Limiting Negations in Non-Deterministic Circuits

Hiroki Morizumi*

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan
morizumi@ecei.tohoku.ac.jp

Abstract

The minimum number of NOT gates in a Boolean circuit computing a Boolean function \( f \) is called the inversion complexity of \( f \). In 1958, Markov determined the inversion complexity of every Boolean function and particularly proved that \( \lceil \log_2(n + 1) \rceil \) NOT gates are sufficient to compute any Boolean function on \( n \) variables. In this paper, we consider circuits computing non-deterministically and determine the inversion complexity of every Boolean function. In particular, we prove that one NOT gate is sufficient to compute any Boolean function in non-deterministic circuits if we can use an arbitrary number of guess inputs.

Keywords: circuit complexity, negation-limited circuit, non-deterministic circuit, inversion complexity

1 Introduction

No superlinear lower bound has been known so far on the size of Boolean circuits computing an explicit Boolean function, while exponential lower bounds have been known on the size of monotone circuits, which consist of only AND and OR gates and do not include NOT gates [1, 4, 12]. It is natural to ask: what happens if a limited number of NOT gates are allowed? This motivates us to study the negation-limited circuit complexity under various scenarios [2, 3, 5, 8, 14].

When we consider Boolean circuits with a limited number of NOT gates, there is a basic question: Can a given Boolean function be computed by a circuit with a limited number of NOT gates? This question was answered by Markov [10] in 1958 and the result plays an important role in the study of the negation-limited circuit complexity. The inversion complexity of a Boolean function \( f \) is the minimum number of NOT gates required to construct a Boolean circuit computing \( f \), and Markov completely determined the inversion complexity of every Boolean function \( f \). In particular, it has been shown that \( \lceil \log_2(n + 1) \rceil \) NOT gates are sufficient to compute any Boolean function.

*Present Affiliation: Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan (morizumi@kuis.kyoto-u.ac.jp)
In restricted circuits this situation changes. After more than 30 years from the result of Markov, Santha and Wilson [13] investigated the inversion complexity in constant depth circuits and showed that under this restriction $\lceil \log_2(n + 1) \rceil$ NOT gates are not sufficient to compute a Boolean function. The result has been extended to bounded depth circuits by Sung and Tanaka [15]. In this paper, we investigate the opposite direction, i.e., whether in more powerful circuits the inversion complexity decreases or not. We consider circuits computing non-deterministically. Non-deterministic circuits appear in the definition of NP/poly and are studied in, e.g., [9]. The definition of non-deterministic circuits will be given in Section 2. We completely determine the inversion complexity of every Boolean function in non-deterministic circuits. In particular, we prove that one NOT gate is sufficient to compute any Boolean function in non-deterministic circuits if we can use an arbitrary number of guess inputs. The comparison of deterministic computation and non-deterministic computation has been studied under various situations, e.g., Turing machines, finite automata and communication complexity. Our results can be considered a comparison of the inversion complexity in deterministic circuits and non-deterministic circuits.

2 Preliminaries

A circuit is an acyclic Boolean circuit which consists of AND gates of fan-in two, OR gates of fan-in two and NOT gates. The size of a circuit is the number of AND gates, OR gates and NOT gates in the circuit. A non-deterministic circuit is a circuit with actual inputs $(x_1, \ldots, x_n) \in \{0, 1\}^n$ and some further inputs $(y_1, \ldots, y_m) \in \{0, 1\}^m$ called guess inputs. A non-deterministic circuit computes a Boolean function $f$ as follows: For $x \in \{0, 1\}^n$, $f(x) = 1$ iff there exists a setting of the guess inputs $(y_1, \ldots, y_m)$ which makes the circuit output 1. In this paper, we call a circuit without guess inputs a deterministic circuit to distinguish it from a non-deterministic circuit, and we call a non-deterministic circuit which has $m$ guess inputs and includes at most $r$ NOT gates an $(m, r)$ non-deterministic circuit.

Let $x$ and $x'$ be Boolean vectors in $(0, 1)^n$. $x \leq x'$ means $x_i \leq x_i'$ for all $1 \leq i \leq n$. $x < x'$ means $x \leq x'$ and $x_i < x'_i$ for some $i$. A Boolean function $f$ is called monotone if $f(x) \leq f(x')$ whenever $x \leq x'$.

The theorem of Markov [10] is in the following. We denote the inversion complexity of a Boolean function $f$ in deterministic circuits by $I(f)$. A chain is an increasing sequence $x^1 < x^2 < \cdots < x^k$ of Boolean vectors in $(0, 1)^n$. The decrease $d_X(f)$ of a Boolean function $f$ on a chain $X$ is the number of indices $i$ such that $f(x^i) \leq f(x^{i+1})$. The decrease $d(f)$ of $f$ is the maximum of $d_X(f)$ over all increasing sequences $X$. Markov gave the tight bound of the inversion complexity for every Boolean function.

**Theorem 1** (Markov[10]). For every Boolean function $f$,

$$I(f) = \lceil \log_2(d(f) + 1) \rceil.$$ 

In Theorem 1, the Boolean function $f$ can also be a multi-output function.
3 Inversion Complexity in Non-Deterministic Circuits

3.1 Result

We denote by $I_{ndc}(f, m)$ the inversion complexity of a Boolean function $f$ in non-deterministic circuits with at most $m$ guess inputs. We consider only single-output Boolean functions since non-deterministic circuits are not defined as ones computing multi-output Boolean functions. We determine $I_{ndc}(f, m)$ for every Boolean function as the following theorem:

**Theorem 2.** For every Boolean function $f$,

$$I_{ndc}(f, m) = \lceil \log_2(d(f)/2^m + 1) \rceil.$$ 

If we do not consider the number of guess inputs, we obtain the following corollary from Theorem 2. We denote by $I_{ndc}(f)$ the inversion complexity of $f$ in non-deterministic circuits with an arbitrary number of guess inputs.

**Corollary 1.** For every Boolean function $f$,

$$I_{ndc}(f) = \begin{cases} 0 & \text{if } f \text{ is monotone;} \\ 1 & \text{otherwise}. \end{cases}$$

**Proof.** If $f$ is monotone, then $d(f) = 0$ and therefore $I_{ndc}(f, m) = 0$ for all $m$.

If $f$ is not monotone, then $d(f) > 0$ and therefore $I_{ndc}(f, m) > 0$ for all $m$. We substitute $\lceil \log_2(d(f)) \rceil$ for $m$, then $I_{ndc}(f, d(f)) = 1$. \hfill $\square$

In the rest of this section, we prove Theorem 2.

3.2 Easy upper bound

Before we prove Theorem 2, we give a simple construction of non-deterministic circuits computing a Boolean function $f$, which construction gives

$$I_{ndc}(f, m) \leq \max\{I(f) - m, 0\} + 1.$$ 

The upper bound of $I_{ndc}(f, m)$ above is at most 1 larger than the one of Theorem 2. The construction and the proof sketch are as follows.

By Theorem 1, for every Boolean function $f$, there is a deterministic circuit $C$ which computes $f$ and includes $I(f)$ NOT gates $N_1, \ldots, N_{I(f)}$. Let $i_k$ and $o_k$ be the input and the output of $N_k$ respectively for $1 \leq k \leq I(f)$. Let $z$ be the output of $C$. We construct the $(m, \max\{I(f) - m, 0\} + 1)$ non-deterministic circuit $C'$ computing $f$ from $C$ as follows. See Fig. 1. Intuitively speaking, $m$ guess inputs are used to guess the outputs of $m$ NOT gates and one additional NOT gate is needed to guarantee correctness of the guess.

1. Remove $N_k$ in $C$ for $1 \leq k \leq m$.

2. Prepare one NOT gate $N$ and guess inputs $y_1, y_2, \ldots, y_m$. 

3
3. Connect $y_k$ to all gates which were connected from the output of $N_k$ for $1 \leq k \leq m$. Thus now, $o_k = y_k$.

4. Compute $z \land (\land_{k=1}^m (i_k \lor y_k)) \land (\neg \lor_{k=1}^m (i_k \land y_k))$ as the output of $C'$.

If $i_k = y_k = 0$ for some $k$, then $(\land_{k=1}^m (i_k \lor y_k)) = 0$. If $i_k = y_k = 1$ for some $k$, then $(\neg \lor_{k=1}^m (i_k \land y_k)) = 0$. Therefore the output of $C'$ depends only on the case with $i_k = \neg y_k$ for all $1 \leq k \leq m$. In this case, $z = f$ since $o_k = y_k = \neg i_k$ for all $1 \leq k \leq m$.

3.3 Tight upper bound

We prove $I_{nec}(f, m) \leq \lceil \log_2(\frac{d(f)}{2^m} + 1) \rceil$. In the simple construction mentioned above, we use one NOT gate $N$ only for guaranteeing that the guess inputs satisfy the condition we hope. In the following proof, for the guarantee we do not use one new NOT gate, but we reuse the other NOT gates.

Proof (the upper bound of $I_{nec}(f, m)$). We use induction on $m$. 

Figure 1: A simple construction. (Actually, the gates with more than two fan-in in the figure consist of some gates with two fan-in.)
Figure 2: The separation of $f$.

**Base:** $m = 0$. A non-deterministic circuit with no guess input is a deterministic circuit. Therefore by Theorem 1,

$$I_{ndc}(f, 0) = \lceil \log_2(d(f) + 1) \rceil.$$

**Induction Step:** Suppose

$$I_{ndc}(f, m') \leq \lceil \log_2(d(f)/2^{m'} + 1) \rceil$$

for all $m' \leq m - 1$.

First we separate $f$ to three functions $f_1, f_2$ and $f_3$ so that $f_1 \lor f_2 \lor f_3 = f$ as follows.

See Fig. 2. Let $S_1$ be the set of all vectors $x \in \{0, 1\}^n$ such that for every chain $X$ starting with $x$,

$$d_X(f) = 0 \text{ and } f(x) = 1.$$

Let $S_2$ be the set of all vectors $x \in \{0, 1\}^n$ such that $x \not\in S_1$ and for every chain $X$ starting with $x$,

$$d_X(f) \leq \lfloor d(f)/2 \rfloor.$$

Let $S_3$ be the set of all vectors $x \in \{0, 1\}^n$ such that $x \not\in S_1 \cup S_2$. For $i = 1, 2, 3$, we define $f_i$ as follows:

$$f_i(x) = \begin{cases} f(x) & \text{if } x \in S_i; \\ 0 & \text{otherwise.} \end{cases}$$

We define $f_t$ as follows:

$$f_t(x) = \begin{cases} 1 & \text{if } x \in S_1 \cup S_2; \\ 0 & \text{otherwise.} \end{cases}$$

Since $f_2 = 0$ outside of $S_2$ and $S_2$ does not include any vector $x \in \{0, 1\}^n$ such that for a chain $X$ starting with $x$, $d_X(f) > \lfloor d(f)/2 \rfloor$,

$$d(f_2) \leq \lfloor d(f)/2 \rfloor.$$
Next we show that
\[ d(f_3) \leq \lceil d(f)/2 \rceil. \]

We assume that \( d(f_3) > \lceil d(f)/2 \rceil \). Since \( f_3 = 0 \) outside of \( S_3 \), there is a chain \( X_1 \) ending in a vector \( x \in S_3 \) and such that \( d_{X_1}(f) > \lceil d(f)/2 \rceil - 1 \). Since \( x \) is not in \( S_2 \), there is a chain \( X_2 \) starting with \( x \) and such that \( d_{X_2}(f) \geq \lceil d(f)/2 \rceil + 1 \). Let \( X' \) be the chain which is obtained by connecting \( X_1 \) and \( X_2 \). Then,

\[
\begin{align*}
    d_{X'}(f) &= d_{X_1}(f) + d_{X_2}(f) \\
    &> (\lceil d(f)/2 \rceil - 1) + (\lceil d(f)/2 \rceil + 1) = d(f).
\end{align*}
\]

Thus a contradiction happens.

Let \( r = \lceil \log_2(\lceil d(f)/2 \rceil / 2^{m-1} + 1) \rceil \). By the supposition, there are \( (m - 1, r) \) non-deterministic circuits \( C_2 \) and \( C_3 \) computing \( f_2 \) and \( f_3 \) respectively. Let \( N_{jk} \) be the \( k \)th NOT gate from the input side in \( C_j \) for \( j = 2, 3 \) and \( 1 \leq k \leq r \). More precisely, we choose each number \( k \) so that there is no path from the output of \( N_{jk} \) to the input of \( N_{jk'} \) if \( k' \leq k \). Let \( i_{jk} \) and \( o_{jk} \) be the input and the output of \( N_{jk} \) respectively. If \( N_{jk} \) does not exist, then we define \( i_{jk} = 0 \). Let \( z_j \) be the output of \( C_j \) for \( j = 2, 3 \). Since \( f_1 \) and \( f_t \) are monotone functions by the definition, there are deterministic circuits \( C_1 \) and \( C_t \) which include no NOT gate and compute \( f_1 \) and \( f_t \) respectively. We construct the \( (m, r) \) non-deterministic circuit \( C \) computing \( f \) from \( C_1, C_2, C_3 \) and \( C_t \) as follows. See Fig. 3. In the figure, \( C_1 \) and \( C_t \) are omitted.

1. Remove all NOT gates in \( C_2 \) and \( C_3 \).
2. Prepare \( r \) NOT gates \( N_1, N_2, \ldots, N_r \) and a new guess input \( y_m \).
3. Connect \((i_{2k} \land f_t) \lor (i_{3k} \land y_m) \lor (f_t \land y_m)\) to the input of \( N_k \) for \( 1 \leq k \leq r \).
4. Connect the output of \( N_k \) to all gates which were connected from the output of \( N_{2k} \) or \( N_{3k} \) for \( 1 \leq k \leq r \). Thus now,

\[
o_{2k} = o_{3k} = \neg((i_{2k} \land f_t) \lor (i_{3k} \land y_m) \lor (f_t \land y_m)).
\]

5. Compute \((z_2 \land f_t) \lor (z_3 \land y_m) \lor f_1\) as the output of \( C \).

Although \( C \) has connections between the gates of \( C_2 \) and the gates of \( C_3 \), \( C \) has no loop, since if there is a path of an order \( i_{jk_1}, o_{jk_2}, i_{jk_3}, o_{jk_4} \), then \( k_1 \leq k_2 \leq k_3 \leq k_4 \). We show that \( C \) computes \( f \) for each of the following three cases.

Case 1: The input \( x \) is in \( S_1 \).

By the definition of \( S_1 \), \( f(x) = 1 \) for all \( x \in S_1 \). Since \( f_1(x) = 1 \) for all \( x \in S_1 \), the output of \( C \) is

\[
(z_2 \land f_t) \lor (z_3 \land y_m) \lor f_1 = 1
\]

for arbitrary \( y_1, \ldots, y_m \). Thus \( C \) computes \( f(x) \) for all \( x \in S_1 \).
Figure 3: The \((m, r)\) non-deterministic circuit computing \(f\). (The subcircuits computing \(f_1(x)\) and \(f_t(x)\) are omitted.)

Case 2: The input \(x\) is in \(S_2\).

By the definition, \(f_t(x) = 1\) and \(f_1(x) = 0\) for all \(x \in S_2\). If \(y_m = 1\), then we can show that the output of \(C\) is 0 as follows. Since \(f_t(x) = 1\) and \(y_m = 1\),

\[
o_{2k} = o_{3k} = -((i_{2k} \land f_t) \lor (i_{3k} \land y_m) \lor (f_t \land y_m)) = 0.
\]

Then, by the following lemma, we can show that the output of \(C\) is 0.

**Lemma 1.** For each \(j = 2, 3\), if \(o_{jk} = 0\) for all \(1 \leq k \leq r\), then \(z_j = 0\) for any inputs \(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_{m-1}\).

*Proof.* We use that \(f_j(1, 1, \ldots, 1) = 0\), which is proved as follows. If \(f(1, 1, \ldots, 1) = 0\), then \(f_j(1, 1, \ldots, 1) = 0\) since \(f_j\) is defined as \(f\) or 0. If \(f(1, 1, \ldots, 1) = 1\), then \((1, 1, \ldots, 1)\) is included in \(S_1\) and is not included in \(S_j\). Therefore \(f_j(1, 1, \ldots, 1) = 0\).

We assume that there are \(x = (x_1, x_2, \ldots, x_n)\) and \(y = (y_1, y_2, \ldots, y_{m-1})\) such that \(z_j = 1\). On the other hand, we consider the original \(C_j\), i.e., \(C_j\) before the construction of \(C\). Since \(f_j(1, 1, \ldots, 1) = 0\), if the inputs of \(C_j\) are \(x' = (1, 1, \ldots, 1)\) and \(y\), then
\( z_j = 0 \). (The output of NOT gates may be 0 and may be 1.) These two cases include a contradiction since \( z_j \) is computed by only AND gates and OR gates from \( x_1, x_2, \ldots, x_n \) and \( y_1, y_2, \ldots, y_{m-1} \) and \( o_{j1}, o_{j2}, \ldots, o_{jr} \).

Therefore the output of \( C \) depends only on the case with \( y_m = 0 \). The output of \( C \) is
\[
(z_2 \land f_1) \lor (z_3 \land y_m) \lor f_1 = z_2.
\]
Since
\[
o_{2k} = \neg((i_{2k} \land f_1) \lor (i_{3k} \land y_m) \lor (f_1 \land y_m))
= \neg i_{2k},
\]
the output of \( C \) is \( z_2 = f_2 = f \) for all \( x \in S_2 \).

Case 3: The input \( x \) is in \( S_3 \).

By the definition, \( f_t(x) = 0 \) and \( f_1(x) = 0 \) for all \( x \in S_3 \). If \( y_m = 0 \), then the output of \( C \) is
\[
(z_2 \land f_1) \lor (z_3 \land y_m) \lor f_1 = 0.
\]
Therefore the output of \( C \) depends only on the case with \( y_m = 1 \). Then the output of \( C \) is
\[
(z_2 \land f_1) \lor (z_3 \land y_m) \lor f_1 = z_3.
\]
Since
\[
o_{3k} = \neg((i_{2k} \land f_1) \lor (i_{3k} \land y_m) \lor (f_1 \land y_m))
= \neg i_{3k},
\]
the output of \( C \) is \( z_3 = f_3 = f \) for all \( x \in S_3 \).

Thus \( C \) is an \((m, r)\) non-deterministic circuit computing \( f \). The number of NOT gates in \( C \) is at most \( r \). If \( d(f) \) is an even number, then
\[
r = \lceil \log_2(\lceil d(f)/2 \rceil/2^{m-1} + 1) \rceil \\
= \lceil \log_2(d(f)/2^m + 1) \rceil.
\]
If \( d(f) \) is an odd number, then
\[
r = \lceil \log_2(\lceil d(f)/2 \rceil/2^{m-1} + 1) \rceil \\
= \lceil \log_2((d(f) + 1)/2^m + 1) \rceil \\
= \lceil \log_2((d(f) + 1 + 2^m)/2^m) \rceil \\
= \lceil \log_2(d(f) + 2^m) \rceil - m \\
= \lceil \log_2(d(f) + 2^m) \rceil - m \\
= \lceil \log_2(d(f)/2^m + 1) \rceil.
\]
Therefore
\[
\text{Indc}(f, m) \leq \lceil \log_2(d(f)/2^m + 1) \rceil.
\]
\( \square \)
3.4 Lower bound

We prove $\text{Indc}(f, m) \geq \lceil \log_2(d(f)/2^m + 1) \rceil$. We denote the output of a circuit $C$ given an input $x$ and a guess input $y$ by $C(x, y)$.

**Proof (the lower bound of $\text{Indc}(f, m)$).** We use induction on $m$.

**Base:** $m = 0$. A non-deterministic circuit with no guess input is a deterministic circuit. Therefore by Theorem 1, 

$$\text{Indc}(f, 0) = \lceil \log_2(d(f) + 1) \rceil.$$ 

**Induction Step:** Suppose $\text{Indc}(f, m') \geq \lceil \log_2(d(f)/2^{m'} + 1) \rceil$ for all $m' \leq m - 1$. Let $C$ be an $(m, r)$ non-deterministic circuit computing a Boolean function $f$. Let $X$ be an increasing sequence $x^1 < x^2 < \cdots < x^k$ of Boolean vectors in $\{0, 1\}^n$ such that $d_X(f) = d(f)$. In other words, we focus on one of the increasing sequences on which the decrease of $f$ is the maximum. For all $i$ such that $f(x^i) \not\leq f(x^{i+1})$, $f(x^i) = 1$ and $f(x^{i+1}) = 0$. Since $C$ is a non-deterministic circuit, $C$ given $x^i$ outputs 1 for some guess input, and $C$ given $x^{i+1}$ outputs 0 for all guess inputs. Let $y^i \in \{0, 1\}^m$ be a setting to guess inputs of the circuit such that $C(x^i, y^i) = 1$ for each $i$. If $y^m = 0$ for at least half of $y^i$'s, then we fix the last guess input $y_m$ of $C$ to 0; otherwise, we fix $y_m$ to 1. Let $C'$ be the obtained $(m - 1, r)$ non-deterministic circuit. $C'$ computes 1 for at least $\lfloor d(f)/2 \rfloor x^i$'s and computes 0 for all $x^{i+1}$'s. Let $f'$ be a Boolean function computed by $C'$. Then $d(f') \geq d_X(f') \geq \lfloor d(f)/2 \rfloor$. By the supposition,

$$r \geq \text{Indc}(f', m - 1) \geq \lceil \log_2(d(f')/2^{m-1} + 1) \rceil \geq \lceil \log_2([d(f)/2]/2^{m-1} + 1) \rceil \geq \lceil \log_2(d(f)/2^m + 1) \rceil.$$ 

\[ \square \]

4 Concluding Remarks

In this paper we completely determined the inversion complexity of every Boolean function in non-deterministic circuits. In particular, we proved that one NOT gate is sufficient to compute any Boolean function, which means that non-deterministic computation can reduce the number of required NOT gates to compute an arbitrary Boolean function from $\lceil \log_2(n + 1) \rceil$ to 1 in the comparison of Theorem 1. Another interesting computation is probabilistic computation. No result is known for the inversion complexity in probabilistic circuits. About probabilistic circuits, see, e.g., Chap. 12 of [16].

Finally, we note that our construction in Section 3.2 imply a relation about the size of non-deterministic circuits and negation-limited non-deterministic circuits. The relation
corresponds to the known relations in deterministic circuits [5, 6, 7, 11]. We denote by \( \text{size}_{ndc}^d(f) \) the size of the smallest non-deterministic circuit computing a function \( f \) (with an arbitrary number of guess inputs) and denote by \( \text{size}_{r\text{ndc}}^d(f) \) the size of the smallest non-deterministic circuit computing \( f \) with at most \( r \) NOT gates. By a similar construction to the one in Section 3.2, we can prove the following theorem. Note that two AND gates and two OR gates are added for each \( N_k \) in the construction.

**Theorem 3.** For every Boolean function \( f \),
\[
\text{size}_{1\text{ndc}}^d(f) \leq 4\text{size}_{r\text{ndc}}^d(f).
\]

By Theorem 3, if there is a non-deterministic circuit which computes \( f \) and has polynomial size, then there is a non-deterministic circuit which computes \( f \) and has polynomial size and at most one NOT gate.

**Acknowledgment**

I would like to thank anonymous referees of an early version of this paper, in which the main result was Corollary 1 of this paper. Two of the referees pointed out that the problem could be extended to one which has a consideration of the number of guess inputs. I especially thank one of the referees who pointed out that the result in the early version could be extended to the one in Section 3.2.

**References**


