Some Results on the Power of Nondeterministic Computation

Hiroki Morizumi
Shimane University, Japan
morizumi@cis.shimane-u.ac.jp

Abstract

In this paper, we consider the power of nondeterministic computation in two directions. In the first part of this paper, we consider nondeterministic computation and advice strings in bounded space computation. More precisely, we prove that \( L/\text{quasipoly} \supset NL/\text{poly} \). As a corollary, we obtain that \( L/\text{quasipoly} \supset NL \). In the proof, we show a relation between nondeterministic branching programs and bounded width Boolean circuits. In the second part of this paper, we consider the power of nondeterministic circuits. We prove that there is a Boolean function \( f \) such that the nondeterministic \( U_2 \)-circuit complexity of \( f \) is at most \( 2^n + o(n) \) and the deterministic \( U_2 \)-circuit complexity of \( f \) is \( 3n - o(n) \).

1 Introduction

Revealing the power of nondeterministic computation is one of the central problems in computational complexity. In this paper, we consider nondeterministic computation in two directions. We describe the two directions in Section 1.1 and Section 1.2, respectively. Section 1.1 and Section 1.2 correspond to Section 2 and Section 3, respectively.

1.1 Nondeterministic computation in bounded space

We prove the following theorem.

Theorem 1. \( L/\text{quasipoly} \supset NL/\text{poly} \).

See Section 2.1 for the definitions. Since \( NL/\text{poly} \supset NL \), the following corollary is immediately obtained.

Corollary 2. \( L/\text{quasipoly} \supset NL \).

We consider the theorem and the corollary from three points of view below.

The \( L \) vs. \( NL \) problem. Savitch’s theorem \cite{Savitch} shows that \( \text{NSPACE}(f(n)) \subseteq \text{SPACE}(f(n)^2) \) for \( f(n) \geq \log n \). While \( \text{PSPACE} = \text{NPSPACE} \) by the theorem, the \( L \) vs. \( NL \) problem is a longstanding central open problem in computational complexity. Corollary 2 may give
some new insight for the \( L \) vs. \( \text{NL} \) problem. Savitch’s theorem means that nondeterministic computation can be replaced by more spaces in this situation. Corollary 2 means that nondeterministic computation can be replaced by advice strings in the situation.

The \( L / \text{poly} \) vs. \( \text{NL/poly} \) problem. This is the nonuniform variant of the \( L \) vs. \( \text{NL} \) problem and also a longstanding open problem in computational complexity. Theorem 1 can be considered as a result related to the \( L / \text{poly} \) vs. \( \text{NL/poly} \) problem.

The power of advice strings. If we consider nonuniform variant of \( L \), then the size of advice strings is polynomial. Therefore, \( L / \text{poly} \) has been well studied. Theorem 1 and Corollary 2 imply the power of advice strings for the case that the size of advice strings is beyond polynomial.

To prove Theorem 1, we prove the following theorem. This theorem is the fundamental main contribution in the first part of this paper.

**Theorem 3.** Any nondeterministic branching programs of size \( s \) can be converted to a Boolean circuit of size \( 2^{O(\log^2 s)} \) and width \( O(\log s) \).

In Section 2.2 and Section 2.3, we prove Theorem 3 and Theorem 1, respectively.

**1.2 The power of nondeterministic circuits**

Nondeterministic circuits are a nondeterministic variant of Boolean circuits as a computation model. While both of nondeterministic computation and Boolean circuit complexity are central topics in computational complexity, the circuit complexity of nondeterministic circuits is not well studied. The author proved a \( 3(n-1) \) lower bound for the size of nondeterministic \( U_2 \)-circuits computing the parity function in his previous paper [1]. It was known that the minimum size of deterministic \( U_2 \)-circuits computing the parity function exactly equals \( 3(n-1) \) [3]. Thus, nondeterministic computation is useless to compute the parity function by \( U_2 \)-circuits.

In this paper, we consider the opposite directions, i.e., the case that nondeterministic computation is useful. We denote by \( \text{size}^{\text{dc}}(f) \) the size of the smallest deterministic \( U_2 \)-circuit computing a function \( f \), and denote by \( \text{size}^{\text{ndc}}(f) \) the size of the smallest nondeterministic \( U_2 \)-circuit computing a function \( f \). We prove the following theorem.

**Theorem 4.** There is a Boolean function \( f \) such that \( \text{size}^{\text{ndc}}(f) \leq 2n + o(n) \) and \( \text{size}^{\text{dc}}(f) = 3n - o(n) \).

To prove Theorem 4, we introduce a simple proof strategy, and call the key idea *nondeterministic selecting*. In Section 3.2 we explain nondeterministic selecting and the proof outline using it.
2 Nondeterministic computation in bounded space

2.1 Preliminaries

Let $n$ be the input size. $L$ is the class of decision problems solvable by a $O(\log n)$ space Turing machine. $NL$ is the nondeterministic variant of $L$. $L/poly$ is the class of decision problems solvable by a $O(\log n)$ space Turing machine with polynomial size advice strings. $NL/poly$ is the nondeterministic variant of $L/poly$. $L/quasipoly$ is the class of decision problems solvable by a $O(\log n)$ space Turing machine with quasipolynomial size advice strings.

A nondeterministic branching program is a directed acyclic graph. The nodes of non-zero out-degree are called inner nodes and labeled by a variable. The nodes of out-degree 0 are called sinks and labeled by 0 or 1. For each inner node, outgoing edges are labeled by 0 or 1. There is a single specific node called the start node. The output of the nondeterministic branching program is 1 if and only if at least one path leads to 1 sink. The size of branching programs is the number of its nodes.

In Section 2 and Section 3, the gates used in circuits are different. We describe all circuit definitions in each section, independently.

Circuits are formally defined as directed acyclic graphs. The nodes of in-degree 0 are called inputs, and each one of them is labeled by a variable or by a constant 0 or 1. The other nodes are called gates, and each one of them is labeled by a Boolean function. In this section, the gates are AND gates of fan-in two, OR gates of fan-in two, and NOT gates. There is a single specific node called output. The size of a circuit is the number of gates in the circuit.

2.2 Proof of Theorem 3

Lemma 5. If any nondeterministic branching programs of size $s$ can be converted to a Boolean circuit of size $s'$ and width $w$, then any nondeterministic branching programs of size $2s$ can be converted to a Boolean circuit of size $2s^2s' + O(s^2)$ and width $w + 2$.

Proof. Let $G$ be a nondeterministic branching programs of size $2s$. Let $G_1$ and $G_2$ be the former $s$ nodes and the latter $s$ nodes, respectively, in arbitrary topological sorted order. Let $E_1$ be the edges between $G_1$ and $G_2$. The number of edges in $E_1$ is at most $s^2$. All paths from the start node to a sink node contain one edge in $E_1$. For each edge in $E_1$, we check the existence of a path from the start node to the 1 sink node. Natural construction of such circuit is enough to prove the lemma.

Proof of Theorem 3 We apply Lemma 5 recursively.

2.3 Proof of Theorem 1

In this subsection, we prove Theorem 1 by Theorem 3.
Proof of Theorem 1. Let \( n \) be the size of the input. We apply polynomial of \( n \) to \( s \) in Theorem 3. Then, we obtain that any nondeterministic branching programs of polynomial size of \( n \) can be converted to a Boolean circuit of size \( 2^{O(\log^2 n)} \) and width \( O(\log n) \). Nondeterministic branching programs of polynomial size correspond to NL/poly. Boolean circuits of size \( 2^{O(\log^2 n)} \) and width \( O(\log n) \) correspond to L/quasipoly.

\[ \square \]

3 The power of nondeterministic circuits

3.1 Preliminaries

3.1.1 definitions

In Section 2 and Section 3, the gates used in circuits are different. We describe all circuit definitions in each section, independently.

Circuits are formally defined as directed acyclic graphs. The nodes of in-degree 0 are called inputs, and each one of them is labeled by a variable or by a constant 0 or 1. The other nodes are called gates, and each one of them is labeled by a Boolean function. The fan-in of a node is the in-degree of the node, and the fan-out of a node is the out-degree of the node. There is a single specific node called output. The size of a circuit is the number of gates in the circuit.

We denote by \( B_2 \) the set of all Boolean functions \( f : \{0,1\}^2 \to \{0,1\} \). By \( U_2 \) we denote \( B_2 - \{\oplus, \equiv\} \), i.e., \( U_2 \) contains all Boolean functions over two variables except for the XOR function and its complement. A Boolean function in \( U_2 \) can be represented as the following form:

\[
f(x, y) = ((x \oplus a) \land (y \oplus b)) \oplus c,
\]

where \( a, b, c \in \{0,1\} \). A \( U_2 \)-circuit is a circuit in which each gate has fan-in 2 and is labeled by a Boolean function in \( U_2 \).

A nondeterministic 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 nondeterministic 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 section, we call a circuit without guess inputs a deterministic circuit to distinguish it from a nondeterministic circuit.

The parity function of \( n \) inputs \( x_1, \ldots, x_n \), denoted by \( \text{Parity}_n \), is 1 iff \( \sum x_i \equiv 1 \) (mod 2).

3.1.2 the gate elimination method

In our proof, we need the gate elimination method and the result by Schnorr using the method. In this subsection, we have a quick look at them.

Consider a gate \( g \) which is labeled by a Boolean function in \( U_2 \). Recall that any Boolean function in \( U_2 \) can be represented as the following form:

\[
f(x, y) = ((x \oplus a) \land (y \oplus b)) \oplus c,
\]
where \( a, b, c \in \{0, 1\} \). If we fix one of two inputs of \( g \) so that \( x = a \) or \( y = b \), then the output of \( g \) becomes a constant \( c \). In such case, we call that \( g \) is \textit{blocked}.

\textbf{Theorem 6} (Schnorr [3]).

\[
\text{size}^{\text{dc}}(\text{Parity}_n) = 3(n-1).
\]

\textit{Proof.} Assume that \( n \geq 2 \). Let \( C \) be an optimal deterministic \( U_2 \)-circuit computing \( \text{Parity}_n \). Let \( g_1 \) be a top gate in \( C \), i.e., whose two inputs are connected from two inputs \( x_i \) and \( x_j \), \( 1 \leq i, j \leq n \). Then, \( x_i \) must be connected to another gate \( g_2 \), since, if \( x_i \) is connected to only \( g_1 \), then we can block \( g_1 \) by an assignment of a constant to \( x_j \) and the output of \( C \) becomes independent from \( x_i \), which contradicts that \( C \) computes \( \text{Parity}_n \). By a similar reason, \( g_1 \) is not the output of \( C \). Let \( g_3 \) be a gate which is connected from \( g_1 \). See Figure 1.

We prove that we can eliminate at least three gates from \( C \) by an assignment to \( x_i \). We assign a constant 0 or 1 to \( x_i \) such that \( g_1 \) is blocked. Then, we can eliminate \( g_1 \), \( g_2 \) and \( g_3 \). If \( g_2 \) and \( g_3 \) are the same gate, then the output of \( g_2 \) (= \( g_3 \)) becomes a constant, which means that \( g_2 \) (= \( g_3 \)) is not the output of \( C \) and we can eliminate another gate which is connected from \( g_2 \) (= \( g_3 \)). Thus, we can eliminate at least three gates and the circuit come to compute \( \text{Parity}_{n-1} \) or \( \neg \text{Parity}_{n-1} \). For deterministic circuits, it is obvious that \( \text{size}^{\text{dc}}(\text{Parity}_{n-1}) = \text{size}^{\text{dc}}(\neg \text{Parity}_{n-1}) \). Therefore,

\[
\text{size}^{\text{dc}}(\text{Parity}_n) \geq \text{size}^{\text{dc}}(\text{Parity}_{n-1}) + 3 \\
\vdots \\
\geq 3(n-1).
\]

\( x \oplus y \) can be computed with three gates by the following form:

\[
(x \land \neg y) \lor (\neg x \land y).
\]

Therefore, \( \text{size}^{\text{dc}}(\text{Parity}_n) \leq 3(n-1) \). \( \square \)
3.2 Nondeterministic selecting

In this subsection, we describe our idea of the proof. We call the key idea nondeterministic selecting.

Let \( f' : \{0, 1\}^{\sqrt{n}} \rightarrow \{0, 1\} \), and

\[
f = \bigvee_{i=0}^{\sqrt{n}-1} f'(x_{\sqrt{n}i+1}, x_{\sqrt{n}i+2}, \ldots, x_{\sqrt{n}i+\sqrt{n}}).
\]

Nondeterministic circuits can compute \( f \) efficiently. We construct a nondeterministic circuit \( C \) computing \( f \) as follows. Firstly, we select \( \sqrt{n} \) inputs nondeterministically. More precisely, we construct a selector circuit \( C' \) which outputs \( x_{\sqrt{n}i+1}, x_{\sqrt{n}i+2}, \ldots, x_{\sqrt{n}i+\sqrt{n}} \) for each \( i, 0 \leq i \leq \sqrt{n} - 1 \), when guess inputs of \( C \) are assigned to an assignment. Then, one circuit \( C'' \) computing \( f' \) is enough in \( C \). \( \sqrt{n} \) variables of the output of \( C' \) are connected to the input of \( C'' \). It is not difficult to confirm that \( C \) computes \( f \) by the definition of nondeterministic circuits.

On the other hand, a trivial construction of deterministic circuits computing \( f \) needs \( \sqrt{n} \) circuits computing \( f' \). Note that it is a complicated problem (called a direct sum) whether \( \sqrt{n} \) circuits are needed. In our proof of Theorem \( 4 \), we choose the parity function as \( f' \) so that we can prove the large lower bound of \( \text{size}_{dc}(f) \).

3.3 Proof of Theorem \( 4 \)

To prove Theorem \( 4 \) we let

\[
f = \bigvee_{i=0}^{\sqrt{n}-1} \text{Parity}(x_{\sqrt{n}i+1}, x_{\sqrt{n}i+2}, \ldots, x_{\sqrt{n}i+\sqrt{n}}),
\]

and prove two lemmas.

**Lemma 7.** \( \text{size}_{adc}(f) \leq 2n + o(n) \).

*Proof.* We construct a nondeterministic circuit computing \( f \) as mentioned in Section \( 3.2 \). We use \( \lfloor \log \sqrt{n} \rfloor \) guess inputs. The number of gates in the selector circuit is \( 2n + o(n) \). The number of gates in one circuit computing \( \text{Parity}(x_{\sqrt{n}i+1}, x_{\sqrt{n}i+2}, \ldots, x_{\sqrt{n}i+\sqrt{n}}) \) is \( o(n) \) by Theorem \( 6 \). □

**Lemma 8.** \( \text{size}_{dc}(f) = 3n - o(n) \).

*Proof.* Since \( \text{size}_{dc}(\text{Parity}) = 3(n - 1) \) by Theorem \( 6 \), \( \text{size}_{dc}(f) \leq 3n - o(n) \).

We prove that \( \text{size}_{dc}(f) \geq 3n - o(n) \). We refer the proof of Theorem \( 6 \). While we eliminate at least three gates from the circuit by an assignment to \( x_i \) as the proof of Theorem \( 6 \), we modify the proof as follows. If \( x_{\sqrt{n}i+1}, x_{\sqrt{n}i+2}, \ldots, x_{\sqrt{n}i+\sqrt{n}} \) have been assigned except one variable for some \( i, 0 \leq i \leq \sqrt{n} - 1 \), then we assign 0 or 1 to the variable so that \( \text{Parity}(x_{\sqrt{n}i+1}, x_{\sqrt{n}i+2}, \ldots, x_{\sqrt{n}i+\sqrt{n}}) = 0 \) and we do not consider the number of eliminated gates. By the modification, we can eliminate at least \( 3n - o(n) \) gates. □

*Proof of Theorem \( 4 \).* By Lemma \( 7 \) and Lemma \( 8 \), the theorem holds. □
4 Concluding Remarks and Open Problems

In this paper, we considered the power of nondeterministic computation in two somewhat new directions. Many open problems are raised after this paper.

In the first part of this paper, we considered nondeterministic computation and advice strings in bounded space computation. We proved that $L^{\text{quasipoly}} \supseteq NL$ (Corollary 2). It remains open whether this result can be improved to $L^{\text{poly}} \supseteq NL$. Another direction is revealing the power of advice strings in $L$. In this paper, we proved that $L$ with quasipolynomial size advice strings has nontrivial computational power. It may be interesting that some relations between $L$ with advice strings beyond polynomial size and other complexity classes ($P$, $PSPACE$ and so on) are proved.

In the second part of this paper, we considered the power of nondeterministic circuits. To prove the main theorem (Theorem 1), we introduced a simple proof strategy using nondeterministic selecting. It remains open that we use the strategy and prove a similar or improved result of Theorem 1 for $U_2$-circuits or other Boolean circuits.

References

[1] Hiroki Morizumi. Lower bounds for the size of nondeterministic circuits. In Proc. of COCOON, pages 289–296, 2015.

[2] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci., 4(2):177–192, 1970.

[3] Claus-Peter Schnorr. Zwei lineare untere schranken für die komplexität boolescher funktionen. Computing, 13(2):155–171, 1974.