Electronic Colloquium on Computational Complexity, Report No. 86 (2011)

A tighter lower bound on the circuit size of the hardest Boolean functions

Masaki Yamamoto *

Abstract

In [IPL2005], Frandsen and Miltersen improved bounds on the circuit size $L(n)$ of the hardest Boolean function on n input bits: for some constant $c > 0$:

$$\left(1 + \frac{\log n}{n} - \frac{c}{n}\right)\frac{2^n}{n} \le L(n) \le \left(1 + 3\frac{\log n}{n} + \frac{c}{n}\right)\frac{2^n}{n}.$$

In this note, we announce a modest improvement on the lower bound: for some constant $c > 0$ (and for any sufficiently large n),

$$L(n) \ge \left(1 + 2\frac{\log n}{n} - \frac{c}{n}\right)\frac{2^n}{n}.$$

1 Introduction

For any positive integer n, let $L(n)$ be the circuit size of the hardest Boolean function on n input bits. We assume that the in-degree of gates of a circuit is at most two, and hence each gate computes a binary function. Shannon [3] proved that for any $\epsilon > 0$ and for any sufficiently large n,

$$(1-\epsilon)\frac{2^n}{n} \le L(n) \le O(1)\frac{2^n}{n}.$$

Here, we note that we can easily improve the lower bound into $2^n/n$ by slightly modifying his proof. Lupanov [2] improved, via a novel representation of a Boolean function, the upper bound into

$$L(n) \le \left(1 + \frac{O(1)}{\sqrt{n}}\right)\frac{2^n}{n}.$$

It means that $L(n)$ is essentially $2^n/n$, that is, $L(n) = (1+o(1))2^n/n$. Most of researchers may regard this bound to be tight, and hence may regard the research on this topic to end with this result.

After Lupanov's result, in 2005, Frandsen and Miltersen [1] developed a novel representation of a circuit, and estimated a more precise value of $L(n)$ hidden by the notation $o(1)$: for some constant $c > 0$ (and for any sufficiently large n),

$$\left(1 + \frac{\log n}{n} - \frac{c}{n}\right)\frac{2^n}{n} \le L(n) \le \left(1 + 3\frac{\log n}{n} + \frac{c}{n}\right)\frac{2^n}{n}.$$

The lower bound was obtained via their representation scheme for circuits. The upper bound was obtained by improving the circuit implementation of Lupanov representation, and by setting parameters optimally.

*Kwansei-Gakuin University, masaki.yamamoto@kwansei.ac.jp

ISSN 1433-8092

In this note, we further improve the lower bound: for some constant $c > 0$,

$$L(n) \geq \left(1 + 2\frac{\log n}{n} - \frac{c}{n}\right)\frac{2^n}{n}.$$

Our idea

Our bound is obtained in the same way as [1]: they showed an algorithm for transforming a circuit into a sequence of instructions for a "stack program" so that circuits are one-to-one mapped to stack programs. Then, they estimated the length of descriptions of such stack programs. (Their lower bound was obtained by comparing the number of stack programs with the total number of Boolean functions of n variables.)

The difference from theirs is the way of analyzing the number of stack programs: We partition the family of circuits of size s into $s+1$ sets, say, $\mathcal{C}(t)$ for $0 \leq t \leq s$, according to the number of gates of out-degree at least two. That is, $\mathcal{C}(t)$ is the set of circuits of size s where the number of gates of out-degree at least two is t. Then, we estimate the length of descriptions of stack programs for $\mathcal{C}(t)$. The crucial point is that by using the fact that the number of gates of out-degree at least two is t for any circuit of $\mathcal{C}(t)$, (1) we can reduce the length of descriptions of stack programs, and (2) we can estimate the number of stack programs that can be constructed from the same circuit. Taking the maximum of $|\mathcal{C}(t)|$ over $0 \leq t \leq s$, we obtain an improved upper bound of the number of stack programs, from which we obtain the desired lower bound on the circuit size.

2 Preliminaries

Let F_n be the family of functions on n variables. (Thus, $|F_n| = 2^{2^n}$.) Let C_n be an arbitrary circuit on n input bits. For any function $s(n)$, let SIZE$(s(n))$ be the set of functions of F_n that can be computed by a circuit C_n of size at most $s(n)$. Then, the definition of $L(n)$ we study here is as follows:

$$L(n) \stackrel{\text{def}}{=} \max_{f \in F_n} \min\{s(n) : f \in \mathrm{SIZE}(s(n))\}.$$

Theorem 2.1 (Frandsen & Miltersen [1]). *There is a constant $c > 0$ such that $L(n) \geq (2^n/n)(1 + \log n/n - c/n)$.*

They proved this theorem by showing an algorithm for transforming a circuit into a sequence of instructions for a "stack program". We briefly review their proof of the theorem.

Let C_n be an arbitrary circuit of size $s = s(n)$. We assume that letting input gates $g_1, \ldots, g_n$, operational gates are labelled with $g_{n+1}, \ldots, g_{n+s}$ arbitrarily, and the output gate with g_s. Moreover, C_n is represented by the set $\{g_i = g_i^{(1)} \mathrm{op}_i\, g_i^{(2)} : n+1 \leq i \leq n+s\}$, where op_i is the operation of g_i. A stack program constructed from a circuit is a sequence of the following types of instructions:

- "`push` i" for some $1 \leq i \leq n+s$
- "`operate` op_i" for some $n+1 \leq i \leq n+s$

Such a stack program is constructed as shown in Fig. 1. Intuitively, given a circuit, the algorithm traverses the circuit staring with the output gate in the depth-first-search manner. In doing so, it simulates the process of evaluating a circuit value: it marks a gate, which means that it implicitly keeps its gate value.

```
transform(C_n)

    Let P_n be empty
    construct(g_{n+s})
    output P_n

construct(g)

    if g = g_i for some 1 ≤ i ≤ n, then add "push i" to the last of P_n
    else // i.e., g = g_i for some n+1 ≤ i ≤ n+s
        if g_i is marked, then add "push i" to the last of P_n
        else // i.e., g_i is not marked
            mark g_i
            construct(g_i^(1))
            construct(g_i^(2))
            add "operate op_i" to the last of P_n
```

Figure 1: The construction of P_n

A stack program constructed in this way is executed as shown in Fig. 2. It is easy to see that $P_n \equiv C_n$, where P_n is constructed from C_n. Moreover, if $C_n \not\equiv C'_n$, then $P_n \not\equiv P'_n$, where P_n (resp. P'_n) is constructed from C_n (resp. C'_n). That is, circuits are mapped to stack programs in the one-to-one sense.

Now, we estimate the length $\ell(s) = \ell_n(s)$ of the description (i.e., the binary representation) of P_n constructed from C_n of size s. (Later, we see that this value is independent of the structure of C_n.) Note here that circuits of size s are one-to-one mapped to stack programs of description length $\ell(s)$. (From the estimation of $\ell(s)$, we obtain the number of possible stack programs, which is an upper bound on the number of circuits of size s.) Recall that P_n is a sequence of two types of instructions. For each type of instructions, we estimate the number of its occurrences in P_n. First, from the construction of P_n, we see that the number of occurrences of type "`operate` op_i" is exactly s. Next, considering the execution of P_n, we can estimate the number of occurrences of type "`push` i" as follows: the size of the stack (in the execution of P_n) increases by one due to "`push` i" while it decreases by one due to "`operate` op_i". Thus, since the stack is empty at the first (i.e., before the execution of P_n) and its size is one at the end, the number of occurrences of type "`push` i" is one more than that of "`operate` op_i", that is, exactly $s+1$.

The description of P_n is a sequence of blocks of bits, which are of fixed lengths according to types of instructions. Thus, we save one bit per one block for recognizing types (and block lengths). A block for type "`push` i" further needs length $\lceil \log(n+s) \rceil$ since i can take its value from one to $n+s$. A block for type "`operate` op_i" further needs length a constant since op_i is a binary function, and hence it can be recognized by a constant bits. Thus, the total length of the description of P_n is

$$\ell(s) = (s+1)(1 + \lceil \log(n+s) \rceil) + O(s),$$

that is, $\ell(s) = s\log(s+n) + O(s)$. From this, the number of circuits of size at most s is at most $s \cdot 2^{\ell(s)} = 2^{\ell(s)+\log s} = 2^{\ell}$, where $\ell = s\log(s+n) + O(s)$. Thus, we need $\ell \geq 2^n$ if any function of F_n can be computed by a circuit of size at most s. Therefore, the theorem is

$P_n(x)$

Let $(g_1, \ldots, g_n) = (x_1, \ldots, x_n)$
Let S be the empty stack

for $j : 1 \leq j \leq |P_n|$ // $|P_n|$ is the number of instructions in P_n
  if $P_n[j]$ is "`push` i", **then** push the value of g_i to S
  if $P_n[j]$ is "`operate` op_i", **then**
    1. pop the top two s_1 and s_2 from S,
    2. let $g_i = s_1 \mathrm{op}_i s_2$
    3. push the value of g_i to S

Output the value of the top of S

Figure 2: The execution of P_n

proved by showing the following for any constant $c > 0$: if $s = (2^n/n)(1 + \log n/n - c/n)$, then $s \log(s + n) + cs < 2^n$. This is checked by an elementary calculation.

3 A tighter lower bound

In the previous section, we see that the stack program constructed from a circuit C_n of size s is described by at most $\ell(s) = s \log(n + s) + O(s)$ bits. Thus, the number of possible stack programs for circuits of size at most s is at most 2^ℓ, where $\ell = s \log(n + s) + O(s)$. In this section, we improve this upper bound, from which we obtain an improved lower bound on $L(n)$.

For any non-negative integer $t : 0 \leq t \leq s$, let $\mathcal{C}(t)$ be the set of circuits of size s such that the number of gates of out-degree at least two is (exactly) t. (Thus, $[\mathcal{C}(t) : 0 \leq t \leq s]$ is a partition of the set of circuits of size s.) We first estimate $|\mathcal{C}(t)|$ for any $t : 0 \leq t \leq s$.

Lemma 3.1.

$$|\mathcal{C}(t)| \leq \frac{2^{s \log(t+n)+O(s)}}{t!}.$$

Proof. We prove it in the similar way to the proof of Theorem 2.1: There are two different things to estimate $|\mathcal{C}(t)|$. One of the two is the way of estimating the length needed for the description of type "`push` i": we have seen that $\lceil \log(n + s) \rceil$ bits are needed to describe the value of i since i can take its value from one to $n+s$. We will shortly see that if P_n is constructed from a circuit $C_n \in \mathcal{C}(t)$, then we can reduce this number by applying a suitable labelling: Let $C_n \in \mathcal{C}(t)$ be an arbitrary circuit of size s, and let P_n be the stack program constructed from C_n. Then, we let $\{g_{n+1}, \ldots, g_{n+t}\}$ be the set of gates of out-degree at least two, and let $\{g_{n+t+1}, \ldots, g_{n+s}\}$ be the set of the other gates. Observe that "`push` i" for $n + 1 \leq i \leq n + s$ appears in P_n if and only if g_i is of out-degree at least two. Thus, i takes its value from one to $n + t$, and hence we only need $\lceil \log(n + t) \rceil$ bits for describing the value of i. By the proof of Theorem 2.1, the length of the description of P_n is

$$(s + 1)(1 + \lceil \log(n + t) \rceil) + O(s),$$

which is $s \log(n + t) + O(s)$. From this, we have $|\mathcal{C}(t)| \leq 2^{s \log(n+t)+O(s)}$.

The other different thing is that we estimate how much we get to over-estimate $|\mathcal{C}(t)|$ if we apply the same analysis as that in the proof of Theorem 2.1. Note here that we already have $|\mathcal{C}(t)| \le 2^{s\log(n+t)+cs}$, which is still an over-estimated bound. Let $C_n \in \mathcal{C}(t)$ be an arbitrary circuit of size s. Recall that $\{g_{n+1}, \dots, g_{n+t}\}$ is the set of gates of out-degree at least two. Consider that we arbitrarily number $\{g_{n+1}, \dots, g_{n+t}\}$ with $n+1, \dots, n+t$, and the other gates with $n+t+1, \dots, n+s$. (In the above, we have numbered g_i with i for $n+1 \le i \le n+s$.) Note that there are $t!(s-t)!$ such numberings for the circuit C_n. Let $C_n^{(1)}$ and $C_n^{(2)}$ be two circuits identical to C_n that have distinct numberings. Let $P_n^{(1)}$ and $P_n^{(2)}$ be the stack programs constructed from $C_n^{(1)}$ and $C_n^{(2)}$, respectively. The constructions are done by the algorithm shown in Fig. 1, but, in case that "`push` i" for $n+1 \le i \le n+s$ is added to P_n, we add "`push` a" to P_n, where g_i is numbered with a. (Note that "`push` a" does not appear in P_n for any $n+t+1 \le a \le s$.) Then, it is easy to see the following claim.

Claim 1. If $C_1^{(1)}$ and $C_n^{(2)}$ have different numberings on gates $g_{n+1}, \dots, g_{n+t}$, then the two descrptions of $P_n^{(1)}$ and $P_n^{(2)}$ are different. Otherwise, these two are same.

Besides this claim, for any two distinct circuits of $\mathcal{C}(t)$, the descriptions of the two stack programs constructed are different (however those gates are numbered). Thus, there are exactly $t!$ (distinct) descriptions for each circuit of $\mathcal{C}(t)$ that are also different from those for the other circuits. Therefore, we conclude that $|\mathcal{C}(t)|$ is at most $2^{s\log(n+t)+O(s)}/t!$. □

From this lemma (and using $t! \ge (t/e)^t = 2^{t\log t - t\log e}$), we see that the total number of circuits C_n of size at most s is at most

$$\begin{aligned} s\cdot\left|\bigcup_{0\le t\le s}\mathcal{C}(t)\right| = s\cdot\sum_{0\le t\le s}|\mathcal{C}(t)| &\le s^2\cdot\max_{0\le t\le s}\left\{\frac{2^{s\log(t+n)+O(s)}}{t!}\right\} \\ &\le \max_{0\le t\le s}\left\{2^{2\log s + s\log(t+n)+O(s)-t\log t+t\log e}\right\} \\ &\le \max_{0\le t\le s}\left\{2^{s\log t - t\log t+O(s)}\right\}. \quad \text{(asumming } t\ge n) \end{aligned}$$

Let $t = \alpha s$ for any $\alpha = \alpha(s) : 0 \le \alpha \le 1$. We estimate the maximum of $f(\alpha) = \log\alpha s - \alpha\log\alpha s$ over $0 \le \alpha \le 1$. By an elementary calculation, $f(\alpha)$ is maximized at $\alpha = c_0/\log s$ for some $c_0 : 1.44 < 1/\ln 2 < c_0 < 1.5$ if s is sufficiently large. Let $t = c_0 s/\log s$. (This value of t is at least n when $s = \Omega(n^2)$.) Since there are 2^{2^n} distinct Boolean functions on n inputs, we must have $s\log t - t\log t + O(s) \ge 2^n$. Then, we derive a contradiction to this inequality if we assume $s = (2^n/n)(1 + 2\log n/n - c/n)$ for some constant $c > 0$:

$$\begin{aligned} s\log\frac{c_0 s}{\log s} - \frac{c_0 s}{\log s}\log\frac{c_0 s}{\log s} + O(s) &= s\left(\log\frac{c_0 s}{\log s} - \frac{c_0}{\log s}\log\frac{c_0 s}{\log s} + O(1)\right) \\ &\le s\left(\log s - \log\log s + O(1)\right) \\ &\le s(n - 2\log n + O(1)). \end{aligned}$$

Applying the above value of s with sufficiently large constant $c > 0$, we have

$$\begin{aligned} \frac{2^n}{n}\left(1 + \frac{2\log n}{n} - \frac{c}{n}\right)(n - 2\log n + O(1)) &\le \frac{2^n}{n}(n - c + O(1)) \\ &< 2^n. \end{aligned}$$

References

[1] G. S. Frandsen and P. B. Miltersen, Reviewing bounds on the circuit size of the hardest functions, Information Processing Letters 95, pp. 354-357, 2005.

[2] O. B. Lupanov, The synthesis of contact circuits, Dokl. Akad. Nauk SSSR (N.S.) 119, pp. 23-26, 1958.

[3] C. E. Shannon, The synthesis of two-terminal switching circuits, Bell System Tech. J. 28, pp. 59-98, 1949.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de