Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 20, no. 1, pp. 115–131 (2016)
DOI: 10.7155/jgaa.00387

Threshold Circuits Detecting Global Patterns in Two-dimensional Maps

Kei Uchizawa[1] *Daiki Yashima*[2] *Xiao Zhou*[2]

[1]Graduate School of Science and Engineering, Yamagata University, Jonan 4-3-16, Yonezawa-shi Yamagata, 992-8510, Japan.
[2]Graduate School of Information Sciences, Tohoku University, Aramaki-aza Aoba 6-6-05, Aoba-ku, Sendai, 980-8579, Japan.

Abstract

In this paper, we consider a biologically-inspired Boolean function P_D^n that models a simple task of detecting global spatial patterns on a two-dimensional map. We prove that P_D^n is computable by a threshold circuit of size (i.e., number of gates) $O(\sqrt{n}\log n)$, which is an improvement on the previous upper bound $O(n)$, while our circuit has larger depth $O(\sqrt{n})$ and total wire length $O(n \log^2 n)$. Moreover, we demonstrate that the size of our circuit is nearly optimal up to a logarithmic factor: we show that any threshold circuit computing P_D^n has size $\Omega(\sqrt{n}/\log n)$.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
March 2015	July 2015	July 2015	September 2015	January 2016
		Published:		
		February 2016		
	Article type:		Communicated by:	
	regular paper		M. S. Rahman and E. Tomita	

The early version of this paper is presented at the 9th International Workshop on Algorithms and Computation (WALCOM 2015).

E-mail addresses: uchizawa@yz.yamagata-u.ac.jp (Kei Uchizawa) yashima.daiki@ec.ecei.ac.jp (Daiki Yashima) zhou@ecei.tohoku.ac.jp (Xiao Zhou)

1 Introduction

A threshold circuit is a combinatorial circuit comprising logic gates computing linear threshold function, and comprises one of the most well-studied computational models in circuit complexity theory. It is known that threshold circuits have surprising computational power: polynomial-size and constant-depth threshold circuits can compute a variety of Boolean functions including basic arithmetic operations such as addition, multiplication, division, sorting, *etc.* [6, 8, 9, 11, 10]. Finding an explicit function that cannot be computed by polynomial-size and constant-depth threshold circuits is a standard approach to resolving the well-known P vs. NP problem, and threshold circuits are currently on its cutting edge (e.g., see [12]).

Threshold circuits have been studied from another perspective: threshold circuits are considered to provide a theoretical model of neural networks in the brain [6, 7, 10]. It is known that a threshold gate, the basic element of a threshold circuit, captures the basic input-output characteristic of a biological neuron. In this line of research, we expect to take a step towards understanding how a neural network realizes complicated information processing (e.g., sensory processing) with remarkably high speed and low energy consumption.

However, classical circuit complexity theory provides little insight; Legenstein and Maass suggested the following two reasons [2]. (1) Conventional complexity theory focuses on a different set of computational problems such as the arithmetic operations above mentioned, and (2) standard complexity measures such as circuit size and depth are *not* tailored to resources that are of primary interest in neuromorphic engineering and the analysis of neural circuits in biological organisms. Motivated by this viewpoint, Legenstein and Maass attempted to resolve the situation by introducing the following setting.

They proposed several biologically-inspired Boolean functions that model simple tasks of visual information processing. They consider the so-called local feature detectors in neural networks, in which a local feature detector is a biological unit detecting the presence of a salient local feature, such as a center that emits higher intensity than its surrounding, a line segment in a particular direction, or even more complex local visual patterns such as an eye or a nose. Intuitively, they define their Boolean functions as problems of asking whether two (or more) types of local feature detectors form a predefined simple configuration. In addition, being motivated by the importance of minimizing wiring in determining brain organization (e.g., see [1] and its references), they propose a complexity measure, total wire length, that is the sum of the lengths of wires in a particular circuit implementation model. They investigate threshold circuits computing their Boolean functions by means of the total wire length [2, 3, 4].

Among the proposed Boolean functions, we focus on one of them, called P_D^n, defined on a two-dimensional map on which local feature detectors of two types reflecting spatial relationship in the outside world are arranged. P_D^n is a simple pattern detection problem querying whether there exists a pair of different local feature detectors satisfying a simple condition: one local feature is detected below and to the left of the other (see Section 2.3 for precise definition).

Ref.	size	depth	total wire length
[2]	$O(n)$	$O(\log n \cdot \log\log n)$	$O(n)$
	$O(n)$	$O\left(\frac{\log n}{\log \Delta}\right)$	$O\left(n \cdot \Delta \cdot \frac{\log n}{\log \Delta}\right)$
	$O(n)$	$O(\log n)$	$O(n)$
Ours	$O(\sqrt{n}\log n)$	$O(\sqrt{n})$	$O(n \log^2 n)$

Table 1: Complexities of prior and our circuits. We omit some advantages of the circuits provided in [2]. The first circuit comprises AND/OR gates with just two inputs and outputs. The second circuit is formulated in terms of Δ, $2 \leq \Delta \leq \sqrt{n}$, which is the maximum number of inputs of the gates. The third circuit comprises threshold gates with at most $O(\log n)$ inputs.

Legenstein and Maass showed that P_D^n is computable by a circuit of reasonably small size, depth, and total wire length [2]. They provide a circuit of size (i.e., number of gates) $O(n)$, depth $O(\log n \cdot \log\log n)$, and total wire length $O(n)$, in which every gate computes the AND or OR of two inputs. They then consider the case in which AND/OR gates are permitted to have larger inputs and total wire length and show that P_D^n is computable by a circuit of size $O(n)$, depth $O(\log n/\log \Delta)$, and total wire length $O((n \cdot \Delta \cdot \log n)/\log \Delta)$ for any integer Δ, $2 \leq \Delta \leq \sqrt{n}$, in which every gate computes AND or OR of at most Δ inputs. They also give a threshold circuit of size (i.e., the number of gates) $O(n)$, depth $O(\log n)$, and total wire length $O(n)$, in which the threshold gates have $O(\log n)$ inputs. (See Table 1.)

Our main result in this paper is an improvement on the size of threshold circuits computing P_D^n. We prove that P_D^n is computable by a threshold circuit of size $O(\sqrt{n}\log n)$, depth $O(\sqrt{n})$, and total wire length $O(n \log^2 n)$. We obtain our circuit by construction, and our proof exhibits the explicit structure of the desired circuit. We note that our circuit comprises a number of copies of subcircuits performing the same task with a different set of inputs.

An optimal size of threshold circuits computing P_D^n is of independent interest. In fact, as a complement to our circuit construction, we further show that any threshold circuit computing P_D^n is of size $\Omega(\sqrt{n}/\log n)$. Thus, the size of our circuit is optimal up to a polylogarithmic factor.

The remainder of the paper is organized as follows. In Section 2, we define threshold circuits and the complexity measures, and provide a formal definition of P_D^n. In Section 3, we show that P_D^n is computable by a threshold circuit of $O(\sqrt{n}\log n)$ gates. In Section 4, we provide a lower bound $\Omega(\sqrt{n}/\log n)$. In Section 5, we finally conclude with some remarks.

2 Definitions

In Section 2.1, we define threshold circuits. In Section 2.2, we define three complexity measures, size, depth, and total wire length. In Section 2.3, we provide the precise definition of P_D^n.

2.1 Threshold Circuits

A *threshold gate* with an arbitrary number z of inputs computes a linear threshold function with z inputs: for every input $\boldsymbol{x} = (x_1, x_2, \ldots, x_z) \in \{0,1\}^z$, the output $g(\boldsymbol{x})$ of a threshold gate g with integer weights $w_1, w_2, \ldots, w_z$ and threshold t is defined as

$$g(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{z} w_i x_i - t\right)$$

where for any number η, $\text{sign}(\eta) = 1$ if $\eta \geq 0$, $\text{sign}(\eta) = 0$, otherwise.

A *threshold circuit* $\mathcal{C}$ is a combinatorial circuit of threshold gates, and is defined by a directed acyclic graph. Let n be the number of input variables to $\mathcal{C}$. Then each node of in-degree 0 in $\mathcal{C}$ corresponds to one of the n input variables $x_1, x_2, \ldots, x_n$, and the other nodes correspond to threshold gates.

Let $\mathcal{C}$ be a threshold circuit with n input variables $x_1, x_2, \ldots, x_n$ and s gates. Let $g_1, g_2, \ldots, g_s$ be the gates in $\mathcal{C}$, which are topologically ordered with respect to the underlying directed acyclic graph of $\mathcal{C}$. We regard the output of g_s as the *output* $C(\boldsymbol{x})$ *of* $\mathcal{C}$, and call the gate g_s the *top gate* of $\mathcal{C}$. A threshold circuit $\mathcal{C}$ *computes* a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ if $\mathcal{C}(\boldsymbol{x}) = f(\boldsymbol{x})$ for every input $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$.

2.2 Complexity Measures for Threshold Circuits

We use three complexity measures, size, depth, and total wire length for threshold circuits. The first two measures are standard in circuit complexity; for a threshold circuit $\mathcal{C}$, the *size* $s(C)$ of $\mathcal{C}$, simply denoted by s, is the number of gates in $\mathcal{C}$, and the *depth* $d(C)$ of $\mathcal{C}$, simply denoted by d, is the number of gates on the longest directed path to the top gate of $\mathcal{C}$ on its underlying directed acyclic graph.

Legenstein and Maass proposed the last measure, total wire length, to evaluate threshold circuits from the biological viewpoint. The measure is based on the following model of a circuit implementation that abstracts the total wire length of a neural network in cortical circuitry. (See [2] and its references for justification. More precisely, we adopt Model (A) in [2].)

We assume that gates, input-ports and output-ports are placed on intersection points on a two-dimensional grid on which two adjacent points are unit-distance apart. We can connect these gates and ports by routing wires in the Euclidean plane; wires can cross and need not run rectilinearly. If a gate g has k inputs, then g occupies a set of k intersection points that are all connected by an undirected wire. Any of these k points can be used to provide one of the k

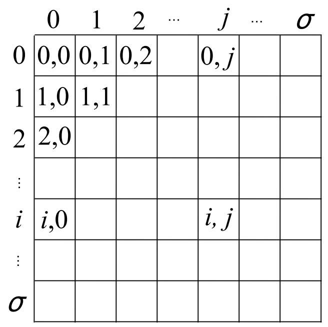


Figure 1: Arrangement of indices of $\boldsymbol{x} = (x_{0,0}, x_{0,1}, \ldots, x_{\sigma,\sigma}) \in \{0,1\}^n$ and $\boldsymbol{y} = (y_{0,0}, y_{0,1}, \ldots, y_{\sigma,\sigma}) \in \{0,1\}^n$ on the 2-dimensional grid.

inputs or one of the outputs of the function computed by g. A wire is permitted to branch and provide inputs to the other gates. We define *total wire length* of a circuit $\mathcal{C}$ as the minimal value of the sum of all the wire lengths of any such arrangement implementing $\mathcal{C}$.

2.3 Target Function P_D^n

Let n be a positive integer. In the rest of the paper, we assume that n is a perfect square, and define $\sigma = \sqrt{n} - 1$. Let $\boldsymbol{x} = \{x_{0,0}, x_{0,1}, \ldots, x_{\sigma,\sigma}\} \in \{0,1\}^n$ and $\boldsymbol{y} = \{y_{0,0}, y_{0,1}, \ldots, y_{\sigma,\sigma}\} \in \{0,1\}^n$ be two input variables, each of which is arranged in an $n \times n$ square. (See Fig. 1.)

The function P_D^n represents a simple task relating global patterns concerning two local features and is defined as

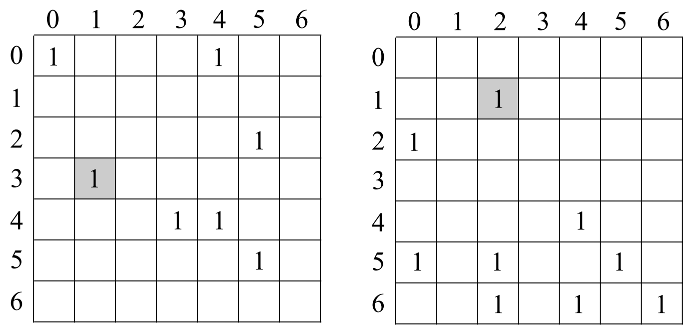
$$P_D^n(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \exists i_1, i_2, j_1, j_2 : x_{i_1,j_1} = y_{i_2,j_2} = 1 \text{ such that } i_1 > i_2 \text{ and } j_1 < j_2; \\ 0 & \text{otherwise} \end{cases}$$

for every pair of $\boldsymbol{x} = (x_{0,0}, x_{0,1}, \ldots, x_{\sigma,\sigma})$ and $\boldsymbol{y} = (y_{0,0}, y_{0,1}, \ldots, y_{\sigma,\sigma})$. In other words, $P_D^n(\boldsymbol{x}, \boldsymbol{y}) = 1$ if and only if there exists a pair of locations x_{i_1,j_1} and y_{i_2,j_2} such that (i) $x_{i_1,j_1} = y_{i_2,j_2} = 1$, and (ii) x_{i_1,j_1} is below and to the left of y_{i_2,j_2}. (See Fig. 2.) The function P_D^n models a task for determining whether a set of locations with $x_{i_1,j_1} = 1$ overlaps those with $y_{i_2,j_2} = 1$.

3 Construction of Circuit

In this section we give an upper bound on the size of threshold circuits computing P_D^n, as in the following theorem.

Theorem 1 *P_D^n is computable by a threshold circuit $\mathcal{C}$ of size $O(\sqrt{n} \log n)$, depth $O(\sqrt{n})$ and total wire length $O(n \log^2 n)$.*



(a) $\boldsymbol{x}$ (b) $\boldsymbol{y}$

Figure 2: Input assignment $(\boldsymbol{x}, \boldsymbol{y})$ for P_D^{49}, where (a) and (b) depict $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Note that we omit 0s in $\boldsymbol{x}$ and $\boldsymbol{y}$ for simplicity. In this case, $P_D^{49}(\boldsymbol{x}, \boldsymbol{y}) = 1$, since $x_{3,1} = 1$ and $y_{1,2} = 1$ (the corresponding locations are shaded).

We prove the theorem by construction. In Section 3.1, we define some terms, and obtain a useful lemma. In Section 3.2, we construct the circuit. In Section 3.3, we evaluate the size, depth and total wire length of the circuit.

3.1 Terms and Idea

Let $\boldsymbol{x} = (x_{0,0}, x_{0,1}, \ldots, x_{\sigma,\sigma}) \in \{0,1\}^n$. For each j, $0 \le j \le \sigma$, consider the j-th column of $\boldsymbol{x}$, and define $\max(\boldsymbol{x}; *, j)$ and $\min(\boldsymbol{x}; *, j)$ as the maximum and minimum indices i, respectively, satisfying $x_{i,j} = 1$:

$$\max(\boldsymbol{x}; *, j) = \max\{i \mid x_{i,j} = 1, 0 \le i \le \sigma\},$$

and

$$\min(\boldsymbol{x}; *, j) = \min\{i \mid x_{i,j} = 1, 0 \le i \le \sigma\};$$

if there is no index i satisfying $x_{i,j} = 1$, we define $\max(\boldsymbol{x}; *, j) = 0$ and $\min(\boldsymbol{x}; *, j) = \sigma$. For example, see Fig. 2(a) where $\max(\boldsymbol{x}; *, 2) = 0$ and $\max(\boldsymbol{x}; *, 5) = 5$, and see Fig. 2(b) where $\min(\boldsymbol{y}; *, 0) = 2$ and $\min(\boldsymbol{y}; *, 2) = 1$.

Let $\boldsymbol{x} = (x_{0,0}, x_{0,1}, \ldots, x_{\sigma,\sigma}) \in \{0,1\}^n$ and $\boldsymbol{y} = (y_{0,0}, y_{0,1}, \ldots, y_{\sigma,\sigma}) \in \{0,1\}^n$ be an arbitrary pair of inputs of P_D^n. Then $P_D^n(\boldsymbol{x}, \boldsymbol{y})$ can be determined by the following lemma.

Lemma 1 $P_D^n(\boldsymbol{x}, \boldsymbol{y}) = 1$ *if and only if there exists an index* j, $0 \le j \le \sigma - 1$, *such that*

$$\min_{j+1 \le q \le \sigma} \min(\boldsymbol{y}; *, q) < \max(\boldsymbol{x}; *, j) \tag{1}$$

Proof: ($\Leftarrow$) Suppose there exists an index j satisfying Eq. (1). Let

$$i_1 = \max(\boldsymbol{x}; *, j) \quad \text{and} \quad i_2 = \min_{j+1 \le q \le \sigma} \min(\boldsymbol{y}; *, q). \tag{2}$$

Clearly,

$$x_{i_1,j} = 1 \tag{3}$$

and there exists an index q, $j + 1 \le q \le \sigma$, satisfying

$$y_{i_2,q} = 1. \tag{4}$$

By Eqs. (1) and (2), we have

$$i_1 > i_2. \tag{5}$$

Furthermore, we have

$$j < j + 1 \le q. \tag{6}$$

Thus, by Eqs. (3)–(6), $P_D^n(\boldsymbol{x}, \boldsymbol{y}) = 1$ holds.

($\Rightarrow$) Suppose $P_D^n(\boldsymbol{x}, \boldsymbol{y}) = 1$, that is, there exist indices i_1, i_2, j_1 and j_2 such that $x_{i_1,j_1} = 1$ and $y_{i_2,j_2} = 1$ satisfying

$$i_1 > i_2 \tag{7}$$

and

$$j_1 < j_2. \tag{8}$$

Since Eq. (8) holds, we have

$$i_1 \le \max(\boldsymbol{x}; *, j_1)$$

and

$$i_2 \ge \min_{j_2 \le q \le \sigma} \min(\boldsymbol{y}; *, q),$$

and hence by Eqs. (7) and (8)

$$\min_{j_1+1 \le q \le \sigma} \min(\boldsymbol{y}; *, q) \le \min_{j_2 \le q \le \sigma} \min(\boldsymbol{y}; *, q) \le i_2 < i_1 \le \max(\boldsymbol{x}; *, j_1)$$

as required when $j = j_1$. □

3.2 Construction of $\mathcal{C}$

Based on Lemma 3.1, we now construct the desired threshold circuit C. Let $\tau = \lceil \log \sqrt{n} \rceil$ for simplicity.

First, for each j, $0 \le j \le \sigma$, we construct a set of τ threshold gates g_j^t, $0 \le t \le \tau - 1$, such that the outputs of τ gates represent $\max(\boldsymbol{x}; *, j)$ in binary system. We utilize a circuit construction used in [2]. For each pair of i, $0 \leq i \leq \sigma$, and t, $0 \leq t \leq \tau$, let

$$p(i,t) = \left\lfloor \frac{i}{2^t} \right\rfloor$$

and

$$u(i,t) = (-1)^{1+p(i,t)} \cdot 2^i. \tag{9}$$

Clearly, if the $(t+1)$st bit of the binary representation of i is 1, then $p(i,t)$ is odd, thus $u(i,t) = 2^i$; and otherwise, $p(i,t)$ is even, thus $u(i,t) = -2^i$. Let j, $0 \leq j \leq \sigma$, be an arbitrarily fixed index. For each t, $0 \leq t \leq \tau - 1$, the gate g_j^t has threshold 1, and receives every input in the jth column: g_j^t receives $x_{i,j}$ with weight $u_{i,t}$ for every i, $0 \leq i \leq \sigma$. Thus, for every $\boldsymbol{x} \in \{0,1\}^n$,

$$g_j^t(\boldsymbol{x}) = \operatorname{sign}\left(-1 + \sum_{i=0}^{\sigma} u(i,t) x_{i,j}\right). \tag{10}$$

Equations (9) and (10) imply that

$$\max(\boldsymbol{x}; *, j) = \sum_{t=0}^{\tau-1} 2^t \cdot g_j^t(\boldsymbol{x}), \tag{11}$$

and thus equals to the $(t+1)$st bit of the binary representation of $\max(\boldsymbol{x}; *, j)$.

Second, for each j, $0 \leq j \leq \sigma$, we similarly construct a set of τ threshold gates h_j^t, $0 \leq t \leq \sigma$, such that the outputs of the τ gates represent $\min(\boldsymbol{y}; *, j)$ in the binary system. For each t, $0 \leq t \leq \tau - 1$, the gate h_j^t has threshold 1, and receives every input in the jth column: h_j^t receives $y_{i,j}$ with weight

$$v(i,t) = (-1)^{1+p(i,t)} \cdot 2^{\sigma - i} \tag{12}$$

for every i, $0 \leq i \leq \sigma$. Thus, for every $\boldsymbol{y} \in \{0,1\}^n$,

$$h_j^t(\boldsymbol{y}) = \operatorname{sign}\left(-1 + \sum_{i=0}^{\sigma} v(i,t) y_{i,j}\right). \tag{13}$$

Then Eqs. (12) and (13) imply that

$$\min(\boldsymbol{y}; *, j) = \sum_{t=0}^{\tau-1} 2^t \cdot h_j^t(\boldsymbol{y}), \tag{14}$$

and hence the output of h_i^t equals to the $(t+1)$st bit of the binary representation of $\min(\boldsymbol{y}; *, j)$.

Using h_j^t, $0 \leq t \leq \tau - 1$, we construct gates $\hat{h}_j^t$, $0 \leq t \leq \tau - 1$, that represent

$$\min_{j \leq q \leq \sigma} \min(\boldsymbol{y}; *, q);$$

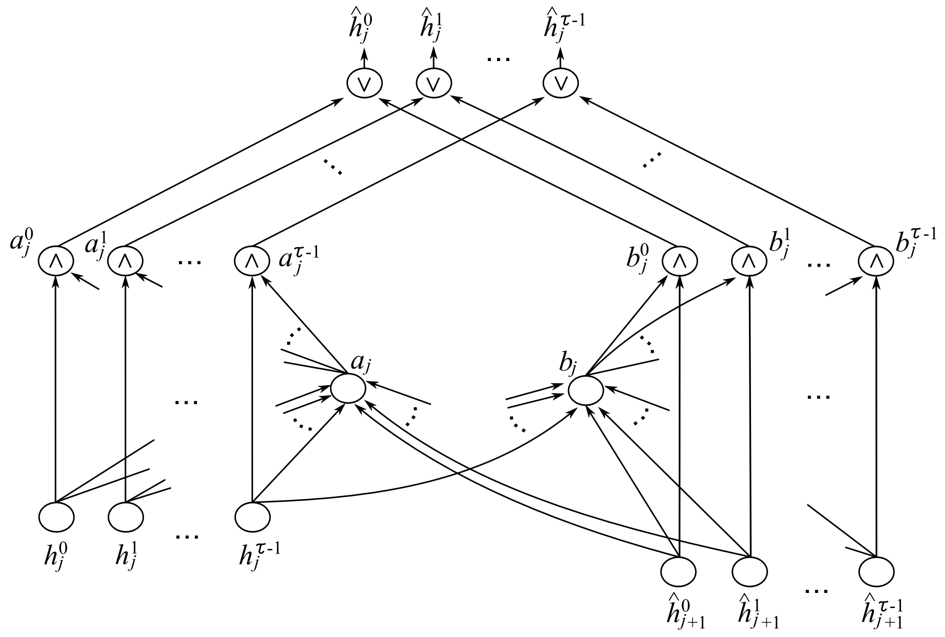


Figure 3: Overview of the circuit for $\hat{h}_j^t$, $0 \le t \le \tau - 1$, where the gates labeled with "$\wedge$" compute the AND of two inputs and the gates labeled with "$\vee$" computes the OR of two inputs.

the construction is inductive on j from σ to 0. For the case in which $j = \sigma$, we do not create any new gate, and simply identify $\hat{h}_\sigma^t$ with h_σ^t for every t, $0 \le t \le \tau - 1$, since

$$\min_{\sigma \le q \le \sigma} \min(\boldsymbol{y}; *, q) = \min(\boldsymbol{y}; *, \sigma).$$

For each j, $\sigma - 1 \ge j \ge 1$, we introduce two gates a_j, b_j and 2τ gates a_j^t and b_j^t, $0 \le t \le \tau - 1$, whose outputs are used for inputs to $\hat{h}_j^t$, $0 \le t \le \tau - 1$. The gates a_j and b_j determine whether $\min(\boldsymbol{y}; *, j)$ is smaller than $\min_{j+1 \le q \le \sigma} \min(\boldsymbol{y}; *, q)$, and are defined as

$$a_j(\boldsymbol{y}) = \text{sign}\left(-1 - \sum_{t=0}^{\tau-1} 2^t h_j^t(\boldsymbol{y}) + \sum_{t=0}^{\tau-1} 2^t \hat{h}_{j+1}^t(\boldsymbol{y})\right)$$

and

$$b_j(\boldsymbol{y}) = \text{sign}\left(\sum_{t=0}^{\tau-1} 2^t h_j^t(\boldsymbol{y}) - \sum_{t=0}^{\tau-1} 2^t \hat{h}_{j+1}^t(\boldsymbol{y})\right).$$

By Eq. (14), a_j outputs 1 if and only if

$$\min(\boldsymbol{y}; *, j) < \min_{j+1 \le q \le \sigma} \min(\boldsymbol{y}; *, q);$$

whereas b_j outputs 1 if and only if

$$\min(\boldsymbol{y};*,j) \geq \min_{j+1\leq q\leq\sigma} \min(\boldsymbol{y};*,q);$$

Thus, exactly one of a_j and b_j outputs 1 for any input. Then, for each t, $0 \leq t \leq \tau-1$, the gate a_l^t computes the AND of the outputs of h_j^t and a_j:

$$a_j^t(\boldsymbol{y}) = \mathrm{sign}(h_j^t(\boldsymbol{y}) + a_j(\boldsymbol{y}) - 2).$$

Clearly, the output of a_j^t is equal to that of h_j^t if $a_j(\boldsymbol{y}) = 1$ (i.e. $\min(\boldsymbol{y};*,j) < \min_{j+1\leq q\leq\sigma}\min(\boldsymbol{y};*,q)$); and equals to 0, otherwise. Similarly, the gate b_j^t computes the AND of the outputs of h_{j+1}^t and b_j:

$$b_j^t(\boldsymbol{y}) = \mathrm{sign}(\hat{h}_{j+1}^t(\boldsymbol{y}) + b_j(\boldsymbol{y}) - 2).$$

The output of b_j^t is equal to that of $\hat{h}_{j+1}^t$ if $b_j(\boldsymbol{y}) = 1$ (i.e. $\min(\boldsymbol{y};*,j) \geq \min_{j+1\leq q\leq\sigma}\min(\boldsymbol{y};*,q)$); and equals to 0, otherwise. For each t, $0 \leq t \leq \tau-1$, we obtain the gate $\hat{h}_j^t$ simply computing the OR of the outputs of a_j^t and b_j^t;

$$\hat{h}_j^t(\boldsymbol{y}) = \mathrm{sign}(a_j^t(\boldsymbol{y}) + b_j^t(\boldsymbol{y}) - 1).$$

Clearly, we have

$$\min_{j\leq q\leq\sigma}\min(\boldsymbol{y};*,q) = \sum_{t=0}^{\tau-1} 2^t \cdot \hat{h}_j^t(\boldsymbol{y}). \tag{15}$$

See Fig. 3 which depicts the circuit for computing $\hat{h}_j^t$, $0 \leq t \leq \tau-1$.

Finally, we construct $\sigma+1$ gates $r_0, r_1, \ldots, r_\sigma$ such that for each j, $0 \leq j \leq \sigma-1$, the gate r_j determines whether there exists an index j satisfying Eq. (1). Equations (11) and (15) imply that we can obtain such r_j, as follows: r_j has threshold 0 and receives the outputs of g_j^t with weight 2^t and $\hat{h}_{j+1}^t$ with weight -2^t for every t, $0 \leq t \leq \tau-1$. More formally,

$$r_j(\boldsymbol{x},\boldsymbol{y}) = \mathrm{sign}\left(-1 - \sum_{t=0}^{\tau-1} 2^t \cdot g_j^t(\boldsymbol{x}) + \sum_{t=0}^{\tau-1} 2^t \cdot \hat{h}_{j+1}^t(\boldsymbol{y})\right).$$

Consequently, Lemma 3.1 implies that $P_D^n(\boldsymbol{x},\boldsymbol{y}) = 1$ if and only if there exists an index j, $0 \leq j \leq \sigma-1$, such that $r_j(\boldsymbol{x},\boldsymbol{y}) = 1$. Therefore, our construction of C is completed by adding the top gate s computing the OR of $r_0, r_1, \ldots, r_{\sigma-1}$:

$$s(\boldsymbol{x},\boldsymbol{y}) = \mathrm{sign}\left(-1 + \sum_{j=0}^{\sigma-1} r_j(\boldsymbol{x},\boldsymbol{y})\right).$$

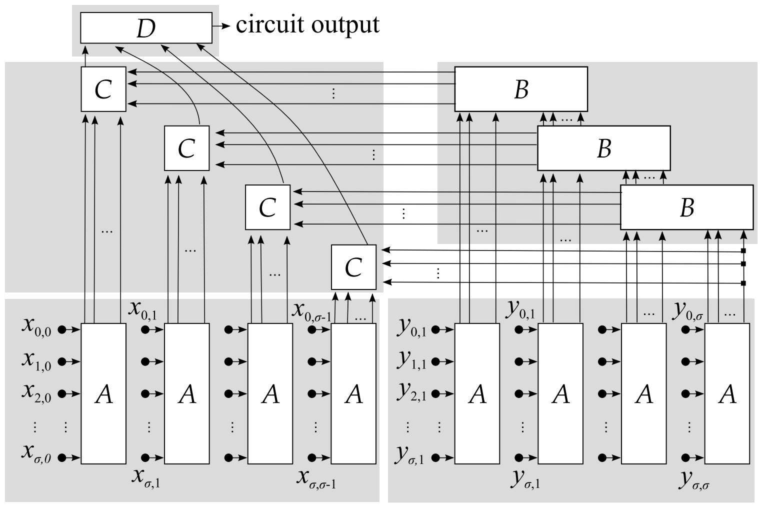


Figure 4: Overview of the layout of $\mathcal{C}$.

3.3 Analysis of $\mathcal{C}$

We now analyze the size, depth and total wire length of $\mathcal{C}$.

For every j, $0 \le j \le \sigma$, we have 3τ gates g^t_j, h^t_j and $\hat{h}^t_j$, $0 \le t \le \tau - 1$. In addition, for every j, $0 \le j \le \sigma$, we have 2τ gates a^t_j and b^t_j, $0 \le t \le \tau - 1$ together with two gates a_j and b_j. Besides, we have $r_0, r_1, \ldots, r_{\sigma-1}$ and s. Consequently, the size of $\mathcal{C}$ is

$$3\tau(\sigma+1) + (2\tau+2)(\sigma+1) + \sigma + 1 = O(\sigma\tau) = O(\sqrt{n}\log n).$$

Moreover, we require at most three layers to obtain $\hat{h}^t_j$, $0 \le t \le \tau - 1$, for each j, $0 \le j \le \sigma$, followed by the two layers containing $r_0, r_1, \ldots, r_{\sigma-1}$ and the top gate s. Thus, the depth is $O(\sigma) = O(\sqrt{n})$.

Finally, we show that the total wire length of $\mathcal{C}$ is $O(n \log n)$. To obtain the desired circuit layout, we use a number of duplicated copies of four basic layouts, A, B and C together with a layout D. An overview of the entire layout depicted with A, B, C, and D is shown in Fig 4. We first evaluate the length of wires within each of the four layouts and then evaluate the length among the layouts.

In the layout A, we place the gates g^t_j, $0 \le t \le \tau - 1$, for every j, $0 \le j \le \sigma$, as follows: we place the inputs $x_{0,j}, x_{1,j}, \ldots, x_{\sigma,j}$, vertically; moreover, to the right of the inputs, we place the gates g^t_j, $0 \le t \le \tau - 1$, each of which occupies σ vertical intersection points (see Fig 5(a)). The length of the wires comprising a gate is $O(\sigma)$; therefore the length for all of the gates is $O(\sigma\tau)$. The length of wires connected to each input is $O(\tau)$; therefore the length for all of the gates is $O(\sigma\tau)$. We consequently have wires of length $O(\sigma\tau)$ in total for each layout. Similarly, the layout A is used to place the gates h^t_j, $0 \le t \le \tau - 1$, for every j,

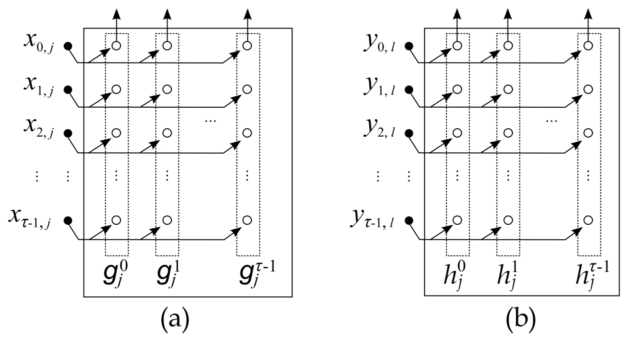


Figure 5: Circuit layout A applied to (a) g_j^t, $0 \leq t \leq \tau - 1$, and to (b) h_j^t, $0 \leq t \leq \tau - 1$.

$0 \leq j \leq \sigma$ (see Fig 5(b)). In total, we have wires of length $O(\sigma\tau)$ within each A. Since we have $O(\sigma)$ copies of the layout A, we have wires of $O(\sigma^2\tau)$ length for the A's.

In the layout B, for every j, $0 \leq j \leq \sigma$, we place the gates a_j, b_j, a_j^t and b_j^t, $0 \leq t \leq \tau - 1$, together with $\hat{h}_j^t$, as follows. We place the gate b_j at the bottom of the layout, and place a_j just above b_j; each of them occupies 2τ horizontal intersection points. We then place a_j^t and b_j^t, $0 \leq t \leq \tau - 1$, each of which occupies two vertical intersection points. Finally, we place $\hat{h}_j^t$, $0 \leq t \leq \tau - 1$, just above a_j^t, $0 \leq t \leq \tau - 1$, each of which occupies two vertical intersection points (see Fig. 6). The length of wires for composing a_j (and b_j) is $O(\tau)$, and the length for the other gates is $O(1)$; therefore, the length for all the gates is $O(\tau)$. The length of wires among a_j, b_j, a_j^t and b_j^t, $0 \leq t \leq \tau - 1$, is clearly $O(\tau)$, whereas the length of wires among b_j^t and $\hat{h}_j^t$, $0 \leq t \leq \tau - 1$, is $O(\tau^2) = O(\log^2 n)$. The length of wires for the outputs of $\hat{h}_j^t$, $0 \leq t \leq \tau - 1$, is $O(\tau^2)$. Consequently, in total, we have wires of length $O(\tau^2)$ within each B. Since we have $O(\sigma)$ copies of the layout B, we have wires of $O(\sigma\tau^2)$ length for the B's.

In the layout C, we place each of the gates r_j, $0 \leq j \leq \sigma - 1$, as follows. We place the gate r_j so that each gate occupies τ horizontal intersection points and τ vertical intersection points (see Fig. 7). The length of wires for composing the gate is clearly $O(\tau)$. Since we use $O(\sigma)$ copies of the layout C, in total, wires of length $O(\sigma\tau)$ for the C's.

In the layout D, we place the gate s as follows. The gate s occupies $O(\sigma)$ horizontal intersection points (see Fig. 8). The length of wires for composing the gate s is clearly $O(\sigma)$, and the length for the output of the gate is $O(1)$. Consequently, in total, we have wires of length $O(\sigma)$ within D.

Consider at last the wires on the outside of the layouts A, B, C and D. The length of a vertical wire between a pair of layouts A and C is $O(\sigma\tau)$; since we have $O(\sigma\tau)$ of such wires, the length of wires for all of the pairs among A's and C's is $O(\sigma^2\tau^2)$. Similarly, we have wires of length $O(\sigma^2\tau^2)$ for all of the

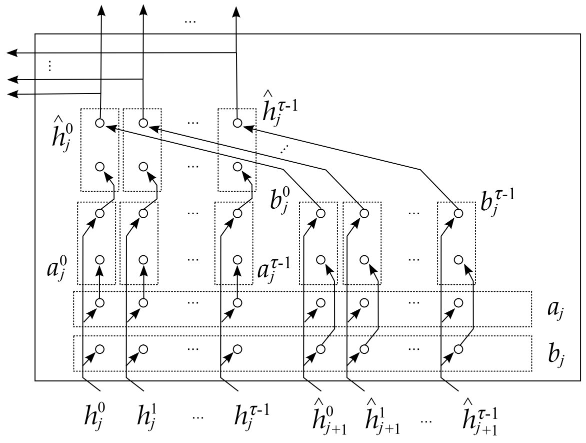


Figure 6: Circuit layout B.

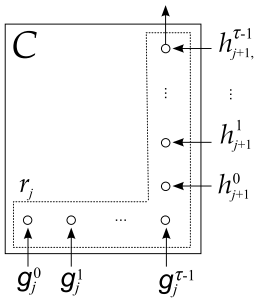


Figure 7: Circuit layout C.

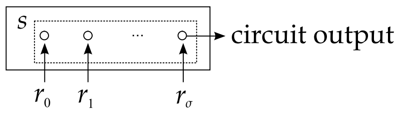


Figure 8: Circuit layout D.

pairs among A's and B's. Since the length of a wire between a pair of B's is $O(1)$, we have wires of length $O(\sigma)$ for all of the pairs among B's. The length of a horizontal wire between a pair of layouts B and C is $O(\sigma\tau)$; since we have $O(\sigma\tau)$ of such wires, the length of wires for all the pairs among B's and C's is $O(\sigma^2\tau^2)$. The length of a wire connecting a C and D pair is $O(\sigma\tau)$; since we have $O(\sigma)$ such wires, the length of wires for all of the pairs is $O(\sigma^2\tau)$. Thus, we have, in total, wires of length $O(\sigma\tau^2)$

Therefore, the layout A has length $O(n\tau)$, B has length $O(\sigma\tau^2)$, C has length $O(\sigma\tau)$, and D has length $O(\sigma)$; the outside of A, B, C and D has length $O(\sigma^2\tau^2)$. Therefore, the total wire length of $\mathcal{C}$ is $O(\sigma^2\tau^2) = O(n\log^2 n)$, as desired.

4 Lower Bound

In this section, we show that the size of the circuit given in Theorem 1 is optimal up to a polylogarithmic factor, as in the following theorem.

Theorem 2 *Let $\mathcal{C}$ be an arbitrary threshold circuit computing P_D^n. Then $\mathcal{C}$ has size $\Omega(\sqrt{n}/\log n)$.*

Let $\mathcal{C}$ be an arbitrary threshold circuit computing P_D^n. We prove the theorem by reducing the disjointness function $DISJ^n$ to our function, where $DISJ^n$ is defined as follows: For every pair of $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \{0,1\}^n$,

$$DISJ^n(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \forall i : x_i = 0 \text{ or } y_i = 0; \\ 0 & \text{otherwise.} \end{cases}$$

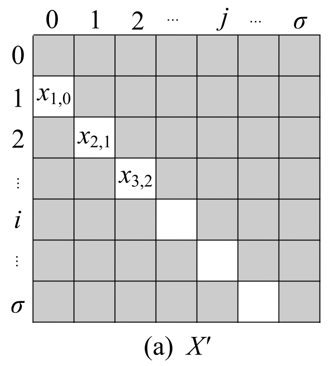
It is known that any threshold circuit computing $DISJ^n$ has almost linear size in n.

Lemma 2 ([5]) *Let $\mathcal{D}^*$ be an arbitrary threshold circuit computing $DISJ^n$. Then $\mathcal{D}^*$ has size $\Omega(n/\log n)$.*

Thus, it suffices to show that we can construct a circuit $\mathcal{D}^*$ computing $DISJ^{O(\sqrt{n})}$ from $\mathcal{C}$ such that $\mathcal{D}^*$ has same size as that of $\mathcal{C}$. We obtain the desired circuit $\mathcal{D}^*$ by fixing some of the input variables of $\mathcal{C}$ to 0s, as follows.

Let

$$X = \{x_{0,0}, x_{0,1}, \ldots, x_{\sigma,\sigma}\}$$



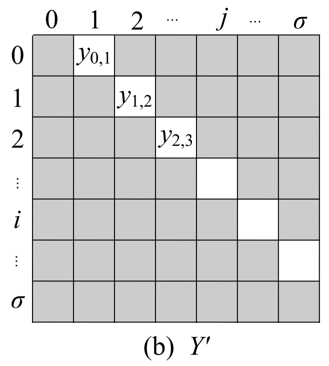


Figure 9: In (a) (and (b), respectively), the unshaded locations indicate X' (and Y'), whereas shaded locations indicate $X\backslash X'$ (and $Y\backslash Y'$).

and

$$Y = \{y_{0,0}, y_{0,1}, \dots, y_{\sigma,\sigma}\}$$

be sets of the input variables $\boldsymbol{x} = (x_{0,0}, x_{0,1}, \dots, x_{\sigma,\sigma})$ and $\boldsymbol{y} = (y_{0,0}, y_{0,1}, \dots, y_{\sigma,\sigma})$ to P_D^n. We define subsets X' and Y' of X and Y as

$$X' = \{x_{i+1,i} \mid 0 \le i \le \sigma - 1\} \subseteq X$$

and

$$Y' = \{y_{i,i+1} \mid 0 \le i \le \sigma - 1\} \subseteq Y.$$

(See Fig. 9.) We then fix every input in $(X\backslash X') \cup (Y\backslash Y')$ of $\mathcal{C}$ to 0. We denote the resulting circuit by $\mathcal{D}$. Clearly, $\mathcal{D}$ computes P_D^n over $X' \cup Y'$. By the definition of P_D^n, $\mathcal{D}^*$ outputs 1 if and only if there exist indices i_x and i_y such that $i_x + 1 > i_y$ and $i_x < i_y + 1$:

$$i_y - 1 < i_x < i_y + 1;$$

therefore $i_x = i_y$. Consequently, $\mathcal{D}$ outputs 1 if and only if there exists an index i_x, $0 \le i \le \sigma - 1$, such that $x_{i+1,i} = y_{i,i+1} = 1$; thus $\mathcal{D}$ computes the complement of $DISJ^\sigma$, that is, $DISJ^{\sqrt{n}-1}$.

We complete the construction of $\mathcal{D}^*$ by replacing the top gate g of $\mathcal{D}$ with a new gate g^* computing its complement. Suppose g has threshold t and weight $w_1, w_2, \dots, w_z$ for a number z of inputs comprising $x_{j+1,j}$ and $y_{j,j+1}$, $0 \le j \le \sqrt{n} - 2$, together with the outputs of the gates in $\mathcal{D}^*$. We then replace g with g^* with threshold $-2t + 1$ and weight $-2w_1, -2w_2, \dots, -2w_z$.

5 Conclusion

In this paper, we consider a Boolean function P_D^n that models a simple information processing task on two-dimensional maps. We demonstrated that P_D^n is computable by a threshold circuit of size $O(\sqrt{n} \log n)$, whereas any threshold circuit computing P_D^n requires size $\Omega(\sqrt{n}/\log n)$. Our circuit has smaller size but greater depth than those given in [2]. It would be interesting to discover a trade-off between the size and depth of threshold circuits computing P_D^n.

References

[1] E. Bullmore and O. Sporns. The economy of brain network organization. *Nature Reviews Neuroscience*, 13(5):336–349, 2012. doi:10.1038/nrn3214.

[2] R. A. Legenstein and W. Maass. Foundations for a circuit complexity theory of sensory processing. *Advances in neural information processing systems*, pages 259–265, 2001.

[3] R. A. Legenstein and W. Maass. Neural circuits for pattern recognition with small total wire length. *Theoretical Computer Science*, 287(1):239 – 249, 2002. Natural Computing. doi:10.1016/S0304-3975(02)00097-X.

[4] R. A. Legenstein and W. Maass. Wire length as a circuit complexity measure. *Journal of Computer and System Sciences*, 70(1):53 – 72, 2005. doi:10.1016/j.jcss.2004.06.001.

[5] N. Nisan. The communication complexity of threshold gates. *Combinatorics, Paul Erdős is Eighty*, 1:301–315, 1993.

[6] I. Parberry. *Circuit Complexity and Neural Networks*. MIT Press, Cambridge, MA, USA, 1994.

[7] J. Šíma and P. Orponen. General-purpose computation with neural networks: A survey of complexity theoretic results. *Neural Computation*, 15(12):2727–2778, 2003.

[8] K.-Y. Siu and J. Bruck. On the power of threshold circuits with small weights. *SIAM Journal on Discrete Mathematics*, 4(3):423–435, 1991. doi:10.1137/0404038.

[9] K.-Y. Siu, J. Bruck, T. Kailath, and T. Hofmeister. Depth efficient neural networks for division and related problems. *Information Theory, IEEE Transactions on*, 39(3):946–956, 1993. doi:10.1109/18.256501.

[10] K.-Y. Siu, V. Roychowdhury, and T. Kailath. *Discrete Neural Computation: A Theoretical Foundation*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[11] K.-Y. Siu and V. P. Roychowdhury. On optimal depth threshold circuits for multiplication and related problems. *SIAM Journal on Discrete Mathematics*, 7(2):284–292, 1994. doi:10.1137/S0895480192228619.

[12] R. Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 194–202. ACM, 2014. doi:10.1145/2591796.2591858.