Complexity of Counting Output Patterns of Logic Circuits

Kei Uchizawa[1] **Zhenghong Wang**[2] **Hiroki Morizumi**[3] **Xiao Zhou**[2]

[1] Graduate School of Science and Engineering
Yamagata University,
Jonan 4-3-16, Yonezawa-shi Yamagata, 992-8510 Japan,
Email: uchizawa@yz.yamagata-u.ac.jp

[2] Graduate School of Information Sciences
Tohoku University,
Aramaki-aza Aoba 6-6-05, Aoba-ku, Sendai, 980-8579, Japan,
Email: wang@nishizeki.ecei.tohoku.ac.jp, zhou@ecei.tohoku.ac.jp

[3] Interdisciplinary Faculty of Science and Engineering
Shimane University,
Nishikawatsu-cho 1060, Matsue, Shimane, 690-8504, Japan,
Email: morizumi@cis.shimane-u.ac.jp

Abstract

Let C be a logic circuit consisting of s gates $g_1, g_2, \ldots, g_s$, then the output pattern of C for an input $\boldsymbol{x} \in \{0,1\}^n$ is defined to be a vector $(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_s(\boldsymbol{x})) \in \{0,1\}^s$ of the outputs of $g_1, g_2, \ldots, g_s$ for $\boldsymbol{x}$. For each $f : \{0,1\}^2 \to \{0,1\}$, we define an f-circuit as a logic circuit where every gate computes f, and investigate computational complexity of the following counting problem: Given an f-circuit C, how many output patterns arise in C? We then provide a dichotomy result on the counting problem: We prove that the problem is solvable in polynomial time if f is PARITY or any degenerate function, while the problem is #P-complete even for constant-depth f-circuits if f is one of the other functions, such as AND, OR, NAND and NOR.

Keywords: Boolean functions, counting complexity, logic circuits, minimum AND-circuits problem.

1 Introduction

Neural circuits in the brain consist of computational units, called *neurons*. Neurons communicate with each other by firing in order to perform various information processing. Many theoretical models of neurons are proposed in the literature, and a circuit consisting of such particular model of neurons is intensively studied (See, for example, a survey (Sima & Orponen 2003)). Among these models, a logic circuit (i.e., a combinatorial circuit consisting of gates, each of which computes a Boolean function) plays a fundamental role; a threshold circuit is an example of such important theoretical models (Parberry 1994, Siu et al. 1995). In these models, gates in a logic circuit C output 0 or 1 for a given input assignment to C, and an output "1" of a gate is considered to represent a "firing" of a neuron. From the viewpoint of neuroscience, such a computation of a logic circuit shows an information coding carried out by a neural circuit: a stimulus to a neural circuit is coded to a set of firing and non-firing neurons, that is, an output pattern. More formally, an output pattern of a logic circuit is defined as follows: For a logic circuit C consisting of s gates $g_1, g_2, \ldots, g_s$ and n input variables, an *output pattern* of C for a circuit input $\boldsymbol{x} \in \{0,1\}^n$ is defined to be a vector $(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_s(\boldsymbol{x})) \in \{0,1\}^s$ of the outputs of $g_1, g_2, \ldots, g_s$ for $\boldsymbol{x}$. In previous research, it turns out that the number of output patterns that arise in a circuit closely related to its computational power: a threshold circuit of $n+1$ patterns can compute the Parity function of n variables, while any threshold circuit of n output patterns cannot (Uchizawa et al. 2011); thus, there exists a gap of computational power between threshold circuits of $n+1$ patterns and those of n patterns. Furthermore, it is known that a threshold circuit C of Γ patterns can be simulated by a threshold circuit C' of $\Gamma + 1$ gates (Uchizawa et al. 2006); thus, if we can decide whether a given circuit C has a small number of patterns, we can construct such C' of small size.

In this paper, we focus more on the number of patterns that arise in a logic circuit. In particular, we investigate the following *counting* problem: Given a circuit C, how many output patterns arise in C? We show that the counting problem can be intractable even for very simple case. Consider a logic circuit C, every gate of which has fan-in two and computes a common function. More specifically, for each $f : \{0,1\}^2 \to \{0,1\}$, we define an *$f$-circuit* as a logic circuit where every gate computes f, and investigate computational complexity of counting output patterns of a given f-circuit. An f-circuit computes an elementary function, but analyzing its computation is not so trivial if it computes a multi-output function. For example, a multi-output $\wedge$-circuit is extensively studied in a context of automated circuit design, and minimizing size of such an $\wedge$-circuit performing a particular task is known to be an APX-hard problem (Arpe & Manthey 2009, Charikar et al. 2005, Morizumi 2011). Let B_2 be a set of all the Boolean functions $f : \{0,1\}^2 \to \{0,1\}$ with two input vari-

This work was supported by JSPS KAKENHI Grant Numbers 23500001, 23700003, 23700020.

f	Complexity
$\mathbf{0}, \mathbf{1}, z_1, z_2, \overline{z_1}, \overline{z_2}, \oplus, \overline{\oplus}$	FP
$\wedge, \vee, \wedge\rceil, \vee\rceil, \lceil\wedge, \lceil\vee$	#P-complete even for depth-2 circuits
$\overline{\wedge}, \overline{\vee}$	#P-complete even for depth-3 circuits

Table 1: Complexity of counting output patterns of a given f-circuit.

ables z_1 and z_2. Clearly, $|B_2| = 2^{2^2} = 16$. We denote the sixteen functions in B_2 as follows:

$$\begin{aligned} \mathbf{0}(z_1, z_2) &= 0, \quad \mathbf{1}(z_1, z_2) = 1, \\ z_1, \quad & z_2, \quad \overline{z_1}, \quad \overline{z_2}, \\ \oplus(z_1, z_2) &= z_1 \oplus z_2, \quad \overline{\oplus}(z_1, z_2) = \overline{z_1 \oplus z_2}, \\ \wedge(z_1, z_2) &= z_1 \wedge z_2, \quad \vee(z_1, z_2) = z_1 \vee z_2, \\ \overline{\wedge}(z_1, z_2) &= \overline{z_1 \wedge z_2}, \quad \overline{\vee}(z_1, z_2) = \overline{z_1 \vee z_2}, \\ \lceil\wedge(z_1, z_2) &= \overline{z_1} \wedge z_2, \quad \lceil\vee(z_1, z_2) = \overline{z_1} \vee z_2, \\ \wedge\rceil(z_1, z_2) &= z_1 \wedge \overline{z_2}, \quad \vee\rceil(z_1, z_2) = z_1 \vee \overline{z_2}. \end{aligned} \tag{1}$$

We make a complete analysis on the problem of computing the number of the output patterns that arise in an f-circuit C for each $f \in B_2$, and provide the dichotomy result as follows: We prove that the problem is solvable in polynomial time if $f \in \{\mathbf{0}, \mathbf{1}, z_1, z_2, \overline{z_1}, \overline{z_2}, \oplus, \overline{\oplus}\}$; while the problem is #P-complete even for constant-depth f-circuits if $f \in \{\wedge, \vee, \wedge\rceil, \vee\rceil, \lceil\wedge, \lceil\vee, \overline{\wedge}, \overline{\vee}\}$ (See Table 1).

The rest of the paper is organized as follows. In Section 2, we define some terms on logic circuits and counting problems. In Section 3, we consider $f \in \{\mathbf{0}, \mathbf{1}, z_1, z_2, \overline{z_1}, \overline{z_2}, \oplus, \overline{\oplus}\}$, and show that counting patterns of an f-circuit is solvable in polynomial time. In Section 4, we prove #P-completeness for $f \in \{\wedge, \vee, \wedge\rceil, \vee\rceil, \lceil\wedge, \lceil\vee, \overline{\wedge}, \overline{\vee}\}$. In Section 5, we conclude with some remarks.

2 Definitions

Let B_2 be a set of all the Boolean functions $f : \{0,1\}^2 \to \{0,1\}$ with two input variables. We denote each of the functions in B_2 by the letter in (1). For $f \in B_2$, an *f-gate* is a logic gate that computes f, and an *f-circuit* C of n input variables is expressed as a directed acyclic graph where each node of in-degree 0 in C corresponds to one of the n input variables $x_1, x_2, \ldots, x_n$, and the other nodes correspond to f-gates. Let C be an f-circuit with n input variables. Let s be the number of gates in C, and let $g_1, g_2, \ldots, g_s$ be the s gates. The *output pattern of C for an input* $\boldsymbol{x} \in \{0,1\}^n$ is defined to be the vector $(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_s(\boldsymbol{x}))$, where $g_i(\boldsymbol{x})$, $1 \le i \le s$, is the output of the gate g_i for $\boldsymbol{x}$. We often abbreviate an output pattern as a *pattern*. We say that a pattern $\boldsymbol{p} = (p_1, p_2, \ldots, p_s) \in \{0,1\}^s$ *arises* in C if there exists an input $\boldsymbol{x} \in \{0,1\}^n$ such that $g_i(\boldsymbol{x}) = p_i$ for every i, $1 \le i \le s$. We denote by $\Gamma(C)$ a set of the patterns that arise in C. Clearly, we have $\Gamma(C) \subseteq \{0,1\}^s$ and hence $|\Gamma(C)| \le 2^s$. For each $f \in B_2$, a *counting* problem #Pat(f) is to compute $|\Gamma(C)|$ for a given f-circuit C.

The *class #P* is defined to be a set of functions that can be expressed as the number of accepting path of a nondeterministic polynomial-time Turing machine. Let FP be a class of functions that can be computed by a polynomial-time deterministic Turing machine. A function f is *#P-hard* if #P $\subseteq FP^f$, where FP^f is a class of functions that can be computed by a polynomial-time deterministic Turing machine with an oracle to f. We say that f is *#P-complete* if $f \in$ #P and f is #P-hard.

3 Tractable cases

In this section, we prove that #Pat(f) is solvable in polynomial time if $f \in \{\mathbf{0}, \mathbf{1}, z_1, z_2, \overline{z_1}, \overline{z_2}, \oplus, \overline{\oplus}\}$. That is, we prove the following theorem.

Theorem 1. *For any $f \in \{\mathbf{0}, \mathbf{1}, z_1, z_2, \overline{z_1}, \overline{z_2}, \oplus, \overline{\oplus}\}$, #Pat($f$) is in FP.*

It is trivially true that #Pat($\mathbf{0}$) and #Pat($\mathbf{1}$) are in FP, since any $\mathbf{0}$-cirucit and $\mathbf{1}$-circuit have only one output pattern. It thus suffices to give proofs for the other six functions $\{z_1, z_2, \overline{z_1}, \overline{z_2}, \oplus, \overline{\oplus}\}$. We first consider the cases for $z_1, z_2, \overline{z_1}$ and $\overline{z_2}$.

Lemma 1. *For any $f \in \{z_1, z_2, \overline{z_1}, \overline{z_2}\}$, #Pat($f$) is in FP.*

Proof. In this proof, we verify the lemma only for #Pat(z_1), since the proofs for the other cases are similar.

Let C be a circuit consisting of a number s of z_1-gates $g_1, g_2, \ldots, g_s$ together with n input variables. Without loss of generality, we assume that $g_1, g_2, \ldots, g_s$ are topologically ordered in the underlying graph of C. Then, for each i, $1 \le i \le s$, we inductively label g_i by an index of an input variable as follows: If the left input of g_i is an input variable x_j, then the label of g_i is j; otherwise, the label of g_i is same as the one for the left input gate of g_i. Clearly, the output of every gate labelled by j equals to x_j for every $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. Let l be the number of distinct labels by which we give at least a gate in C, then we have $|\Gamma(C)| = 2^l$. □

We then consider the cases for $\oplus$ and $\overline{\oplus}$.

Lemma 2. *For any $f \in \{\oplus, \overline{\oplus}\}$, #Pat($f$) is in FP.*

Proof. We first consider $\oplus$-circuits. Let C be a $\oplus$-circuit with n input variables $x_1, x_2, \ldots, x_n$. Without loss of generality, we assume that every input variable is connected to some gate in C. Let A be a set of gates whose inputs are both input variables. We denote by $G(C) = (V, E)$ a graph obtained from C as follows:

$$V = \{1, 2, \ldots, n\}$$

and

$$\begin{aligned} E = \{(i_1, i_2) \in V \times V \mid \\ \text{a gate } g \in A \text{ receives } x_{i_1} \text{ and } x_{i_2} \text{ s.t. } i_1 < i_2\}. \end{aligned}$$

Note that $G(C)$ is an undirected graph. We denote by r the number of connected components in $G(C)$, and by $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_r = (V_r, E_r)$ the r connected components of $G(C)$. Note that some connected component may consist of only a single node. Let B be a set of gates whose inputs are an input variable and an output of a gate. We now define r Boolean values $\beta_1, \beta_2, \dots, \beta_r \in \{0, 1\}$ as follows: For each j, $1 \le j \le r$, we have $\beta_j = 1$ if there is a gate $g \in B$ that receives x_i, $i \in V_j$, as its input; and $\beta_j = 0$ otherwise. Below we prove that

$$|\Gamma(C)| = \prod_{j=1}^{r} 2^{|V_j|+\beta_j-1} \tag{2}$$

by induction on r; Eq. (2) implies that $\#\text{PAT}(\oplus)$ is in FP, since we can obtain $V_1, V_2, \dots, V_r$ and $\beta_1, \beta_2, \dots, \beta_r$ in polynomial time.

For each j, $1 \le j \le r$, let $A_j \subseteq A$ be a set of gates whose input variables are x_{i_1} and x_{i_2}, $i_1, i_2 \in V_j$. Clearly, $A_1, A_2, \dots, A_r$ compose a partition of A. We use the following simple claim in the inductive proof.

Claim. *Let j, $1 \le j \le r$, be a positive integer, and let $\boldsymbol{a} = (a_1, a_2, \dots, a_n), \boldsymbol{b} = (b_1, b_2, \dots, b_n) \in \{0, 1\}^n$ be a pair of input assignments to C. Then $g(\boldsymbol{a}) = g(\boldsymbol{b})$ for every $g \in A_j$ if and only if either $a_i = b_i$ for every $i \in V_j$ or $a_i \ne b_i$ for every $i \in V_j$.*

Proof. Sufficiency. Assume that either $a_i = b_i$ for every $i \in V_j$ or $a_i \ne b_i$ for every $i \in V_j$. Then, we have $a_i + a_j \equiv b_i + b_j \pmod 2$ for any pair of $i, j \in V_j$. Since every gate $g \in A_j$ is a $\oplus$-gate, we clearly have $g(\boldsymbol{a}) = g(\boldsymbol{b})$ for every $g \in A_j$.

Necessity. Assume that

$$g(\boldsymbol{a}) = g(\boldsymbol{b}) \tag{3}$$

for every $g \in A_j$, and suppose for the sake of contradiction that there exist indices $i_1, i_2 \in V_j$ such that $a_{i_1} = b_{i_1}$ and $a_{i_2} \ne b_{i_2}$. Consider the jth connected component $G_j = (V_j, E_j)$ of G, then there is a path p between $i_1 \in V_j$ and $i_2 \in V_j$; we denote by l the length of p, and by $(i_1, v_1), (v_1, v_2), \dots, (v_{l-1}, i_2)$ be the edges on p. Since $a_{i_1} = b_{i_1}$, Eq. (3) implies that, for every z, $1 \le z \le l-1$, we have $a_{v_z} = b_{v_z}$, and thus $a_{i_2} = b_{i_2}$. This contradicts the fact that $a_{i_2} \ne b_{i_2}$. □

The claim implies that a same output pattern arises for $\boldsymbol{a}, \boldsymbol{b} \in \{0, 1\}^n$ only if $\boldsymbol{a} = \overline{\boldsymbol{b}}$ where $\overline{\boldsymbol{b}}$ is the bitwise complement of $\boldsymbol{b}$.

For the basis, we verify that Eq. (2) holds for $r = 1$. Clearly, the claim implies that, for any pair of input assignment $\boldsymbol{a}, \boldsymbol{b} \in \{0, 1\}^n$, $g(\boldsymbol{a}) = g(\boldsymbol{b})$ for every gate $g \in A_1 (= A)$ if and only if $\boldsymbol{a} = \overline{\boldsymbol{b}}$. Consider the case where $\beta_1 = 0$. In this case, we have $B = \emptyset$, and thus every gate $g \notin A$ in C receives two inputs from outputs of gates. Therefore, any pattern that arises in C is determined by the outputs of the gates in A. Consequently, for any pair of input assignment $\boldsymbol{a}, \boldsymbol{b} \in \{0, 1\}^n$, $g(\boldsymbol{a}) = g(\boldsymbol{b})$ for every gate g in C if and only if $\boldsymbol{a} = \overline{\boldsymbol{b}}$. Thus, we have

$$|\Gamma(C)| = 2^{|V_1|-1} = 2^{|V_1|+\beta_1-1}.$$

Consider the other case where $\beta_1 = 1$. In this case, we have $B \ne \emptyset$. Let g' be an arbitrary gate in B. The gate g' receives an input variable, say x_i, and an output of a gate, say g''. If $\boldsymbol{a} = \overline{\boldsymbol{b}}$, then $g'(\boldsymbol{a}) = g'(\boldsymbol{b})$ if and only if $g''(\boldsymbol{a}) \ne g''(\boldsymbol{b})$. Furthermore, by the claim, we have $g(\boldsymbol{a}) = g(\boldsymbol{b})$ for every gate $g \in A_1 (= A)$ if and only if $\boldsymbol{a} = \overline{\boldsymbol{b}}$. Thus, we have

$$|\Gamma(C)| = 2^{|V_1|} = 2^{|V_1|+\beta_1-1}.$$

For inductive hypothesis, assume that Eq. (2) holds for $r - 1$, and that the graph $G(C)$ has r connected components $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_r = (V_r, E_r)$. For each j, $1 \le j \le r$, we define $k_j = |V_j|$, and then relabel the k_j input variables whose indices are in V_j as $x_{j,1}, x_{j,2}, \dots, x_{j,k_j}$. Consider a circuit C' given by applying any partial input assignment to the variables $x_{r,1}, x_{r,2}, \dots, x_{r,k_r}$ and removing the gates whose outputs are determined. Let $G(C')$ be a graph obtained from C'. Clearly, $G(C')$ has $r - 1$ connected components, and hence the induction hypothesis implies that C' has

$$\prod_{j=1}^{r-1} 2^{|V_j|+\beta_j-1} \tag{4}$$

patterns. Note that there are $2^{|V_r|}$ partial input assignments to $x_{r,1}, x_{r,2}, \dots, x_{r,k_r}$. Therefore, if $\beta_r = 0$, the claim and Eq. (4) imply that C has

$$2^{|V_r|-1} \times \prod_{j=1}^{r-1} 2^{|V_j|+\beta_j-1} = \prod_{j=1}^{r} 2^{|V_j|+\beta_j-1} \tag{5}$$

patterns, as required. On the other hand, if $\beta_r = 1$, we have a gate $g' \in B$ that receives an input variable, say x_i, $i \in V_r$, and an output of a gate, say g''. If $\boldsymbol{a} = \overline{\boldsymbol{b}}$, then $g'(\boldsymbol{a}) = g'(\boldsymbol{b})$ if and only if $g''(\boldsymbol{a}) \ne g''(\boldsymbol{b})$. By the claim, we have $g(\boldsymbol{a}) = g(\boldsymbol{b})$ for every gate $g \in A_r$ if and only if $\boldsymbol{a} = \overline{\boldsymbol{b}}$. Therefore, C has

$$2^{|V_r|} \times \prod_{j=1}^{r-1} 2^{|V_j|+\beta_j-1} = \prod_{j=1}^{r} 2^{|V_j|+\beta_j-1} \tag{6}$$

patterns. Consequently, by Eqs. (5) and (6), Eq. (2) holds for r.

We can similarly compute the number of patterns of a $\overline{\oplus}$-circuit, and so omit the proof. □

4 Intractable cases

In this section, we consider eight functions $\wedge, \vee, \overline{\wedge}, \overline{\vee}, \lceil\wedge, \lceil\vee, \wedge\rceil, \vee\rceil$, and show that every case is intractable even for constant-depth circuits.

Theorem 2. *For any $f \in \{\wedge, \vee, \lceil\wedge, \lceil\vee, \wedge\rceil, \vee\rceil\}$, $\#\text{PAT}(f)$ is $\#\text{P}$-complete even for circuits of depth two. For any $f \in \{\overline{\wedge}, \overline{\vee}\}$, $\#\text{PAT}(f)$ is $\#\text{P}$-complete even for circuits of depth three.*

In the following lemma, we show that counting patterns of a circuit C is contained in #P.

Lemma 3. *For any $f \in \{\wedge, \vee, \lceil\wedge, \lceil\vee, \wedge\rceil, \vee\rceil, \overline{\wedge}, \overline{\vee}\}$, $\#\text{PAT}(f)$ is contained in $\#\text{P}$.*

Proof. Let $f \in \{\wedge, \vee, \lceil\wedge, \lceil\vee, \wedge\rceil, \vee\rceil, \overline{\wedge}, \overline{\vee}\}$, and let C be an f-circuit of s gates and n input variables. It is enough to show that, given a pattern $\boldsymbol{p} = (p_1, p_2, \dots, p_s) \in \{0, 1\}^s$ of C, we can decide in polynomial time if the pattern $\boldsymbol{p}$ arises in C: If $\boldsymbol{p}$ arises in C, we decide to *accept*, and otherwise *reject*. Since the satisfiability of 2CNF formula is decidable in polynomial time (Johnson 1990, p. 86), it suffices to construct a formula ϕ such that ϕ is satisfiable if and only if $\boldsymbol{p}$ arises in C, and each clause of ϕ contains at most two literals. We below construct the desired formula ϕ.

Let $g_1, g_2, \ldots, g_s$ be the gates in C. We initially have an empty set $S(=\emptyset)$ of clauses, and for each i, $1 \le i \le s$, we repeat the following procedure:

(1) If g_i has two inputs from the outputs of two gates, say g_{i_1} and g_{i_2}, then check whether $p_i = g_i(p_{i_1}, p_{i_2})$. If $p_i \ne g_i(p_{i_1}, p_{i_2})$, then $\boldsymbol{p}$ never arises in C, and hence we decide to reject; otherwise, we continue the procedure.

(2) If g_i has an input from an output of a gate, say g_{i_1}, and the other input from an input variable, say x_{i_2}, then check whether $p_i = g_i(p_{i_1}, y)$ for each $y \in \{0,1\}$. If $p_i \ne g_i(p_{i_1}, y)$ for all $y \in \{0,1\}$, then we decide to reject. If $p_i = g_i(p_{i_1}, y)$ for exactly one of $y = 0$ and $y = 1$, we add a literal of x_{i_2} to S so that the following equation holds

$$p_i = g_i(p_{i_1}, x_{i_2}). \tag{7}$$

That is, if $p_i = g_i(p_{i_1}, 1)$, then we add a positive literal of x_{i_2} to S; and if $p_i = g_i(p_{i_1}, 0)$, then we add a negative literal $\neg x_{i_2}$ of x_{i_2} to S. If $p_i = g_i(p_{i_1}, y)$ for all $y \in \{0,1\}$, we do nothing and continue the procedure.

(3) If both of the inputs of g_i are input variables, say x_{i_1} and x_{i_2}, then we add clauses to S so that the following equation holds

$$p_i = g_i(x_{i_1}, x_{i_2}). \tag{8}$$

Since g_i computes a function of two inputs, it is sufficient to add at most two clauses, each of which contains at most two literals.

Then we construct ϕ by taking a conjunction of all the clauses in S. Since we add at most two clauses to S for each i, $1 \le i \le s$, ϕ is a 2CNF formula of at most $2s$ input variables and at most $2s$ clauses. Clearly, ϕ is satisfiable if and only if Eqs. (7) and (8) holds for every i, $1 \le i \le s$. Thus, ϕ is satisfiable if and only if $\boldsymbol{p}$ arises in C. □

We now show that the problems are #P-hard. We first prove that #PAT(∧) and #PAT(∨) are #P-hard by a reduction from a counting problem for graphs. Let $G = (V, E)$ be a graph with a vertex set V and an edge set E. A subset $V' \subseteq V$ is called an *independent set* if for any pair of $u, v \in V'$, $(u, v) \notin E$. We employ the following canonical #P-complete problem (Provan & Ball 1983).

#INDEPENDENT SET
Input: A graph G
Output: The number of independent sets in G

Lemma 4. *For any $f \in \{\wedge, \vee\}$, #PAT(f) is #P-hard even for circuits of depth two.*

Proof. We first give a proof for #PAT(∧). Let $G = (V, E)$ be an instance of #INDEPENDENT SET, where $V = \{1, 2, \ldots, n\}$ and $E \subseteq V \times V$. For each assignment $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n$, we define $S_{\boldsymbol{a}} = \{i \in V \mid a_i = 1\}$, and let

$$I = \{\boldsymbol{a} \in \{0,1\}^n \mid S_{\boldsymbol{a}} \text{ is an independent set of } G\}.$$

We prove the lemma by constructing in polynomial time a depth-2 ∧-circuit C_G such that

$$|\Gamma(C_G)| = 2^n - |I| + 1. \tag{9}$$

If Eq. (9) holds, we can obtain $|I|$ by just subtracting $|\Gamma(C_G)|$ from $2^n + 1$ and hence complete the proof.

The desired circuit C_G is given as follows. C_G receives n input variables $x_1, x_2, \ldots, x_n$. In the first layer, C_G has a gate g_e for every $e = (i_1, i_2) \in E$ that computes "x_{i_1} and x_{i_2}." In the second layer, C_G has a gate $g_{i,e}$ for each pair of i, $1 \le i \le n$, and $e \in E$ that computes "x_i and the output of g_e." Clearly, we can construct C_G in polynomial time, and C_G is a depth-2 ∧-circuit. We note that if g_e outputs one for an assignment $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n$, then we have $g_{i,e}(\boldsymbol{a}) = a_i$ for each i, $1 \le i \le n$.

We now verify that Eq. (9) holds. Consider an arbitrary assignment $\boldsymbol{a} \in I$. Clearly, every gate in the first layer of C_G outputs zero for $\boldsymbol{a}$, and hence every gate in the second layer of C_G outputs zero for $\boldsymbol{a}$ too. Thus, the pattern $(0, 0, \ldots, 0)$ arises in C_G for $\boldsymbol{a}$. Moreover, $(0, 0, \ldots, 0)$ arises only for an input $\boldsymbol{a} \in I$, since, otherwise, we have $g_e(\boldsymbol{z}) = 1$ for some $e \in E$.

Let $\bar{I} = \{0,1\}^n \backslash I$. We show that C_G has an unique output pattern for each $\boldsymbol{a} \in \bar{I}$. Consider an arbitrary pair of assignments $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \bar{I}$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n) \in \bar{I}$, where $\boldsymbol{a} \ne \boldsymbol{b}$. Let $E_{\boldsymbol{a}}$ (resp., $E_{\boldsymbol{b}}$) be a set of the edges in a subgraph of G induced by $S_{\boldsymbol{a}}$ (resp., $S_{\boldsymbol{b}}$). One may assume without loss of generality that $|E_{\boldsymbol{a}}| \ge |E_{\boldsymbol{b}}|$. If $E_{\boldsymbol{a}} \ne E_{\boldsymbol{b}}$, then, for any edge e such that $e \in E_{\boldsymbol{a}}$ and $e \notin E_{\boldsymbol{b}}$, we have $g_e(\boldsymbol{a}) \ne g_e(\boldsymbol{b})$, and hence the pattern for $\boldsymbol{a}$ is different from the one for $\boldsymbol{b}$. If $E_{\boldsymbol{a}} = E_{\boldsymbol{b}}$, then $g_e(\boldsymbol{a}) = g_e(\boldsymbol{b})$ for every $e \in E$, and hence $g_{i,e}(\boldsymbol{a}) = a_i$ and $g_{i,e}(\boldsymbol{b}) = b_i$ for each i, $1 \le i \le n$. Since $\boldsymbol{a} \ne \boldsymbol{b}$, there is an index i^* such that $a_{i^*} \ne b_{i^*}$, and hence we have $a_{i^*} = g_{i^*,e}(\boldsymbol{a}) \ne g_{i^*,e}(\boldsymbol{a}') = b_{i^*}$. Therefore, the pattern for $\boldsymbol{a}$ is different from the one for $\boldsymbol{b}$. Consequently, we have $|\Gamma(C_G)| = |\bar{I}| + 1$. Since $|I| + |\bar{I}| = 2^n$, Eq. (9) holds.

A proof for #PAT(∨) can be easily obtained by C_G and De Morgan's law, that is, we can construct an ∨-circuit C'_G from C_G so that $|\Gamma(C'_G)| = |\Gamma(C_G)|$ as follows: (1) negate every input variable of C_G by NOT-gates, (2) push the NOT-gates forward to the outputs of the gates in the second layer, and (3) remove the NOT-gates. Since each of (1), (2) and (3) preserves the number of output patterns, we complete the proof. □

We can prove #P-hardness of #PAT(f) for every $f \in \{\lceil\wedge, \lceil\vee, \wedge\rceil, \vee\rceil\}$ in a very similar manner to the proof of Lemma 4. Below, we give proofs for the completeness. We employ the following variant of #INDEPENDENT SET (Provan & Ball 1983).

#BIPARTITE INDEPENDENT SET
Input: A bipartite graph G
Output: The number of independent sets in G

Lemma 5. *For any $f \in \{\lceil\wedge, \lceil\vee, \wedge\rceil, \vee\rceil\}$, #PAT($f$) is #P-hard even for circuits of depth two.*

Proof. We give a proof only for #PAT(⌈∧), since proofs for the other cases are obtained by the hardness of #PAT(⌈∧) and De Morgan's law.

Let $G = (U, V, E)$ be an instance of #BIPARTITE INDEPENDENT SET, where $U = \{1, 2, \ldots, n\}$, $V = \{n+1, n+2, \ldots, 2n\}$ and $E \subseteq U \times V$. For each

$\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \{0,1\}^{2n}$, we define $S_{\boldsymbol{a}} = \{i \in U \mid a_i = 0\} \cup \{i \in V \mid a_i = 1\}$, and let

$$I = \{\boldsymbol{a} \mid \ S_{\boldsymbol{a}} \text{ is an independet set of } G.\}.$$

We now construct a depth-2 $\lceil\wedge$-circuit C_G such that

$$|\Gamma(C_G)| = 2^n - |I| + 1. \qquad (10)$$

The desired circuit C_G receives $2n$ input variables $x_1, x_2, \ldots, x_{2n}$. In the first layer, C_G has a gate g_e for every $e = (i_1, i_2) \in E$ that computes "$\overline{x_{i_1}}$ and y_{i_2}," where $\overline{x_{i_1}}$ is the negation of x_{i_1}. In the second layer, C_G has a gate $g_{i,e}$ for each pair of $i \in U \cup V$ and $e \in E$ that computes "$\overline{x_i}$ and the output of g_e." Clearly, we can construct C_G in polynomial time, and C_G is a depth-2 circuit consisting of only $\lceil\wedge$-gates.

We can now verify that Eq. (10) holds; we can prove that C_G has the output pattern $(0, 0, \ldots, 0)$ for every $\boldsymbol{a} \in I$, and has an unique output pattern for each $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \bar{I}$. We omit the detail, since the rest of the proof is same as the one for Lemma 4. □

We lastly verify that #PAT($\overline{\wedge}$) and #PAT($\overline{\vee}$) are #P-hard:

Lemma 6. *#PAT($\overline{\wedge}$) and #PAT($\overline{\vee}$) are #P-hard even for circuits of depth three.*

Proof. Note that an $\wedge$-gate g can be replaced by two $\overline{\wedge}$-gates g' and g'' such that g' receives same inputs as ones of g, and g'' receives two copies of the output of g'; similarly, $\vee$-gate can be replaced by two $\overline{\vee}$-gates.

We prove the lemma by the fact above and Lemma 4 as follows. Recall that the circuit C_G given in the proof of Lemma 4 is a depth-2 $\wedge$-circuit. By replacing each $\wedge$-gate in the first layer of C_G with two $\overline{\wedge}$-gates, we obtain a depth-3 circuit whose number of patterns is same as C_G. Then we can safely replace each $\wedge$-gate in the third layer with a $\overline{\wedge}$-gate, and obtain C'_G. Clearly, C'_G consists of only $\overline{\wedge}$-gates, and $|\Gamma(C'_G)| = |\Gamma(C_G)|$. Thus we complete the proof for #PAT($\overline{\wedge}$). We can similarly prove the hardness of #PAT($\overline{\vee}$), and so omit the proof. □

5 Conclusions

In this paper, we investigate computational complexity of counting output patterns of a given f-circuit, and give a a complete analysis for the counting problem on $f \in B_2$. More formally, we prove that the problem of counting the number of the outputs patterns that arise in an f-circuit is solvable in polynomial time if $f \in \{\mathbf{0}, \mathbf{1}, a_1, a_2, \overline{a_1}, \overline{a_2}, \oplus, \overline{\oplus}\}$; while the problem is #P-complete even for constant-depth f-circuits if $f \in \{\wedge, \vee, \wedge\rceil, \vee\rceil, \lceil\wedge, \lceil\vee, \overline{\wedge}, \overline{\vee}\}$.

References

Arpe, J. & Manthey, B. (2009), 'Approximability of minimum AND-circuits', *Algorithmica* **53**(3), 337–357.

Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A. & Shelat, A. (2005), 'The smallest grammar problem', *IEEE Transactions on Information Theory* **51**(7), 2554–2576. Cited By (since 1996): 35.

Johnson, D. S. (1990), A catalog of complexity classes, *in* 'Handbook of Theoretical Computer Science', Vol. A, Elsevier Science Publishers, chapter 2, pp. 67–161.

Morizumi, H. (2011), 'Improved approximation algorithms for minimum AND-circuits problem via k-set cover', *Information Processing Letters* **111**(5), 218 – 221.

Parberry, I. (1994), *Circuit Complexity and Neural Networks*, MIT Press, Cambridge, MA.

Provan, J. S. & Ball, M. O. (1983), 'The complexity of counting cuts and of computing the probability that a graph is connected', *SIAM Journal on Computing* **12**(4), 777–788.

Sima, J. & Orponen, P. (2003), 'General-purpose computation with neural networks: A survey of complexity theoretic results', *Neural Computation* **15**, 2727–2778.

Siu, K. Y., Roychowdhury, V. & Kailath, T. (1995), *Discrete Neural Computation; A Theoretical Foundation*, Prentice-Hall, Inc., Upper Saddle River, NJ.

Uchizawa, K., Douglas, R. & Maass, W. (2006), 'On the computational power of threshold circuits with sparse activity', *Neural Computation* **18**(12), 2994–3008.

Uchizawa, K., Takimoto, E. & Nishizeki, T. (2011), 'Size-energy tradeoffs of unate circuits computing symmetric Boolean functions', *Theoretical Computer Science* **412**, 773–782.