2. Lower bounds on the network complexity of Boolean functions

References

- Claus-Peter Schnorr, Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen, Computing 13 (1974), 155 - 171.
- Wolfgang Paul, A 2.5n-lower bound on the combinational complexity of boolean functions, SIAM J. Comput. 6 (1977), 427 - 443.
- Larry Stockmeyer, On the combinational complexity of certain symmetric Boolean functions, Math. Systems Theory 10 (1977), 323 - 336.
- Norbert Blum, A Boolean function requiring 3n network size, TCS 28 (1984), 337 - 345.
- Uri Zwick, A 4n lower on the combinational complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions, SIAM J. Comput. 20 (1991), 499 - 505.

Although the network complexity of almost all Boolean functions is exponential, no nonlinear lower bound of an explicit defined Boolean function in NP is known. We shall present some of the few known linear lower bound

proofs. First, we shall consider the base B_2.

A function $f \in B_n$ is called <u>degenerated</u> if f does not depend on all variables $x_1, x_2, \ldots, x_n$. This means that there is a variable x_i such that the subfunctions of f after setting $x_i := 0$ and $x_i := 1$ are equal.

<u>Theorem 2.1</u>

$C_{B_2}(f) \geq n-1$ for all nondegenerated functions in B_n.

<u>Proof:</u>

Let β be an optimal B_2-network for f.

<u>Observation</u>

- The outdegree of each input node x_i in β is at least one. (Otherwise, f would not depend on x_i).
- Each gate which is not the output node has outdegree at least one. (Otherwise, we could eliminate this gate without changing the function computed at the output node.)

$\Rightarrow$

Number of edges $\geq n + c - 1$,

where c is the number of gates of indegree two in β.

Each edge has a gate of indegree two as end node.

$\Rightarrow$

$$\text{Number of edges} = 2 \cdot c$$

$\Rightarrow$

$$2c \geq n + c - 1$$

$$\Leftrightarrow \quad c \geq n - 1.$$

■

The proof of Theorem 2.1 is only based on the graph structure of the network β. A better bound cannot be obtained in this way since there are nondegenerated functions in B_n which have network size $n-1$ (e.g., $f: \{0,1\}^n \to \{0,1\}$ with $f(x_1, x_2, \ldots, x_n) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$).

$\Rightarrow$

We need further techniques to prove larger lower bounds.

All proofs of larger lower bounds use the so-called <u>gate-elimination method</u>. The gate-elimination method uses induction. By an assignment of some variables with values from $\{0,1\}$, a specific number of gates are eliminated in each step and the resulting function is of the same type as the function before the assignment.

We shall demonstrate the gate-elimination method proving some lower bounds. First, we need a definition.

A Boolean function $f \in B_n$ belongs to the class $Q^n_{2,3}$ if for all different $i, j \in \{1, 2, \ldots, n\}$, the following is fulfilled.

i) After the replacement of x_i and x_j by constants, we obtain at least three different subfunctions.

ii) If $n \geq 4$ then there is a constant c_i such that after the replacement of x_i by c_i, we obtain a subfunction in $Q^{n-1}_{2,3}$.

Schnorr has defined the class $Q^n_{2,3}$ in such a way that a lower bound $2n-3$ for each function in this class can be proved easily.

Theorem 2.2

For all $f \in Q^n_{2,3}$ there holds $C_{B_2}(f) \geq 2n-3$.

Proof:

Observe that each function $f \in Q^n_{2,3}$ is node-generated. If f would not depend on x_i then f could have at most two different subfunctions with respect to x_i and any $x_j \neq x_i$.

Let β be an optimal B_2-network for f.

Consider a gate v in $\mathcal{B}$ such that both direct predecessors of v are input nodes. Obviously, such a gate exists.

Optimality of $\mathcal{B}$ $\Rightarrow$
Both input nodes correspond to different variables x_i and x_j.

If both input nodes have outdegree one, then f depends on x_i and x_j only via the gate v. v can only compute 0 or 1.

$\Rightarrow$

f can have at most two different subfunctions with respect to x_i and x_j. This contradicts the definition of $Q^n_{2,3}$.

$\Rightarrow$

At least one of the two input nodes has outdegree at least two.

W.l.o.g. we can assume that the input node with label x_i has outdegree ≥ 2.

We distinguish two cases.

Case 1: $n = 3$

Then at least four edges leave the input nodes. Analogous to the proof of Theorem 2.1, we can show the existence of 3 gates in $\mathcal{B}$. exercise

Case 2: $n \geq 4$

Assume that the assertion is proved for $\ell < n$.
Replace x_i by c_i.

$\Rightarrow$

- At least two successors of the input node x_i can be eliminated.
- The resulting network β' computes a function in $Q^{n-1}_{2,3}$.

$\Rightarrow$

assumption

β' contains at least $2(n-1)-3$ gates.

$\Rightarrow$

β contains at least $2n-3$ gates.

■

Example:

Counting function:

$$C^n_{k,m} : \{0,1\}^n \longrightarrow \{0,1\}$$

where

$$C^n_{k,m}(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i = k \bmod m \\ 0 & \text{otherwise} \end{cases}$$

Let us consider $C^n_{k,3}$.

After the replacement of x_i and x_j by constants, we obtain the different functions

$$\begin{cases} C_{k,3}^{n-2} & \text{if } x_i + x_j = 0 \\ C_{k-1,3}^{n-2} & \text{if } x_i + x_j = 1 \\ C_{k-2,3}^{n-2} & \text{if } x_i + x_j = 2 \end{cases}$$

For $n \geq 4$, we define $c_i := 0$ for all i. After the replacement of x_i by c_i, we obtain the subfunction $C_{k,3}^{n-1}$ which is in $Q_{2,3}^{n-1}$.

The so-called storage access function $SA_n \in B_{n+k}$ where $n = 2^k$ is defined on a k-bit number $a = a_1 a_2 \dots a_k$ and n variables $x_1, x_2, \dots, x_n$ by

$$SA_n(a, x) = x_{(a)} \quad \text{where}$$

(a) is the binary number represented by a_{n+1}

Wolfgang Paul has proved a $2n-2$ lower bound for the network complexity of SA_n.

Theorem 2.3

$$C_{B_2}(SA_n) \geq 2n - 2.$$

Proof:

Our goal is to give an inductive proof. But the replacement of x-variables does not lead to

storage access functions. Hence, we define a larger class of functions.

Let $s \in \{1,2,\ldots,n\}$. F_s contains all functions $f \in B_{n+2}$ such that there exists $S \subseteq \{1,2,\ldots,n\}$, $|S| = s$ such that for all a with $(a) \in S$

$$f(a, x_1, x_2, \ldots, x_n) = x_{(a)}.$$

Since $SA_n \in F_n$, it suffices to prove a $2s-2$ lower bound for all functions in F_s.

$s = 1$: trivial since $2s-2 = 0$.

Let $s \geq 2$. Assume that the lower bound holds for $\ell < s$, i.e., $C_{B_2}(f) \geq 2\ell - 2$ for all $f \in F_\ell$.

Consider any function $f \in F_s$. Let β be an optimal B_2-network for f. We prove that we obtain a network for a subfunction in F_{s-1} after the elimination of at least two gates.
Three cases are possible:

Case 1: $\exists i \in S$: input node x_i has outdegree ≥ 2.

Setting x_i to a constant eliminates at least two gates. Replace S by $S \setminus \{i\}$. The resulting network computes a function $f' \in F_{s-1}$. By assumption $C_{B_2}(f') \geq 2(s-1) - 2$.

$$\Rightarrow C_{B_2}(f) \geq 2s - 2.$$

Case 2: $\exists i \in S$: input node x_i has outdegree one and the edge from x_i goes into an $\wedge$-type gate v; i.e., $res_\beta(v) = (x_i^b \wedge g^c)^d$ with $b,c,d \in \{0,1\}$ and g is a function of the other input variables.

If we replace x_i by $\bar{b}$ then the output of v is replaced by the constant 0^d. Furthermore, the resulting network computes a function in F_{S-1}.

$\Rightarrow$

v cannot be the output gate.

At least the gate v and its successor have been eliminated. As in Case 1, we obtain

$$C_{B_2}(f) \geq 2s - 2.$$

Case 3: $\exists i \in S$: input node x_i has outdegree one and the edge from x_i goes into a $\oplus$-type gate v.

Then $res_\beta(v) = x_i \oplus g \oplus b$ for some function g of the other variables and $b \in \{0,1\}$.

Furthermore, v cannot be the output gate. To see this, consider $j \in S \setminus \{i\}$ and $\sigma(a) = j$. The network has to compute x_j independently from the value of x_i. A change of the value of x_i changes the result of the gate v.

The value of x_i has no influence on the output for all $(a) \in S \setminus \{i\}$.

$\Rightarrow$

After replacing x_i by an arbitrary function, we obtain a network for a function in F_{S-1}.

$\rightsquigarrow$

Choose the function g to replace x_i. Then the gate v computes the constant 6.

$\Rightarrow$

v and its successors are eliminated.

$\Rightarrow$

At least two gates are eliminated.

As in Case 1, we obtain

$$C_{B_2}(f) \geq 2s-2.$$

□

Wolfgang Paul has also proved a $2.5n$ lower bound for the B_2-complexity for a function

$$f: \{0,1\}^{n+2\log n+1} \longrightarrow \{0,1\},$$

defined in the following way:

Let

$$a_1 = a_1, a_2, \ldots, a_{\log n}$$
$$a_2 = a_{\log n+1}, \ldots, a_{2\log n}$$

and

$$f(a_1, a_2, \ldots, a_{2\log n}, q, x_1, x_2, \ldots, x_n)$$

$$= \begin{cases} x_{(\sigma_1)} \wedge x_{(\sigma_2)} & \text{if } q = 1 \\ x_{(\sigma_1)} \oplus x_{(\sigma_2)} & \text{if } q = 0. \end{cases}$$

Larry Stockmeyer has applied the ideas of Paul to prove a $2.5n$ lower bound for several fundamental symmetric functions.

A function $f \in B_{n,m}$ is <u>symmetric</u> if

$$f(x_1, x_2, \ldots, x_n) = f(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}, \ldots, x_{\pi(n)})$$

for all permutations π of $1, 2, \ldots, n$.

We have considered the function

$$f : \{0,1\}^{n + 3\log n + 1} \longrightarrow \{0,1\}$$

where

$$f(a_1, a_2, \ldots, a_{3\log n}, r, x_1, x_2, \ldots, x_n)$$

$$= x_{(\sigma_1)}^{r} \wedge (x_{(\sigma_2)} \oplus x_{(\sigma_3)})$$

with

$$\sigma_3 = a_{2\log n + 1}, \ldots, a_{3\log n}.$$

We have extended the method of Paul and proved that $C_{B_2}(f) \geq 3n - 3$.

Note that the function f is similar to the function of Paul.

- If $r = 1$ and $x_{(\sigma_1)} = 1$ then

$$f(\sigma_1, \sigma_2, \sigma_3, r, x_1, x_2, \ldots, x_n) = x_{(\sigma_2)} \oplus x_{(\sigma_3)}.$$

- If $r = 1$ and $x_{(\sigma_3)} = 0$ then

$$f(\sigma_1, \sigma_2, \sigma_3, r, x_1, x_2, \ldots, x_n) = x_{(\sigma_1)} \wedge x_{(\sigma_2)}.$$

We shall present the proof of this lower bound.

Throughout the proof, we shall use the following lemma

Lemma 2.1

Let β be a network computing $f \in B_n$. Let $v \in \beta$ be an $\wedge$-type gate or a $\oplus$-type gate. If one input function of v is constant then we can eliminate gate v and the reduced network still computes f.

Proof:

exercise

∎

Let $U \subset V_n$ and $\alpha : U \to \{0,1\}$ be a mapping. Frequently, we shall consider the restriction f_α of $f \in B_n$ under the assignment α. More precisely, f_α is defined by