Reports of the Academy of Sciences of the USSR

1969

SovietRxiv

Abstract

Full Text

Reports of the Academy of Sciences of the USSR
1969. Volume 189, No. 4

UDC 519.95

CYBERNETICS AND CONTROL THEORY

M. I. DEKHTYAR

ON THE IMPOSSIBILITY OF ELIMINATING EXHAUSTIVE SEARCH IN COMPUTING FUNCTIONS RELATIVE TO THEIR GRAPHS

(Presented by Academician P. S. Novikov on 21 IV 1969)

In the works of S. V. Yablonskii ([1]) and Yu. I. Zhuravlev ([2]), certain special problems are considered in whose solution it is presumably impossible to do without complicated algorithms of the "search through all variants" type. Justification of such propositions is carried out by them in terms of "proper algorithms" ([1]) and "local algorithms of finite rank" ([2]). In ([3]) B. A. Trakhtenbrot considered another approach to the matter, based on estimating the complexity of computations by means of signaling functions, and in these terms analyzed some aspects of S. V. Yablonskii' s problem. The same approach is used in the present paper to establish the non-eliminability of search in solving another model problem, namely: the problem of computing a function relative to its graph. Its preliminary formulation (for exact definitions see §§ 1°, 2°) is as follows. Suppose that A has conceived some function (possibly non-effective), whose values B tries to guess; in doing so he is allowed to ask questions of the form "$f(y) = z$?" , to which A is obliged to give correct answers ("yes" or "no"). The "exhaustive search" strategy for B consists in the fact that, for each x, he successively asks "$f(x) = 0$?" , "$f(x) = 1$?" , and so on, until a positive answer is obtained. In what cases and in what sense is a better strategy impossible (i.e., search is non-eliminable)?

It is clear that it is unreasonable to estimate the complexity of a strategy only by the number of questions asked, since this does not take into account the complexity of the questions themselves (one could, for example, for an effective function f, always restrict oneself to a single question "$f(x) = y$?" , where y is the value of the function at the point x, computed in advance).

In §§ 1°, 2° exact definitions are given of the concepts "guessing strategy," "exhaustive search," and their complexity in terms of computations on Turing machines with an "oracle," and of computational complexity. In §§ 3°, 4° two

situations are considered (Theorems 1, 2) confirming B. A. Trakhtenbrot' s hypothesis on the non-eliminability of exhaustive search.

1°. To compute functions we shall use special Turing machines. The external alphabet of these machines A is conveniently regarded as the product of two alphabets: $A = A_1 \times A_2$, where $A_1 = \{\Lambda, 1, 0\}$ is the alphabet of the "lower story" of the tape, and some finite alphabet A_2 is the alphabet of the "upper story" of the tape. Each machine has a finite set of states $Q = \{q_0, q_1, \dots, q_s\}$, in which a subset $\widetilde{Q} = \{q_{i_1}, q_{i_2}, \dots, q_{i_r}\}$ $(r < s)$ of question states is distinguished ($\widetilde{Q}$ may be empty). For each question state $q_\alpha \in \widetilde{Q}$, the program of the machine contains special commands of the form

$$q_\alpha a \to q_i,\ q_j \qquad (a \in A;\ q_i,\ q_j \in Q). \tag{1}$$

When computing a function f on some machine $\mathfrak{M}$, an "f-oracle" is attached to the machine; it "knows" the graph of the function and, for any pair

numbers (z, y) "is able" to answer the question: is the value of the function f at the point y equal to the number z ("$f(y) = z$?"). The operation of the machine $\mathfrak{M}$ proceeds as follows: at the initial moment the argument x is written on the lower level of the tape (in the binary system); the head of the machine is in the cell containing the least significant digit of x, in the state q_1. Then the machine works according to its program until it first enters some question state q_α. The execution of a command of type (1) depends on the f-oracle. Suppose that by this moment the inscription on the lower level of the tape has the form

$$N \wedge z \wedge y \wedge M, \tag{2}$$

where N and M are arbitrary words in the alphabet A_1; z and y are binary numbers, and the head of the machine scans one of the cells containing the inscription z. Such a situation is interpreted by the f-oracle as the question "$f(y) = z$?" . The answer to the question of the machine is given in the following form: a) if $f(y) = z$, then the f-oracle puts the machine into the state q_i; b) if $f(y) \neq z$, then the machine is put into the state q_j. If, at the moment of execution of command (1), the inscription on the tape does not have the form (2), then the machine remains in the state q_α and thereafter performs no actions.

Having received the answer of the f-oracle to the first question, the machine $\mathfrak{M}$ continues to work according to its program. In the course of its work it may again address the f-oracle with questions of the form "$f(y) = z$?" and receive an answer in the form indicated above.

Let $f(x)$ be an arbitrary function. Denote by D_f its domain of definition.

Definition. The machine $\mathfrak{M}$ computes the function $f(x)$ relative to the graph of the function $f(x)$ if, having begun its work with the f-oracle in the initial configuration, it: 1) for every $x \in D_f$, after a finite number of steps arrives

at the final state q_ω, and the inscription on the lower level of the tape at this moment has the form

$$N \wedge z \wedge M, \tag{3}$$

where N, M are arbitrary words in the alphabet A_1; z is a binary number, with $z = f(x)$, and the head of the machine is in one of the cells containing the inscription z; 2) for every $x \notin D_f$, either it never stops, or the final configuration does not have the form (3).

We shall denote by $t_{\mathfrak{M}f}(x)$ the number of steps of the operation of $\mathfrak{M}$ in computing f at the point x. If, in the course of working with the f-oracle at the point x, the machine never enters the final configuration (3), where $z = f(x)$, then by definition we set $t_{\mathfrak{M}f}(x) = \infty$.

In what follows we shall say that the machine $\mathfrak{M}$ computes the function $f(x)$ if $\mathfrak{M}$ computes $f(x)$ relative to the graph of $f(x)$.

2°. There exist many machines implementing complete enumeration. In this paragraph we shall fix one of them, $\mathfrak{M}_0$, which is the "fastest" (see Lemma 2) among such machines. The alphabet A_2 of the upper level of the tape of $\mathfrak{M}_0$ consists of the single symbol $*$. Having begun work with the argument x on the tape, the head of $\mathfrak{M}_0$ passes over the inscription x from right to left, leaves one cell empty, and in the next one prints 0 and enters a question state. The question put to the f-oracle is: "$f(x) = 0$?" . If the answer is negative, 0 is replaced by 1 and the question "$f(x) = 1$?" is asked, and so on. The symbol $*$ on the upper level marks the cell with the least significant digit of the question. For any $n > 0$, the question "$f(x) = n$?" is asked by the machine $\mathfrak{M}_0$ when its head is in the cell containing the rightmost one in the binary representation of the number n. If n is odd, then this will be the marked cell. Then, in order to pass to the question "$f(x) = (n + 1)$?" , the head of $\mathfrak{M}_0$ moves from right to left, replacing all ones by zeros until it first reaches a cell not containing 1. In this cell a 1 is written, and the machine enters a question state. The question "$f(x) = (n + 1)$?" is asked.

If n is even, then the cell in which the head of $\mathfrak{M}_0$ is located at the moment of the query "$f(x) = n$?" does not contain $*$. Then, in order to write the next query, the head moves to the right to the marked cell, replaces the 0 in it by 1, and asks the query "$f(x) = (n+1)$?" . The machine $\mathfrak{M}_0$ stops when it receives a positive answer from the f-oracle to its query. It is easy to estimate the running time of the machine $\mathfrak{M}_0$.

Lemma 1. Let f be an arbitrary function defined at the point x. Then

$$t_{\mathfrak{M}_0 f}(x) < 8[f(x) + 1] + \log_2 x.$$

Let f be an arbitrary function, and $\mathfrak{M}$ one of the machines. Suppose that, in computing f at some point x, the machine $\mathfrak{M}$ carries out exhaustive search, i.e.,

it asks successively the queries "$f(x) = 0?$," "$f(x) = 1?$," etc. The following lemma shows that $\mathfrak{M}$ carries out the search no faster than the machine $\mathfrak{M}_0$.

Lemma 2. One of the following two assertions holds:

1) the machine $\mathfrak{M}$, in computing f at the point x, asks all its queries in the same time as the machine $\mathfrak{M}_0$ in computing $f(x)$;

2) there exists a number n such that the query "$f(x) = n?$" is asked by the machine $\mathfrak{M}_0$ earlier than by the machine $\mathfrak{M}$, while all the queries "$f(x) = i?$" $(i = 0, 1, \dots, n-1)$ are asked by the machines $\mathfrak{M}_0$ and $\mathfrak{M}$ simultaneously.

Generally speaking, there are many machines whose work at all or almost all points coincides with the work of the machine $\mathfrak{M}_0$. Naturally, such machines should not be distinguished from $\mathfrak{M}_0$. We shall call a machine $\mathfrak{N}$ equivalent to the machine $\mathfrak{M}_0$ if, at all points, with the exception, perhaps, of a finite number, $\mathfrak{N}$ carries out exhaustive search in the same time as the machine $\mathfrak{M}_0$ $(\mathfrak{N} \sim \mathfrak{M}_0)$.

3°. For sufficiently broad classes of functions there exist machines computing them at all points faster than the machine $\mathfrak{M}_0$. Such are, for example, all monotone functions, functions with a "rare," easily enumerable set of values, and functions computable on ordinary Turing machines (without f-oracles) with temporal signaling $t_f(x) < f(x)$. The last class includes, in particular, all polynomials. The following assertion shows that there are also functions that cannot be computed at all points faster than exhaustive search.

Theorem 1. There exists a general recursive function (g.r.f.) $f(x)$ such that, for any machine $\mathfrak{N}$, $\mathfrak{N} \nsim \mathfrak{M}_0$, the inequality

$$t_{\mathfrak{N}f}(x) > t_{\mathfrak{M}_0 f}(x)$$

holds for infinitely many points x.

The proof of this theorem is carried out by diagonal methods and uses Lemma 2.

Remark 1. From any g.r.f. $g(x)$, by changing it on a "rare" set, one can obtain a function $f(x)$ satisfying the assertion of Theorem 1. Denote by $\lambda f g(y)$ the number of those values of the argument x, $x \leq y$, for which $f(x)$ differs from $g(x)$. Let $\varepsilon(y)$ be an arbitrary nondecreasing, unbounded g.r.f. Then, for the given g.r.f. $g(x)$, there exists a g.r.f. $f(x)$ satisfying Theorem 1 and such that

$$\lambda f g(y) \leq \varepsilon(y)$$

for all y.

Remark 2. Theorem 1 cannot be strengthened, since for every function $f(x)$ there exists a machine $\mathfrak{M}_f$ which, at infinitely many points x, computes $f(x)$ faster than the machine $\mathfrak{M}_0$. Of course, if $f(x)$ satisfies Theorem 1, then on another infinite set of points the machine $\mathfrak{M}_0$ computes $f(x)$ faster than the machine $\mathfrak{M}_f$.

4°. In this section we shall consider the time of computation of functions up to constant factors (in order). It follows from Lemma 1 that, for computing sufficiently "large" functions f, exhaustive search spends, in order, $f(x)$ steps. There exist functions for which this estimate is best possible.

Theorem 2. For an arbitrary g.r.f. $g(x)$ there exists a g.r.f. $f(x)$ such that:

1) $\forall x[f(x) > g(x)]$;

2) every machine $\mathfrak{N}$ computes $f(x)$ no faster (in order) than the machine $\mathfrak{M}_0$, i.e.,
$$\forall \mathfrak{N}\ \exists c(\mathfrak{N})\ \forall x\,[\,t_{\mathfrak{N}f}(x) \geq c(\mathfrak{N}) \cdot f(x)\,],$$
where $c(\mathfrak{N})$ is a constant depending only on the machine $\mathfrak{N}$.

The assertion of Theorem 2 can be refined if one takes into consideration the number of questions asked by the machine $\mathfrak{N}$ when computing the function f. There exist functions such that, if in computing them $\mathfrak{N}$ asks, in order, fewer questions than the machine $\mathfrak{M}_0$, then the time spent by the machine $\mathfrak{N}$ will be significantly greater than the time of the machine $\mathfrak{M}_0$.

Theorem 2′. Let $h(x, y)$ be an arbitrary general recursive function, nondecreasing in y. Then for any general recursive function $g(x)$ there exists a general recursive function $f(x)$, $f(x) > g(x)$ at all points, with the following property: for every machine $\mathfrak{N}$ there exists a constant $c(\mathfrak{N})$ such that, at an arbitrary point x, one of the following two assertions holds:

1) in computing f at the point x, the machine $\mathfrak{N}$ asks the f-oracle at least $c(\mathfrak{N}) \cdot f(x)$ questions;

2) $t_{\mathfrak{N}f}(x) \geq h(x, f(x))$.

It is clear that, for $h(x, y) \geq y$, the functions satisfying Theorem 2′ also satisfy Theorem 2.

In conclusion the author thanks B. A. Trakhtenbrot for posing the problem and for useful remarks.

Novosibirsk
State University

Received
17 IV 1969

REFERENCES

1. S. V. Yablonskii, in *Problems of Cybernetics*, 2, 75 (1959).

2. Yu. I. Zhuravlev, ibid., 8, 5 (1962).

3. B. A. Trakhtenbrot, *Algebra and Logic*, Seminar, 4, 5, 79 (1965).

Note: Figure translations are in progress. See original paper for figures.

Source: Math-Net.Ru and CyberLeninka. Machine translation. Verify with the original.