Lower Bounds for Threshold Circuits of Bounded Energy

Kei UCHIZAWA*

Faculty of Engineering, Yamagata University, Yonezawa, Yamagata 992-8510, Japan

Received October 4, 2013; final version accepted February 10, 2014

A threshold circuit is a combinatorial circuit consisting of logic gates computing linear threshold functions. A threshold circuit is one of the most well-studied computational models in circuit complexity theory, and is commonly viewed as an abstract model of a neural network in the brain. In this paper, we investigate threshold circuits from the viewpoint of a biologically-inspired complexity measure, called energy complexity. Following basic definitions and examples of threshold circuits, we observe three lower bound results for threshold circuits of bounded energy.

KEYWORDS: circuit complexity, logic circuits, threshold circuits, neural networks

1. Introduction

Computation is performed by a combination of simple well-defined operations, each of which takes unit time to carry out. Computational model specifies what operations are allowable. Consider, for example, the following two fundamental computational models: the Turing machine and the logic circuit. Computation of a Turing machine is of several operations such as reading a bit from the input tape, moving a head on work tape to right, changing the current state, etc., while the counterpart of a logic circuit is of gates computing basic logic operations such as AND, OR, and NOT (i.e., AND gates, OR gates, and NOT gates). We know that computational model affects its computational power, as in the following extreme example: there is a problem, so-called UNARY HALTING, that cannot be computed by any Turing machine, but can be computed by a logic circuit (see, for example, [3]). Thus, different computational models provide slightly different worlds of computation, which can be critical for a specific computational task.

In this paper, we focus on a particular computational model, called a threshold circuit, where computation of a threshold circuit is of logic gates computing linear threshold functions, called threshold gates. A threshold circuit receives significant attention in computer science; there are mainly two reasons.

One reason is that a threshold circuit is a theoretical model of a neural network in the brain [17, 19, 23]. While there are many theoretical models of a neuron [22], a threshold gate is considered to capture a basic input-output property of a biological neuron. Besides, it is known that neurons communicate with each other by emitting an electrical signal (i.e., “firing” or “non-firing”). This is similar to logic gates outputting a Boolean value (i.e., “one” or “zero”). Thus, by studying a threshold circuit, we hopefully make a step towards understanding characteristic of computation realized by a neural network.

The other reason is that a threshold circuit is on the cutting edge of our understanding for limits of computation. Historically, logic circuits have been considered to open new avenues for resolving P vs. NP problem, since we can conclude P ⊈ NP if we could obtain a superpolynomial lower bound on the size of logic circuits of AND, OR, and NOT gates for a problem in NP. Towards the ultimate goal, the research started off from a restricted model of logic circuits.

Let us observe how the research proceeded along some major results for restricted complexity classes; we focus on $\mathsf{AC}^0$, $\mathsf{AC}^{0}[p]$, and $\mathsf{ACC}^0$. The class $\mathsf{AC}^0$ is a set of Boolean functions computable by constant-depth and polynomial-size logic circuits of AND, OR, and NOT gates. Furst et al., Ajtai, and Hastad independently proved that PARITY is not contained in $\mathsf{AC}^0$ [2, 8, 16]. The class $\mathsf{AC}^{0}[p]$ is a more general class than $\mathsf{AC}^0$: the basic operations include MOD$[p]$ for some prime $p$, where a MOD$[p]$ gate outputs zero if and only if the sum of the inputs bits is zero modulo $p$; and $\mathsf{AC}^{0}[p]$ is defined to be a set of Boolean functions computable by constant-depth and polynomial-size logic circuits of AND, OR, NOT, and MOD$[p]$ gates. Clearly, $\mathsf{AC}^{0} \subseteq \mathsf{AC}^{0}[p]$ holds. Razborov and Smolensky proved that MOD$[q]$ is not contained in $\mathsf{AC}^{0}[p]$ if $p$ and $q$ are coprime [21, 25]. Recently, Williams considered more general class $\mathsf{ACC}^0$ such that the basic operations include MOD$[m]$ for any fixed positive integer $m$, and proved that there exists a Boolean function in NEXP (i.e., the exponential analogue of NP) that is not contained in $\mathsf{ACC}^0$ [32]. Thus, through the last three decades, we have arrived at a superpolynomial lower bound for $\mathsf{ACC}^0$. The candidate class which researchers

---

2010 Mathematics Subject Classification: Primary 62F12, Secondary 62F05.
*Corresponding author. E-mail: uchizawa@yz.yamagata-u.ac.jp
consider to be the next target is given by threshold circuits: The class \( \text{TC}^0 \) is a set of Boolean functions computable by constant-depth and polynomial-size threshold circuits. It is known that \( \text{ACC}^0 \subseteq \text{TC}^0 \), and \( \text{TC}^0 \) seems to be the smallest natural circuit class containing \( \text{ACC}^0 \). Currently, no superpolynomial lower bound is known for \( \text{TC}^0 \) even in NEXP, although superpolynomial lower bounds have been obtained for threshold circuits of depth two or three under some restriction [7, 9, 10, 20]. Thus, a superpolynomial lower bound for \( \text{TC}^0 \) would be a surprising breakthrough following Williams’ result.

The motivation of the research presented in this paper stems from both reasons given above. We consider the following interesting fact about the energy consumption of firing of a neuron: firing consumes substantially more energy than non-firing, and the energy cost of a firing is relatively high. Since energy supplied to the brain is limited, the fact leads the fraction of neurons firing concurrently to be possibly fewer than 1% [4, 15]. Consequently, this asymmetricity of energy consumption can influence the way of information processing of a neural network, and the brain has countered this metabolic constraint by adopting energy-efficient circuit [6, 14, 18, 30]. Uchizawa et al. focused on this fact from the view point of theoretical computer science, and introduce a new complexity measure called the energy complexity of threshold circuits [27]. More formally, we define energy of a threshold circuit \( C \) as the maximum number of gates outputting “ones,” where the maximum is taken over all input assignments to \( C \). We investigate the computational power of a threshold circuit of small energy in the context of computational complexity theory. We note that the mechanism of energy consumption of an electrical circuit is quiet different from a neural network: an electrical gate consumes almost the same amount of energy in either case of outputting “one” and outputting “zero” [1, 12]. Thus, if a neural network realizes its computation under severe energy constraint for firing, this new complexity measure helps us highlight a difference between the computation of normal logic circuits and that of neural networks, and explore how a neural network processes information. Moreover, the energy complexity provides a new perspective to research on threshold circuits, which would help us push our understanding of the computation.

In this paper, we select three lower bound results for threshold circuits with regard to the energy complexity, each of which is derived by a different proof technique. The lower bounds are presented in a line of papers [26, 28, 29], but we simplify the original statements so that the proofs highlight ideas and are self-contained. We mainly consider two Boolean functions, PARITY and INNER-PRODUCT. We first give two lower bounds for PARITY. We show that, while it is known that PARITY can be computed by a logarithmic-size threshold circuit if there is no restriction on the energy, (i) any threshold circuit computing PARITY needs exponential size if its energy is restricted to one; and (ii) any threshold circuit computing PARITY needs sublinear size if its energy is restricted to some constant. We then consider INNER-PRODUCT, and show that, while it is known that INNER-PRODUCT can be computed by a linear-size and depth-3 threshold circuit if there is no restriction on the energy, (iii) a constant-depth and polynomial-weight threshold circuit computing INNER-PRODUCT needs exponential size if its energy is restricted to sublinear. These lower bounds imply that restriction on the energy strongly influences the computation realized by threshold circuits.

The rest of the paper is organized as follows. In Sections 2, we start from the basic beginning: we give definitions of a threshold gate and a threshold circuit, and then give several simple examples. In Section 3, we consider the two target functions, PARITY and INNER-PRODUCT, and show several threshold circuits computing them. In Section 4, we present the two lower bounds on the size of a threshold circuits computing PARITY. In Section 5, we present the lower bound for INNER-PRODUCT. In Section 6, we conclude with some open problems.

2. Preliminaries

In this section, we present basic definitions and some terms on a threshold gate and a threshold circuit. In Sections 2.1 and 2.2, we consider a threshold gate and a threshold circuit, respectively. In Section 2.3, we prove a theorem presented in Section 2.1.

2.1 Threshold gate

We consider a threshold gate with an arbitrary number \( n \) of inputs. A threshold gate \( g \) is identified by integer weights \( w_1, w_2, \ldots, w_n \) and an integer threshold \( t \); we define the output \( g(x) \) of \( g \) as follows: For every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \),

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

(2.1)

where, for any number \( \eta \), \( \text{sign} (\eta) \) is defined to be one if \( \eta \geq 0 \), and zero otherwise. One may think that the weights and threshold in the definition can be easily extended to arbitrary real numbers (e.g., \( 1/3, \sqrt{2}, \pi \), etc.), and a threshold gate with real number weights has more computational power than one with integer weights; that is, there exists a Boolean function that can be computed by a threshold gate with real number weights but cannot by one with integer weights. However, we can safely restrict our attention to only integer weights as in the following theorem whose proof is given in Section 2.3.
Theorem 2.1. Any Boolean function computed by a threshold gate with real number weights can also be computed by one with integer weights.

Thus, throughout the paper, we consider only threshold gates with integer weights. For any number \( w \), let \( |w| \) be the absolute value of \( w \). We then denote by \( W(g) \) the sum of the absolute values of the weights of \( g \): \[ W(g) = \sum_{i=1}^{n} |w_i|. \]

We below present several examples of threshold gates. The following three examples show that a threshold gate computes any of the three basic Boolean operations, AND, OR, and NOT.

Example 1 We give a threshold gate \( g \) computing AND. For every index \( i \), \( 1 \leq i \leq n \), we set the weight \( w_i = 1 \), and \( t = n \) (see Fig. 1). Then, for every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \),
\[ g(x) = \text{sign} \left( \sum_{i=1}^{n} x_i - n \right) = \begin{cases} 1 & \text{if } \forall i: x_i = 1; \\ 0 & \text{otherwise}. \end{cases} \]

Example 2 We give a threshold gate \( g \) computing OR. For every index \( i \), \( 1 \leq i \leq n \), we set the weight \( w_i = 1 \), and \( t = 1 \) (see Fig. 2). Then, for every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \),
\[ g(x) = \text{sign} \left( \sum_{i=1}^{n} x_i - 1 \right) = \begin{cases} 1 & \text{if } \exists i: x_i = 1; \\ 0 & \text{otherwise}. \end{cases} \]

Example 3 We give a threshold gate \( g \) computing NOT. We set the weight \( w_1 = -1 \), and \( t = 0 \) (see Fig. 3). Then, for every \( x_1 \in \{0, 1\} \),
\[ g(x_1) = \text{sign}(-x_1) = \begin{cases} 1 & \text{if } x_1 = 0; \\ 0 & \text{otherwise}. \end{cases} \]

The examples above show that any logic circuit consisting of AND, OR, and NOT gates can be simulated by a threshold circuit, and hence we immediately have \( AC^0 \subseteq TC^0 \).

A threshold gate can compute a Boolean function other than AND, OR, and NOT. Let \( \text{MAJORITY} \) be a Boolean function whose output is one if and only if at least half of the input bits are ones. More formally, we define \( \text{MAJORITY} \) as follows: For every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \), \( \text{MAJORITY}(x) = 1 \) if and only if \( \sum_{i=1}^{n} x_i \) is greater than or equal to \( \lceil n/2 \rceil \). The following example shows that a threshold gate computes \( \text{MAJORITY} \):

Example 4 We give a threshold gate \( g \) computing \( \text{MAJORITY} \). For every index \( i \), \( 1 \leq i \leq n \), we set the weight \( w_i = 1 \), and \( t = \lceil n/2 \rceil \) (see Fig. 4). Then, for every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \),
\[ g(x) = \text{sign} \left( \sum_{i=1}^{n} x_i - \left\lceil \frac{n}{2} \right\rceil \right) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq \left\lceil \frac{n}{2} \right\rceil; \\ 0 & \text{otherwise}. \end{cases} \]

It is worth noting that \( \text{MAJORITY} \) is not contained in \( AC^0 \), which implies that \( AC^0 \subseteq TC^0 \) [21].

Note that the threshold gates given in Examples 1–4 have weights \(-1, 0, 1\), and hence \( W(g) = O(n) \). It is known that exponential weights give more computational power to a threshold gate. Let \( \text{COMPARISON} \) be a Boolean...
function that compares two \( n \) bit integers of binary representation. More formally, we define COMPARISON as follows: For every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \) and \( y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n \), \( \text{COMPARISON}(x, y) = 1 \) if and only if \( \sum_{i=1}^{n} 2^{i-1} x_i \) is greater than or equal to \( \sum_{i=1}^{n} 2^{i-1} y_i \). The COMPARISON function can be computed by a threshold gate \( g \) with \( W(g) = O(2^n) \), as follows.

**Example 5** We give a threshold gate \( g \) computing COMPARISON. Let \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \) and \( y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n \) be inputs to a threshold gate \( g \). For every \( i \), \( 1 \leq i \leq n \), we set the weight \( 2^{i-1} \) for \( x_i \), the weight \( -2^{i-1} \) for \( y_i \), and set \( t = 0 \) (see Fig. 5). Then, for every \( x = (x_1, x_2, \ldots, x_n) \), \( y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n \), we have

\[
g(x, y) = \text{sign}\left( \sum_{i=1}^{n} 2^{i-1} x_i - \sum_{i=1}^{n} 2^{i-1} y_i \right) = \begin{cases} 
1 & \text{if } \sum_{i=1}^{n} 2^{i-1} x_i \geq \sum_{i=1}^{n} 2^{i-1} y_i; \\
0 & \text{otherwise}. 
\end{cases}
\]

Lastly, we show that the following simple theorem captures the limitation of the computational power of a threshold gate. We define PARITY of \( n \) variables as follows: For every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \), \( \text{PARITY}(x) = 1 \) if and only if \( \sum_{i=1}^{n} x_i \) is odd.

**Theorem 2.2.** For any integer \( n \geq 2 \), \( \text{PAR}_n \) cannot be computed by a threshold gate.

**Proof.** We first consider the case of two variables. Suppose for the sake of contradiction that there exists a threshold gate \( g \) computing PARITY of two variables \( x_1 \) and \( x_2 \). Let \( w_1 \) and \( w_2 \) be weights of \( g \) for \( x_1 \) and \( x_2 \), respectively; and let \( t \) be the threshold. Since \( \text{PAR}_2(x) = 1 \) for \( x = (x_1, x_2) = (1, 0) \) or \( x = (x_1, x_2) = (0, 1) \), we have

\[
w_1 - t \geq 0 \text{ and } w_2 - t \geq 0. 
\]

On the other hand, since \( \text{PAR}_2(x) = 0 \) for \( x = (x_1, x_2) = (0, 0) \) or \( x = (x_1, x_2) = (1, 1) \), we have

\[
-t < 0 \text{ and } w_1 + w_2 - 2t < 0.
\]

The inequalities of (2.2) imply that \( w_1 + w_2 - 2t \leq 0 \), while the inequalities of (2.3) imply that \( w_1 + w_2 - 2t < 0 \); this is a contradiction.

Consider then the case of \( n \geq 3 \) variables. Suppose for the sake of contradiction that there exists a threshold gate computing \( \text{PAR}_n \). In this case, we can obtain a threshold gate computing \( \text{PARITY} \) of two variables by putting zeros into the variables \( x_3, x_4, \ldots, x_n \), which contradicts the argument for two variables. Thus we have completed the proof.

### 2.2 Threshold circuit

A **threshold circuit** \( C \) is a combinatorial circuit consisting of threshold gates, and is expressed by a directed acyclic graph. Let \( n \) be the number of inputs to \( C \), and then \( C \) has \( n \) input nodes of in-degree 0, each of which corresponds to one of the \( n \) input variables \( x_1, x_2, \ldots, x_n \); and the other nodes correspond to threshold gates. We define **size** \( s \) of a threshold circuit \( C \) as the number of threshold gates in \( C \).

Let \( C \) be a threshold circuit of size \( s \), let \( g_1, g_2, \ldots, g_s \) be the gates in \( C \), and let \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \) be an input to \( C \). Then the input \( z_i \) to a gate \( g_i \), \( 1 \leq i \leq s \), either consists of the inputs \( x_1, x_2, \ldots, x_i \) to \( C \) and the outputs of the gates other than \( g_i \) (or part of them). Let \( g_i \) be one of the gates of out-degree 0, and we regard the output \( g_i(x) \) of \( g_i \) as the **output** \( C(x) \) of \( C \). Thus, \( C(x) = g_s(x) \) for every input \( x \in \{0, 1\}^n \). The gate \( g_i \) is called the **top gate** of \( C \). We say that a threshold circuit \( C \) **computes** a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) if \( C(x) = f(x) \) for every input \( x \in \{0, 1\}^n \).

We say that a gate \( g_i \), \( 1 \leq i \leq s \), is in the \( l \)-th layer of a circuit \( C \) if there are \( l \) gates (including \( g_i \)) on the longest path from an input node to \( g_i \) in the underlying graph of \( C \). The **depth** \( d \) of \( C \) is the number of gates on the longest path to the output gate \( g \). We define the **energy complexity** \( e \) of \( C \) as

\[
e = \max_{x \in \{0, 1\}^n} \sum_{i=1}^{s} g_i(x).
\]

Clearly \( 0 \leq e \leq s \) holds.

For a circuit \( C \), we define \( W(C) \) as the maximum value of \( W(g) \), where the maximum is taken over all the gates \( g \) in \( C \): \( W(C) = \max_{1 \leq i \leq s} W(g_i) \). We say that \( C \) is of **polynomial-weight** if \( W(C) \) is bounded by a polynomial in the number \( n \) of input variables.
Throughout the paper, we denote by \( n \) the number of input variables to a threshold circuit \( C \), by \( s \) the size of \( C \), by \( d \) the depth of \( C \), and by \( e \) the energy complexity of \( C \). We may assume without loss of generality that \( n, s, d, e \geq 1 \).

We below observe some examples of threshold circuits. We define SUM-EQUALITY of \( 2n \) variables as follows: For every \( x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \) and \( y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n \), SUM-EQUALITY\( (x, y) = 1 \) if and only if \( \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \). One can construct a threshold circuit computing SUM-EQUALITY as below.

**Example 6** We construct a threshold circuit \( C \) of three gates \( g_1, g_2 \) and \( g_3 \) that computes SUM-EQUALITY. The gate \( g_1 \) has weights \(+1\) for every \( x_i \), \( 1 \leq i \leq n \), and \(-1\) for every \( y_i \), \( 1 \leq i \leq n \) together with threshold zero. That is, we have

\[
g_1(x) = \text{sign} \left( \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \right).
\]

The gate \( g_2 \) has weights \(-1\) for every \( x_i \), \( 1 \leq i \leq n \), and \(+1\) for every \( y_i \), \( 1 \leq i \leq n \) together with threshold zero:

\[
g_2(x) = \text{sign} \left( -\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i \right).
\]

We can easily observe that both of \( g_1 \) and \( g_2 \) output ones if and only if \( \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \). The top gate \( g_3 \) then computes AND of the two outputs; \( g_3 \) receives the outputs of \( g_1 \) and \( g_2 \) with weights one and threshold two (see Fig. 6).

Clearly, \( C \) computes SUM-EQUALITY function; and has size three, depth two and energy three.

We then show in the following example that any Boolean function can be computed by a threshold circuit of extremely small energy, energy one, if exponential size is allowed. For any set \( S \), we define \( \#S \) as the cardinality of \( S \).

**Example 7** Let \( f \) be an arbitrary Boolean function of \( n \) variables. We show that \( f \) can be computed by a threshold circuit \( C \) of energy one. Let

\[
S_0(f) = \{ x \in \{0, 1\}^n \mid f(x) = 0 \}.
\]

Then, for every input assignment \( x = (x_1, x_2, \ldots, x_n) \in S_0(f) \), we construct a threshold gate \( g_x \) in the first layer so that \( g_x \) has threshold \( \sum_{i=1}^{n} x_i \) and weight

\[
w_i = \begin{cases} 1 & \text{if } x_i = 1; \\ -1 & \text{otherwise} \end{cases}
\]

for every \( i, 1 \leq i \leq n \). Clearly, the output of \( g_x \) is one if and only if each corresponding input bit equals to \( x_i \) for every \( i, 1 \leq i \leq n \). We then complete the construction of \( C \) by adding the top gate \( g \) that has threshold zero and receives every output of \( g_x \) for \( x \in S_0 \) with weight \(-1\). Clearly, \( C \) computes \( f \) and has size \( \#S_0(f) + 1 \leq 2^n + 1 \) and depth two. Moreover, for input \( x \in S_0(f) \), only the gate \( g_x \) in the first layer outputs one and the top gate \( g \) outputs zero, while, for input \( x \in \{0, 1\}^n \setminus S_0(f) \), no gate in the first layer outputs one and the top gate \( g \) outputs one. Thus energy of \( C \) is one. Figure 7 depicts such a circuit for a Boolean function \( f \) of three variables such that \( S_0(f) = \{(1,0,0),(1,0,1), (0,1,1),(1,1,1)\} \).

We can formalize the statement in Example 7 as follows.

**Theorem 2.3.** Any Boolean function \( f \) of \( n \) variables can be computed by a threshold circuit of size at most \( 2^n + 1 \), depth two and energy one.

### 2.3 Proof of Theorem 2.1

In this section, we prove Theorem 2.1 by construction: We show that any given threshold gate with real number weights can be simulated by a threshold gate with integer weights. Let \( g \) be a threshold gate with weights \( w_1, w_2, \ldots, w_n \) and threshold \( t \), where \( w_1, w_2, \ldots, w_n \) and \( t \) are arbitrary real numbers:
Since \( n > /C_{21} \), we have

\[
\begin{align*}
\text{Fig. 7. Circuit of energy one that computes } f : \{0, 1\}^3 \to \{0, 1\} \text{ such that } S_0(f) = \{(1, 0, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)\}.
\end{align*}
\]

Let \( p(x) \) be the value inside the sign function in Eq. (2.5), that is,

\[
p(x) = \sum_{i=1}^{n} w_i x_i - t.
\]  \hspace{1cm} (2.6)

Without loss of generality, we can assume that \( p(x) \neq 0 \) for every \( x \in \{0, 1\}^n \); if this is not the case, we can add a sufficiently small real number to the threshold. We show that there is a threshold gate \( g^* \) which computes the same Boolean function as \( g \) does, and has integer weights \( w_1^*, w_2^*, \ldots, w_n^* \) and threshold \( t^* \).

Let

\[
\delta = \min_{x \in \{0, 1\}^n} |p(x)|.
\]  \hspace{1cm} (2.7)

We define \( w_i' = w_i / \delta \) for each \( i, 1 \leq i \leq n \), and \( t' = t / \delta \). Let

\[
p'(x) = \sum_{i=1}^{n} w_i' x_i - t' = \frac{1}{\delta} \left( \sum_{i=1}^{n} w_i x_i - t \right).
\]

Since \( \delta > 0 \), we have \( p(x) \geq 0 \) if and only if \( p'(x) \geq 0 \); moreover, (2.7) implies that

\[
|p'(x)| \geq 1.
\]  \hspace{1cm} (2.8)

We then define \( w_i'' = (n + 1) w_i' \) for each \( i, 1 \leq i \leq n \), and \( t'' = (n + 1) t' \). Let

\[
p''(x) = \sum_{i=1}^{n} w_i'' x_i - t'' = \frac{n + 1}{\delta} \left( \sum_{i=1}^{n} w_i x_i - t \right).
\]

Since \( n \geq 1 \), we have \( p'(x) \geq 0 \) if and only if \( p''(x) \geq 0 \); moreover, we have, by (2.8),

\[
|p''(x)| \geq n + 1
\]  \hspace{1cm} (2.9)

for every \( x \in \{0, 1\}^n \).

We finally give the desired weights and threshold as follows: \( w_i^* = \lfloor w_i'' \rfloor \) for each \( i, 1 \leq i \leq n \), and \( t^* = \lfloor t'' \rfloor \). Clearly,

\[
w_i^* - 1 \leq w_i^* \leq w_i'' \text{ and } t_i^* - 1 \leq t'' \leq t''.
\]  \hspace{1cm} (2.10)

Let

\[
p^*(x) = \sum_{i=1}^{n} w_i^* x_i - t^*.
\]  \hspace{1cm} (2.11)

It suffices to show that \( p(x) \geq 0 \) if and only if \( p^*(x) \geq 0 \). We consider the following two cases.

**Case 1: \( p(x) > 0 \)**

In this case, we have, by the inequalities of (2.10),

\[
p^*(x) = \sum_{i=1}^{n} w_i^* x_i - t^*
\]

\[
\geq \sum_{i=1}^{n} (w_i^* - 1) x_i - t'' - 1
\]

\[
\geq \sum_{i=1}^{n} w_i'' x_i - t'' - (n + 1)
\]

\[
= p''(x) - (n + 1).
\]
Since the assumption and (2.9) imply that \( p'(x) \geq n + 1 \), we have \( p^*(x) \geq 0 \) as desired.

**Case 2:** \( p(x) < 0 \)

Similarly to the other case, the inequalities of (2.10) imply that

\[
p^*(x) = \sum_{i=1}^{n} w_i^* x_i - t^*
\]

\[
\leq \sum_{i=1}^{n} w_i^* x_i - t^*
\]

\[
= p^*(x).
\]

Since \( p(x) \geq 0 \) if and only if \( p'^*(x) \geq 0 \), the assumption implies \( p^*(x) < 0 \).

### 3. Target Functions: PARITY and INNER-PRODUCT

In this section, we give definitions of the two Boolean functions, PARITY and INNER-PRODUCT, for which we derive lower bounds. We also present several threshold circuits computing PARITY and INNER-PRODUCT. These examples help us observe that we have many circuits for PARITY and INNER-PRODUCT with various size, depth, and energy.

#### 3.1 Circuits for PARITY

For every \( x \in \{0, 1\}^n \), we denote by \(|x|\) the number of ones in \( x \). As mentioned in the last section, we abbreviate PARITY of \( n \) variables as \( PAR_n \), and define

\[
PAR_n(x) = \begin{cases} 
1 & \text{if } |x| \equiv 1 \pmod{2} \\
0 & \text{otherwise}
\end{cases}
\]

for every \( x \in \{0, 1\}^n \). In this section, we present three circuits computing PARITY: The first one is of size \( n + 1 \), depth two, and energy at most \( 3 \). The second is of size \( \lceil \log n \rceil + 1 \), depth \( \lceil \log n \rceil + 1 \), and energy \( \lceil \log n \rceil + 1 \); and the third is of size \( n + 1 \), depth \( n + 1 \), and energy two.

**Example 8** We present a threshold circuit \( C \) computing \( PAR_n \), where its size is \( n + 1 \), its depth is two, and its energy is at most \( n + 1 \). The first layer of the circuit \( C \) contains \( n \) gates \( g_1, g_2, \ldots, g_n \), where the gate \( g_j, 1 \leq j \leq n \), has weight one for every \( x_i, 1 \leq i \leq n \), and threshold \( j \) for every \( x \in \{0, 1\}^n \),

\[
g_j(x) = \text{sign}(|x| - j).
\]

Clearly, \( g_j \) outputs one if and only if \( |x| \geq j \). The top gate, denoted by \( g_{n+1} \), has threshold one, and receives the output of \( g_j \) with weight

\[
w_j = \begin{cases} 
1 & \text{if } j \text{ is odd}; \\
-1 & \text{otherwise}
\end{cases}
\]

for each \( j, 1 \leq j \leq n \). Thus,

\[
g_{n+1}(x) = \text{sign}\left( \sum_{j=1}^{n} w_j g_j(x) - 1 \right).
\]

See Fig. 8 depicting the circuit \( C \).

Let us confirm that \( C \) computes \( PAR_n \). For any input \( x \in \{0, 1\}^n \), the gates \( g_1, g_2, \ldots, g_{|x|} \) output ones, and hence we have, for every \( x \in \{0, 1\}^n \),

\[
\sum_{j=1}^{n} w_j g_j(x) = \begin{cases} 
1 & \text{if } |x| \equiv 1 \pmod{2}; \\
0 & \text{otherwise}.
\end{cases}
\]

Since \( g \) has threshold one, \( C \) computes \( PAR_n \). Clearly, \( C \) has size \( n + 1 \) and depth two. Every gate outputs one for \( x = (1, 1, \ldots, 1) \) if \( n \) is odd, and hence \( C \) has energy at most \( n + 1 \).

**Example 9** We next present a threshold circuit \( C \) computing \( PAR_n \) that has size \( \lceil \log n \rceil + 1 \), depth \( \lceil \log n \rceil + 1 \) and energy \( \lceil \log n \rceil + 1 \). To simplify the presentation, we only consider the case for \( n = 2^k \) for some positive integer \( k \). We say that a threshold circuit \( C \) is **regular** if every gate in \( C \) receives each of \( x_1, x_2, \ldots, x_n \) with weight one. We construct the desired regular threshold circuits by induction on \( k \).

Consider the base case where \( k = 1 \); that is, we have \( n = 2 = 2^1 \) variables \( x_1 \) and \( x_2 \). In this case, we construct the desired circuit \( C_1 \) consisting of two threshold gates \( g_1 \) and \( g_2 \). Each of \( g_1 \) and \( g_2 \) has weight one for each of the two input variables, but \( g_1 \) has threshold one, while \( g_2 \) has threshold two. We complete the construction of \( C_1 \) by connecting the output of \( g_2 \) to \( g_1 \) with weight \(-2\) (see Fig. 9(a)). Clearly, \( C_1 \) is a regular circuit of size \( \log 2 + 1 = 2 \), and of depth...
log 2 + 1 = 2. Moreover, we have \( g_1(x) = g_2(x) = 0 \) for \( x = (x_1, x_2) = (0, 0) \), and \( g_1(x) = 0 \) and \( g_2(x) = 1 \) for \( x = (x_1, x_2) = (1, 1) \); while \( g_1(x) = 1 \) and \( g_2(x) = 0 \) for \( x = (x_1, x_2) = (0, 1) \) and \( (x_1, x_2) = (1, 0) \). Thus, \( C_1 \) computes \( \text{PAR} \) of two variables, and has energy \( \log 2 + 1 = 2 \).

Let \( k \) be an integer such that \( k \geq 2 \), and \( n = 2^k \). By induction hypothesis, we have a regular threshold circuit \( C_{k-1} \) computing \( \text{PAR} \) of \( 2^{k-1} = n/2 \) variables \( x_1, x_2, \ldots, x_{n/2} \) such that \( C_{k-1} \) is of size \( \log n \), depth \( \log n \) and energy \( \log n \). Using \( C_{k-1} \), we below construct the desired threshold circuit \( C_k \) computing \( \text{PAR}_n \) of \( 2^k = n \) input variables: \( x_1, x_2, \ldots, x_{n/2} \) together with \( x_{n/2+1}, x_{n/2+2}, \ldots, x_n \).

We first construct an intermediate circuit \( C' \) from \( C_{k-1} \) by connecting each of \( x_{n/2+1}, x_{n/2+2}, \ldots, x_n \) to every gate in \( C_{k-1} \) with weight one. Note that \( C' \) and \( C_{k-1} \) are both regular, and hence, if \( 0 \leq |x| \leq 2^{k-1} \) holds for \( x \in [0, 1]^n \), the output of every gate in \( C' \) equals that of the corresponding gate in \( C_{k-1} \) for any input \( x' \in [0, 1]^{n/2} \) satisfying \( |x| = |x'| \). Therefore, for every \( x \in [0, 1]^n \) satisfying \( |x| \leq 2^{k-1} \),

\[
C(x) = C_{k-1}(x') = \text{PAR}_{n/2}(x') = \text{PAR}_n(x)
\]

where \( x' \in [0, 1]^{n/2} \) satisfies \( |x| = |x'| \).

Let \( g_k \) be a threshold gate with threshold \( 2^{k-1} \) and weight one for each of the \( n \) input variables, \( x_1, x_2, \ldots, x_n \):

\[
g_k(x) = \text{sign}(x) - 2^{k-1}.
\]

We then construct the desired circuit \( C_k \) by combining \( C' \) and \( g_k \) as follows: We connect the outputs of \( g_k \) to every gate in \( C' \) with weight \(-2^{k-1}\) (see Fig. 9(b)). Clearly, \( C_k \) is a regular threshold circuit; and, since \( C_{k-1} \) (and hence also \( C' \)) is of size \( \log n \) and depth \( \log n \), \( C_k \) has size \( \log n + 1 \) and depth \( \log n + 1 \). Since \( C_k \) has size \( \log n + 1 \), \( C_k \) trivially has energy \( \log n + 1 \).

Let us confirm that \( C_k(x) = \text{PAR}_n(x) \) for every \( x \in [0, 1]^n \). We verify the claim by considering the following two cases: \( |x| \leq 2^{k-1} - 1 \), and \( |x| \geq 2^{k-1} \). For the former case, the gate \( g_k \) outputs zero; and hence we have \( C_k(x) = C'(x) \). Thus, (3.1) imply that \( C_k(x) = \text{PAR}_n(x) \), as desired. For the latter case, the gate \( g_k \) outputs one; and hence, every gate originally in \( C' \) receives \(-2^k\) from \( g_k \). Consequently, since \( C' \) is regular, the output of every gate in \( C' \) for \( x \) equals to that for any input \( x' \in [0, 1]^n \) satisfying...
and hence (3.1) imply that
\[ C_k(x) = C_0(x_0) = \text{PAR}_n(x_0). \]  

Besides, since \( 2^{k-1} \) is even, (3.2) implies that
\[ |x| \equiv |x| - 2^{k-1} \equiv |x'| \pmod{2}, \]
and hence
\[ \text{PAR}_n(x) = \text{PAR}_n(x'). \]  

Consequently, (3.3) and (3.4) imply \( C_k(x) = \text{PAR}_n(x) \), as desired. \( \square \)

**Example 10**  
We lastly give a threshold circuit \( C \) computing \( \text{PAR}_n \) that has size \( n + 1 \), depth \( n + 1 \) and energy just two. Similarly to Example 9, we obtain \( C \) by inductively constructing gates \( g_1, g_2, \ldots, g_{n+1} \).

Firstly, we construct a threshold gate \( g_n \) which has a threshold \( n \) and receives an input \( x_i \) with weight one for every \( i \), \( 1 \leq i \leq n \):
\[ g_n(x) = \text{sign}(-n + |x|). \]

We then inductively construct a threshold gate \( g_k \) from \( n - 1 \) to 1, as follows: the gate \( g_j \) has threshold \( j \), and receives an input \( x_i \) with weight one for every \( i \), \( 0 \leq i \leq n \); moreover, \( g_j \) receives an output of \( g_{j'} \) with weight \( j' \) for every \( j < j' \), \( 1 \leq j' \leq n \) (see Fig. 10). That is, for each \( j \), \( 0 \leq j \leq n - 1 \), we have
\[ g_j(x) = \text{sign}(-j + |x| - \sum_{j' = j+1}^{n} j' \cdot g_{j'}(x)). \]

We use the following simple claim with regard to the gates \( g_1, g_2, \ldots, g_n \).

**Claim.** For every \( x \in \{0, 1\}^n \), \( g_{|x|}(x) = 1 \) while \( g_j(x) = 0 \) for every \( j \), \( 0 \leq j \leq |x| - 1 \) and \( |x| + 1 \leq j \leq n \).

**Proof.** We denote by \( p_j(x) \) the value in the sign function of the gate \( g_j \) for \( x \):
\[ p_j(x) = \begin{cases} 
-|x| & \text{if } j = n; \\
-j + |x| - \sum_{j' = j+1}^{n} j' \cdot g_{j'}(x) & \text{if } 1 \leq j \leq n - 1. 
\end{cases} \]  

(3.5)

For each \( j \), \( |x| + 1 \leq j \leq n \), we have \( p_j(x) \leq -j + |x| \), and hence \( g_j(x) = 0 \) which implies that
\[ p_{|x|}(x) = -|x| + |x| = 0, \]
and hence \( g_{|x|}(x) = 1 \). Consequently, for each \( j \), \( 0 \leq j \leq |x| - 1 \),
We then give another polynomial-size threshold circuit of slightly smaller energy: a threshold circuit of size \(O(x)\) outputs one when \(x\) is sufficiently small [24].

The claim implies that \(x\) is odd if and only if only the gate \(g_j\) for some odd \(j\), \(1 \leq j \leq n\), outputs one for \(x \in \{0, 1\}^n\).

We complete the construction by adding the top gate \(g_{n+1}\) computing OR of outputs of the gates \(g_j\) for every odd \(j\), \(1 \leq j \leq n\): \(g_{n+1}\) has threshold one and weight one for every output of the gate \(g_j\) for every odd \(j\) (see Fig. 10). By the claim above, \(C\) computes \(PAR_n\) and has energy two. Clearly, \(C\) has size \(n + 1\) and depth \(n + 1\).

There are some known lower bounds for threshold circuits computing \(PAR_n\). Sin et al. proved that any depth-\((d + 1)\) threshold circuit \(C\) must have size at least \(dn^{1/d}/\log^2 n\) if \(W(C)\) is sufficiently small [24]. Besides, Impagliazzo et al. obtained a lower bound \((n/2)^{1/(2d-1)}\) even if \(W(C)\) is arbitrarily large [11].

### 3.2 Circuits for INNER-PRODUCT

We abbreviate INNER-PRODUCT of \(2n\) variables as \(IP_n\), and define

\[
IP_n(x, y) = \left\{ \begin{array}{ll}
1 & \text{if } \sum_{i=1}^{n} x_i y_i \equiv 1 \pmod{2}; \\
0 & \text{otherwise}.
\end{array} \right.
\]

for every pair of \(x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n\).

Similarly to the last section, we present two threshold circuits computing \(IP_n\) each of which has different size, depth and energy.

**Example 11** We first present a threshold circuit \(C^*\) of size \(2n + 1\), depth three and energy \(2n + 1\). The first layer of \(C^*\) contains \(n\) gates \(h_1, h_2, \ldots, h_n\) such that, for every \(i, 1 \leq i \leq n\), \(h_i\) computes AND of \(x_i\) and \(y_i\): \(h_i\) has threshold two and weights one for \(x_i\) and \(y_i\). Thus we have

\[
h_i(x, y) = \text{sign}(x_i + y_i - 2).
\]

Let \(C\) be the depth-2 threshold circuit computing \(PAR_n\) given in Example 8. Thus, \(C\) has the gates \(g_1, g_2, \ldots, g_n\) in the first layer, and the top gate \(g_{n+1}\) in the second layer. We simply combine \(C\) with \(h_1, h_2, \ldots, h_n\) so that each of \(g_1, g_2, \ldots, g_n\) receives, instead of the input variables, every output of \(h_1, h_2, \ldots, h_n\) with weights one (see Fig. 11). The resulting circuit \(C^*\) clearly computes \(IP_n\), and has size \(n + (n + 1) = 2n + 1\) and depth three. Since every gate in \(C^*\) outputs one when \(x = (1, 1, \ldots, 1)\) and \(y = (1, 1, \ldots, 1)\), \(C\) has energy \(2n + 1\).

**Example 12** We then give another polynomial-size threshold circuit of slightly smaller energy: a threshold circuit \(C^*\) of size \(O(n^3/\log n)\), depth four, and energy \(O(n/\log n)\). Let \(I = \lfloor \log n \rfloor\) and \(m = \lceil n/I \rceil\), and \(S_1, S_2, \ldots, S_m\) be an arbitrary partition of the set \(\{1, 2, \ldots, n\}\), where \(\#S_j \leq I\) for every \(1 \leq j \leq m\). For every \(j, 1 \leq j \leq m\), we consider the following Boolean function: for every \(x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n\),

\[
f_j(x, y) = \left\{ \begin{array}{ll}
1 & \text{if } \sum_{i \in S_j} x_i y_i \equiv 1 \pmod{2} \\
0 & \text{otherwise}.
\end{array} \right.
\]

(3.6)
Since $f_j$ is a function depending on at most $2l$ variables, Theorem 2.3 implies that we can construct a threshold circuit $C_j$ computing $f_j$ such that its size is at most $2^{2l} + 1 = O(n^3)$, its depth is two, and its energy is one.

Let $C$ be a threshold circuit computing $\text{PARITY}$ of $m$ variables which we construct similarly to the one given in Example 8. We then construct $C^*$ by combining $C$ with $C_1, C_2, \ldots, C_m$ so that each of $g_1, g_2, \ldots, g_m$ in $C$ receives, instead of the input variables, every output of $C_1, C_2, \ldots, C_m$ with weights one (see Fig. 12). Since we have

$$\text{IP}_n(x, y) = \sum_{j=1}^{m} f_j(x, y) \pmod{2},$$

the resulting circuit computes $\text{IP}_n$. Clearly, $C^*$ has size $O(n^2) \cdot m + (m + 1) = O(n^3/\log n)$, depth four, and energy $m + m + 1 = O(n/\log n)$.

We note that $\text{INNER-PRODUCT}$ is often used in the literature to derive lower bounds on the size of depth-2 or depth-3 threshold circuits. Hajnal et al. proved that any depth-2 threshold circuit $C$ computing $\text{IP}_n$ has exponential size if the top gate of $C$ has polynomial weight [9]. Håstad and Goldmann obtained an exponential lower bound for depth-3 threshold circuits $C$ such that every gate in $C$ has polynomial weight, and every gate in the first layer of $C$ has a bounded number of inputs [10]. Razborov and Wigderson generalized the result of [10], and show that any depth-3 threshold circuit computing $\text{IP}_n$ has size $\Omega(n^{\log n})$ if every gate in $C$ has polynomial weight, and every gate in the first layer of $C$ are $\text{AND}$ gates. Forster et al. then considered depth-2 threshold circuits of which the gate in the first layer has polynomial weight, and obtained an exponential lower bound [7].

4. Lower Bounds for $\text{PARITY}$

In this section, we present two lower bounds for $\text{PARITY}$. In Section 4.1, we prove that a threshold circuit needs an exponential size if its energy is one. In Section 4.2, we give a lower bound for threshold circuits of energy $e \geq 2$. In Section 4.3, we prove a technical lemma presented in Section 4.2.

4.1 Circuits of energy one

In Example 10 of the last section, we observe that $\text{PAR}_n$ can be computed by a threshold circuit of size $O(n)$, depth $O(n)$ and energy two. The following simple question then comes up: is $\text{PAR}_n$ computable by a threshold circuit of polynomial (or even linear) size and energy one? The following theorem gives a negative answer to this question.

**Theorem 4.1.** If a threshold circuit $C$ of energy one computes $\text{PAR}_n$, then the size of $C$ is at least $2^{n-1}$.

We know from Example 7 that, for any Boolean function $f$ of $n$ variables, one can construct a threshold circuit of size $\#S_0(f) + 1$ and energy one, where $S_0(f) = \{ x \in \{0, 1\}^n | f(x) = 0 \}$. Since $\#S_0(\text{PAR}_n) = 2^{n-1}$, a threshold circuit of size $2^{n-1} + 1$ and energy one computes $\text{PAR}_n$. Theorem 4.1 thus implies that the construction given in Example 7 gives an almost optimal circuit. In the rest of the section, we prove Theorem 4.1. The proof is similar to the one for Theorem 2.2.

Let $C$ be a threshold circuit of size $s$ and energy $e = 1$ that computes $\text{PAR}_n$. We denote by $g_1, g_2, \ldots, g_s$ the gates in $C$, and assume that $g_1, g_2, \ldots, g_s$ are topologically ordered with respect to the underlying directed acyclic graph of $C$. Then $g_s$ is the top gate of $C$. Let
Suppose for the sake of contradiction that \( s < \#S_0(\text{PAR}_n) - 1 \). Since \( e = 1 \) and the top gate \( g_t \) outputs zero for any input \( x \in S_0(\text{PAR}_n) \), either exactly one of \( g_1, g_2, \ldots, g_{t-1} \) outputs one or none of the gates outputs one for any \( x \in S_0(\text{PAR}_n) \). Since \( s \leq \#S_0(\text{PAR}_n) - 1 \), the pigeonhole principle implies that there exists a pair of inputs \( x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in S_0(\text{PAR}_n) \) that satisfies one of the following two conditions:

(i) there exists only one index \( k, 1 \leq k \leq s - 1 \), such that \( g_k(x) = g_k(y) = 1 \);

(ii) none of \( g_1, g_2, \ldots, g_s \) outputs one for each of \( x \) and \( y \).

Thus, it suffices to derive a contradiction for each of (i) and (ii). We below give a proof only for the case (i), since the proof for the other case is similar.

Since \( g_1, g_2, \ldots, g_s \) are topologically ordered, the inputs of \( g_k \) consist of some of the outputs of \( g_1, g_2, \ldots, g_{k-1} \) and \( x_1, x_2, \ldots, x_n \). However, since \( g_k(x) = g_k(y) = 1 \) and \( e = 1 \), the gates \( g_1, g_2, \ldots, g_{k-1} \) output zeros for each of \( x \) and \( y \). Let \( w_i, 1 \leq i \leq n \), be the weight of \( g_k \) for the \( i \)th input variables, and let \( t_k \) be the threshold of \( g_k \). Then, since \( g_k \) outputs one for \( x \) and \( y \), we have

\[
g_k(x) = \text{sign} \left( \sum_{i=1}^{n} w_i x_i - t_k \right) = 1
\]

and

\[
g_k(y) = \text{sign} \left( \sum_{i=1}^{n} w_i y_i - t_k \right) = 1,
\]

which imply that

\[
\sum_{i=1}^{n} w_i x_i - t_k \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} w_i y_i - t_k \geq 0.
\]

Summing the two inequalities, we obtain

\[
\sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} w_i y_i - 2t_k \geq 0. \tag{4.2}
\]

On the other hand, since \( x \not= y \), there must exist an index \( j \) such that \( x_j \not= y_j \). Consider a pair of inputs \( x' \) and \( y' \) obtained from \( x \) and \( y \) by exchanging the \( j \)th component of \( x \) for the counterpart of \( y \):

\[
x' = (x_1, x_2, \ldots, x_{j-1}, y_j, x_{j+1}, \ldots, x_n)
\]

and

\[
y' = (y_1, y_2, \ldots, y_{j-1}, x_j, y_{j+1}, \ldots, y_n). \tag{4.3}
\]

Both \(|x|\) and \(|y|\) are even and we have either \( 0 = x_j \not= y_j = 1 \) or \( 1 = x_j \not= y_j = 0 \), and hence both \(|x'|\) and \(|y'|\) are odd. Thus, \( g_k(x') = g_k(y') = 1 \), and, since \( e = 1 \), we have \( g_k(x') = g_k(y') = 0 \) for every \( j, 1 \leq j \leq s - 1 \). Consequently, \( g_k(x') = g_k(y') = 0 \) for every \( j, 1 \leq j \leq k \). We thus have

\[
g_k(x') = \text{sign} \left( \sum_{i=1}^{n} w_i x_i - w_j x_j + w_j y_j - t_k \right) = 0
\]

and

\[
g_k(y') = \text{sign} \left( \sum_{i=1}^{n} w_i y_i - w_j y_j + w_j x_j - t_k \right) = 0,
\]

which imply that

\[
\sum_{i=1}^{n} w_i x_i - w_j x_j + w_j y_j - t_k < 0 \quad \text{and} \quad \sum_{i=1}^{n} w_i y_i - w_j y_j + w_j x_j - t_k < 0.
\]

Summing the two inequalities, we obtain

\[
\sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} w_i y_i - 2t_k < 0,
\]

which contradicts (4.2).
4.2 Circuits of energy $e \geq 2$

In the last section, we observe that, if a threshold circuit of energy one computes $\text{PAR}_n$, it requires an exponential size in $n$. In this section, we consider the case where energy $e \geq 2$. Recall that, in Example 10, we construct a threshold circuit of size $n + 1$ and energy two. The next theorem rules out the possibility that we can much improve the circuit for the case of energy two.

**Theorem 4.2.** Let $C$ be a threshold circuit computing $\text{PAR}_n$ variables. If $C$ has energy $e$, then the size $s$ of $C$ satisfies

$$s \geq \frac{e(n + 1)^{1/e}}{c}$$

(4.5)

for some positive constant $c$.

Thus, if $C$ has constant energy, the lower bound on the size of $C$ is sublinear. In particular, if $C$ has energy two, then $C$ must have size $\Omega(\sqrt{n})$. Below, we provide a proof of Theorem 4.2.

Let a threshold circuit $C$ consist of gates $g_1, g_2, \ldots, g_s$, and let $g_s$ be the top gate of $C$: $g_s(x) = C(x)$ for every $x \in \{0, 1\}^n$. For an input $x \in \{0, 1\}^n$, we define a pattern $p_C(x) \in \{0, 1\}^s$ of $C$ for $x$ as

$$p_C(x) = (g_1(x), g_2(x), \ldots, g_s(x)).$$

We denote by $P(C)$ the set of all the patterns that arise in $C$:

$$P(C) = \{p_C(x) \in \{0, 1\}^s \mid x \in \{0, 1\}^n\}.$$

The number $|P(C)|$ is closely related to the size $s$ and the energy $e$. Using Stirling's formula [5, p. 102, p. 122], one can easily prove the following upper bound on $|P(C)|$.

$$|P(C)| \leq \left(\frac{c \cdot x}{e}ight)^x$$

(4.6)

where $c$ is some positive constant.

For a threshold circuit $C$ with $n \geq 2$ inputs, we denote by $C_0$ a circuit obtained from $C$ by fixing the $n$-th input $x_n$ of $x = (x_1, x_2, \ldots, x_n)$ to the constant 0. One can obtain $C_0$ from $C$ by deleting the $n$-th input node for $x_n$ and all the wires linked from the node. Note that the circuit $C_0$ has $n - 1$ inputs, but the size of $C_0$ is the same as that of $C$, because we do not delete any gate in $C$. Let

$$X_0 = \{(x_1, x_2, \ldots, x_{n-1}) \in \{0, 1\}^{n-1} \mid x_n = 0\}.$$

Then, for every input $x = (x_1, x_2, \ldots, x_{n-1}) \in \{0, 1\}^{n-1}$ to $C_0$, we denote by $\text{ext}(x) = (x_1, x_2, \ldots, x_{n-1}, 0) \in X_0$ an input to $C$. Note that the number of ones in $x \in \{0, 1\}^{n-1}$ equals to that in $\text{ext}(x)$, and we have $p_{C_0}(x) = p_C(\text{ext}(x))$. Therefore,

$$P(C_0) = \{p_{C_0}(x) \mid x \in \{0, 1\}^{n-1}\} = \{p_C(\text{ext}(x)) \mid x \in X_0\} \subseteq P(C).$$

(4.7)

If a threshold circuit $C$ computes $\text{PAR}_n$, then clearly $C_0$ computes $\text{PAR}_{n-1}$. We now have the following key lemma on $P(C)$ and $P(C_0)$.

**Lemma 4.3.** Let $n \geq 2$ be an integer. If a threshold circuit $C$ computes $\text{PAR}_n$, then

$$|P(C_0)| + 1 \leq |P(C)|.$$  

(4.8)

We will prove the lemma in the next section. From Lemma 4.3, one can easily derive a linear lower bound on $|P(C)|$.

**Lemma 4.4.** If a threshold circuit $C$ computes $\text{PAR}_n$, then

$$n + 1 \leq |P(C)|.$$  

(4.9)

**Proof.** For the inductive base, we consider $n = 1$. Clearly, there exists a pair of inputs $x, y \in \{0, 1\}^n$ such that $|x|$ is even and $|y|$ is odd. Thus $|P(C)| \geq 2 = n + 1$, and hence (4.9) holds.

For the inductive hypothesis, we assume that $n \geq 2$ and that (4.9) holds for every threshold circuit computing $\text{PAR}_{n-1}$. Let $C$ be an arbitrary threshold circuit computing $\text{PAR}_n$. Since the circuit $C_0$ obtained from $C$ computes $\text{PAR}_{n-1}$, the induction hypothesis implies that

$$|P(C)| \geq (n - 1) + 1 = n \leq |P(C_0)|.$$  

(4.10)

Lemma 4.3 and (4.10) yields

$$n + 1 \leq |P(C_0)| + 1 \leq |P(C)|,$$

as required. □
By (4.6) and Lemma 4.4, we have

$$n + 1 \leq \#P(C) \leq \left( \frac{e - s}{e} \right)^e,$$

which clearly implies Theorem 4.2.

We recommend readers to verify that a threshold circuit $C$ given in Example 8 satisfies that $P(C) \leq n + 1$. Thus, the bound in Lemma 4.4 is, in fact, the best possible. Moreover, the bound given in Theorem 4.2 is almost optimal, as shown in the following theorem [26].

**Theorem 4.5** ([26]). *PARITY of $n \geq 2$ variables is computable by a threshold circuit of energy at most $e$ and size

$$\left( e - 1 \right) \left[ \left( \frac{n + 1}{2} \right)^{1/(e-1)} \right].$$

Lastly, we provide a lower bound on the size of a threshold circuit of unbounded energy, as a byproduct of Lemma 4.4. The following corollary implies that a threshold circuit given in Example 9 has an almost optimal size. We mention that this gives an alternative proof of a lower bound on the size of an arbitrary threshold circuit computing PARITY obtained by Wegener [31].

**Corollary 4.6.** Any threshold circuit computing PAR$_n$ has size $\log(n + 1)$.

Proof. Let $C$ be an arbitrary threshold circuit that computes PAR$_n$ and has size $s$. Since a pattern of $C$ is of length $s$, we trivially have

$$\#P(C) \leq 2^s.$$  

Therefore, Lemma 4.4 and (4.13) imply

$$n + 1 \leq 2^s;$$

and the statement follows. $\square$

### 4.3 Proof of Lemma 4.3

In this section, we give a proof of Lemma 4.3. Suppose that a threshold circuit $C$ computes PAR$_n$, and that $C$ consists of $s$ gates $g_1, g_2, \ldots, g_n$, which are topologically ordered with respect to the underlying directed acyclic graph of $C$. Thus, $g_s$ is the top gate of $C$. For notational simplicity, we write $p(x)$ for $p_C(x)$.

Assume for the sake of contradiction that (4.8) does not hold, that is,

$$\#P(C_0) \geq \#P(C).$$

Since $P(C_0) \subseteq P(C)$, (4.14) implies that

$$P(C_0) = P(C).$$

Let

$$X_1 = \{(x_1, x_2, \ldots, x_n) \in \{0, 1\}^n | x_n = 1\}.$$  

Let $P_1$ be a subset of $P(C)$ such that

$$P_1 = \{p(x) \in P(C) | x \in X_1\}.$$  

Since $P_1 \subseteq P(C)$, we have from (4.15) that

$$P_1 \subseteq P(C_0).$$

In the rest of the section, we derive a contradiction from (4.17).

Let $h = \#P(C)$. We arbitrarily choose $x_0$ from the set $X_1$. The following sequence of inputs to $C$ plays a key role to derive a contradiction:

$$x_0 \rightarrow y_0 \rightarrow x_1 \rightarrow y_1 \rightarrow \cdots \rightarrow x_{h-1} \rightarrow y_{h-1} \rightarrow x_h,$$

where, for every index $j$, we obtain $x_j \in X_1$ and $y_j \in X_0$ by repeatedly applying the following procedures.

**Procedure 1:** $x_j \rightarrow y_j$

Let

$$S = \{y \in X_0 | p(y) = p(x_j)\}.$$  

Since $p(x_j) \in P_1 \subseteq P(C_0)$, we have $S \neq \emptyset$. We then choose an arbitrary input $y_j$ from $S$. Clearly,

$$p(x_j) = p(y_j).$$
**Procedure 2:** $y_j \rightarrow x_{j+1}$

We obtain $x_{j+1} \in X_1$ from $y_j \in X_0$ simply by flipping the $n$-th input of $y_j \in X_0$. Thus, if

$$y_j = (y_1, y_2, \ldots, y_{n-1}, 0),$$

(4.20)

then

$$x_{j+1} = (y_1, y_2, \ldots, y_{n-1}, 1).$$

(4.21)

Repeating the two procedures above $h$ times, we obtain the sequence (4.18). We then consider the following sequence of patterns:

$$p(x_0) \rightarrow p(y_0) \rightarrow p(x_1) \rightarrow p(y_1) \rightarrow \cdots \rightarrow p(x_{h-1}) \rightarrow p(y_{h-1}) \rightarrow p(x_h).$$

(4.22)

Since Procedure 2 implies that $|y_j| + 1 = |x_{j+1}|$, we have $|y_j| \not\equiv |x_{j+1}| \pmod 2$, and hence $g_l(y_j) \neq g_l(x_{j+1})$. Therefore, for every $j$, $0 \leq j \leq h - 1$,

$$p(y_j) \neq p(x_{j+1}).$$

(4.23)

(4.19) and (4.23) imply that the sequence (4.22) contains $h + 1$ patterns $p(x_0), p(x_1), \ldots, p(x_h)$. Since $h = \#P(C)$, there is a pair of indices $l$ and $r$, $0 \leq l < r \leq h$, such that

$$p(x_l) = p(x_r).$$

(4.24)

We now consider the following subsequence of (4.22):

$$p(x_l) \rightarrow p(y_l) \rightarrow p(x_{l+1}) \rightarrow p(y_{l+1}) \rightarrow \cdots \rightarrow p(x_{r-1}) \rightarrow p(y_{r-1}) \rightarrow p(x_r).$$

(4.25)

The inequality (4.23) implies that, for each $j$, $l \leq j \leq r - 1$, there exists at least one gate that output $b \in \{0, 1\}$ for $y_j$ but output the complement $\overline{b}$ of $b$ for $x_{j+1}$. Let $g_l$ be the gate with the smallest index among all such gates. Thus, for every index $j$, $l \leq j \leq r - 1$,

$$g_l(y_j) \neq g_l(x_{j+1}).$$

(4.26)

and for every index $k$, $1 \leq k \leq i_j - 1$,

$$g_l(y_j) = g_l(x_{j+1}).$$

(4.27)

Moreover, we define $i_l, l \leq t \leq r - 1$, as the smallest index among $i_l, i_{l+1}, \ldots, i_{r-1}$. Then for all indices $j$, $l \leq j \leq r - 1$, and $k, 1 \leq k \leq i_j - 1$, we have

$$g_l(x_j) = g_l(y_j) = g_l(x_{j+1}).$$

(4.28)

Thus the outputs of the gates $g_1, g_2, \ldots, g_{n-1}$ are same throughout the sequence (4.25). Therefore, we can conclude that the $n$-th input node $x_n$ of $C$ is directly connected to the gate $g_b$ with non-zero weight; otherwise, the output of $g_b$ never changes.

Let $w_n$ be the weight of $g_b$ for $x_n$. If

$$g_b(x_l) \neq g_b(x_r),$$

(4.29)

then (4.29) contradicts (4.24). Thus it suffices to prove (4.29). We consider the following two cases.

**Case 1:** $w_n > 0$.

When applying Procedure 1, (4.19) implies that the output of $g_b$ does not change. When applying Procedure 2, Eqs. (4.20) and (4.21) imply that the sum of the products of inputs and weights of the gate $g_b$ increases by exactly $w_n > 0$, since the input to the gate $g_b$ consists of (part of) the outputs of $g_1, g_2, \ldots, g_{n-1}$ and (part of) the inputs $x_1, x_2, \ldots, x_n$, and (4.28) implies that the outputs of $g_1, g_2, \ldots, g_{n-1}$ do not change throughout the sequence (4.25). Thus, for every $j$, $l \leq j \leq r - 1$, we have

$$g_b(y_j) \leq g_b(x_{j+1}).$$

(4.30)

Consequently, Eqs. (4.26) and (4.30) imply that

$$0 = g_b(x_l) = g_b(y_j) \neq g_b(x_{j+1}) = 1.$$

(4.31)

Furthermore, Eqs. (4.30) and (4.31) imply that

$$g_b(x_j) = 0$$

(4.32)

for every $j$, $l \leq j \leq t - 1$, and

$$g_b(x_j) = 1$$

(4.33)

for every $j$, $t + 2 \leq j \leq r$. Equations (4.31)–(4.33) imply that

$$0 = g_b(x_l) \neq g_b(x_r) = 1,$$

and hence (4.29) holds.
Case 2: $w_n < 0$.
Similarly to Case 1, since $w_n < 0$, we have
\[ g_i(y) \geq g_i(x_{j+1}) \]  (4.34)
for every $j, l \leq j \leq r - 1$. Thus, (4.26) and (4.34) imply that
\[ 1 = g_0(x_t) = g_0(y_t) \neq g_0(x_{t+1}) = 0, \]
while (4.34) and (4.35) imply that
\[ g_0(x_{j}) = 1 \]  (4.36)
for every $j, l \leq j \leq t - 1$, and
\[ g_0(x_{j}) = 0 \]  (4.37)
for every $j, t + 2 \leq j \leq r$. By Eqs. (4.35)–(4.37), we have
\[ 1 = g_i(x) \neq g_i(x_r) = 0, \]
and hence (4.29) holds.

5. Lower Bound for INNER-PRODUCT

In this section, we derive a lower bound for INNER-PRODUCT. In Section 5.1, we give a theorem implying the lower bound, and prove the theorem using a technical lemma. In Section 5.2, we prove the technical lemma.

5.1 Circuits of polynomial weight

In the last section, we have focused on threshold circuits $C$ computing PARITY, and observe that any lower bound on the size of $C$ is exponential if $C$ has energy one, while it is sublinear if $C$ has energy $e \geq 2$. In this section, we prove that a threshold circuits $C$ needs, in some case, exponential size to compute INNER-PRODUCT even if $C$ has energy $e \geq 2$. We give a lower bound in the following theorem; recall that $W(C)$ is the maximum number of $W(g)$, where the maximum is taken over all the gates $g$ in $C$.

Theorem 5.1. Let $C$ be a threshold circuit computing $IP_n$. Then, the size $s$, depth $d$ and energy $e$ of $C$ satisfy
\[ s = \Omega \left( \frac{2^{n^2}}{(2W(C) + 1)^{1/2(e+1)^{d+1}}} \right). \]  (5.1)

The theorem immediately implies the following corollary.

Corollary 5.2. Let $C$ be a polynomial-weight threshold circuit computing $IP_n$. Then, the size $s$ of $C$ satisfies
\[ s = 2^{O(n/(e+1)^{d+1})}. \]

Thus, if $(e + 1)^{d+1} = n^{o(1)}$ (e.g., $e = O(\log n)$ and $d = O(\log \log n)$), we obtain an exponential lower bound for polynomial-weight threshold circuits. In the rest of the section, we prove Theorem 5.1.

Let $C$ be an arbitrary threshold circuit that computes $IP_n$ and has size $s$, depth $d$, and energy $e$. The first step of our proof is to obtain a set of depth-2 threshold circuits from $C$. The following technical lemma gives its details, whose proof will be given in the next section.

Lemma 5.3. Assume that $IP_n$ can be computed by a threshold circuit $C$ of size $s$, depth $d$ and energy $e$. Let
\[ m = \sum_{i=0}^{s} \binom{s}{i}. \]  (5.2)
Then $IP_n$ can be represented as
\[ IP_n(x, y) = \text{sign} \left( \sum_{j=1}^{m} w_j C_j(x, y) \right), \]  (5.3)
for every $x, y \in \{0, 1\}^n$ by using $m$ threshold circuits $C_1, C_2, \ldots, C_m$ and weights $w_1, w_2, \ldots, w_m$ satisfying the following conditions: for every $x, y \in \{0, 1\}^n$,
\[ 1 \leq \left| \sum_{j=1}^{m} w_j C_j(x, y) \right|; \]  (5.4)
for every $j, 1 \leq j \leq m$, $C_j$ is a depth-2 threshold circuit computing AND of at most $e$ gates, and satisfies $W(C_j) \leq W(C)$; and
\[
\sum_{j=1}^{m} |w_j| \leq (s^e + 1)^{2(s+1)^e}.
\]  
(5.5)

We use Lemma 5.3, and obtain \(m\) threshold circuits \(C_1, C_2, \ldots, C_m\) from \(C\). For each of \(C_1, C_2, \ldots, C_m\), we define sets \(S^{(j)}_0\) and \(S^{(j)}_1\), \(1 \leq j \leq m\), as follows:

\[
S^{(j)}_0 = \{(x, y) \in \{0, 1\}^n \mid C_j(x, y) = 1 \text{ and } IP_n(x, y) = 0\}
\]
and

\[
S^{(j)}_1 = \{(x, y) \in \{0, 1\}^n \mid C_j(x, y) = 1 \text{ and } IP_n(x, y) = 1\}.
\]

The values \(|S^{(j)}_0| - |S^{(j)}_1|, 1 \leq j \leq m\), play key role in our proof: We prove the theorem by providing lower and upper bounds of \(|S^{(j)}_0 - S^{(j)}_1|\). The following lemma give the lower bound, and implies that there exists an index \(j^*\), \(1 \leq j^* \leq m\), such that the output of \(C_{j^*}\) is well-correlated to \(IP_n\) (or the negation of \(IP_n\)).

**Lemma 5.4.** There exists an index \(j^*\), \(1 \leq j^* \leq m\), such that

\[
\frac{2^{2n}}{(s^e + 1)^{2(s+1)^e}} \leq |S^{(j^*)}_1 - S^{(j^*)}_0|.
\]  
(5.6)

**Proof.** Let \(I_1\) and \(I_0\) be sets of inputs \((x, y)\) such that \(IP_n(x, y) = 1\) and \(IP_n(x, y) = 0\), respectively. The inequality (5.4) implies that, for any \((x, y) \in I_1\),

\[
1 \leq \sum_{j=1}^{m} w_j C_j(x, y),
\]
and, for any \((x, y) \in I_0\),

\[
\sum_{j=1}^{m} w_j C_j(x, y) \leq -1.
\]

Therefore, by summing over all the corresponding inputs, we have

\[
\#I_1 \leq \sum_{(x,y) \in I_1} \sum_{j=1}^{m} w_j C_j(x, y)
\]
and

\[
\sum_{(x,y) \in I_0} \sum_{j=1}^{m} w_j C_j(x, y) \leq -\#I_0.
\]

Thus

\[
2^{2n} = \#I_1 + \#I_0 \leq \sum_{(x,y) \in I_1} \sum_{j=1}^{m} w_j C_j(x, y) - \sum_{(x,y) \in I_0} \sum_{j=1}^{m} w_j C_j(x, y)
\]

\[
= \sum_{j=1}^{m} w_j \left( \sum_{(x,y) \in S^{(j)}_0} C_j(x, y) - \sum_{(x,y) \in S^{(j)}_1} C_j(x, y) \right)
\]

\[
\leq \sum_{j=1}^{m} w_j \left( |S^{(j)}_0| - |S^{(j)}_1| \right)
\]

\[
\leq \sum_{j=1}^{m} |w_j| \cdot \left| |S^{(j)}_1| - |S^{(j)}_0| \right|.
\]  
(5.7)

Fix

\[
j^* = \arg \max_{1 \leq j \leq m} \left| |S^{(j)}_1| - |S^{(j)}_0| \right|,
\]
and then (5.7) implies that

\[
2^{2n} \leq \sum_{j=1}^{m} |w_j| \cdot \left| |S^{(j^*)}_1| - |S^{(j^*)}_0| \right|.
\]

Thus, by (5.5), the lemma holds. \(\square\)

Fix the index \(j^*\) given in Lemma 5.4. Then we can obtain the upper bound of \(|S^{(j^*)}_1 - S^{(j^*)}_0|\), as follows.
Lemma 5.5.

\[
\left| \#S^{(r)}_1 - \#S^{(r)}_0 \right| \leq (2W(C) + 1)^y 2^{3n/2}.
\] (5.8)

Proof. Let \( r (\leq e) \) be the number of the gates in the first layer of \( C_r \), and we denote by \( g_1, g_2, \ldots, g_r \) the gates in \( C_r \), that is, \( C_r \) computes AND of the outputs of \( g_1, g_2, \ldots, g_r \). For each \( i, 1 \leq i \leq n \) and \( k, 1 \leq k \leq r \), we denote by \( \alpha_{i,k} \) and \( \beta_{i,k} \) the weight of \( g_k \) for the input \( x_i \) and \( y_i \), respectively, and \( t_k \) the threshold of \( g_k \). Thus

\[
g_k(x, y) = \text{sign}(-t_k + \sum_{i=1}^{n} \alpha_{i,k} x_i + \sum_{i=1}^{n} \beta_{i,k} y_i).
\]

We then partition the set

\[
S^{(r)}_0 \cup S^{(r)}_1 = \{(x, y) \in \{0, 1\}^n \times \{0, 1\}^n \mid C_r(x, y) = 1\}
\]

into a number of sets according to the values \( \sum_{i=1}^{n} \alpha_{i,k} x_i \) and \( \sum_{i=1}^{n} \beta_{i,k} y_i \), as follows. We denote by \([-W(C), W(C)]\) a set of integers between \(-W(C)\) and \( W(C)\). Then, for every \( u = (u_1, u_2, \ldots, u_n) \in [-W(C), W(C)]^n \), we define a set \( X(u) \) and \( Y(u) \) as

\[
X(u) = \left\{ x \in \{0, 1\}^n \mid \forall k : \sum_{i=1}^{n} \alpha_{i,k} x_i = u_k \right\}
\]

and

\[
Y(u) = \left\{ y \in \{0, 1\}^n \mid \forall k : \sum_{i=1}^{n} \beta_{i,k} y_i \geq t_k - u_k \right\}.
\]

Clearly, \( X(u) \times Y(u) \) for every \( u \in [-W(C), W(C)]^n \) compose a partition of \( S^{(r)}_0 \cup S^{(r)}_1 \). Let

\[
I_1(u) = \{(x, y) \in X(u) \times Y(u) \mid IP_x(x, y) = 1\}
\]

and

\[
I_0(u) = \{(x, y) \in X(u) \times Y(u) \mid IP_x(x, y) = 0\},
\]

then we have

\[
\left| \#S^{(r)}_1 - \#S^{(r)}_0 \right| = \left| \sum_{u \in [-W(C), W(C)]^n} \#I_1(u) - \sum_{u \in [-W(C), W(C)]^n} \#I_0(u) \right| \\
\leq \sum_{u \in [-W(C), W(C)]^n} \left| \#I_1(u) - \#I_0(u) \right|.
\] (5.11)

A trivial bound on the value \( \#I_1(u) - \#I_0(u) \) is \( 2^{2n} \), since we have \( 0 \leq \#I_1(u), \#I_0(u) \leq 2^{2n} \). By a more careful analysis, we can prove that \( \#I_1(u) - \#I_0(u) \) cannot be such large:

Claim 5.6. For any pair of \( X(u) \subseteq [0, 1]^n \) and \( Y(u) \subseteq [0, 1]^n \), define the sets \( I_1(u) \) and \( I_0(u) \) by Eqs. (5.9) and (5.10). Then

\[
\left| \#I_1(u) - \#I_0(u) \right| \leq \sqrt{2^n \sqrt{\#X(u) \#Y(u)}}.
\]

Proof. We prove the lemma by matrix analysis. Let \( H \) be a \( 2^n \times 2^n \) matrix where each row is identified by \( x \in [0, 1]^n \) and each column is by \( y \in [0, 1]^n \). For every \( x, y \in [0, 1]^n \), we set \( H(x, y) = -1 \) if \( IP_x(x, y) = 0 \), and \( H(x, y) = 1 \) if \( IP_x(x, y) = 1 \). Fix an arbitrary \( u \in [-W(C), W(C)]^n \), then we also define a row vector \( 1_X(u) \in [0, 1]^{2^n} \) as the characteristic vector of \( X(u) \), that is, for each \( x \in [0, 1]^n \), the \( x \)-th entry of \( 1_X \) is one if \( x \in X(u) \) and zero otherwise. Similarly, we define a row vector \( 1_Y \) as the characteristic vector of \( Y(u) \). Clearly, we have

\[
\#I_1(u) - \#I_0(u) = \sum_{x \in X(u)} \sum_{y \in Y(u)} H(x, y) = 1_X \cdot H \cdot 1_Y^T,
\]

where \( 1_Y^T \) is the transpose vector of \( 1_Y \). For any row vector \( a \in [0, 1]^n \), we define Euclidian norm of \( a \) as \( \|a\| = \sqrt{a \cdot a^T} \). Cauchy-Schwarz inequality states that, for any row vectors \( a \) and \( b \),

\[
a \cdot b^T \leq \|a\| \cdot \|b\|,
\]

and hence

\[
1_X \cdot H \cdot 1_Y \leq \|1_X \cdot H\| \cdot \|1_Y\|.
\]

Since

\[
\left| \#I_1(u) - \#I_0(u) \right| \leq \sqrt{2^n \sqrt{\#X(u) \#Y(u)}}.
\]
By the definition of the Euclidian norm, we have
\[ |\#I_1(u) - \#I_0(u)| = \|1_X \cdot H\| \cdot \sqrt{\#Y(u)}. \quad (5.12) \]

By the definition of the Euclidean norm, we have
\[ \|1_X \cdot H\| = \sqrt{1_X \cdot H \cdot (1_X \cdot H)^T} = \sqrt{1_X \cdot H \cdot H^T \cdot 1_X^T}. \quad (5.13) \]

Since \( H \) is a symmetric matrix, for any \( x, y \in \{0, 1\}^n \), the \((x, y)\)-th entry of the matrix \( H \cdot H^T \) is
\[ \sum_{z \in \{0, 1\}^n} H(x, z)H^T(z, y) = \sum_{z \in \{0, 1\}^n} H(x, z)H(z, y). \]

Consider the case where \( x = y \), then \( H(x, z) = H(x, y) \) for every \( z \in \{0, 1\}^n \), and hence the entry is clearly \( 2^n \). Consider the other case where \( x \neq y \), then we have \( H(x, z) = H(x, y) \) for exactly half of \( z \)-s, while \( H(x, z) \neq H(x, y) \) for the other half of \( z \)-s; thus, the entry is zero. Consequently, we have
\[ H \cdot H^T = 2^n I \quad (5.14) \]

where \( I \) denotes the \( 2^n \times 2^n \) identity matrix. By Eqs. (5.13) and (5.14),
\[ \|1_X \cdot H\| \leq \sqrt{2^n} \cdot \sqrt{1_X \cdot 1_X^T} = \sqrt{2^n \#X(u)}. \quad (5.15) \]

The claim follows from Eqs. (5.12) and (5.15) \( \square \)

Since both of \( \#X(u) \) and \( \#Y(u) \) are bounded above by \( 2^n \), Claim 5.6 clearly implies that
\[ |\#I_1(u) - \#I_0(u)| \leq 2^{3n/2} \quad (5.16) \]

which is strictly smaller than the trivial upper bound \( 2^{2n} \). By Eqs. (5.11) and (5.16), we have
\[ |\#S_{j_1} - \#S_{0_{j_1}}| \leq (2W(C) + 1)2^{3n/2} \leq (2W(C) + 1)2^{3n/2}, \]

as required. \( \square \)

By combining Lemmas 5.4 and 5.5, we immediately obtain
\[ \frac{2^n}{(s^e + 1)^{2e+1}e^2} \leq (2W(C) + 1)^{2^{3n/2}}, \]

and hence we have
\[ \left( \frac{2^{n/2}}{(2W(C) + 1)^e} \right)^{1/(2e+1)^d} - 1 \leq s. \]

Thus, the theorem follows.

### 5.2 Proof of Lemma 5.3

In this section, we prove Lemma 5.3. Let \( C \) be a threshold circuit computing \( IP_n \), and \( s, d, \) and \( e \) be size, depth and energy of \( C \), respectively. Let \( C \) consist of the gates \( g_1, g_2, \ldots, g_n \), which are topologically ordered with respect to the underlying directed acyclic graph of \( C \), and that \( g_i \) is the top gate of \( C \).

For each index \( j, 1 \leq j \leq s \), we denote by \( w_{j,1}, w_{j,2}, \ldots, w_{j,\alpha} \) the weights of the gate \( g_j \) for the inputs \( x_1, x_2, \ldots, x_n \); and by \( w_{j,\beta}, w_{j,\gamma}, \ldots, w_{j,\delta} \), the weights of \( g_j \) for the outputs of the gates \( g_1, g_2, \ldots, g_s \). Since the gates \( g_1, g_2, \ldots, g_s \) are topologically ordered, we have \( w_{j,\alpha} = 0 \) if \( k > j \), and hence the output of \( g_j \) for \( x \in \{0, 1\}^n \) is given as follows:
\[ g_j(x) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{k=1}^{j-1} w_{j,\beta}g_k(x) - t_j \right). \quad (5.17) \]

For each gate \( g_j, 1 \leq j \leq s \), we denote by \( \text{lev}(g_j) \) the layer of the gate \( g_j \) in \( C \). We now construct threshold circuits and weights satisfying Eqs. (5.2)–(5.5).

Define \( S \) as a family of the subsets \( S \subseteq \{1, 2, \ldots, s\} \) such that each \( S \) contains at most \( e \) elements from \( \{1, 2, \ldots, s\} \):
\[ S = \{ S \subseteq \{1, 2, \ldots, s\} \mid 0 \leq |S| \leq e \}. \]

For every \( S \in S \), we construct a depth-2 threshold circuit \( C_S \) consisting of \( |S| + 1 \) gates as follows. In the first level, the circuit \( C_S \) contains a gate \( g_j^S \) for every index \( j \in S \) that computes
\[ g^S_j(x) = \text{sign} \left( \sum_{i=1}^{n} u_{j,i}x_i + \sum_{k \in S} u_{j,k} - t_i \right) \] (5.18)

for every input \( x \in \{0, 1\}^n \). Note that the second term in the argument of the sign function of (5.18) is just the sum of the weights for \( g_k \) for \( k \in S \). In the second level, \( C_3 \) contains a gate \( g_S \) computing AND of the outputs of the gates \( g^S_k \) for every \( k \in S \). If \( S = \emptyset \), then \( C_3(x) = \text{sign}(0) = 1 \) for every input \( x \in \{0, 1\}^n \).

We then define weight \( w_S \) of \( C_3 \) for each set \( S \in \mathcal{S} \) as follows. For \( S = \emptyset \), we set \( w_S = 1 \). For a set \( S \in \mathcal{S}\setminus\{\emptyset\} \), assume that \( S \) contains the indices \( i_1, i_2, \ldots, i_{k_S} \) such that \( 1 \leq i_1 < i_2 < \ldots < i_{k_S} \leq s \). Then we denote by \( v = (v_1, v_2, \ldots, v_d) \) the weight vector for \( S \), and define

\[ v_l = \# \{ i \in S \mid \text{lev}(g_i) = l \} \]

for each index \( l, 1 \leq l \leq d \). Clearly, we have \( 0 \leq v_l \leq e \), for every \( l, 1 \leq l \leq d \). Then we define \([S]\) as the integer whose \((e + 1)\)-nary representation of \( d \) digits is \( v \):

\[ [S] = \sum_{l=1}^{d} v_l(e + 1)^{d-l}. \]

The weight \( w_S \) for \( C_3 \) is defined to be

\[ w_S = \begin{cases} \begin{array}{ll}
  m^{[S]} & \text{if } g_S \in S; \\
 -m^{[S]} & \text{otherwise},
\end{array} \end{cases} \] (5.19)

where

\[ m = \#S = \sum_{k=0}^{e} \binom{s}{k}. \]

Consequently, we obtain the following threshold function:

\[ \text{sign} \left( \sum_{S \in \mathcal{S}} w_S C_3(x) \right). \] (5.20)

In the rest of the section, we prove that the threshold function (5.20) satisfies Eqs. (5.2)–(5.5). Since

\[ m = \#S = \sum_{k=0}^{e} \binom{s}{k}, \]

(5.2) holds. Moreover, it holds that

\[ [S] \leq \sum_{l=1}^{d} e(e + 1)^{d-l} \leq (e + 1)^d \]

for every \( S \in \mathcal{S} \). Since \( m \leq s^e + 1 \),

\[ \sum_{S \in \mathcal{S}} |w_S| = \sum_{S \in \mathcal{S}} m^{[S]} \]

\[ \leq (s^e + 1)(s^e + 1)^{(e + 1)d} \]

\[ \leq (s^e + 1)^{2(e + 1)d}. \]

We have thus proved (5.5).

Below, we verify Eqs. (5.3) and (5.4). Consider an arbitrary input \( x \in \{0, 1\}^n \). Let

\[ S^* = \{ j \in \{1, 2, \ldots, s\} \mid \text{the gate } g_j \text{ outputs } 1 \text{ for } x \text{ in the original circuit } C \}. \]

Thus, for every \( j \in S^* \),

\[ g_j(x) = 1, \] (5.21)

while, for every \( j \not\in S^* \),

\[ g_j(x) = 0. \] (5.22)

Let

\[ \mathcal{F} = \{ S \in \mathcal{S} \mid \forall j \in S, \text{ the gate } g^S_j \text{ outputs } 1 \text{ for } x \text{ in } C_3 \}, \]

that is, for every \( S \in \mathcal{F} \),

\[ C_3(x) = g_s(x) = 1, \] (5.23)

while, for every \( S \not\in \mathcal{F} \),
Equations (5.17), (5.18), (5.21), and (5.22) imply that, for every index \( j \in S^* \),
\[
g_j(x) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{k \in S} w_{j,k}g_k(x) - t_j \right). \tag{5.25}
\]
Since \( w_{j,k} = 0 \) for every \( k > j \), we can rewrite (5.25) as
\[
g_j(x) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{j \in S^*} w_{j,k}g_j(x) - t_j \right) = g^S_j(x) \tag{5.26}
\]
and hence (5.21) implies that \( g^S_j(x) = 1 \). We thus have
\[
S^* \subseteq F. \tag{5.27}
\]
Consequently, by Eqs. (5.23) and (5.27),
\[
g_{S^*}(x) = 1. \tag{5.28}
\]
Then the following claim holds.

**Claim 5.7.** For every set \( S \in F \setminus \{S^*\}, |S| \leq |S^*| - 1 \).

**Proof.** Let \( S \) be an arbitrary set in \( F \) such that
\[
S \neq S^*. \tag{5.29}
\]
Thus, for every index \( j \in S \),
\[
g^S_j(x) = 1. \tag{5.30}
\]
For each index \( l, 1 \leq l \leq d \), we define \( S_l \) as a set of indices \( j \) such that the gate \( g_j \) is in the level \( l \) of \( C \), that is,
\[
S_l = \{ j \in S \mid \text{lev}(g_j) = l \}.
\]
Similarly, we define
\[
S^*_l = \{ j \in S^* \mid \text{lev}(g_j) = l \}.
\]
By (5.29), there exists an index \( h \) such that, for every index \( l, 1 \leq l \leq h - 1 \),
\[
S_l = S^*_l \tag{5.31}
\]
and \( S_h \neq S^*_h \). Let \( j \) be an arbitrary index in \( S_h \), and then we have \( w_{j,k} = 0 \) for any \( k \in S_h \cup S_{h+1} \cup \ldots \cup S_d \subseteq S \). Therefore
\[
g^S_j(x) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{k \in S} w_{j,k}g_k(x) - t_j \right) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{k \in S} w_{j,k}g_k(x) - t_j \right). \tag{5.32}
\]
Similarly,
\[
g^S_j(x) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{k \in S^*} w_{j,k}g_k(x) - t_j \right) = \text{sign} \left( \sum_{i=1}^{n} w_{j,i}x_i + \sum_{k \in S^*} w_{j,k}g_k(x) - t_j \right). \tag{5.33}
\]
By Eqs. (5.31), (5.32), and (5.33), we have \( g^S_j(x) = g^S_j(x) \) and hence (5.26) implies that
\[
g^S_j(x) = g_j(x). \tag{5.34}
\]
Therefore, by (5.30) we have \( g(x) = 1 \), and thus \( j \in S^*_h \). Consequently, we have
\[
S_h \subseteq S^*_h. \tag{5.35}
\]
The equation (5.31) implies that for every index \( l, 1 \leq l \leq h - 1 \),
\[
|S_l| = |S^*_l|, \tag{5.36}
\]
and (5.35) implies that
\[
|S_h| \leq |S^*_h| - 1. \tag{5.37}
\]
Let $\mathbf{v} = (v_1, v_2, \ldots, v_d)$ be the weight vector for $S$, and let $\mathbf{v}^* = (v_1^*, v_2^*, \ldots, v_d^*)$ be the weight vector for $S^*$. Then Eqs. (5.36) and (5.37) imply that, for every index $l$, $1 \leq l \leq h - 1$,

$$v_l = v_l^*,$$

and

$$v_h < v_h^*.$$

Thus, $\mathbf{v}^*$ is lexicographically larger than $\mathbf{v}$, and hence the claim follows.

We are now ready to verify Eqs. (5.3) and (5.4). There are two cases to consider: (i) $C(x) = 1$, and (ii) $C(x) = 0$.

Case 1: $C(x) = 1$.

In this case, it suffices to prove that

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \geq 1. \tag{5.38}$$

Equations (5.23), (5.24), and (5.28) imply that

$$\sum_{S \in \mathbb{F}} w_SC_S(x) = w_{S'} + \sum_{S \in \mathbb{F} \setminus \{S'\}} w_SC_S(x) \geq w_{S'} - \sum_{S \in \mathbb{F} \setminus \{S'\}} |w_S|. \tag{5.39}$$

Since $C(x) = g_s(x) = 1$, we have $s \in S^*$, and hence Eqs. (5.19) and (5.39) imply that

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \geq m^{[S']} - \sum_{S \in \mathbb{F} \setminus \{S'\}} m^{[S]}. \tag{5.40}$$

If $\mathbb{F} = \{S'\}$, then we have

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \geq m^{[S']} \geq 1,$$

and hence (5.38) holds. If $\mathbb{F} \neq \{S'\}$, then Claim 5.7 implies that $[S] \leq [S'] - 1$ for every $S \in \mathbb{F} \setminus \{S'\}$. Therefore, by (5.40) we have

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \geq m^{[S']} - (m - 1)m^{[S']-1}$$

$$= m^{[S']-1} \geq 1,$$

and hence (5.38) holds.

Case 2: $C(x) = 0$.

In this case, it suffices to prove that

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \leq -1. \tag{5.41}$$

Similarly as in Case 1, we have

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \leq -w_{S'} + \sum_{S \in \mathbb{F} \setminus \{S'\}} |w_S|$$

$$\leq -m^{[S']} + \sum_{S \in \mathbb{F} \setminus \{S'\}} m^{[S]}. \tag{5.42}$$

If $\mathbb{F}^* = \{S^*\}$, then by (5.42) we have

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \leq -m^{[S']} \leq -1.$$

Otherwise,

$$\sum_{S \in \mathbb{F}} w_SC_S(x) \leq -m^{[S']} + (m - 1)m^{[S']-1}$$

$$= -m^{[S']-1} \leq -1.$$

Hence (5.41) holds.

6. Conclusions and Open Problems

In this paper, we have provided three lower bounds for threshold circuits with regard to the energy complexity. The first and second lower bounds are derived for PARITY: We proved that any threshold circuit $C$ computes PARITY of $n$ variables has size at least $2^{n-1}$ if $C$ has energy one; and that any threshold circuit $C$ computing PARITY of $n$ variables
has size $s = \Omega(e \cdot (n + 1)^{1/e})$ if $C$ has energy $e$ ($\geq 2$). The third lower bound is for INNER-PRODUCT: any threshold circuit $C$ computing INNER-PRODUCT of $2n$ variables has size more than

$$\Omega\left(\frac{2^{n/2}}{(2W(C) + 1)^{1/(2(e + 1)^{(e+1)})}}\right)$$

if $C$ has depth $d$ and energy $e$. As mentioned in Introduction, these statements are simplified from the original ones in order to highlight the proof ideas. In this section, we overview how the lower bounds can be generalized, and also give three open problems with regard to the three lower bounds.

The first lower bound is known to be generalized to MOD$[m]$ for any positive integer $m$, as follows: if a threshold circuit $C$ of energy one computes MOD$[m]$ of $n$ variables, then the size of $C$ is at least $2^{(n-m)/2}$ [26]. We are interested in whether the lower bound can be generalized to the so-called symmetric functions, where a Boolean function is called symmetric if its output value depends only on the number of ones in the input; thus, PARITY, MOD$[m]$ and MAJORITY are all symmetric.

We can also obtain a more generalized version of the second lower bound by almost the same argument: a similar lower bound holds not only for PARITY but for all the symmetric functions [29]. The generalized lower bound is given as follows. Note that a symmetric function $f$ of $n$ variables can be represented by an $(n + 1)$-vector $v(f)$ whose $i$-th entry, $0 \leq i \leq n$, is the output value (0 or 1) of $f$ on an input with $i$ ones.

**Theorem 6.1** ([29]). Let $m_0$ and $m_1$ be the maximum numbers of consecutive 0’s and consecutive 1’s in $v(f)$, respectively, and let $l = \min\{m_0, m_1\}$ and $m = \max\{m_0, m_1\}$. Then, for any threshold circuit $C$ computing $f$, the size $s$ and energy $e$ of $C$ satisfies

$$s = \Omega\left(e \cdot \left(\frac{n + 1 - l}{m}\right)^{1/e}\right),$$

(6.1)

For example, if $f$ is PARITY, then $l = m = 1$, and hence we obtain almost the same bound presented in this paper. We conjecture that (6.1) is far from optimal, and the optimal lower bound will be given in terms of the alternating variables has size more than

**REFERENCES**


