definition 6 *$\widetilde{\mathrm{LT}}_d$ is the class of functions computable by depth d, polynomial-size, circuits where the top gate has polynomially bounded weights.*

We will show the following theorem.

Theorem 11 *For any depth d, possibly depending on n, $\widetilde{\mathrm{LT}}_d = \widehat{\mathrm{LT}}_d$.*

Moreover, given a $\widetilde{\mathrm{LT}}_d$ circuit, for input size n, with weights bounded by $2^{p(n)}$ for some polynomial p, there is an explicit $\widehat{\mathrm{LT}}_d$ circuit that computes the same function.

Corollary 12 *For any depth d, possibly depending on n, $\mathrm{LT}_d = \widehat{\mathrm{LT}}_{d+1}$.*

Moreover, given a LT_d circuit, for input size n, with weights bounded by $2^{p(n)}$ for some polynomial p, there is an explicit $\widehat{\mathrm{LT}}_{d+1}$ circuit that computes the same function.

Proof. Make the LT_d circuit into a $\widetilde{\mathrm{LT}}_{d+1}$ circuit by adding a dummy gate with the output-gate of the LT_d circuit as its only input and give this input weight 1. By Theorem 11 the $\widetilde{\mathrm{LT}}_{d+1}$ circuit can be turned into an $\widehat{\mathrm{LT}}_{d+1}$ circuit. □

It remains to prove the theorem.

Proof of Theorem 11. Let us look at a circuit C for an $\widetilde{\mathrm{LT}}_d$-function. For simplicity, assume that the weights at the top gate are all 1. Let the top gate be given by

$$g(x) = \operatorname{sign}\left(\sum_{i=1}^{t} C_i(x)\right),$$

where the C_i are depth $d-1$ threshold circuits. Let s be the size of C. For each circuit C_i we construct an approximator $\Gamma_i^{\alpha,p}$. Below we describe how this is done.

Let $w_{max}(C)$ be the largest magnitude of a weight of the circuit C, and let $W = \max\{w_{max}(C), 2^s\}$. Let C_k be one of the subcircuits of g. Let the gates of C_k be $f_1, \ldots, f_r$, where $r \le s$. The fan-in of any f_i is trivially bounded by $\log W$, and all gates have their weights bounded by W. Set the parameter m of the previous section as follows.

$$m = 256 s\, t \log^3 W.$$

For $(\alpha, p) \in [m] \times \mathcal{PR}^m$, construct a gate approximator $\varphi_i^{\alpha,p}$ for each gate f_i. Now, connect the approximators in the same way as the gates. That is, if the output of f_i is the k'th input to f_j, then the output of $\varphi_i^{\alpha,p}$ is the k'th input to $\varphi_j^{\alpha,p}$. We call the constructed "circuit" $\Gamma_k^{\alpha,p}$. Note that for any fixed input x, if (α, p) is simultaneously good for x in all $\varphi_i^{\alpha,p}$ of $\Gamma_k^{\alpha,p}$, then $\Gamma_k^{\alpha,p}(x) = C_k(x)$.

Lemma 13 *For any fixed input x, if (α, p) is chosen uniformly at random from $[m] \times \mathcal{PR}^m$, then* $\Pr\left[\Gamma_k^{\alpha,p}(x) \neq C_k(x)\right] \le (8t \log W)^{-1}$.

Proof. We know that each gate f_i of C_k, has fan-in bounded by $\log W$, and will give the correct output if (α, p) is good. By Lemma 9, the probability that (α, p) is bad is $(8s\, t \log W)^{-1}$ by the choice of m. Since there are at most s gates in C_k, the lemma follows. □

We say that (α, p) is good for x and $\Gamma_k^{\alpha,p}$, if it is good for all gate approximators $\varphi_i^{\alpha,p}$ of $\Gamma_k^{\alpha,p}$ simultaneously.

What happens when (α, p) is bad for $\Gamma_k^{\alpha,p}$? The magnitude of the output of $\Gamma_k^{\alpha,p}$ is bounded by the maximum output of the approximator of the top gate of C_k. Using Corollary 5, we have for all x and all (α, p),

$$\left|\Gamma_k^{\alpha,p}(x)\right| \le 2 + 3 \log W. \tag{10}$$

Since the corollary holds even when the inputs to the top gate are integers, this bound holds even when some other approximator in $\Gamma_k^{\alpha,p}$ fails.

If we combine Lemma 13 and (10) we get

Lemma 14 *For any fixed input x, if (α, p) is chosen uniformly at random from $[m] \times \mathcal{PR}^m$, then* $\left|\mathrm{E}\left[\Gamma_k^{\alpha,p}(x)\right] - C_k(x)\right| \le \frac{1}{2t}$.

Do the same for all the circuits C_i to get approximators $\Gamma_i^{\alpha,p}$. Consider the following function:

$$h(x) = \operatorname{sign}\left(\sum_{i=1}^{t} \sum_{(\alpha,p)} \Gamma_i^{\alpha,p}(x)\right).$$

Observe that this is equivalent to

$$h(x) = \operatorname{sign}\left(\mathrm{E}\left[\sum_{i=1}^{t} \Gamma_i^{\alpha,p}(x)\right]\right). \tag{11}$$

We claim the following.

Proposition 15 *For all inputs x, $h(x) = g(x)$.*

PROOF. By(11) and the fact that $\left|\sum_{i=1}^{t} C_i(x)\right| \geq 1$ it is sufficient to show that

$$\left|\mathrm{E}\left[\sum_{i=1}^{t} \Gamma_i^{\alpha,p}(x)\right] - \sum_{i=1}^{t} C_i(x)\right| \leq 1/2.$$

This inequality follows by straightforward application of Lemma 14. □

An argument analogous to that in the previous section shows that $g(x)$ can be implemented as a depth d threshold circuit. We call the circuit $\widehat{C}$. It is not hard to see that each wire in C corresponds to a polynomially (in $\log W$) many wires in $\widehat{C}$.

To get integer weights in the circuit it is sufficient to multiply all weights by 2. The weights all have magnitude bounded by order of the largest prime used multiplied by s. The largest prime in $\mathcal{PR}^m$ has size $O(m^2 \log m)$, so the weights are all $O(s^3 t^2 \log^6 W(\log(st) + \log\log W))$. This completes the proof of Theorem 11.

Just as before, the construction is polynomial in n as long as $\log W$ is polynomial in n. By Theorem 1, this is always possible. □

6 Uniform Computations

In addition to being explicit, the constructions given in this paper also preserve the uniformity conditions of the classes of circuits. The constructions of Section 4 and 5 yield the following.

Theorem 16 *Given any q–uniform class of threshold circuits $\{C_n\}$ of depth d and polynomial size, $q \in \{L, P\}$. There exists an algorithm running in logspace that given a description of a circuit $\overline{C}_n$ of depth d and polynomial size will output the description of a small-weight threshold circuit $\overline{C}'_n$ of depth $d+1$ and a polynomial size such that $f_{C_n} = f_{C'_n}$ for all $n \geq 0$.*

We have also the following corollary:

Corollary 17

$$\begin{aligned} L\text{–uniform } LT_d &\subseteq L\text{–uniform } \widehat{LT}_{d+1} \\ P\text{–uniform } LT_d &\subseteq P\text{–uniform } \widehat{LT}_{d+1}. \end{aligned}$$

□

PROOF. For P–uniformity the proof is very simple: generate the large-weight circuit and then translate it to a small-weight circuit using Theorem 16.

To prove the result for L–uniformity we use the following lemma.

Lemma 18 *A family of circuits is L–uniform if and only if there is a Turing machine that on input n outputs $\overline{C}_n$ using space $O(\log n)$.*

PROOF. Assume that we have an L–uniform family of circuits $\mathcal{C}$, that is, we have a machine M that recognizes the language $L_{\mathcal{C}}$. Let $\overline{C}_n$ be the description of the circuit for input size n.

There is are constants c and c' such that each $(n, g) \in L_G$ has at most $c(1+\log(n+1))$ bits, and each $(n, e, v, g, k, \delta) \in L_E$ has at most $c'(1+\log(n+1))$ bits. Consider the machine M' that given n runs through all possible tuples with n as the first component and outputs those that belong to $L_{\mathcal{C}}$. This will output $\overline{C}_n$. The space required is $O(\log n)$ plus the space used by M

and the time is polynomial in n times the time used by M. Thus, we have proved the "only if" part of the lemma.

Next we consider the case when there is a machine M that on input n generates $\overline{C}_n$. Let us construct a machine M' that works as follows. When given an input tuple, say (n, g) for instance, it checks if $|(n, g)| > c(1 + \log(n + 1))$ and if so it rejects, otherwise it runs M on input n and checks the output of M "on-line" and if (n, g) is produced then M' accepts, otherwise it rejects. Clearly, M' recognizes $L_{\mathcal{C}}$. Also, M uses space $O(\log n)$ plus the space used by M and time polynomial in n plus the time used by M. We have now proved the "if" part of both statements and the proof of the lemma is complete. □

The following argument completes the proof of the corollary. If a family of (large-weight) threshold circuits is L–uniform, then by the lemma there is a machine that generates $\overline{C}_n$ in space $O(\log n)$. By Theorem 16 this description can be translated to a description of an equivalent small-weight circuit in space $O(\log n)$. Applying the lemma again, in the other direction, we see that the family of small-weight circuits is also L–uniform. □

It remains to prove Theorem 16. Below we explain how, given a polynomial-size description of a large-weight threshold circuit, one can construct a description of an equivalent small-weight threshold circuit with one extra level of gates. It will be clear that all the necessary computations can be carried out in space $O(\log n)$. In order to conform with the notation used so far, we allow weights of the form $w/2$ where w is an integer. However, as stated before, if all weights in the circuit are multiplied by two, then the circuit has integral weights.

In what follows
α ranges over $[m]$,
p ranges over $\mathcal{PR}^m$,
l ranges from 0 to $\lfloor 3 \log W \rfloor + 1$,
j ranges from $-\lfloor (n+1)/2 \rfloor$ to $\lceil (n+1)/2 \rceil$,
β ranges over $\{-1, 1\}$,
γ ranges over $\{1, 2\}$
g is a gate label in a large-weight circuit,
and e is an edge label in a large-weight circuit.
The parameters α through γ correspond to the threshold gates in an approximator for a large-weight gate, since the approximator can be written on the following form (for fixed α and p):

$$\sum_{l,j,\beta,\gamma} \frac{3 - 2\gamma}{2} \operatorname{sign}\left(\Phi_{(l)}^{\alpha,p} + jp + \beta\gamma m - \frac{\gamma}{2}\right). \tag{12}$$

Note that the summation over β and γ corresponds to the four terms in the function M^m. The idea now is that for each (large-weight) gate in the original circuit construct the collection of gates in (12), and for each edge between two gates in the original circuit construct and edges between the corresponding sets of small-weight gates in the new (small-weight) circuit.

We use g and e for gate labels and edge labels of the large-weight circuit, and we use $\hat{g}$ and $\hat{e}$ to denote labels in the small-weight circuit. To describe the construction it is convenient to define *labeling functions*. We have functions $\lambda_g(n, g, \alpha, p, l, j, \beta, \delta)$ (to generate new gate labels), $\lambda_e(n, e, \hat{g}', \hat{g})$ (to generate new edge labels), $\lambda_1(n, \hat{g})$, and $\lambda_t(n, \hat{g})$ (which also generate edge labels). We also use $\hat{g}_t$ as the label of the top gate of the new circuit. We will assume that

1. the functions are linear space computible (i.e., space $O(\log n)$),

2. the size of the output is linear in the size of the input (i.e., size $O(\log n)$),

3. the functions λ_e, λ_1, and λ_t have pairwise disjoint images,
4. the set I_n is disjoint from the image of λ_g,
5. $\hat{g}_t$ is not in I_n or in the image of λ_g,
6. the functions are one-to-one.

Such functions are easy to find and we will not concern ourselves with the exact implementation of them.

The translation works as follows:

1. Compute the size s of the large-weight circuit and find the largest k-value of any edge and call this k_{max}.
2. Compute $L = \max\{s, k_{max}\}$ (note that $W = 2^L$).
3. Compute $m = 256sL^3$.
4. Output $(n, \hat{g}_t)$.
5. For each gate (n, g) we output a collection of gates that correspond to the threshold functions of (12) for each value of α and p:

```
for each (α,p) ∈ [m] × PR^m
    for each l ∈ {0,1,...,3L+1}
        for each j ∈ {−⌊(n+1)/2⌋,...,⌈(n+1)/2⌉}
            for each (β,γ) ∈ {−1,1} × {1,2}
                ĝ ← λ_g(n,g,α,p,l,j,β,γ)
                output (n,ĝ)
                ê ← λ_1(n,ĝ)
                w ← jp + βγm − β/2
                output (n,ê,1,ĝ,|w|,sign(w))
```

6. For each edge (n, e, v, g, k, δ) construct a collection of new edges. How this is done depends on whether v is an input or a gate.

   If v is an input, then it must be fed, with the appropriate weight, to all of the gates that are constructed from g in the previous step.

```
for each (α,p) ∈ [m] × PR^m
    for each l ∈ {0,1,...,3L+1}
        for each j ∈ {−⌊(n+1)/2⌋,...,⌈(n+1)/2⌉}
            for each (β,γ) ∈ {−1,1} × {1,2}
                ĝ ← λ_g(n,g,α,p,l,j,β,γ)
                ê ← λ_e(n,e,v,ĝ)
                w ← ⌊αγδm2^(k−l)⌋ rem p
                output (n,ê,v,ĝ,|w|,sign(w))
```

   If, on the other hand, we have an edge (n, e, g', g, k, δ) between two gates, the situation is quite similar, except that many of the gates that g' gives rise to are connected to many of the gates that g gives rise to.

```
for each (α,p) ∈ [m] × PR^m
    for each (l,l') ∈ {0,1,...,3L+1}^2
        for each (j,j') ∈ {−⌊(n+1)/2⌋,...,⌈(n+1)/2⌉}^2
            for each (β,β',γ,γ') ∈ {−1,1}^2 × {1,2}^2
                ĝ ← λ_g(n,g,α,p,l,j,β,γ)
                ĝ' ← λ_g(n,g',α,p,l',j',β',γ')
                ê ← λ_e(n,e,ĝ',ĝ)
                w ← ⌊α(3−2γ)γδm2^(k−l−1)⌋ rem p
                output (n,ê,ĝ',ĝ,|w|,sign(w))
```

7. Finally, let g_t be the top gate of the original circuit. Each gate that is produced from g_t in step 5 is to be fed with weight 1 to $\hat{g}_t$, the top gate of the new circuit.

```
for each (α,p) ∈ [m] × PR^m
    for each l ∈ {0,1,...,3L+1}
        for each j ∈ {−⌊(n+1)/2⌋,...,⌈(n+1)/2⌉}
            for each (β,γ) ∈ {−1,1} × {1,2}
                ĝ ← λ_g(n,g_t,α,p,l,j,β,γ)
                ê ← λ_t(n,ĝ)
                output (n,ê,ĝ,ĝ_t,1,1)
```

Except for the calculation of w in step 6 it is immediately clear that the computations involved can be carried out in space $O(\log n)$. This is true also for the weight computation. For $\lfloor \alpha\gamma\delta m 2^{k-l} \rfloor$ rem p, note that $\alpha\gamma m$ can be represented with $O(\log n)$ bits, and that when $k < l$ then so can $\lfloor \alpha\gamma\delta m 2^{k-l} \rfloor$. When $k \geq l$, then $\alpha\gamma\delta m 2^{k-l}$ is an integer, and computing 2^{k-l} rem p is easily done in space $O(p+k+l) = O(\log n)$.

7 Conclusions and Open Problems

Our results entail the first explicit constructions for the optimal depth, polynomial-size majority circuits for the number of basic functions including, among others, *powering* (depth 3), *integer multiplication* and *integer division* (depth 3), see [27] and [4].

More generally, our results entail the *uniformity* of the classes of majority circuits simulating the corresponding classes of *threshold circuits*. We look at the following functions.

ADDITION: given two n-bit numbers, compute their sum.

MULTIPLE ADDITION: given n n-bit numbers, compute their sum.

MULTIPLICATION: given two n-bit numbers, compute their product.

MULTIPLE MULTIPLICATION: given n n-bit numbers, compute their product.

DIVISION: given a $2n$-bit numbers x and an n-bit number y, compute $\lfloor x/y \rfloor$.

SQUARING: given an n-bit number x, compute x^2.

POWERING: given an n-bit number x and a $\log n$-bit number y, compute x^y.

COMPARISON: given two n-bit numbers x and y, decide if $x \geq y$.

MAXIMUM: given n n-bit numbers, output the largest one.

SORTING: given n n-bit numbers, output them in sorted order.

The following table surveys the uniform upper bounds known before and stemming from the present paper and compares it with the best known (nonuniform) lower bounds. Most of the constructions follow from non-uniform versions in [27] and [26] (see also [22]). The uniform constructions for ADDITION,COMPARISON, and SORTING follow from [3].

Function	Uniform Depth Upper Bound	Lower Bound
ADDITION	2,L–uniform	2
MULTIPLE ADDITION	2,L–uniform	2
MULTIPLICATION	3,L–uniform	3 [11]
MULTIPLE MULTIPLICATION	4,L–uniform	3 [11]
DIVISION	3,P–uniform	3 [33]
SQUARING	3,L–uniform	3 [33]
POWERING	3,P–uniform	2 [33]
COMPARISON	2,L–uniform	2 [25]
MAXIMUM	3,L–uniform	2
SORTING	3,L–uniform	3 [26]

We conclude with the list of open problems:

1. If the original circuit is monotone, our construction yields a non-monotone majority circuit. It is an open question if an arbitrary monotone threshold gate can be simulated by constant depth monotone majority circuits of polynomial size.

2. Alternating Turing machines are closely connected to circuits with AND-gates and OR-gates, and counting Turing machines are connected to majority circuits. Is there a reasonable machine model that has such a relationship to threshold circuits? If so, what would be the corresponding notion of uniformity, and would our simulation still work?

3. Any strong lower bounds for depth two threshold circuits or depth three majority circuits computing some explicit function?

Acknowledgement

We are grateful to Johan Håstad and Ingo Wegener for many valuable comments on earlier versions of this paper. We also thank Jens Lagergren, Sasha Razborov, and Avi Wigderson for several helpful discussions on the topic of this paper. Finally, we thank the referees for careful reading of the paper and numerous helpful suggestions and comments.

References

[1] E. Allender. A note on the power of threshold circuts. In *Proc. 30th IEEE Symposium on Foundations of Computer Science*, pages 580–584, 1989.

[2] N. Alon and R. B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7:1–22, 1987.

[3] N. Alon and J. Bruck. Explicit constructions of depth-2 majority circuits for comparison and addition. Technical Report RJ 8300 (75661), IBM Research Division, August 1991. To appear in *SIAM J. Disc. Math.*

[4] P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM J. Comp.*, 15:994–1003, 1986.

[5] R. Beigel and J. Tarui. On ACC. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 783–792, 1991.

[6] J. Bruck. Harmonic analysis of polynomial threshold functions. *SIAM J. Disc. Math.*, 3(2):168–177, May 1990.

[7] J. Bruck and R. Smolensky. Polynomial threshold functions, AC^0 functions and spectral norms. In *Proc. 31st IEEE Symposium on Foundations of Computer Science*, pages 632–641, 1990.

[8] A. Chandra, L. Stockmeyer, and U. Vishkin. Constant depth reducibility. *SIAM J. Comp.*, 13:423–439, 1984.

[9] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[10] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[11] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. In *Proc. 28th IEEE Symposium on Foundations of Computer Science*, pages 99–110, 1987.

[12] J. Håstad. On the size of weights for threshold gates. *SIAM. J. Disc. Math.*, 7:484–492, 1994.

[13] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

[14] T. Hofmeister and P. Pudlak. A Proof that Division Is Not in TC_2^0. *Research Report No. 447*, Dept. of Computer Science, Univ. of Dortmund, 1992.

[15] R. Impagliazzo, R. Paturi, and M. Saks. Size-depth trade-offs for threshold circuits. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 541–550, 1993.

[16] S. Muroga. *Threshold Logic and its Applications.* Wiley-Interscience, 1971.

[17] J. Myhill and W. H. Kautz. On the size of weights required for linear-input switching functions. *IRE Trans. on Electronic Computers*, EC-10(2):288–290, 1961.

[18] P. Orponen. Neural networks and complexity theory. In *Proc. 17th International Symposium on Mathematical Foundations of Computer Science*, pages 50–61. Springer-Verlag, 1992. Lecture Notes in Computer Science 629.

[19] I. Parberry. A primer on the complexity theory of neural networks. In *Formal Techniques in Artificial Intelligence: A Sourcebook.* Elsevier, 1990.

[20] I. Parberry and G. Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, 36(3):278–302, June 1988.

[21] N. Pippenger. The complexity of computation by networks. *IBM J. of Research and Development*, 31(2):235–243, March 1987.

[22] A. A. Razborov. On small depth threshold circuits. In *Proc. 3rd Scandinavian Workshop on Algorithm Theory*, pages 42–52. Springer-Verlag, 1992. Lecture Notes in Computer Science 621.

[23] A. A. Razborov and A. Wigderson. $n^{\Omega(\log n)}$ lower bounds on the size of depth 3 threshold circuits with AND gates at the bottom. *Information Processing Letters*, 45(6):303–307, 1993.

[24] W. L. Ruzzo. On uniform circuit complexity. *J. Computer and System Sciences*, 22:365–383, 1981.

[25] K. Y. Siu and J. Bruck. On the power of threshold circuits with small weights. *SIAM J. Disc. Math.*, 4:423–435, 1991.

[26] K. Y. Siu, J. Bruck, T. Kailath, and T. Hofmeister. Depth-efficient neural networks for division and related problems. Technical Report RJ 7946, IBM Research Division, January 1991. *IEEE Trans. Information Theory*, 39:946–956, 1993.

[27] K. Y. Siu and V. Roychowdhury. On optimal depth threshold circuits for multiplication and related problems. *SIAM J. Disc. Math.*, 7:284–292, 1994.

[28] K. Y. Siu, V. Roychowdhury, and T. Kailath. Computing with optimal size threshold circuits. Technical Report, Stanford University, 1990.

[29] S. Toda. On the computational power of PP and $\oplus P$. In *Proc. 30th IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[30] J. Torán. An oracle characterization of the counting hierarchies. In *Proc. 3rd Annual Structure in Complexity Theory Conference*, pages 213–223, 1988.

[31] J. Torán. A combinatorial technique for separating counting complexity classes. In *Proc. 16th International Colloquium on Automata, Languages and Programming*, pages 732–744. Springer-Verlag, 1989. Lecture Notes in Computer Science 372.

[32] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

[33] I. Wegener. Optimal Lower Bounds on the Depth of Polynomial Size Threshold Circuits for some Arithmetic Functions. *Information Processing Letters*, 46:85–87, 1993.

[34] A. C. Yao. On ACC and threshold circuits. In *Proc. 31st IEEE Symposium on Foundations of Computer Science*, pages 619–627, 1990.