q

Simulating Threshold Circuits by Majority Circuits

Mikael Goldmann* Marek Karpinski †

TR-92-080

December 1992

Abstract

We prove that a single threshold gate can be simulated by an explicit polynomial size depth 2 majority circuit. In general we show that a depth d threshold circuit can be simulated uniformly by a majority circuit of depth $d+1$. Goldmann, Håstad and Razborov showed in [9] that a non-uniform simulation exists. Our construction answers two open questions posed in [9]: we give an explicit construction whereas [9] 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 [9] gives polynomial size circuits only when d is constant).

*Dept. of Computer Science, Royal Institute of Technology, S-100 44 Stockholm. Email: migo@nada.kth.se. Part of this work was done while visiting the Department of Computer Science at University of Bonn.

†Department of Computer Science, University of Bonn, 5300 Bonn 1, and International Computer Science Institute, Berkeley, California. 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 input 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.

Small weight threshold gates are a restricted type of threshold gates. In this case the magnitude of the (integer) weights of the gate is polynomially bounded. The corresponding circuits are called small weight threshold circuits. 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 [23]) shows that any function in can be computed by depth three majority circuits of quasi-polynomial size. Yao [27] extends this to all of (see also [5]).

There are some strong lower bounds for majority circuits of very small depth. Hajnal [10] prove exponetial lower bounds on the size of depth two majority circuits computing "inner product mod 2." These results were extended in [12] to depth three majority circuits where the gates on the bottom level have very small fanin, and recently superpolynomial bounds were proved for depth three majority circuits where the gates on the bottom level are arbitrary gates of fan-in $n^{1-\epsilon}$ [19]. For depth three majority circuits with no extra restrictions, no super-linear lower bounds are known.

If one considers threshold circuits with arbitrary weights, even less is known. There is no super-linear lower bound for depth two threshold circuits computing some function in . As we mentioned above, such bounds exist for majority circuits [10]. It is therefore interesting to explore the power of large weights in threshold circuits.

It is well known that a single threshold gate is strictly more powerful than a single small-weight threshold gate. Also, it was recently shown that depth 2, polynomial-size threshold circuits are more powerful than depth 2, polynomial-size, small-weight threshold circuits [9].

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 [17]). This implies that depth d, polynomial-size threshold circuits can be simulated by depth $O(d)$, polynomial-size, small-weight threshold circuits.

In [20] Siu and Bruck gave a non-constructive proof that depth d threshold circuits can be simulated by depth $2d+1$ majority circuits. Alon and Bruck gave a uniform construction in [3] achieving this. Goldmann, Håstad, and Razborov showed in [9] 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 [9]: Can one make an explicit construction, and can one make it work also for non-constant depth? ([9] uses a probablilistic argument, and the blowup in size is super-polynomial if the depth depends on the number of variables.) We give positive answers to

both questions.

For a thourough survey of complexity theoretic results on threshold circuits, see [18]

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, and does not affect the power of threshold gates. A threshold gate computes its output as the sign of a linear form:

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

where the w_i are the *weights*. We assume that the argument to the sign function is nonzero for all $x \in 1, -1^n$. The following well known result by Muroga [13] gives a bound on the magnitude of the weights. [[13, Theorem 9.3.2.1]] Let $f(x)$ be an arbitrary n-variable threshold function. Then $f(x)$ can be written as

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

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

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

A recent result by Håstad [11] gives a lower bound on the weights required for a particular threshold function. The lower bound nearly matches the upper bound in Theorem 2. The size of a circuit is the number of wires.

We use the following notation from [6].

- $[d]$ is the class of functions computable by depth d polynomial-size threshold circuits.
- $[d]$ is the class of functions computable by depth d polynomial-size threshold circuits with polynomially bounded integer weights.

We define the following operator. The operator is defined as follows. For integers a and b,

$$ab = c,$$

where c is the unique integer such that

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

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

$$\begin{aligned} {}_j(y) &= \frac{1}{2}y + jp - + \frac{1}{2} - \frac{1}{2}y + jp - 2 + \frac{1}{2} \\ &\quad + \frac{1}{2}y + jp + - \frac{1}{2} - \frac{1}{2}y + jp + 2 - \frac{1}{2} \end{aligned}$$

The function $_j(y)$ has some nice properties. If $p > 4$ then for any integer y the following is true.

If $y + jp = yp$ and $\leq yp < 2$, then $_j(y) = yp$. In all other cases $_j(y) = 0$. It is easy to see that the first two terms contribute 1 if $\leq y + jp < 2$ and 0 otherwise. Similarly, the last two terms contribute -1 if $-2 < y + jp \leq -$ and 0 otherwise. This implies that we can get a nonzero contribution only when $\leq y + jp < 2$.

If $y + jp \neq yp$ then $y + jp \geq p/2 \geq 2$, and thus $_j(x) = 0$. This proves the first part of the lemma.

The second part of the lemma follows by the choice of j_0.

If $yp \notin [, 2)$ then for all j we have $y - jp \notin [, 2)$, and thus $_j(y) = 0$ for all j.

3 The Idea Behind the Construction

Let f be an arbitrary threshold gate given by

$$f(x) = F(x),$$

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 '' 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 fixed input, the approximator is good for randomly chosen parameters. To show the intuition behind the construction, we build an approximator for f that computes correctly for a random input.

In the construction it will be convenient to use rational weights. Any circuit we construct will have weights 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 . It will be convenient to assume the following:

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

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

The parameter controls how good the approximator is. If is big, then the approximator is good, but the weights in the approximator are polynomial in .

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

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

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

$$\leq |(x)| < 2.$$

It is plausible that for most values of $F(x)$, the truncation error will not matter. Note that $F(x) = (x)$. To get the weights small, we just look at (x) modulo some small prime $p >^2$.

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

$$\begin{aligned} w_i^{(l)} &= \lfloor w_i/2^l \rfloor p, \\ (x) &= w_0^{(l)} + \sum_{i=1}^{n} w_i^{(l)} x_i, \\ (x) &= \sum_{j=-n-1}^{n+1} {}_j(x). \end{aligned} \tag{3}$$

Let us establish some properties of as defined by (3). For $p >^2$ the following holds.

1. For any x, $(x) \in 0, 1$.
2. If $(x)p \notin [, 2)$ then $(x) = 0$.
3. If $\leq (x) < 2$ then $(x) = (x)$.

All statments follow from Lemma 2. The first one is immediate. The second follows by the observation that $(x) \equiv (x) \pmod{p}$.

For the third statement we have the following. Assume that $\leq (x) < 2$. We then have

$$(x)p = (x).$$

Let j_0 be the integer that satisfies

$$(x)p = (x) + j_0 p.$$

Since $(x) \leq (n+1)(p-1)/2$, j_0 occurs in the sum in (3). The third statement now follows from Lemma 2.

It is tempting to set

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

and hope that $(x) = F(x) = f(x)$. 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 $\leq (x) < 2$ and $(x) = F(x)$. By Proposition 3, $(x) = (x)$ and for all other l we would probably have $(x) = 0$. Then we would have $(x) = (x) = F(x) = f(x)$. 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* (when $2^l \leq F(x)$ but $(x) <$), 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, or rather, for most values of $F(x)$.

4 The Approximator

To get an approximator based on the ideas in the previous section we want to spread the value $F(x)$ in some random fashion. This was done in [9] by considering $\alpha F(x)$ for a randomly chosen integer $\alpha \in \{1, 2, \ldots, 2^{2n}\}$. Since $\alpha > 0$ one has $F(x) = \alpha F(x)$. It was shown that for any x, the approximator would compute $\alpha F(x)$ 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.

We will modify the construction. Just like in [9] 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. We define the following sets. Let

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

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 $\alpha F(x) = F(x) = f(x)$. We call the corresponding truncated linear form (x), that is

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

Now we are ready to define the parametrized approximator (x).

$$\begin{aligned} w_i^{(l)} &= \left\lfloor \alpha w_i / 2^l \right\rfloor p \\ (x) &= w_0^{(l)} + \sum_{i=1}^{n} w_i^{(l)} x_i \\ (x) &= \sum_{j=-n-1}^{n+1} {}_j((x)) \\ (x) &= \sum_{l=0}^{\lfloor 3 \log W \rfloor + 1} (x), \end{aligned} \tag{5}$$

where the upper bound of the last summation follows from an argument analogous to Remark 3, the assumptions (1) and (2), and the fact that $\alpha \leq$.

The following proposition is the (α, p)-version of Proposition 3. The proof is completely analogous. For $p >^2$ the following holds.

1. For any x, $(x) \in 0, 1$.
2. If $(x)p \notin [, 2)$ then $(x) = 0$.
3. If $\leq (x) < 2$ then $(x) = (x)$.

If we assume (1) and (2), then for all $x \in^n$ we have $(x) \in$ and $(x) \leq 2 + 3 \log W$. It is not hard to see that statement 1 of Proposition 4 holds even for $x \in^n$. 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.

1. We say that α is *bad* for x if there is some l such that $\alpha F(x) - 2^l \leq 2^{l+1} n/$. Otherwise α is *good* for x.
2. We say that p is *bad* for x and α if there is an l such that $(x) \geq 2$ but $(x)p < 2$. 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 $(x) = f(x)$. We will also show that a for any fixed x, a random pair (α, p) is likely to be good.

First we show that $(x) = f(x)$ when (α, p) is good. Let x be an arbitrary fixed input. The following is true. if (α, p) is good for x, then $(x) = f(x)$. [(Sketch)] If (α, p) is good for x then, by the definition of "good," there will not be any truncation errors or modular errors. Thus, the only contribution we get in (5) is the contribution from the "right" l, which is $\alpha F(x)$.

We will show that for any fixed x most α's are good, and for any fixed x and α most p's are good. If W and satisfy (1) and (2), then for any x

$$\alpha \text{ is bad for } x < (16 \log^2 W)/.$$

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

$$r < 2^{l+1}, \tag{6}$$
$$r > 2^{l-1}/. \tag{7}$$

The bad interval for α has length $2^{l+2} n/(r)$. If r and l do not satisfy (7), then all α's are too small to be bad. On the other hand, if r and l satisfy (7), there are at most $8n$ different α's that fit in the bad interval.

For any r, there are less than $2 \log$ different l that satisfy (6) and (7). Thus, any x has at most $16n \log$ bad α. The lemma follows by our assumption that (1) and (2) hold.

If W and satisfy (1) and (2), then for any x and α

$$p \text{ is bad for } x \text{ and } \alpha < (16 \log^2 W)/.$$

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

$$(x) \leq \alpha F(x) \leq^2 (n+1)W \ll^{2 \log W} .$$

This implies that $k = (x)$ which means that $(x) < 2$, and we have a contradiction. Since there are at most $2 + 3 \log W < 4 \log W$ different l, the lemma follows.

Lemma 4 and Lemma 4 imply the following lemma. If W and satisfy (1) and (2), then for an arbitrary fixed input x,

$$(\alpha, p) \text{ is bad for } x \leq (32 \log^2 W)/.$$

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 [9]. [2]. We will prove the inclusion. That it is strict follows from the fact that parity is in [2]\. Assume $n \geq 100$, smaller inputs are handled by writing the function on disjuntive normal form.

We are given a threshold gate

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

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

$$g(x) = \sum_{(\alpha,p)} (x).$$

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

$$f(x)(x) > 0, \tag{8}$$

where x is arbitrary and fixed, and we take expectation over the uniform distribution on pairs (α, p). Let us prove (8).

$$\begin{aligned} f(x)(x) &\geq (\alpha, p) \text{ good} - (2 + 3 \log W)(\alpha, p) \text{ bad} \\ &\geq 1 - \frac{1}{4 \log W} - \frac{2 + 3 \log W}{4 \log W} \\ &> 0, \end{aligned}$$

where the first inequality follows from Corollary 4 and the second inequality follows from Lemma 4. This shows that $f(x) = g(x)$ for all x.

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

$$g(x) = \sum_{(\alpha,p)} \sum_{l} \sum_{j} \frac{1}{2} F^{(l)}_{(\alpha,p)}(x) \cdot \cdots \cdots .$$

Since we only have two levels of sign-functions, and one is the outmost operator, $g(x)$ can be computed by a depth 2 threshold circuit. The size is the total number of terms in the summations (recall that each $F^{(l)}_{(\alpha,p)}(x)$ has fan-in $n \leq \log W$). There are $m^3 = O\log^9 W$ 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 the $w^{(l)}_1, \ldots, w^{(l)}_n$ and $w^{(l)}_0 + jp$ weights. All the $w^{(l)}_i$ are reduced modulo some prime p that is among the first $2^2 = O(\log^6 W)$ primes. This implies that $p \leq O(\log^7 W)$. Since $-(n+1) \leq j \leq n+1$, 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 $\log W \leq O n \log n$, 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 [9] Goldmann, Håstad, and Razborov introduced a circuit class that mixes small and large weights. $[d]$ 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. For any depth d, possibly depending on n, $[d] = [d]$.

Moreover, given a $[d]$ circuit, for input size n, with weights bounded by $2^{p(n)}$ for some polynomial p, there is an explicit $[d]$ circuit that computes the same function. Let us look at a circuit C for an $[d]$-function. For simplicity, assume that the weights at the top gate are all 1. Let the top gate be given by

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

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 $_k$. Below we describe how this is done.

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 $r \leq s$. 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 of the previous section as follows.

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

For $(\alpha, p) \in [] \times \mathcal{PR}$, construct a gate approximator $_i$ 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 input to f_j, then the output of $_i$ is the k input to $_j$. We call the constructed "circuit" $_k$. Note that for any fixed input x, if (α, p) is simultaneously good for all $_i$ of $_k$, then $_k(x) = C_k(x)$. For any fixed input x, if (α, p) is chosen uniformly at random from $[] \times \mathcal{PR}$, then $_k(x) \neq C_k(x) \leq (8t \log W)^{-1}$. 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 4, the probability that (α, p) is bad is $(8s\,t \log W)^{-1}$ by the choice of . Since there are at most s gates in C_k, the lemma follows. We say that (α, p) is good for x and $_k$, if it is good for all gate approximators $_i$ of $_k$ simultaneously.

What happens when (α, p) is bad for $_k$? The magnitude of the output of $_k$ is bounded by the maximum output of the approximator of the top gate of C_k. Using Corollary 4, we have for all x and all (α, p),

$$_k(x) \leq 2 + 3 \log W. \tag{9}$$

Since the corollary holds even when the inputs to the top gate are integers, this bound holds even when some other approximator in $_k$ fails.

If we combine Lemma 5 and (9) we get For any fixed input x, if (α, p) is chosen uniformly at random from $[] \times \mathcal{PR}$, then $_k(x) - C_k(x) \leq \frac{1}{2t}$.

Do the same for all the circuits C_i to get approximators $_i$. Consider the following function:

$$h(x) = \sum_{i=1}^{t} \sum_{(\alpha,p)} {}_i(x).$$

Observe that this is equivalent to

$$h(x) = \sum_{i} {}_i(x). \tag{10}$$

We claim the following. For all inputs x, $h(x) = g(x)$. We will prove this by showing that for any fixed x,

$$\sum_{i=1}^{t} {}_i(x) - \sum_{i=1}^{t} C_i(x) \leq 1/2. \tag{11}$$

This is sufficient by (10) and the fact that $\sum_{i=1}^{t} C_i(x) \geq 1$. The inequality (11) follows by straightforward application of Lemma 5.

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 . It is not hard to see that each wire in C corresponds to a polynomially (in $\log W$) many wires in .

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 $\mathcal{PR}$ has size $O(^2\log)$, so the weights are all $O(s^3 t^2 \log^2 W(\log(st) + \log\log W))$. This completes the proof of Theorem 5.

Just as before, the construction is polynomial in n as long as $\log W$ is polynomial in n. By Theorem 2, this is always possible.

6 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 2), *integer multiplication* and *integer division* (depth 3), see [22].

More generally, our results entail the *uniformity* of the classes of majority circuits simulating the corresponding classes of *threshold circuits*. (We address this question in the full version of this paper.)

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 for many valuable comments on an earlier version of this paper. We also thank Jens Lagergren, Sasha Razborov, and Avi Wigderson for several interesting discussions on the topic of this paper.

References

[1] E. Allender. A note on the power of threshold circuts. In *Proc. 30 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. In *Proc. 25 IEEE Symposium on Foundations of Computer Science*, pages 1–6, 1984.

[5] R. Beigel and J. Tarui. On ACC. In *Proc. 32 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. 31 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] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. In *Proc. 7 Annual Structure in Complexity Theory Conference*, pages 2–13, 1992. To appear in *Computational Complexity*.

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

[11] J. Håstad. On the size of weights for threshold gates. Manuscript, 1992.

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

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

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

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

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

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

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

[19] 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. Manuscript, 1992.

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

[21] 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. To appear in IEEE Trans. Information Theory.

[22] K. Y. Siu and V. Roychowdhury. On optimal depth threshold circuits for multiplication and related problems. Manuscript, 1992.

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

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

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

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

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