A Note on Polynomial Size Circuits with Low Resource-Bounded Kolmogorov Complexity

Montserrat Hermo

Facultad de Informática
Universidad del País Vasco
Apdo. 649
20080 San Sebastián, Spain
E-mail: montse@si.ehu.es

Elvira Mayordomo

Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo 5
08028 Barcelona, Spain
E-mail: mayordomo@lsi.upc.es

Abstract

It is well-known that the class P/poly *can be characterized in terms of polynomial size circuits. We obtain a characterization of the class* $P/\log$ *using polynomial size circuits with low resource-bounded Kolmogorov Complexity.*

The concept of "small circuits with easy descriptions" has been introduced in the literature as a candidate to characterizing $P/\log$. *We prove that this concept corresponds exactly to the class* $P/O(\log n * \log(\log n))$, *and that this is different from* $P/\log$.

Generalizations of this result are also obtained.

Introduction

The study of nonuniform complexity classes stems from the comparison between uniform models of computation, in which a program is valid for arbitrarily long inputs, and nonuniform models in which each program is valid only for inputs of a fixed length. There are many well-known models for both. Typical examples of uniform models are the Turing machine and other equivalent models such as the RAM; the most important representative of the nonuniform family is the boolean circuit model, although there are many others.

There are several connections between them. Uniformity conditions can be set on the nonuniform model to obtain uniform complexity classes, or vice-versa advice functions can be used to extend uniform classes into nonuniform ones. Our framework is this second approach.

In this context, the most widely studied class is P/poly, which consists of the sets decidable in polynomial time with the help of polynomially long advice strings. This class can be characterized in various manners, one of them as the class of languages decidable by boolean circuits whose size grows polynomially on the length of the input. This important class has been studied in depth (see for instance [1] and the references there). On the contrary, the class $\mathrm{P}/\log$, corresponding to polynomial time with the help of a logarithmically long advice string, has received up to now much less attention. We study it here.

Both classes P/poly and $\mathrm{P}/\log$ are defined with the help of advice functions and were introduced in [4] in order to find relationships between uniformity and nonuniformity. There, two main classes were selected as bounds for the length of the advice words: polynomial bounds and logarithmic bounds, describing respectively P/poly and $\mathrm{P}/\log$. In the same paper, the concept of languages having "small circuits with easy descriptions" was introduced attempting to characterize $\mathrm{P}/\log$. After the results of [4], additional results regarding this class have been much more infrequent. In [1] a characterization is given in terms of uniformly computable functions. Other results were presented in [2], where the authors consider classes of the more general form $C/\log$. We address here the question of finding a characterization of the class $\mathrm{P}/\log$ in terms of circuits.

We also study the above mentioned concept of "small circuits with easy descriptions". Such easy descriptions are again circuits, of logarithmic size, describing the interconnection pattern of the circuit. In [4] it was proved that this class lies between $\mathrm{P}/\log$ and $\mathrm{P}/O(\log n * \log(\log n))$ leaving for the journal version the precise characterization of $\mathrm{P}/\log$. However this journal version [5] did not treat $\mathrm{P}/\log$ in depth. Here we explain why this concept does not characterize precisely $\mathrm{P}/\log$: we show that it corresponds exactly to $\mathrm{P}/O(\log n * \log(\log n))$. Our proof shows that the reason for the circuits with easy descriptions not corresponding to $\mathrm{P}/\log$ is essentially the Shannon-Lupanov effect [8], by which boolean circuits are able to encode a nontrivial amount of information into their interconnection pattern. A boolean function of m inputs can be defined with 2^m bits, but there always exists a $2^m m^{-1} + o(2^m m^{-1})$ size circuit computing the function. The fact that $O(m)$ is exactly the number of bits that are needed to describe each circuit gate and its connections indicates that the interconnection between gates contains relevant information. Thus, this suggests that in order to characterize $\mathrm{P}/\log$, Turing programs must be used instead of logarithmic size circuits.

We are using here Kolmogorov complexity theory. On the one hand, the existence and abundance of high Kolmogorov complexity strings allow us to show limitations in the computational power of a machine type and thus to ensure differences between complexity classes. In most cases this kind of results can be obtained by other methods like diagonalizations, but Kolmogorov complexity provides the tools to get the same results with shorter and cleaner arguments. On the other hand, the concept of "resource-bounded Kolmogorov complexity class" has proved to be very useful in complexity theory because it provides a formal definition of the amount of information needed to construct a particular string. We present here two characterizations of $\mathrm{P}/\log$ with circuits of low resource-bounded Kolmogorov complexity. Our results show that if we choose small circuits generated by Turing machine programs with "short seeds", instead of small circuits with "easy descriptions", then we obtain exactly $\mathrm{P}/\log$.

Preliminaries

Let Σ be the alphabet $\{0,1\}$. Σ^* is the set of all words over Σ. $\Sigma^{=n}$ is the set of all words of length n, and if A is a subset of Σ^*, $A^{=n} = A \cap \Sigma^{=n}$. The strings of Σ^* are ordered by lexicographical order.

The characteristic string of $A^{=n}$ is defined in the standard way: the i^{th} bit of the string is 1 if and only if the i^{th} word of length n of Σ^* is in A. The length of each word $w \in \Sigma^*$ is noted as $|w|$.

We want to be able to encode several words into one, in such a way that both computing the encoding and recovering the coded words can be easily done. We have chosen a pairing function $\langle\rangle \colon \Sigma^* \times \Sigma^* \to \Sigma^*$. Given x and y, the word $\langle x, y\rangle$ is obtained by duplicating each bit of x, duplicating every bit of of y, and appending the duplication of y to that of x inserting a 01 in between. The computations of $\langle\rangle$ and its inverses use a minimum of resources. The pairing function can be generalized to encode tuples as follows: $\langle x, y, z\rangle = \langle\langle x, y\rangle, z\rangle$ and so on.

Our uniform model corresponds to that of "multitape off-line" Turing machine. Let us introduce the description of the nonuniform computation model called circuit [1]. We fix the following family of boolean functions: the 0-ary functions "true" and "false"; the unary function of negation, denoted $\neg$; and the binary functions $\wedge$ and $\vee$. With these functions as basis, we can compute any n-ary boolean function.

Definition 1. Given a set $\delta = \{x_1, x_2, \ldots, x_n\}$ of n boolean variables, a computation chain over δ is a sequence $g_1, g_2, \ldots, g_k$, in which each g_j is

- Either a source element: an element of δ or a boolean constant,

- or a gate: a pair $(\neg, g_l)$ for $1 \leq l \leq j-1$, or a triple (b, g_l, g_m) for $1 \leq l, m \leq j-1$ and $b \in \{\vee, \wedge\}$. The inputs to a gate g_j are the (one or two) smaller elements of the computation chain which appear in g_j.

With each element g_j of a computation chain over $\delta = \{x_1, x_2, \ldots, x_n\}$, we can associate an n-ary function $result(g_j)$ which represents the boolean value computed at element g_j. It is defined as follows.

Definition 2.

$$result(g_j) = \begin{cases} g_j & \text{if } g_j \text{ is a source} \\ \neg result(g_l) & \text{if } g_j = (\neg, g_l) \\ result(g_l) \vee result(g_m) & \text{if } g_j = (\vee, g_l, g_m) \\ result(g_l) \wedge result(g_m) & \text{if } g_j = (\wedge, g_l, g_m) \end{cases}$$

Representing computation chains by acyclic graphs, circuits are obtained:

Definition 3. A boolean circuit is an acyclic graph representation of a computation chain, which is constructed by associating to each step g_j of the chain a node labeled with the variable, constant, or function present in g_j , and joining node g_j to node g_i by a directed edge if g_j is an input to g_i.

Given a circuit C, an encoding of C could be the set B_C defined as follows:

$\{\langle \hat{\imath}, b, \hat{\jmath}, \hat{k}\rangle /$ the gate g_i of C has input gates g_j and g_k, and performs function $b\}$

where $\hat{\imath}, \hat{\jmath}, \hat{k}$ are strings that encode the index of the gates g_i, g_j, g_k respectively. The value of $\hat{\jmath}$ is arbitrary if b is a negation.

Definition 4. Given a boolean function $f \colon \Sigma^n \to \Sigma^m$, which can be expressed as an m-tuple $f = (f_1, \ldots, f_m)$ of boolean functions from Σ^n to Σ, we say that a circuit C computes f if for every $r, 1 \leq r \leq m$, there exists an s, such that $f_r = result(g_s)$.

The cost or size of a circuit is the number of gates it has. Given a boolean function f, its boolean cost is the size of the smallest circuit computing it. A set A has polynomial size circuits if and only if there is a polynomial p such that for each length n the boolean cost of the characteristic function of $A^{=n}$ is bounded by $p(n)$.

Definition 5. [4] A language S has small circuits with easy descriptions iff there are sequences of circuits $\{C_n\}$ and $\{D_n\}$ such that:

i) For all n, C_n accepts $S^{=n}$ and $\{C_n\}$ has polynomial size.

ii) For all n, D_n has less than $c \log n$ gates and accepts B_{C_n}.

The idea behind this is that $\{D_n\}$ is a family of circuits that describes $\{C_n\}$.

We will use the idea, based on results from Lupanov and Shannon, that boolean circuits are able to encode a nontrivial amount of information into their interconnection pattern. Formally:

Theorem 6. [8] Given a boolean function f from $\{0,1\}^m$ to $\{0,1\}$, the size of the smallest circuit $C(f)$ computing f is less or equal to $2^m m^{-1} + o(2^m m^{-1})$.

The next theorem from Savage relates size complexity and the time of a Turing machine computation:

Theorem 7. [7] If a function f is computed by a Deterministic Turing Machine (DTM) in time $T(n)$, then the restriction of f to $\{0,1\}^n$ can be computed by a circuit of size $O(T(n)^2)$.

Definition 8. The circuit value problem, CVP for short, is the set of all pairs $\langle x, y\rangle$ where $x \in \Sigma^*$ and y encodes a circuit with $|x|$ source elements which outputs 1 on input x.

The task of evaluating a circuit on a given input is not too difficult to do.

Theorem 9. [6] CVP is in P.

The evaluation of a circuit on a given input can be done in time bounded by a polynomial in the size of the circuit.

We fix a Universal Turing Machine U. Using it we can denote the unbounded Kolmogorov Complexity of a word w, and the unbounded Kolmogorov Complexity of w relative to another word y as follows:

Definition 10.

i/ $\mathrm{K}(w) = \min\{|x| / U(x) = w\}$,

ii/ $\mathrm{K}(w|y) = \min\{|x| / U(x,y) = w\}$.

The Kolmogorov complexity of a string w (resp. relative to another word y) is the length of the shortest program (resp. pair $\langle \text{program}, y\rangle$) which, when given as input to U, will lead U to write down w as output. The choice of U as the base Universal Machine is irrelevant, as long as a Universal Machine is used, since the Kolmogorov complexity would change only by an additive constant.

Proposition 11. For each length n, and each natural number m, there are at most $2^{n-m+1}-1$ strings w , with $w \leq n$, having Kolmogorov complexity $\mathrm{K}(w) \leq n - m$.

Hartmanis introduces in [3] a tool we will use in this paper: resource-bounded Kolmogorov Complexity. We follow the notation in [3]. Using U and functions f and

$g : N \to N$ define the classes of bounded Kolmogorov complexity sets $\mathrm{K}[f(n), g(n)]$ and $\mathrm{KS}[f(n), g(n)]$ as follows:

Definition 12.

i/ $L \in \mathrm{K}[f(n), g(n)]$ iff $\forall x \in L\ \exists w,\ |w| \leq f(|x|)$ such that $U(w) = x$ in time $\leq g(|x|)$.

ii/ $L \in \mathrm{KS}[f(n), g(n)]$ iff $\forall x \in L\ \exists w,\ |w| \leq f(|x|)$ such that $U(w) = x$ in space $\leq g(|x|)$.

Thus, $\mathrm{K}[f, g]$ (resp. $\mathrm{KS}[f, g]$) is the class of the sets whose strings can be compressed by a factor of f, and which can also be recovered from their compressed form within the time (resp. space) bound g.

Observe that, despite the similarity of notation, $\mathrm{K}(w)$ and $\mathrm{K}(w|y)$ denote functions from words to nonnegative integers, while $\mathrm{K}[f, g]$ and $\mathrm{KS}[f, g]$ are classes of languages.

In the same way as unbounded Kolmogorov complexity, resource-bounded Kolmogorov classes can be relative to a word z

Definition 13.

i/ $L \in \mathrm{K}[f(n), g(n)|z]$ iff $\forall x \in L\ \exists w,\ |w| \leq f(|x|)$ such that $U(w, z) = x$ in time $\leq g(|x|)$.

ii/ $L \in \mathrm{KS}[f(n), g(n)|z]$ iff $\forall x \in L\ \exists w,\ |w| \leq f(|x|)$ such that $U(w, z) = x$ in space $\leq g(|x|)$.

All along this paper we will use the classes $\mathrm{K}[\log, \mathrm{poly}]$, and $\mathrm{KS}[\log, \log]$, where $\log = O(\log n)$ and $\mathrm{poly} = \{n^{O(1)}\}$.

We will work with classes defined by advice functions. Formally:

Definition 14. Given a class of sets C and a class of bounding functions F, the class C/F is formed by the sets L such that for some $B \in C$ and $f \in F$

$$\forall n\ \exists w_n(|w_n| \leq f(n)) \forall x(|x| = n), x \in L \iff \langle x, w_n \rangle \in B.$$

Each word w_n is frequently called the "advice" for the length n. The corresponding function mapping each n into an appropriate advice w_n is called "advice function". C is usually a uniform complexity class. We work here with the class P, and most frequently with the classes poly and log as bounding functions. Thus from the above definition we find as particular interesting cases:

Definition 15.

i/ $L \in \mathrm{P/poly}$ iff there exist $B \in \mathrm{P}$ and a polynomial p, such that

$$\forall n\ \exists w_n(|w_n| \leq p(n))\forall x(|x| = n), x \in L \iff \langle x, w_n \rangle \in B.$$

ii/ $L \in \mathrm{P}/\log$ iff there exists $B \in \mathrm{P}$ and a constant c, such that

$$\forall n\ \exists w_n(|w_n| \leq c\log n)\forall x(|x| = n), x \in L \iff \langle x, w_n \rangle \in B.$$

Main Results

Karp and Lipton stated the following result:

Theorem 16. [4] Let S be a language. Then $S \in \mathrm{P}/\log$ implies that S has small circuits with easy descriptions.

The proof of this is an application of Theorem 7. They didn't present a full converse of this theorem. Their problem was that the encoding of D_n requires $O(\log n * \log(\log n))$ space, so D_n can't be taken as an advice function. Thus they only achieve:

Theorem 17. [4] Sets having small circuits with easy descriptions are in $\mathrm{P}/O(\log n * \log(\log n))$.

We explain why the converse of Theorem 16 does not hold, by proving first, that the converse of Theorem 17 holds, and second, that $\mathrm{P}/\log$ and $\mathrm{P}/O(\log n * \log(\log n))$ differ.

Theorem 18. Let $A \in \mathrm{P}/O(\log n * \log(\log n))$. Then A has small circuits with easy descriptions.

Proof:

Let $A \in \mathrm{P}/O(\log n * \log(\log n))$. This is to say, we have $B \in \mathrm{P}$ such that $\exists c\ \forall n\ \exists w_n$ such that $\forall x(|x| = n), x \in A$ iff $\langle x, w_n \rangle \in B$, with $|w_n| \leq c\log n * \log(\log n))$.

Using Theorem 7 it is possible to construct the polynomial size circuit C_n that recognizes $A^{=n}$. An interesting characteristic of these circuits is that they are generated having a very regular interconnection pattern —a full construction is given in Balcázar, Díaz, and Gabarró [1]—. This happens because to fix a gate to the circuit, in connection with the others, is trivial knowing the number of this gate.

For each x, y the pairing function $\langle x, y\rangle$ is polynomial time computable. We take the circuit that computes it. The inputs corresponding to y are then fixed to the bits of $w_{|x|}$.

Next, since $B \in$P, another circuit generated in the same way simulates the DTM that recognizes B for inputs of size $|\langle x, w_{|x|}\rangle|$.

Composing both circuits we obtain C_n.

The gates of C_n are of two types:

α - the constant gates corresponding to the advice.

β - the gates produced by the simulation of DTM, with a regular interconnection pattern.

These gates can be codified placing gates of type α first, so that they can be distinguished easily.

The easy description D_n of C_n is built as follows:

1) Given $\langle \hat{\imath}, b, \hat{\jmath}, \hat{k}\rangle$ input, we identify the type of the gate g_i.

2) In case that it is a type α gate we notice that $\hat{\imath}$ has but $\log(\log n)+\log(\log(\log n))+\log c$ significant bits. The bits for the gates g_j and g_k are irrelevant, but b indicates whether the gate g_i is zero or one. In this case D_n has to compute a function $f : \{0,1\}^{\log(\log n)+\log(\log(\log n))+\log c} \rightarrow \{0,1\}$. Using Theorem 6 it is possible to have a circuit with size bounded by

$$O\left(\frac{\log n * \log(\log n)}{\log(\log n)+\log(\log(\log n))}\right) \leq O(\log n).$$

3) In case that the gate is type β the circuits C_n obtained using Theorem 7 are generated having a very regular structure (recall the observation at the beginning of this proof), therefore the size of the describing circuit is $O(\log n)$.

Hence, the total number of gates of the corresponding circuit D_n is logarithmic, and C_n has easy descriptions. □

From Theorem 17 and Theorem 18 we obtain:

Corollary 19. The following are equivalent:

i/ $A \in \mathrm{P}/O(\log n * \log(\log n))$.

ii/ A has small circuits with easy descriptions.

On the other hand, we can assure that $\mathrm{P}/O(\log n * \log(\log n))$ and $\mathrm{P}/\log$ are different. In fact, we prove a more general result. When a class uses a longer advice, the class necessarily increases.

Theorem 20. Let D be any recursive class with $\mathrm{DTIME}(O(n))$ included in D and REC the class of all recursive sets. Let F and G be function classes such that $\exists s \in G, s \in o(2^n)$, such that $\forall r \in F, r \in o(s)$. Then D/G is not included in REC/F.

Proof:

The proof is based on an argument from [2] and shows how words with high Kolmogorov complexity can be used to ensure differences between complexity classes

We are going to define a set A in D/G but not in REC/F. Let γ_n be a string such that $|\gamma_n| = s(n)$ and $\mathrm{K}(\gamma_n|n) \geq |\gamma_n|$. The theory of Kolmogorov complexity provides a precise definition words with high complexity, and makes the existence and abundance of them certain. In particular, using Proposition 11 the existence of such a word γ_n is ensured. Let the characteristic string of $A^{=n}$ be $\gamma_n 10^{2^n - |\gamma_n| - 1}$.

It is easy to prove that A is in D/G, and moreover that A is not in REC/F. Indeed, if $\exists r \in F$ and $\exists B \in \mathrm{REC}$, $B = L(M_i)$ such that $\forall n\ \exists w_n, |w_n| \leq r(n)$, for which $x \in A \iff \langle x, w_n \rangle \in B$, then there exists a program that constructs γ_n from the information: i, w_n and n. Then $\mathrm{K}(\gamma_n|n) \leq |w_n| + c \leq r(n) + c < s(n)$. This is a contradiction. □

Therefore $\mathrm{P}/\log$ and $\mathrm{P}/O(\log n * \log(\log n))$ are different, and thus small circuits with easy descriptions do not provide a characterization of $\mathrm{P}/\log$.

One way to avoid the problem raised by the Shannon-Lupanov effect is using Turing programs instead of circuits. We obtain the following characterization of $\mathrm{P}/\log$ in terms of resource-bounded Kolmogorov Complexity:

Theorem 21. The following are equivalent:

i/ $A \in \mathrm{P}/\log$.

ii/ There exists a family $\{C_n\}$ of circuits of polynomial size that accepts A, such that $\{C_n\} \in \mathrm{KS}[\log, \log]$.

iii/ There exists a family $\{C_n\}$ of circuits of polynomial size that accepts A, such that $\{C_n\} \in \mathrm{K}[\log, \mathrm{poly}]$.

Proof:

i/ $\Rightarrow$ ii/:

Let $A \in \mathrm{P}/\log$. Given n, the polynomial size circuit C_n that recognizes $A^{=n}$ is built as in theorem 18 .

We use again Savage's result that allows us to construct, for f computed by a DTM in $T(n)$ time, a circuit of size $O(T(n)^2)$. It is easy to check that this circuit can be generated with a DTM in $O(T(n)^2)$ time and $O(\log(T(n)))$ space (see again [1]).

As C_n is built composing two circuits of that type for polynomial time DTM, it can be generated by a DTM in $O(p(n)^2)$ time and $O(\log p(n)) = O(\log n)$ space, for a polynomial p, from seeds n and w_n. Thus $\{C_n\} \in \mathrm{KS}[\log, \log]$.

ii/ $\Rightarrow$ iii/:

is straightforward.

iii/ $\Rightarrow$ i/:

Let $\{C_n\}$ be the polynomial size family that accepts A. Since $\{C_n\} \in \mathrm{K}[\log, \mathrm{poly}]$, then for all n $\exists w_n$, $|w_n| \leq c \log n$, such that $U(w_n) = C_n$ in time bounded by n^k. We define $B = \{\langle x, y\rangle / \langle x, z\rangle \in \mathrm{CVP}$, where $z = U(y)$ in time $\leq |z|^k\}$.

As CVP is in P it is clear that B is in P. Finally, $x \in A$ iff $\langle x, w_{|x|}\rangle \in B$. $\square$

Further results

By the same arguments, we can generalize these results to characterize all advice classes of the form P/F, for F between $O(1)$ and poly.

For example it holds:

- $P/O(1)$ is exactly the class of sets having a family of polynomial size (p.s.) circuits $\{C_n\}$ in $\mathrm{K}[O(1), \mathrm{poly}|n]$, and in $\mathrm{KS}[O(1), \log|n]$.
- $P/O(\log^{(m)} n)$ is exactly the class of sets having a family of p.s. circuits $\{C_n\}$ in $\mathrm{K}[O(\log^{(m)} n), \mathrm{poly}|n]$, and in $\mathrm{KS}[O(\log^{(m)} n),\ log|n]$.
- $\mathrm{P}/\log$ is exactly the class of sets having a family of p.s. circuits $\{C_n\}$ in $\mathrm{K}[\log, \mathrm{poly}]$, and in $\mathrm{KS}[\log, \log]$.
- $\mathrm{P}/O(\log * \log^{(m)} n)$ is exactly the class of sets having a family of p.s. circuits $\{C_n\}$ in $\mathrm{K}[O(\log * \log^{(m)} n),\ \mathrm{poly}]$, and in $\mathrm{KS}[O(\log * \log^{(m)} n),\ log]$.

- $\mathrm{P}/O(\log * \log(\log n))$ is exactly the class of sets having a family of p.s. circuits $\{C_n\}$ in $\mathrm{K}[O(\log * \log(\log n)),\ \mathrm{poly}]$, in $\mathrm{KS}[O(\log * \log(\log n)), \log]$, and is also exactly the class of sets having a family of p.s. circuits $\{C_n\}$ with circuit descriptions $\{D_n\}$ with size bounded by $O(\log n)$.
- $\mathrm{P}/O((\log n)^k)$ is exactly the class of sets having a family of p.s. circuits $\{C_n\}$ in $\mathrm{K}[O((\log n)^k), \mathrm{poly}]$, in $\mathrm{KS}[O((\log n)^k), \log]$, and is also exactly the class of sets having a family of p.s. circuits $\{C_n\}$ with circuit descriptions $\{D_n\}$ with size bounded by $O((\log n)^k / \log(\log n))$.
- P/poly is exactly the class of sets having p.s. circuits.

Notice that for sublogarithmic bounds the Kolmogorov Complexity is necessarily taken relative to n.

In fact, we can be certain that the classes $\mathrm{P}/O(1)$, $\mathrm{P}/O(\log^{(m)} n)$, $\mathrm{P}/O(\log * \log^{(m)} n)$, $\mathrm{P}/O(\log *(\log(\log n)))$, $\mathrm{P}/O((\log n)^k)$, and P/poly are all different, where we are using the notation $\log^{(m)} n$ to denote the m-fold application of log.

Acknowledgements

We wish to thank José Luis Balcázar for many helpful ideas and discussions on this subject. We also thank Ricard Gavaldà for proofreading the initial version.

References

[1] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I*, EATCS Monographs on Theoretical Computer Science, Vol. 11, Springer-Verlag, 1988.

[2] J.L.Balcázar, U.Schöning, *Logarithmic advice classes*, Theoretical Computer Science 99, 1992, pp.279–290.

[3] J. Hartmanis, *Generalized Kolmogorov Complexity and the Structure of Feasible Computations*, Proc. 24th IEEE Symposium on Foundations of Computer Science, 1983, pp.439–445.

[4] R.M.Karp and R.J.Lipton, *Some Connections Between Nonuniform and Uniform Complexity Classes*, Proc. 12th Annual Symposium on Theory of Computing, 1980, pp.302–309.

[5] R.M.Karp and R.J.Lipton, *Turing Machines That Take Advice*, L'Enseignement Mathematique (2), vol. 28, 1982, pp. 191–209.

[6] R.Ladner, *The circuit value problem is log space complete for P*, SIGACT News 7, 1975, pp.18–20.

[7] J.E.Savage, *Computational work and time of finite machines*, Journal ACM 19, 1972, pp.660–674.

[8] I.Wegener, *The Complexity of Boolean Functions* , Wiley-Teubner Series in Computer Science, 1987.