

## On the Computational Power of Threshold Circuits with Sparse Activity

**Kei Uchizawa**

*uchizawa@maruoka.ecei.tohoku.ac.jp*

*Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan*

**Rodney Douglas**

*rjd@ini.phys.ethz.ch*

*Institute of Neuroinformatics, ETH Zürich, CH-8057 Zürich, Switzerland*

**Wolfgang Maass**

*maass@igi.tugraz.at*

*Institute for Theoretical Computer Science, Technische Universität Graz,  
A-8010 Graz, Austria*

Circuits composed of threshold gates (McCulloch-Pitts neurons, or perceptrons) are simplified models of neural circuits with the advantage that they are theoretically more tractable than their biological counterparts. However, when such threshold circuits are designed to perform a specific computational task, they usually differ in one important respect from computations in the brain: they require very high activity. On average every second threshold gate fires (sets a 1 as output) during a computation. By contrast, the activity of neurons in the brain is much sparser, with only about 1% of neurons firing. This mismatch between threshold and neuronal circuits is due to the particular complexity measures (circuit size and circuit depth) that have been minimized in previous threshold circuit constructions. In this letter, we investigate a new complexity measure for threshold circuits, energy complexity, whose minimization yields computations with sparse activity. We prove that all computations by threshold circuits of polynomial size with entropy  $O(\log n)$  can be restructured so that their energy complexity is reduced to a level near the entropy of circuit states. This entropy of circuit states is a novel circuit complexity measure, which is of interest not only in the context of threshold circuits but for circuit complexity in general. As an example of how this measure can be applied, we show that any polynomial size threshold circuit with entropy  $O(\log n)$  can be simulated by a polynomial size threshold circuit of depth 3.

Our results demonstrate that the structure of circuits that result from a minimization of their energy complexity is quite different from the structure that results from a minimization of previously considered

**complexity measures, and potentially closer to the structure of neural circuits in the nervous system. In particular, different pathways are activated in these circuits for different classes of inputs. This letter shows that such circuits with sparse activity have a surprisingly large computational power.**

## 1 Introduction

---

The active outputs of neurons are stereotypical electrical pulses (action potentials, or spikes). The stereotypical form of these spikes suggests that the output of neurons is analogous to the 1 of a threshold gate. In fact, historically and even currently, threshold circuits are commonly viewed as abstract computational models for circuits of biological neurons. Nevertheless, it has long been recognized by neuroscientists that neurons are generally silent and that information processing in the brain is usually achieved with a sparse distribution of neural firing.<sup>1</sup> One reason for this sparse activation may be metabolic cost. For example, a recent biological study on the energy cost of cortical computation (Lennie, 2003) concludes that “the cost of a single spike is high, and this limits, possibly to fewer than 1%, the number of neurons that can be substantially active concurrently.” The metabolic cost of the active (1) state of a neuron is very asymmetrical. The production of a spike consumes a substantial amount of energy (about  $2.4 \times 10^9$  molecules of ATP according to Lennie (2003)), whereas the energy cost of the no-spike rest state is substantially less. In contrast to neuronal circuits, computations in feedforward threshold circuits (and many other circuit models for digital computation) have the property that a large portion, usually around 50%, of gates in the circuit output a 1 during any computation. Common abstract measures for the energy consumption of electronic circuits treat the cost of the two output states 0 and 1 of a gate symmetrically and focus instead on the required number of switchings between these two states (see Kissin, 1991, as well as Reif, & Tyagi, 1990). Exceptions are Weinzweig (1961), Kasim-Zade (1992), Cheremisin (2003), which provide Shannon-type results for the number of gates that output a 1 in Boolean circuits consisting of gates with bounded fan-in. Circuits of threshold gates (= linear threshold gates = McCulloch-Pitts neurons) are an important class of circuits that are frequently used as simplified models for computations in neural circuits (Minsky & Papert, 1988; Roychowdhury, Siu, & Orlicsky, 1994; Parberry, 1994; Siu, Roychowdhury, & Kailath, 1995). Their output is binary, like that of a biological neuron (which outputs a spike or no spike), but they work in discrete time. In this letter, we consider how investigations of such

---

<sup>1</sup> According to recent data (Margrie, Brecht, & Sakmann, 2002) from whole cell recordings in awake animals, the spontaneous firing rates are on average below 1 Hz.

abstract threshold circuits can be reconciled with actual activity characteristics of biological neural networks.

In section 2 we give a precise definition of threshold circuits and also define their energy complexity, whose minimization yields threshold circuits that carry out computations with sparse activity: on average, few gates output a 1 during a computation. In section 2 we introduce another novel complexity measure: the entropy of a computation. This measure is interesting for many types of circuits beyond the threshold circuits discussed in this letter. It measures the total number of different patterns of gate states that arise during computations on different circuit inputs. We show in section 3 that the entropy of circuit states defines a coarse lower bound for its energy complexity. This result is relevant for any attempt to simulate a given threshold circuit by another threshold circuit with lower energy complexity, since the entropy of a circuit is directly linked to the algorithm that it implements. Therefore, it is unlikely that there exists a general method permitting any given circuit to be simulated by one with smaller entropy. In this sense, the entropy of a circuit defines a hard lower bound for any general method that aims to simulate any given threshold circuit using a circuit with lower energy complexity. However, we will prove in section 3 that there exists a general method that reduces—if this entropy is  $O(\log n)$ —the energy complexity of a circuit to a level near the entropy of the circuit. Since the entropy of a circuit is a complexity measure that is interesting in its own right, we also offer in section 4 a first result on the computational power of threshold circuits with low entropy. Some open problems related to the new concepts introduced in this letter are listed in section 5.

## 2 Definitions

---

A threshold gate  $g$  (with weights  $w_1, \dots, w_n \in \mathbb{R}$  and threshold  $t \in \mathbb{R}$ ) gives as output for any input  $X = (x_1, \dots, x_n) \in \mathbb{R}^n$

$$g(X) = \text{sign} \left( \sum_{i=1}^n w_i x_i - t \right) = \begin{cases} 1, & \text{if } \sum_{i=1}^n w_i x_i \geq t \\ 0, & \text{otherwise,} \end{cases}$$

where we set  $\text{sign}(z) = 1$  if  $z \geq 0$  and  $\text{sign}(z) = 0$  if  $z < 0$ .

For a threshold gate  $g_i$  within a feedforward circuit  $C$  that receives  $X = (x_1, \dots, x_n)$  as circuit input, we write  $g_i(X)$  for the output that the gate  $g_i$  gives for this circuit input  $X$  (although the actual input to gate  $g_i$  during this computation will in general consist of just some variables  $x_i$  from  $X$ , and in addition, or even exclusively, of outputs of other gates in the circuit  $C$ ).

We define the energy complexity of a circuit  $C$  consisting of threshold gates  $g_1, \dots, g_m$  as the expected number of 1's that occur in a computation

for some given distribution  $Q$  of circuit inputs  $X$ , that is,

$$EC_Q(C) := E \left[ \sum_{i=1}^m g_i(X) \right],$$

where the expectation is evaluated with regard to the distribution  $Q$  over  $X \in \mathbb{R}^n$  (or  $X \in \{0, 1\}^n$ ). Thus, for the case where  $Q$  is the uniform distribution over  $\{0, 1\}^n$ , we have, for example,

$$EC_{\text{uniform}} := \frac{1}{2^n} \sum_{X \in \{0,1\}^n} \sum_{i=1}^m g_i(X).$$

In some cases it is also of interest to consider the maximal energy consumption of a circuit for any input  $X$ , defined by

$$EC_{\text{max}}(C) := \max \left( \sum_{i=1}^m g_i(X) : X \in \mathbb{R}^n \right).$$

We define the entropy of a (feedforward) circuit  $C$  by

$$H_Q(C) := - \sum_{A \in \{0,1\}^m} P_C(A) \cdot \log P_C(A),$$

where  $P_C(A)$  is the probability that the internal gates  $g_1, \dots, g_m$  of the circuit  $C$  assume the state  $A \in \{0, 1\}^m$  during a computation of circuit  $C$  (for some given distribution  $Q$  of circuit inputs  $X \in \mathbb{R}^n$ ). We often write  $H_{\text{max}}(C)$  for the largest possible value that  $H_Q(C)$  can assume for any distribution on a given set of circuit states  $A$ . If  $MAX(C)$  is defined as the total number of different circuit states that circuit  $C$  assumes for different inputs  $X \in \{0, 1\}^n$ , then one has  $H_Q(C) = H_{\text{max}}(C)$  if  $Q$  is such that these  $MAX(C)$  circuit states all occur with the same probability and  $H_{\text{max}}(C)$  is then equal to  $\log_2 MAX(C)$ .

We write  $size(C)$  for the number  $m$  of gates in a circuit  $C$  and  $depth(C)$  for the length of the longest path in  $C$  from an input to its output node (which is always assumed to be the node  $g_m$ ).

### 3 Construction of Threshold Circuits with Sparse Activity \_\_\_\_\_

It is obvious that the number of 1's in a computation limits the number of states that the circuit can assume:

$$\begin{aligned} H_Q(C) &\leq \log(MAX(C)) \leq \log \sum_{j=0}^{EC_{\text{max}}(C)} \binom{size(C)}{j} \\ &\leq \log(size(C)^{EC_{\text{max}}(C)}) = EC_{\text{max}}(C) \cdot \log size(C) \end{aligned} \tag{3.1}$$

(for sufficiently large values of  $EC_{\max}(C)$  and  $size(C)$ ;  $\log$  always stands for  $\log_2$  in this letter; the first inequality follows from the previously discussed equality  $H_{\max}(C) = \log_2(MAX(C))$ ). Hence,

$$EC_{\max}(C) \geq H_Q(C) / \log size(C). \quad (3.2)$$

In fact, this argument shows that

$$EC_{\max}(C) \geq H_{\max}(C) / \log size(C). \quad (3.3)$$

Every Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  can be computed by a threshold circuit  $C$  of depth 2 that represents its disjunctive normal form in such a way that for every circuit input  $X$ , at most a single gate on level 1 outputs a 1. This circuit  $C$  has the property that  $EC_{\max}(C) = 2$ . Furthermore, it is an easy exercise to construct a distribution  $Q$  such that this circuit has  $H_Q(C) = \log(size(C) - 1)$ . Hence it is in some cases possible to achieve  $EC_{\max}(C) < H_Q(C)$ , and the factor  $\log size(C)$  in equations 3.2 and 3.3 cannot be eliminated or significantly reduced.

Threshold circuits that represent a Boolean function  $f$  in its disjunctive normal form allow us to compute any Boolean function with a circuit  $C$  that achieves  $EC_{\max}(C) = 2$ . However, these circuits  $C$  in general have exponential size in  $n$ . Therefore, the key question is whether one can also construct polynomial size circuits  $C$  with small  $EC_Q$  or  $EC_{\max}$ . Because of the a priori bounds of equations 3.2 and 3.3, this is possible only for those functions  $f$  that can be computed with a low entropy of circuit states. The following results show that the existence of a circuit  $C$  that computes  $f$  with  $H_{\max}(C) = O(\log n)$  is sufficient to guarantee the existence of a circuit that computes  $f$  with low energy complexity.

**Theorem 1.** *Assume that a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  can be computed by some polynomial size threshold circuit  $C$  with  $H_{\max}(C) = O(\log n)$ . Then  $f$  can also be computed by some polynomial size threshold circuit  $C'$  with*

$$EC_{\max}(C') \leq H_{\max}(C) + 1 = O(\log n). \quad (3.4)$$

*Furthermore, if  $Q$  is any distribution of inputs  $X \in \{0, 1\}^n$ , then one can construct a polynomial size threshold circuit  $C''$  with*

$$EC_Q(C'') \leq \frac{H_Q(C)}{2} + 2 = O(\log n). \quad (3.5)$$

**Remark 1.** The subsequent proof shows that in fact the following more general statements hold for any function  $f$  and any distribution  $Q$ :

If  $f$  can be computed by some arbitrary (feedforward) threshold circuit  $C$ , then  $f$  can also be computed by a threshold circuit  $C'$  with  $size(C') \leq 2^{H_{\max}(C)}$ ,  $depth(C') \leq size(C) + 1$ ,  $H_{\max}(C') \leq H_{\max}(C)$ , and  $EC_{\max}(C') \leq H_{\max}(C) + 1$ . Furthermore,  $f$  can also be computed by a threshold circuit  $C''$  with  $size(C'') \leq 2^{H_{\max}(C)+1}$ ,  $depth(C'') \leq size(C) + 1$ ,  $H_Q(C'') \leq H_Q(C)$ , and  $EC_Q(C'') \leq \frac{H_Q(C)}{2} + 2$ .

**Remark 2.** The assumption  $H_{\max}(C) = O(\log n)$  is satisfied by standard constructions of threshold circuits for many commonly considered functions  $f$ . Examples are all symmetrical functions (hence, in particular, PARITY of  $n$  bits), COMPARISON of binary numbers, and BINARY ADDRESSING (routing) where the first  $k$  input bits represent an address for one of the  $2^k$  subsequent input bits (thus,  $n = k + 2^k$ ). In fact, to the best of our knowledge, there is no function known that can be computed by polynomial size circuits but not by polynomial size circuits  $C$  with  $H_{\max}(C) = O(\log n)$ .

**Proof of Theorem 1.** The proof is split up into a number of lemmas (see lemmas 1 to 6). The idea is to first simulate in lemma 1 the given circuit  $C$  by a threshold decision tree (i.e., by a decision tree  $T$  with threshold gates at the nodes; see definition 1), which has at most  $2^{H_{\max}(C)}$  leaves. Then this threshold decision tree is restructured in lemma 3 in such a manner that every path in the tree from the root to a leaf takes the right branch at an internal node at most  $\log(\# \text{ of leaves})$  times. Hence, the path can take the right branch at most  $H_{\max}(C)$  times. Obviously such an asymmetrical cost measure is of interest when one wants to minimize an asymmetrical complexity measure such as  $EC$ , which assigns different costs to gate outputs 0 and 1. Finally we show in lemma 5 that the computations of the resulting threshold decision tree can be simulated by a threshold circuit where some gate outputs a 1 whenever the simulated path in the decision tree moves into the right subtree at an internal node of the tree. The proof of lemma 5 has to take into account that the control structures of decision trees and circuits are quite different: a gate in a decision tree is activated only when the computation path happens to arrive at the corresponding node of the decision tree, but a gate in a threshold circuit is activated in any computation of that circuit. Hence, a threshold decision tree with few threshold gates that output 1 does not automatically yield a threshold circuit with low energy complexity. However, we show that all gates in the simulating threshold circuit that do not correspond to a node in the decision tree where the right branch is chosen receive an additional input with a strongly negative weight (see lemma 4), so that they output a 0 when they get activated.

Finally, we show in lemma 6 that the threshold decision tree can be restructured alternatively, so that the average number of times when a computation path takes the right subtree at a node remains small (instead of the maximal number of taking the right subtree). This maneuver yields the proof of the second part of the claim of theorem 1.

**Definition 1.** A threshold decision tree  $T$  (called a linear decision tree in Gröger and Turán (1991)) is a binary tree in which each internal node has two children, a left and a right one, and is labeled by a threshold gate that is applied to the input  $X \in \{0, 1\}^n$  for the tree. All the leaves of threshold decision trees are labeled by 0 or 1. To compute the output of a threshold decision tree  $T$  on an input  $X$ , we apply the following procedure from the root until reaching a leaf: we go left if the gate at a node outputs 0; otherwise, we go right. If we reach a leaf labeled by  $l \in \{0, 1\}$ , then  $l$  is the output of  $T$  for input  $X$ .

Note that the threshold gates in a threshold decision tree are applied only to input variables from the external input  $X \in \{0, 1\}^n$ , not to outputs of preceding threshold gates. Hence, it is obvious that computations in threshold decision trees have a quite different structure from computations in threshold circuits, although both models use the same type of computational operation at each node.

The depth of a threshold decision tree is the maximum number of nodes from the root to a leaf. We assign binary strings to nodes of  $T$  in the usual manner:

- $\hat{g}_\varepsilon$  denotes the root of the tree (where  $\varepsilon$  is the empty string).
- For a binary string  $s$ , let  $\hat{g}_{s0}$  and  $\hat{g}_{s1}$  be the left and right child of the node with label  $\hat{g}_s$ .

For example, the ancestors of a node  $\hat{g}_{1011}$  are  $\hat{g}_\varepsilon$ ,  $\hat{g}_1$ ,  $\hat{g}_{10}$ , and  $\hat{g}_{101}$ . Let  $S_T$  be the set of all binary strings  $s$  that occur as indices of nodes  $\hat{g}_s$  in a threshold decision tree  $T$ . Then all the descendants of node  $\hat{g}_s$  in  $T$  can be represented as  $\hat{g}_{s*}$  for  $s* \in S_T$ .

The given threshold circuit  $C$  can be simulated in the following way by a threshold decision tree:

**Lemma 1.** Let  $C$  be a threshold circuit computing a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  with  $m$  gates. Then one can construct a threshold decision tree  $T$  with at most  $2^{H_{\max}(C)}$  leaves and  $\text{depth}(T) \leq m$ , which computes the same function  $f$ .

**Proof.** Assume that  $C$  consists of  $m$  gates. We number the gates  $g_1, \dots, g_m$  of  $C$  in topological order. Since  $g_i$  receives the circuit input  $X$  and the outputs of  $g_j$  only for  $j < i$  as its inputs, we can express the output  $g_i(X)$  of  $g_i$  for circuit input  $X = \langle x_1, \dots, x_n \rangle$  as

$$g_i(X) = \text{sign} \left( \sum_{j=1}^n w_j^i x_j + \sum_{j=1}^{i-1} w_{g_j}^i g_j(X) + t_i \right),$$

where  $w_{g_j}^i$  is the weight that  $g_i$  applies to the output of  $g_j$  in circuit  $C$ .

Let  $S$  be the set of all binary strings of length up to  $m - 1$ . We define threshold gates  $\hat{g}_s : X \rightarrow \{0, 1\}$  for  $s \in S$  by

$$\hat{g}_s(X) = \text{sign} \left( \sum_{j=1}^n w_j^{|s|+1} x_j + t_s \right) \text{ with}$$

$$t_s = \sum_{j=1}^{|s|} w_{g_j}^{|s|+1} s_j + t_{|s|+1},$$

where  $s_j$  is the  $j$ th bit of string  $s$  and  $|s|$  is the length of  $s$ . Obviously these gates  $\hat{g}_s$  are variations of gate  $g_i$  with different built-in assumptions  $s$  about the outputs of preceding gates.

Let  $T$  be the threshold decision tree consisting of gates  $\hat{g}_s$  for  $s \in S$ . That is, gate  $\hat{g}_\epsilon = g_1$  is placed at the root of  $T$ . We let the left child of  $\hat{g}_s$  be  $\hat{g}_{s0}$  and the right child of  $\hat{g}_s$  be  $\hat{g}_{s1}$ . We let each  $\hat{g}_s$  with  $|s| = m - 1$  have a leaf labeled by 0 as left child and a leaf labeled 1 as right child. Since  $\hat{g}_s$  computes the same function as  $g_{|s|+1}$  if the preceding gates  $g_i$  output  $s_i$  for  $1 \leq i \leq |s|$ ,  $T$  computes the same function  $f$  as  $C$ . We then remove all leaves from  $T$  for which the associated paths correspond to circuit states  $A \in \{0, 1\}^m$  that do not occur in  $C$  for any circuit input  $X \in \{0, 1\}^n$ . This reduces the number of leaves in  $T$  to  $2^{H_{\max}(C)}$ . Finally, we iteratively remove all nodes without children, and replace all nodes below which there exists just a single leaf by a leaf. In this way, we arrive again at a binary tree.

We now introduce a cost measure  $\text{cost}(T)$  for trees  $T$  that, like the energy complexity for circuits, measures for threshold decision trees how often a threshold gate outputs a 1 during a computation:

**Definition 2.** We denote by  $\text{cost}(T)$  the maximum number of times where a path from the root to a leaf in a binary tree  $T$  goes to the right. If  $T$  is a leaf, then  $\text{cost}(T) = 0$ .

We will show in lemma 5 that one can simulate any threshold decision tree  $T'$  by a threshold circuit  $C_{T'}$  with  $EC_{\max}(C_{T'}) \leq \text{cost}(T') + 1$ . Hence, it suffices for the proof of theorem 1 to simulate the threshold decision tree  $T$  resulting from lemma 1 by another threshold decision tree  $T'$  for which  $\text{cost}(T')$  is small. This is done in lemma 4, where we will construct a tree  $T'$  that reduces  $\text{cost}(T')$  down to another cost measure  $\text{rank}(T)$ . This measure  $\text{rank}(T)$  always has a value  $\leq \log(\# \text{ of leaves of } T)$  according to lemma 2; hence,  $\text{rank}(T) \leq H_{\max}(C)$  for the tree  $T$  constructed in lemma 1.

**Definition 3.** *The rank of a binary tree  $T$  is defined inductively as follows:*

- If  $T$  is a leaf, then  $\text{rank}(T) = 0$ .
- If  $T$  has subtrees  $T_l$  and  $T_r$ , then

$$\text{rank}(T) = \begin{cases} \text{rank}(T_l), & \text{if } \text{rank}(T_l) > \text{rank}(T_r) \\ \text{rank}(T_r) + 1, & \text{if } \text{rank}(T_l) = \text{rank}(T_r) \\ \text{rank}(T_r), & \text{if } \text{rank}(T_l) < \text{rank}(T_r). \end{cases}$$

**Lemma 2.** *Let  $T$  be any binary tree. Then  $\text{rank}(T) \leq \log(\# \text{ of leaves of } T)$ .*

**Proof.** We proceed by induction on the depth of  $T$ . If  $\text{depth}(T) = 0$ , then  $T$  consists of a single node; hence,

$$\text{rank}(T) = 0 = \log 1 = \log(\# \text{ of leaves of } T).$$

Assume now that  $\text{depth}(T) > 0$ , and let  $T_l$  and  $T_r$  be the left and right subtree of the root of  $T$ .

**Case 1:**  $\text{rank}(T_l) \neq \text{rank}(T_r)$ . Then the claim follows immediately from the induction hypothesis, since  $\text{rank}(T) = \text{rank}(T_l)$  or  $\text{rank}(T) = \text{rank}(T_r)$ .

**Case 2:**  $\text{rank}(T_l) = \text{rank}(T_r)$ . Assume without loss of generality that  $(\# \text{ of leaves of } T_l) \leq (\# \text{ of leaves of } T_r)$ . Then the induction hypothesis implies that  $\text{rank}(T) = \text{rank}(T_l) + 1 \leq \log(\# \text{ of leaves of } T_l) + 1 = \log(2 \cdot (\# \text{ of leaves of } T_l)) \leq \log(\# \text{ of leaves of } T)$ .

**Lemma 3.** *Let  $T$  be a threshold decision tree computing a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . Then  $f$  can also be computed by a threshold decision tree  $T'$  that has the same depth and the same number of leaves as  $T$  and satisfies  $\text{cost}(T') = \text{rank}(T)$ .*

**Proof.** Let  $T$  consist of gates  $g_s$  for  $s \in S_T$ . We define  $T^s$  as the subtree of  $T$  whose root is  $g_s$ . Let  $T_l^s$  (respectively,  $T_r^s$ ) denote the left(right) subtree below the root of  $T^s$ . We modify  $T$  inductively by the following procedure, starting at the nodes  $g_s$  of largest depth. If  $\text{cost}(T_l^s) < \text{cost}(T_r^s)$ , we replace  $g_s$  by its complement and swap the left subtree and the right subtree. The complement of  $g_s$  is another threshold gate  $g$  that outputs 1 if and only if  $g_s$  outputs 0. Such a gate  $g$  exists since  $\sum_{i=1}^n w_i x_i < t \Leftrightarrow \sum_{i=1}^n (-w_i) x_i > -t \Leftrightarrow \sum_{i=1}^n (-w_i) x_i \geq T'$  for another threshold  $T'$  (which always exists if the  $x_i$  assume only finitely many values). Let  $\hat{T}^s$  be the threshold decision tree produced from  $T^s$  by this procedure. By construction, it has the following properties:

- If the children of  $g_s$  both are leaves, then we have  $\text{cost}(\hat{T}^s) = 1$ .

- Otherwise,

$$\text{cost}(\hat{T}^s) = \begin{cases} \text{cost}(\hat{T}_l^s), & \text{if } \text{cost}(\hat{T}_l^s) > \text{cost}(\hat{T}_r^s) \\ \text{cost}(\hat{T}_r^s) + 1, & \text{if } \text{cost}(\hat{T}_l^s) = \text{cost}(\hat{T}_r^s) \\ \text{cost}(\hat{T}_l^s), & \text{if } \text{cost}(\hat{T}_l^s) < \text{cost}(\hat{T}_r^s), \end{cases}$$

where  $\hat{T}^s$  has subtrees  $\hat{T}_l^s$  and  $\hat{T}_r^s$ .

Since this definition coincides with the definition of the rank, we have constructed a tree  $T'$  with  $\text{cost}(T') = \text{rank}(T)$ . This procedure preserves the function that is computed, the depth of the tree, and the number of leaves.

We now show that the threshold decision tree that was constructed in lemma 3 can be simulated by a threshold circuit with low energy complexity. As preparation, we first observe in lemma 4 that one can “veto” any threshold gate  $g$  through some extra input. This will be used in lemma 5 in order to avoid the event that gates in the simulating circuit that correspond to gates in an inactive path of the simulated threshold decision tree increase the energy complexity of the resulting circuit.

**Lemma 4.** *Let  $g(x_1, \dots, x_n) = \text{sign}(\sum_{i=1}^n w_i x_i - t)$  be a threshold gate. Then one can construct a threshold gate  $g'$  using an additional input  $x_{n+1}$  with the following property:*

$$g'(x_1, \dots, x_n, x_{n+1}) = \begin{cases} 0, & \text{if } x_{n+1} = 1 \\ g(x_1, \dots, x_n), & \text{if } x_{n+1} = 0. \end{cases}$$

**Proof.** We set  $w_{n+1} := -(\sum_{i=1}^n |w_i| + |t| + 1)$ . Apart from that  $g'$  uses the same weights and threshold as  $g$ , it is obvious that the resulting gate  $g'$  has the desired property.

**Lemma 5.** *Let  $T$  be a threshold decision tree that consists of  $k$  internal nodes and computes a function  $f$ . Then one can construct a threshold circuit  $C_T$  with  $EC_{\max}(C_T) \leq \text{cost}(T) + 1$  that computes the same function  $f$ . In addition,  $C_T$  satisfies  $\text{depth}(C_T) \leq \text{depth}(T) + 1$  and  $\text{size}(C_T) \leq k + 1$ .*

**Proof.** We can assume without loss of generality that every leaf with label 1 in  $T$  is the right child of its parent (if this is not the case, swap this leaf with the right subtree of the parent, and replace the threshold gate at the parent node as in the proof of lemma 3 by another threshold gate that always outputs the negation of the former gate; this procedure does not increase the cost of the tree or its depth or number of internal nodes). Now let

$$g_s(X) = \text{sign} \left( \sum_{j=1}^n w_j^s x_j - t_s \right)$$

be the threshold gate in  $T$  at the node with label  $s \in S_T$ . Let  $w_{n+1}^s$  be the weight constructed in lemma 4 for an additional input that can force gate  $g_s$  to output 0. Set  $W := \max\{|w_{n+1}^s| : s \in S_T\}$ .

The threshold circuit  $C_T$  that simulates  $T$  has a gate  $g'_s$  for every gate  $g_s$  in  $T$  and, in addition, an OR-gate that receives inputs from all gates  $g'_s$  so that  $g_s$  has a leaf with label 1. (Because of our assumption, this leaf is reached whenever the gate  $g_s$  at node  $s \in S_T$  gets activated and  $g_s$  outputs a 1.) We make sure that any gate  $g'_s$  in  $C_T$  outputs 1 for a circuit input  $X$  if and only if the gate  $g_s$  in  $T$  gets activated for this input  $X$  and outputs 1. Therefore, a gate  $g'_s$  in  $C_T$  can output 1 only if it corresponds to a gate  $g_s$  in  $T$  with output 1 that lies on the single path of  $T$  that the circuit input  $X$  activates. Hence, this construction automatically ensures that  $EC_{\max}(C_T) \leq \text{cost}(T) + 1$  (where the +1 arises from the additional OR-gate in  $C_T$ ).

In order to achieve this objective,  $g'_s$  gets additional inputs from all gates  $g'_s$  in  $C_T$  such that  $\bar{s}$  is a proper prefix of  $s$ . The weight for the additional input from  $g'_s$  is  $-W$  if  $\bar{s}0$  is a prefix of  $s$ , and  $W$  otherwise. In addition, the threshold of  $g'_s$  is increased by  $l_s \cdot W$ , where  $l_s$  is the number of 1's in the binary string  $s$ . In this way,  $g'_s$  can output 1 if and only if  $g_s$  outputs 1 for the present circuit input  $X$ , and all gates  $g_{\bar{s}}$  of  $T$  for which  $g_s$  lies in the right subtree below  $g_{\bar{s}}$  output 1, and all gates  $g_{\bar{s}}$  of  $T$  for which  $g_s$  lies in the left subtree below  $g_{\bar{s}}$  output 0. Thus,  $g'_s$  outputs 1 if and only if the path leading to gate  $g_s$  gets activated in  $T$  and  $g_s$  outputs 1.

The proof of the first claim of theorem 1 follows now immediately from lemmas 1 to 5. Note that the number  $k$  of internal nodes in a binary tree is equal to (# of leaves)−1; hence,  $k \leq 2^{H_{\max}(C)} - 1$  in the case of the decision tree  $T$  resulting from applications of lemmas 1 and 3. This yields  $\text{size}(C_T) \leq 2^{H_{\max}(C)}$  for the circuit  $C_T$  that is constructed in lemma 5 for this tree  $T$ .

The proof of the second claim of theorem 1 follows by applying the subsequent lemma 6 instead of lemma 3 to the threshold decision tree  $T$  resulting from lemma 1. In addition, a minor modification is needed in the proof of lemma 5. The threshold decision tree  $T'$  that results from lemma 6 is constructed to have the property that each gate in  $T'$  outputs 1 with probability  $\leq 1/2$ . This may require that the left child of a node is a leaf with label 1, causing in lemma 5 a potential doubling of the circuit size and an additive increase by 1 of the energy complexity.

**Lemma 6.** *Let  $T$  be a threshold decision tree computing  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . Then for any given distribution  $Q$  of circuit inputs, there exists a threshold decision tree  $T'$  computing  $f$  such that the expected number of 1's with regard to  $Q$  is at most  $H_Q(C)/2$ .*

**Proof.** Let  $\text{cost}_Q(s)$  be the expected number of times where one goes to the right in a subtree of  $T'$  whose root is the node labeled by  $s$ . Let  $P(s)$  be the probability (with regard to  $Q$ ) that gate  $g_s$  outputs 1. We construct  $T'$

by modifying  $T$  inductively (starting at the nodes of the largest depth  $m$  in  $T$ ) through the following procedure: if  $P(s) > 1/2$ , replace  $g_s$  by a threshold gate that computes its negation and swap the left and right subtree below this node.

By construction, we have  $P(s) \leq 1/2$  for every gate  $g_s$  in  $T'$ . Furthermore, we have:

- If  $|s| = m - 1$ , then  $cost_Q(s) = P(s)$ .
- If  $0 \leq |s| < m - 1$ , then  $P(s) \leq 1/2$  and  $cost_Q(s) = P(s) + P(s)cost_Q(s1) + (1 - P(s))cost_Q(s0)$ .

One can prove by induction on  $|s|$  that  $cost_Q(s) \leq H_Q(s)/2$  for all  $s \in S_{T'}$ , where  $H_Q(s)$  is the entropy of states of the ensemble of gates of  $T'$  in the subtree below gate  $g_s$ .

For the induction step, one uses the convexity of the log function (which implies that  $P(s) = -P(s) \cdot (-1) = -P(s) \cdot \log \frac{P(s)+(1-P(s))}{2} \leq -P(s)(\frac{\log(P(s))+\log(1-P(s))}{2}))$  and the fact that  $P(s) \leq 1 - P(s)$  to show that

$$\begin{aligned} cost_Q(s) &\leq P(s) + P(s) \cdot \frac{H_Q(s1)}{2} + (1 - P(s)) \cdot \frac{H_Q(s0)}{2} \\ &\leq -P(s) \cdot \left( \frac{\log P(s) + \log(1 - P(s))}{2} \right) \\ &\quad + P(s) \frac{H_Q(s1)}{2} + (1 - P(s)) \cdot \frac{H_Q(s0)}{2} \\ &\leq -\frac{P(s)}{2} \log P(s) - \frac{(1 - P(s))}{2} \log(1 - P(s)) + P(s) \frac{H_Q(s1)}{2} \\ &\quad + (1 - P(s)) \frac{H_Q(s0)}{2} \leq \frac{H_Q(s)}{2}. \end{aligned}$$

**Remark 3.** The results of this section can also be applied to circuits that compute arbitrary functions  $f : D \rightarrow \{0, 1\}$  for some arbitrary finite set  $D \subseteq \mathbb{R}^n$  (instead of  $\{0, 1\}^n$ ). For domains  $D \subseteq \mathbb{R}^n$  of infinite size, a different proof would be needed, since then one can no longer replace any given threshold gate by another threshold gate that computes its negation (as used in the proofs of lemmas 3, 5, and 6).

#### 4 On the Computational Power of Circuits with Low Entropy \_\_\_\_\_

The concepts discussed in this letter raise the question which functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  can be computed by polynomial size threshold circuits  $C$  with  $H_{\max}(C) = O(\log n)$ . There is currently no function  $f$  in  $P$  (or even in  $NP$ ) known for which this is provably false. But the following result

shows that if all functions that can be computed by layered<sup>2</sup> polynomial size threshold circuits of bounded depth can be computed by a circuit  $C$  of the same type that satisfies in addition  $H_{\max}(C) = O(\log n)$ , then this implies a collapse of the depth hierarchy for polynomial-size threshold circuits:

**Theorem 2.** *Assume that a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  (or  $f : \mathbb{R}^n \rightarrow \{0, 1\}$ ) can be computed by a threshold circuit  $C$  with polynomially in  $n$  many gates and  $H_{\max}(C) = O(\log n)$ . Then one can compute  $f$  with a polynomial size threshold circuit  $C'$  of depth 3.*

**Proof.** According to lemma 1, there exists a threshold decision tree  $T$  with polynomially in  $n$  many leaves and  $\text{depth}(T) \leq \text{size}(C)$ . Design (as in Gröger & Turán, 1991) for each path  $p$  from the root to a leaf with output 1 in  $T$  a threshold gate  $g_p$  on layer 2 of  $C'$  that outputs 1 if and only if this path  $p$  becomes active in  $T$ . The output gate on layer 3 of  $C'$  is simply an OR of all these gates  $g_p$ .

## 5 Discussion

---

In this letter, we have introduced an energy complexity measure for threshold circuits that reflects the biological fact that the firing of a neuron consumes more energy than its nonfiring. We also have provided methods for restructuring a given threshold circuit with high energy consumption by a threshold circuit that computes the same function but has brain-like sparse activity. Theorem 1 in combination with remark 2 implies that the computational power of such circuits is quite large. The resulting circuits with sparse activity may help us to elucidate the way in which circuits of neurons are designed in biological systems. In fact, the structure of computations in the threshold circuits with sparse activity that were constructed in the proof of theorem 1 is reminiscent of biological results on the structure of computations in cortical circuits of neurons, where there is concern for the selection of different pathways (dynamic routing) in dependence of the stimulus (Olshausen, Anderson, & Essen, 1995). In addition, our constructions provide first steps toward the design of algorithms for future extremely dense VLSI implementations of neurally inspired circuits, where energy consumption and heat dissipation become critical factors.

It is well known (see, e.g., Hajnal, Maass, Pudlak, Szegedy, & Turán, 1993) that threshold circuits can be made robust against random failure of gates with a moderate increase in circuit size. Such methods can also be

---

<sup>2</sup> A feedforward circuit is said to be layered if its gates can be partitioned into layers so that edges go from only one layer to the next. We actually need to assume here only that edges from circuit inputs go only to gates on layer 1.

applied to the sparsely active threshold circuits that were constructed in this letter, maintaining their sparse activity feature. For example, one can replace each threshold gate by an odd number  $k$  of identical copies of this gate and take their majority vote with the help of another threshold gate. This increases the circuit size by a factor of only  $k + 1$ , but preserves their sparse activity. Furthermore, the resulting circuit computes correctly as long as the majority of gates in each group of  $k$  gates computes without a fault. Additional noise suppression could exploit that all legitimate activation patterns of gates in the circuit  $C_T$  that was constructed in lemma 5 have a quite specific structure, since they simulate an activation path in a tree  $T$ .

The new concepts and results of this letter suggest a number of interesting open problems in computational complexity theory. At the beginning of section 3, we showed that the energy complexity of a threshold circuit that computes some function  $f$  cannot be less than the a priori bound given by the minimal required circuit entropy for computing such a function. This result suggests that the entropy of circuit states required for various practically relevant functions should be investigated. Another interesting open problem is the trade-off between energy complexity and computation speed in threshold circuits, both in general and for concrete computational problems. Finally, we consider that both the energy complexity and the entropy of threshold circuits are concepts of interest in their own right. They give rise to interesting complexity classes that have not been considered previously in computational complexity theory. In particular, it may be possible to develop new lower-bound methods for circuits with low entropy, thereby enlarging the reservoir of lower-bound techniques in circuit complexity theory.

## Acknowledgments

---

We thank Michael Pfeiffer, Pavel Pudlak, and Robert Legenstein for helpful discussions; Kazuyuki Amano and Eiji Takimoto for their advice; and Akira Maruoka for making this collaboration possible. This work was partially supported by the Austrian Science Fund FWF, project S9102-N04, and projects FP6-015879 (FACETS) and FP6-2005-015803 (DAISY) of the European Union.

## References

---

- Cheremisin, O. V. (2003). On the activity of cell circuits realising the system of all conjunctions. *Discrete Mathematics and Applications*, 13(2), 209–219.
- Gröger, H. D., & Turán, G. (1991). On linear decision trees computing Boolean functions. *Lecture Notes in Computer Science*, 510, 707–718.
- Hajnal, A., Maass, W., Pudlak, P., Szegedy, M., & Turán, G. (1993). Threshold circuits of bounded depth. *J. Comput. System Sci.*, 46, 129–154.

- Kasim-Zade, M. (1992). On a measure of the activeness of circuits made of functional elements. *Mathematical Problems in Cybernetics*, 4, 218–228. (in Russian)
- Kissin, G. (1991). Upper and lower bounds on switching energy in VLSI. *J. Assoc. Comp. Mach.*, 38, 222–254.
- Lennie, P. (2003). The cost of cortical computation. *Current Biology*, 13, 493–497.
- Margrie, T. W., Brecht, M., & Sakmann, B. (2002). In vivo, low-resistance, whole-cell recordings from neurons in the anaesthetized and awake mammalian brain. *Pflugers Arch.*, 444(4):491–498.
- Minsky, M., & Papert, S. (1988). *Perceptrons: An introduction to computational geometry*. Cambridge, MA: MIT Press.
- Olshausen, B. A., Anderson, C. H., & Essen, D. C. V. (1995). A multiscale dynamic routing circuit for forming size- and position-invariant object representations. *J. Comput. Neurosci.*, 2(1):45–62.
- Parberry, I. (1994). *Circuit complexity and neural networks*. Cambridge, MA: MIT Press.
- Reif, J. H., & Tyagi, A. (1990). Energy complexity of optical computations. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing* (pp. 14–21). Piscataway, NJ: IEEE.
- Roychowdhury, V. P., Siu, K. Y., & Orlitsky, A. (1994). *Theoretical advances in neural computation and learning*. Boston: Kluwer Academic.
- Siu, K.-Y., Roychowdhury, V., & Kailath, T. (1995). *Discrete neural computation; A theoretical foundation*. Upper Saddle River, NJ: Prentice Hall.
- Weinzweig, M. N. (1961). On the power of networks of functional elements. *Dokl. Akad. Nauk SSSR*, 139, 320–323.

---

Received September 14, 2005; accepted May 5, 2006.