Size of IK00 Branching Program

Yupu Hu[1], Xingting Dong[1], and Baocang Wang[1]

State Key Laboratory of Integrated Services Networks,
Xidian University, Xi'an, Shaanxi, China
yphu@mail.xidian.edu.cn

Abstract. Branching program is an important component of indistinguishability obfuscation (IO) schemes, its size greatly influences the efficiencies of corresponding IO schemes. There are two major candidates of branching programs, Bar86 branching program and IK00 branching program. Bar86 branching program was shown to efficiently recognize NC^1 circuits. In specific, a Boolean circuit of depth d can be recognized by a Bar86 branching program of length not larger than 4^d (Therefore of size not larger than $5^2 \times 4^d$).

In this short paper we present similar result about IK00 branching program. We show that IK00 branching program efficiently recognizes NC^1 circuits, that is, a Boolean circuit of depth d can be recognized by an IK00 branching program of size not larger than $(n+1) \times 4^d$, where n is input length.

Our result may be a negative evidence for IK00 branching program to efficiently recognize polynomial-time computable functions.

Keywords: Indistinguishability Obfuscation · Branching Programs · NC^1 Circuits.

1 Introduction

Branching program [1–4] is an important component of indistinguishability obfuscation (IO) schemes [5–7], its size greatly influences the efficiencies of corresponding IO schemes. There are two major candidates of branching programs, Bar86 branching program [1,2] and IK00 branching program [3,4]. Bar86 branching program was shown [1] to efficiently recognize NC^1 circuits. In specific, a Boolean circuit of depth d can be recognized by a Bar86 branching program of length not larger than 4^d (That is, of not more than 4^d Boolean matrices of 5×5, therefore of size not larger than $5^2 4^d$). Up to now, there is no clear conclusion about the size of IK00 branching program, although there is a negative feeling (see final paragraph of section 2 of [3]) and an open question (see final graph of page 116 of [8]).

In this short paper we present similar result about IK00 branching program. We show that IK00 branching program efficiently recognizes NC^1 circuits, that is, a Boolean circuit of depth d can be recognized by an IK00 branching program of size not larger than $(n+1) \times 4^d$, where n is input length.

Our result may be a negative evidence for IK00 branching program to efficiently recognize polynomial-time computable functions.

Paper organization is the follow. In section 2 we review generic expression of Boolean circuit, under $\{+, \times\}$ description and $\{or, and\}$ description respectively. Section 3 is a careful review of IK00 branching program. In section 4 we show our result, that is, a Boolean circuit of depth d can be recognized by an IK00 branching program of size not larger than $(n+1) \times 4^d$.

2 Preliminary: generic expression of Boolean circuit

Generally speaking, a polynomial-time-computable Boolean function cannot be expressed by geometrical normal form, because the number of terms may be exponentially large. From [9], a reasonable expression is sequentially giving each step, say $\{input\ a, input\ b, operation\ f, output\ f(a,b)\}$. There are two different sets of operations, $\{+, \times\}$ and $\{or, and\}$, where " $+$ " is exclusive-or, " $\times$ " is multiplication. We know two simple relations: x "and" $y = x \times y$, x "or" $y = x + y + (x \times y)$. For a fixed set of operations, O, n-dimensional Boolean function $c(x)$ is expressed as follow.

- input: $x_0, x = (x_1, x_2, \cdots, x_n)$, where $x_0 = 1$ is constant.
- Sequential steps: For $i = n+1, n+2, \cdots, N$, let $x_i = c_i\left(x_{a(i)}, x_{b(i)}\right)$, where $a(i) \in \{0, 1, 2, \cdots, i-2\}, b(i) \in \{1, 2, \cdots, i-1\}$, $a(i) < b(i), c_i \in O$. A special case for $O = \{+, \times\}$ is that, if $a(i) = 0$, c_i is only " $+$ ". Similarly, a special case for $O = \{or, and\}$ is that, if $a(i) = 0$, c_i is only "or".
- Output: x_N.(We know $x_N = c(x)$)

We call $\{x_0, x = (x_1, x_2, \cdots, x_n); (a(i), b(i), c_i, x_i), i = n+1, n+2, \cdots, N\}$ generic expression of n-dimensional Boolean function $c(x)$, N the circuit size of $c(x)$, $\{(a(i), b(i)), i = n+1, n+2, \cdots, N\}$ the topology of $c(x)$, $\{c_i, i = n+1, n+2, \cdots, N\}$ the operation series of $c(x)$. To ensure that there is no redundant operation, each position $j \in \{1, 2, \cdots, N-1\}$ should be an input position $j \in \{a(i), b(i)\}$, where $i = \{n+1, n+2, \cdots, N\}$. If so, we call n not only the dimension but also the locality of $c(x)$.

3 IK00 branching program [3, 4]

Suppose n-dimensional Boolean function $c(x)$ has a generic expression $\{x_0, x = (x_1, x_2, \cdots, x_n); (a(i), b(i), c_i, x_i), i = n+1, n+2, \cdots, N\}$, under either $O = \{+, \times\}$ or $O = \{or, and\}$.

3.1 Directed graphs

We define each graph corresponding to each x_i, denoted as $G(i)$, where $i \in \{0, 1, \cdots, N\}$. Each $G(i)$ is an acyclic directed graph, with single starting vertex and single ending vertex, each of its edges is assigned an affine functions of x,

and any two vertexes have at most one directed edge. For $i \in \{0, 1, 2, \cdots, n\}$, each $G(i)$ has only two vertexes, starting vertex and ending vertex; the directed edge from starting vertex to ending vertex is assigned x_i. Now take $i \in \{n+1, n+2, \cdots, N\}$, and suppose $\{G(1), G(2), \cdots, G(i-1)\}$ have been defined. We construct $G(i)$.

Case 1: $c_i = +$**.** We have three steps.

(1) First take $G(i)^*$ as parallel connection of $G(a(i))$ and $G(b(i))$, that is, starting vertexes of $G(a(i))$ and $G(b(i))$ are merged into starting vertex of $G(i)^*$, ending vertexes of $G(a(i))$ and $G(b(i))$ are merged into ending vertex of $G(i)^*$.

(2) Second, take $G(i)^{**}$ as vertex reduction of $G(i)^*$ (Sometimes a vertex of $G(a(i))$ and a vertex of $G(b(i))$ can be merged into one vertex).

(3) Finally, take $G(i)$ as edge reduction of $G(i)^{**}$ (If two vertexes of $G(i)^{**}$ has more than one directed edge, merge them into one directed edge, with assigned value which is the sum).

Case 2: $c_i = or$**.** We have four steps.

(1) First, take $G(i)^*$ as parallel connection of $G(a(i))$ and $G(0)$.

(2) Second, take $G(i)^{**}$ as parallel connection of $G(b(i))$ and $G(0)$.

(3) Third, take $G(i)^{***}$ as series connection of $G(i)^*$ and $G(i)^{**}$, that is, ending vertex of $G(i)^*$ and starting vertex of $G(i)^{**}$ are merged into one vertex of $G(i)^{***}$.

(4) Finally, take $G(i)$ as parallel connection of $G(i)^{***}$ and $G(0)$.

Case 3: $c_i = and$ **(**$c_i = and$**).** Define $G(i)$ as series connection of $G(a(i))$ and $G(b(i))$, that is, ending vertex of $G(a(i))$ and starting vertex of $G(b(i))$ are merged into one vertex of $G(i)$.

The number of vertices of $G(i)$ is denoted as $|G(i)|$. It is easy to see the correctness of Lemma 3.1.

Lemma 3.1 *$|G(i)| = 2$ for $i \in \{0, 1, 2, \cdots, n\}$, and for $i \in \{n+1, n+2, \cdots, N\}$,*

$$\begin{cases} \max\{|G(a(i))|, |G(b(i))|\} \le |G(i)| \le |G(a(i))| + |G(b(i))| - 2, & if \ \ c_i = +, \\ |G(i)| = |G(a(i))| + |G(b(i))| - 1, & if \ \ c_i = \times (c_i = \ and), c_i = or. \end{cases}$$

The final graph $G(N)$ is called IK00 branching program of $c(x)$.

3.2 Matrix of degree-3 randomized polynomials [3, 4]

Suppose $|G(N)| = T$, T is polynomially large. Arrange vertexes in partial order, and take $T \times T$ matrix M_0 as such: its (i, i) entry is constant 1, and (i, j) entry $(i \neq j)$ is assigned value of (i, j) edge. Then M_0 is an upper triangular matrix, with main diagonal entries which are constant 1.

Take $(T-1) \times (T-1)$ matrix M_1 as M_0 crossing the first column and final row. It is easy to show that, for any x, determinant of M_1 equals $c(x)$.

Take two $(T-1) \times (T-1)$ random matrices L and R with determinants 1. Fix L and R, and denote $M = LM_1R$. Then for any x, modular 2 determinant of M equals $c(x)$. Besides, M is the matrix of affine functions of x, therefore the size of M is $(n+1) \times (T-1)^2$. Now we call M IK00 branching program of circuit $c(x)$.

4 IK00 branching program efficiently recognize NC^1 circuits

Lemma 4.1 *Suppose $c(x)$ is of depth d. Then $|G(N)| \leq 2^d + 1$.*

Proof. We make use of mathematical induction. If $d = 0$, $c(x)$ is just some $x_i, i = 0, 1, \cdots, n$. In such case $N = 1$, and $|G(N)| = |G(1)| = 2 = 2^0 + 1$. Suppose all circuits of depth d, where $d \leq D$, satisfy $|G(N)| \leq 2^d + 1$, and $c(x)$ is of depth $D + 1$. We know that $c(x)$ is an operation of two sub-circuits $c^*(x)$ and $c^{**}(x)$, where $c^*(x)$ and $c^{**}(x)$ are of depth not larger than D. According to Lemma 3.1, for $c(x)$ we have $|G(N)| \leq 2(2^D + 1) - 1 = 2^{D+1} + 1$. □

Theorem 4.1 *Suppose $c(x)$ is of depth d. Then M has size $(n + 1) \times 4^d$.*

Proof. Just notice Lemma 4.1 and the fact that M has size $(n+1)\times(|G(N)|-1)^2$. □

References

1. Barrington, D.A.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In: Proceedings of the eighteenth annual ACM STOC, pp 1-5, 1986.
2. Barrington, D.A.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In: Journal of computer and system sciences, vol. 38, no. 1, pp 150-164 (1989), Elsevier, 1989.
3. Ishai, Y., Kushilevitz, E.: Randomizing Polynomials: A new representation with applications to round-efficient secure computation. In: Proceedings of the 41st FOCS, pp. 294-304 (2000).
4. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M., (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244-256. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45465-9_22.
5. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October 2013, Berkeley, CA, USA, pp. 40-49 (2013).
6. Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016. LNCS, vol.9665, pp. 28-57. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_2.
7. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol.10401, pp. 599-629. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-63688-7_20.
8. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. In: Computational Complexity, vol. 15 (2006), pp. 115 – 162. 1016-3328/06/020115–48. https://doi.org/10.1007/s00037-006-0211-8.

9. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 545-554. ACM Press, June 2013.