A Survey of Russian Approaches to *Perebor* (Brute-Force Search) Algorithms

B. A. TRAKHTENBROT

Concerns about computational problems requiring brute-force or exhaustive search methods have gained particular attention in recent years because of the widespread research on the "P = NP?" question. The Russian word for "brute-force search" is "perebor." It has been an active research area in the Soviet Union for several decades. Disputes about approaches to perebor had a certain influence on the development, and developers, of complexity theory in the Soviet Union. This paper is a personal account of some events, ideas, and academic controversies that surrounded this topic and to which the author was a witness and—to some extent—a participant. It covers a period that started in the 1950s and culminated with the discovery and investigation of nondeterministic polynomial (NP)-complete problems independently by S. Cook and R. Karp in the United States and L. Levin in the Soviet Union.

*Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]—graph and tree search strategies; K.2 [**History of Computing**]—people, software*
General Terms: Algorithms, Theory, Verification
Additional Key Words and Phrases: brute-force search algorithms, perebor

Introduction

A *perebor* algorithm, or *perebor* for short, is Russian for what is called in English a "brute-force" or "exhaustive" search method. Other combinations of words also occur in translations from Russian, such as "successive trials," "sequential searching," and "thorough searching." To keep the historical flavor, I prefer to preserve in this paper the original term *perebor* and use such expressions as "*perebor* problems" (problems that are solvable by *perebor*), "impossibility of eliminating *perebor*," etc.

Example. Consider SAT, the satisfiability problem for formulas of the propositional logic. (1) *Existential version:* Given an arbitrary formula $A(X_1, \ldots, X_n)$, determine whether there exists an n-tuple of truth values that satisfies A. (2) *Constructive version:* In the case of an affirmative answer to (1), an n-tuple should be produced.

The obvious *perebor* algorithm that solves both the existential and constructive versions of the problem considers all the n-tuples of truth values in some order (say, lexicographical order). The first time an n-tuple that satisfies A is discovered, the algorithm stops and delivers the right result; of course, this could happen in an early stage of the process. Otherwise, after unsuccessfully considering all the 2^n possible n-tuples, the negative answer is produced. Clearly, a high price is to be paid, in general, for the conceptual simplicity—even triviality—of the *perebor* algorithm, because the number of cases to be tried grows exponentially with respect to the number n of propositional variables. There are many other natural problems, mainly combinatorial and logical ones, for which *perebor* algorithms are quite evident. Often, as for SAT, the problem is to *decide* if some property under consideration holds for arbitrary inputs of the suitable type. For example, in the Hamiltonian circuit problem, the input is graph G, and the property to be checked is "does G have a simple cycle that goes through all vertices?" Similarly, for a given family of games, the problem arises of determining whether an arbitrary

Author's Address: School of Mathematical Sciences, Tel Aviv University, Ramat Aviv, 69 978 Tel Aviv, Israel.
© 1984 AFIPS 0164-1239/84/040384–400$01.00/00

game has a winning strategy. In such cases, both the existential and constructive versions are clearly meaningful. Formally, the first one is the decidability problem for some set (i.e., the task of computing a predicate—the characteristic function of this set), whereas the second deals with the computation of a function.

It may happen that the problem solvable by *perebor* looks somewhat different. Nevertheless, it is often (but not always) clear how to reformulate it (in some sense equivalently) to the preceding standard. Consider, for example, the following two problems: (1) given a graph G, compute its chromatic number (i.e., the minimum number of colors needed to paint each node of the graph so that no two adjacent nodes have the same color); (2) given a finite automaton M, minimize it (i.e., construct an automaton M' that is equivalent to M and has the minimal number of states). Here there is no explicit decidability problem; instead, the computation of a total function (mapping) is required. Again, the solution via *perebor* is obvious—for example, for problem 1, check all of the possible colorings with two colors, then with three colors, etc.

Of course, for some combinatorial problems of the kinds described (e.g., for the minimization of finite automata), more sophisticated algorithms have been discovered that are considerably more efficient than the trivial *perebor*. That is not the case for the other problems formulated here. Moreover, the experience accumulated in this area indicates that one cannot expect more efficient algorithms for such problems, because in some sense *perebor* must occur (perhaps implicitly) in any algorithm that solves them. This is the conjecture on "the impossibility of eliminating *perebor*." Clearly, to attack it one needs a suitable formalization of the intuitive ideas under consideration. Since 1971–1972 there has been a broad consensus in the computer science community about such a formalization based on the idea of computability and reducibility *in polynomial time*. Let us recall the main points about the seminal work of S. A. Cook (1971), R. M. Karp (1972), and L. Levin (1973).

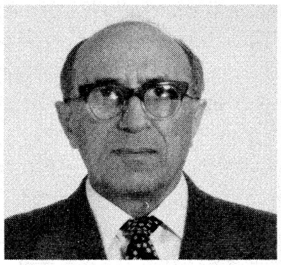
Let $x,y, \ldots$ denote binary strings, $\ell(x)$, $\ell(y)$, $\ldots$ their lengths, and $A(x,y)$ a predicate that is computable in polynomial time with respect to $\ell(x) + \ell(y)$. Note that the existence of a polynomial upper bound just stipulated does not depend essentially on what computing model is assumed: Turing machines with many tapes and heads, random-access machines, etc. Note also that formalization of the intuitive notion "feasible computation" is widely accepted in terms of polynomial time. Hence, the real intention is to consider feasibly computable predicates $A(x,y)$. From the vague class of *perebor* problems one selects the following well-defined subclass of *perebor* problems, called nondeterministic polynomial (NP) problems.

Given $A(x,y)$ as above and a polynomial r,

1. *Existential version.* For arbitrary x determine whether there exists a y such that $A(x,y)$ is true and $\ell(y) \leq r(\ell(x))$.
2. *Constructive version.* In the case of an affirmative answer, produce such a y.

It is immediately clear that SAT turns out to be an NP problem up to some natural encoding of the formulas; moreover, one can manage with the polynomial $r(n) \equiv n$. It is also clear that for each NP problem a solution is available via *perebor*, as it was for SAT. Namely, for a given x_0, one has to check $A(x_0,y)$ for all y of length $\leq r(\ell(x_0))$: altogether, about $2^{r(\ell(x_0))}$ trials—a number that is exponential relative to $\ell(x_0)$—each of which needs only a feasible (polynomial on $\ell(x_0)$) computation of A.

The notation NP indicates that an affirmative answer in the existential version, as well as the value required in the constructive version, can be certified by a nondeterministic procedure in polynomial time as follows: first, guess a correct y and then check in polynomial time that $A(x,y)$ is true. But note that nothing is said about the complexity of getting a negative answer! Now, as suggested by Cook and Levin, "eliminate *perebor* for an NP problem" is to be interpreted as "find for this problem a deterministic algorithm that works in polynomial time (or, in short, a P algorithm)." So the "NP = P?" question arises: "Does there exist for each NP problem a P algorithm?" The conjectured answer is "No"; in solving the problems under consideration, guessing cannot be systematically eliminated without some essential exhausting of all possible guesses. This conjecture remains open, but Cook and Levin discovered the NP-*complete* problems—the most plausible candidates for which no P



Boris A. Trakhtenbrot was born in Brichevo, Moldavia (then Romania, now U.S.S.R.) in 1921. He received his Ph.D. in mathematical logic and the theory of algorithms under P. S. Novikov in 1950 at the Institute of Mathematics in Kiev. Until 1960 he was in teaching and research on computability and automata theory at the Pedagogical and Polytechnical Institutes of Penza. He spent 1960–1980 at the Mathematical Institute of the Siberian Branch of the U.S.S.R. Academy of Sciences and in Novosibirsk University. He emigrated to Israel in 1980 and is professor of computer science at Tel Aviv University.

algorithm exists. By definition, an NP problem Q is *complete* if and only if (iff) for each other NP problem R there exists a mapping of binary strings f such that

1. f is computable in polynomial time.
2. For each x, the answer for x is affirmative in R iff the answer for $f(x)$ is affirmative in Q.

Hence, if there exists in general some NP problem (even if artificially formulated) that is not solvable by P algorithms, this is the case for the NP-complete problems as well. By the way, SAT and a lot of other classical combinatorial problems are NP complete. So if *perebor* is inevitable in solving any particular problem in NP, it is surely inevitable for SAT. Because the "NP = P?" question has so far resisted all attacks, a new trend of reseach has appeared dealing with the challenging question (Hartmanis and Hopcroft 1976): "Is 'NP = P' in general provable or refutable in the frame of reasonable formal theories?"

The whole story belongs—if one may say so—to the post-NP period that was started by the discovery of Cook and Levin (and who knows when it will be over!). As to this paper, it is mainly a survey of the pre-NP period in the Soviet Union, in particular of the events to which I was a witness and to some extent a participant. This period may be divided into two parts.

1. The first part was from the early 1950s until the 1960s. At that time a large spectrum of research was started; the development included switching theory, minimization of boolean functions, automata, program schemes, coding, etc. Nowadays, these activities would be classified as "theoretical computer science," but at that time we used the more general and vague rubric "theoretical cybernetics." Complexity theory was essentially investigated in connection with switching circuits—and so was the *perebor* topic.
2. The second part began in the 1960s, when there was intensive development of algorithmic complexity—that is, of computational complexity and complexity of finite objects by means of the theory of algorithms. Accordingly, some new approaches to *perebor* appeared.

In his *Annals* paper describing related research at the same time in the United States, Hartmanis (1981, pp. 44, 46) wrote:

> My thinking was very strongly influenced by Claude Shannon's work . . . [which] suggested to me that there may be a quantitative theory of computing. . . . We must be able to measure the "computing work" done in computing and classify computations by their complexity.

That is just what was happening with my colleagues and myself! Hartmanis goes on.

> During the late 1950s my colleagues . . . and I knew very little about effective computability and Turing machines. . . . We started reading about [them] in 1962. . . . The beauty and simplicity of the concept impressed us deeply.

For many people in the Soviet Union (including myself), the situation was different. We started solely with a background in mathematical logic and the theory of algorithms before joining computer science. Research in the theory of algorithms began soon after World War II almost simultaneously under P. S. Novikov (Steklov Mathematical Institute in Moscow), A. N. Kolmogorov (Moscow University), and A. A. Markov (Leningrad University). The first generation of "theoretical cyberneticians" was educated in these traditions and was considerably influenced by them. Therefore the interest in switching theory, automata, boolean functions, etc., never did mean a break with effective computability topics; the 1960s marked in some sense a return to the theory of algorithms.

Even a superficial examination of the titles in the References shows that in each of the two pre-NP periods the impossibility of eliminating *perebor* was explicitly proclaimed as a proved fact—the first time by Yablonski (1959*a*) and the second by Dekhtiar (1969). Less explicit claims of this sort could be found in other papers as well. It is clear that at that time there were accepted formalizations of *perebor* problems and of the *perebor* conjecture that were different from the P = NP problem. As a matter of fact, in both of the pre-NP periods not only was *perebor* an active topic of research, but it also influenced the development and developers of complexity theory. Some ideas, results, and controversies of these periods will be surveyed in the following sections.

In many respects, research in the field of complexity theory was performed independently in the U.S.S.R. and in the West, and sometimes the results were similar. Even in these cases, however, the primary motivations and stimuli were sometimes different, especially for *perebor*. Hence there was some divergence in the chronology of events and in their influence. For example, during almost all of the pre-NP period, some versions of a special task related to minimization of circuits (see Tasks 4 and 5 in Sections 1 and 2) were considered as the main models for which presumably it would be impossible to manage without *perebor*. In contrast to SAT, they were not proved to be NP complete, and most likely they are not! In this sense, to persist with them might have looked like a false strategy. On the other hand, these tasks helped to define the developing area of complexity theory, and they especially called attention to problems concerning the role of sparse sets, oracles, immunity, frequency algorithms, probabilistic algorithms, etc. (Trakhtenbrot 1973*a*; 1975).

Section 1 of this paper deals with the 1950s, especially with the first attempt by Yablonski to formalize and to prove the *perebor* conjecture and with the controversies stirred by this attempt.

Section 2 is a superficial survey of the investigations in algorithmic complexity in the U.S.S.R. that created a background for new approaches to *perebor* or were influenced by it.

Section 3 deals with some attempts to formalize the *perebor* phenomena in the framework of the theory of algorithmic complexity that preceded Levin's discovery and to some extent promoted it.

Of course, investigations on *perebor* do not necessarily need complexity theory, just as elaboration and analysis of algorithms may not involve the theory of algorithms. In both cases, the conceptual framework of the theory is relevant when negative results are expected—for example, the inevitability of *perebor*, the nonexistence of an algorithm, etc. On the other hand, when the aim is to obtain positive results, one can manage with direct constructions and ingenious tricks, for which the preceding conceptual framework is irrelevant. Although a review of activities of this sort in the U.S.S.R. is beyond the scope of this paper, I will make a few comments on related work.

Tasks arising from game-playing programs and linear programming were always a source of inspiration as to how one might compete with *perebor*. The book by Adelson-Velski et al. (1976) is a nice account of the investigations on this topic and reflects the essential contributions of the authors. Their main philosophy is formulated as follows: "One does not fear the *perebor* but rather uses it reasonably via a realistic estimation of the dimensions of the disaster it may imply." In the frame of this general approach, effective methods (including heuristics) of cutting down exhaustive search through all a priori possibilities were created. In general, these methods do not always give evidence of correct results within feasible (say, polynomial time) computations; nevertheless, they are useful and interesting in both theoretical and practical aspects.

Because of its practical importance, linear programming was investigated for a long time by representatives of many generations, who contributed to the elaboration and improvement of the respective algorithms. The main task happens to be an NP problem: Given a system of linear inequalities with integral coefficients, (1) *Existential version:* Decide whether it is solvable; (2) *Constructive version:* If it is solvable, exhibit a solution. In Karp (1972) this task is listed among the most important NP problems that are not known to be complete.

Khachijan (1979), relying on the contributions of his predecessors (especially A. Nemirovski and B. Yudin), formulated explicitly a polynomial-time algorithm for the linear programming task. Thus it turns out that in one of the most famous *perebor* problems, one can nevertheless avoid *perebor*.

Section 1. The Cynbernetic Period

From the very beginning of the 1950s, activities in theoretical cybernetics were energetically promoted in the U.S.S.R. by A. A. Liapunov and S. V. Yablonski. For about two decades earlier, Liapunov had closely collaborated with P. S. Novikov in the area of set theory, but now he was mainly attracted by the theory of programming, formal linguistics, and the mathematics of biology. Yablonski finished his Ph.D. dissertation under Novikov on the topic of completeness criteria for boolean and many-valued logics. He became interested in such topics as representation of boolean functions by switching circuits, minimization of disjunctive normal forms, coding, automata, etc. The seminars of Liapunov and Yablonski at Moscow University attracted many students and scholars and soon became important centers of research in these new and exciting areas. At that time, after completing my Ph.D. under Novikov, I held a position in the Pedagogical Institute of Penza, a provincial city east of Moscow, and I was happy to join the cybernetics community through correspondence and visits to Moscow. The general atmosphere within this fresh and energetic community was very friendly, and I benefited much from it.

Clearly, minimization of circuits and disjunctive normal forms, choice of optimal codes, etc., become trivial tasks as soon as *perebor* is allowed. It is therefore not surprising that the *perebor* subject came to light, especially in connection with Shannon's work on the complexity of switching circuits. I learned about this work in a detailed letter from Yablonski, who informed me about the seminar. This episode is char-

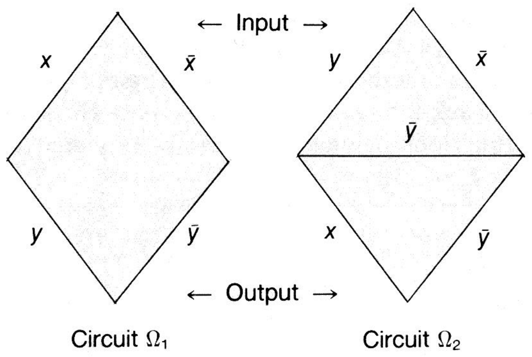


Figure 1.

acteristic of the kind and thoughtful relations within the cybernetics community at that time. Let me recall some notations and facts concerning this topic.

A switching circuit Ω is essentially an undirected graph labeled by boolean variables and their negations, with a designated input and a designated output vertex, as in Figure 1. The paths in Ω from its input vertex to the output vertex induce conjunctions of variables and/or their negations. Hence the whole circuit is associated with a boolean function—the disjunction of these conjunctions. This function is said to be realized by the circuit; for example, for the circuits Ω_1 and Ω_2 from Figure 1, the associated disjunctions of conjunctions are, respectively,

$$xy \vee \bar{x}\bar{y} \quad \text{and} \quad yx \vee y\bar{y}\bar{y} \vee \bar{x}\bar{y}x \vee \bar{x}\bar{y}$$

Therefore, they realize in fact the same function, which is tabulated in Figure 2. As usual, a boolean function of n arguments will be identified with a binary string of length 2^n—namely, the value column in its table (in this case, the string 1001). Additional notations are:

$\Phi(\Omega)$	The function associated with (realized by) the circuit Ω.
$L(\Omega)$	The complexity of circuit Ω (i.e., the number of edges in the graph).
$L(f)$	The complexity of the function f (i.e., min $\{L(\Omega) : \Phi(\Omega) = f\}$).
$P(n)$	The set of all boolean functions of n arguments.
$L(n)$	max $\{L(f) : f \in P(n)\}$.

The following tasks arise quite naturally.

Task 1. Given an arbitrary Boolean function $f(x_1, \ldots, x_n)$,

1. *First version.* Compute $L(f)$, the complexity of the function f.
2. *Second version.* Find a minimal circuit Ω that realizes f; that is, $\Phi(\Omega) = f$ and $L(\Omega) = L(f)$.

Task 2. Given an arbitrary natural n,

1. *First version.* Compute $L(n)$.
2. *Second version.* Find an $f \in P(n)$ such that $L(f) = L(n)$.

Clearly, Task 1 is solvable by *perebor* through the set of circuits ordered with respect to their complexity, and so is Task 2 by an additional *perebor* through all the 2^{2^n} functions belonging to $P(n)$. As a matter of fact, by means of some refined *perebor*, the values $L(2) = 4$, $L(3) = 8$, $L(4) = 13$ were computed, but $L(5)$ is still unknown. On the other hand, Shannon's theorem, later improved by Lupanov (1958), states:

$$L(n) \sim \frac{2^n}{n} \tag{1}$$

where "$\sim$" means asymptotic equivalence.

Since the function $2^n/n$ is easy to compute, no *perebor* is needed in the first version of Task 2 if the asymptotic approximation of $L(n)$ is allowed instead of its exact value. For the second version of Task 2, it is not clear how to manage without *perebor* even in the less strict case when an f is to be produced with $L(f)$ merely close to $L(n)$.

From 1953–1954 Yablonski emphasized the conjecture that even in this less strict case and in some other related tasks, the use of *perebor* is inevitable. Finally, he claimed the proof of the conjecture (Yablonski 1959*a*). Before we continue this topic, let us recall some details concerning Shannon's lower bound for $L(n)$.

Let $P_\varepsilon^-(n)$, $P_\varepsilon^+(n)$ partition $P(n)$ into "simple" and "complex" functions—specifically,

$$f \in P_\varepsilon^-(n) \Leftrightarrow L(f) < \frac{2^n}{n}(1 - \varepsilon)$$

and

$$f \in P_\varepsilon^+(n) \Leftrightarrow L(f) \geq \frac{2^n}{n}(1 - \varepsilon)$$

Let $|V|$ denote the cardinality of a set V.
Then, for each fixed $\varepsilon > 0$, as $n \to \infty$

$$\frac{|P_\varepsilon^-(n)|}{|P(n)|} \to 0 \quad \text{and} \quad \frac{|P_\varepsilon^+(n)|}{|P(n)|} \to 1 \tag{2}$$

Hence each $\varepsilon > 0$ induces a splitting P_ε^-, P_ε^+ of the set of all boolean functions, where $P_\varepsilon^- \underset{\text{def}}{=} \bigcup_n P_\varepsilon^-(n)$ is the "thin" set of "simple" functions, and $P_\varepsilon^+ \underset{\text{def}}{=} \bigcup_n P_\varepsilon^-(n)$ is the "thick" set of "complex" functions.

There is some gap between the technical results of Yablonski's 1959 papers (1959*a*; 1959*b*) and their interpretation concerning *perebor*. Since these circumstances stirred both interest and controversy, I shall try to reproduce the main points as accurately as possible; the interested reader is urged to consult the original texts if needed. Yablonski considered the following task.

Task 3. Construct a sequence M^0 of boolean functions

$$f_1^0(x_1), f_2^0(x_1,x_2), \ldots, f_n^0(x_1,x_2, \ldots, x_n), \ldots \tag{3}$$

such that for some subsequence

$$L(f_{n_k}^0)/L(n_k) \to 1 \tag{4}$$

x	y	f(x,y)
0	0	1
0	1	0
1	0	0
1	1	1

Figure 2. Table of the function realized by both Ω_1 and Ω_2.

holds.

Clearly, this is the same as to require that for each $\varepsilon > 0$

$$M^0 \cap P_\varepsilon^+ \tag{4'}$$

is infinite.

Note that Task 3 is a weaker form of Task 2 (second version), and therefore its solution by *perebor* is obvious. As for a solution of Task 3 without *perebor*, Yablonski considers first the case in which one allows algorithms with random steps. Suppose that for each n one constructs a function $f(x_1, \ldots, x_n)$ by throwing a regular coin to obtain a string of length 2^n, which is taken as the value column of the function. Then an accurate estimate of the ratio $|P_\varepsilon^-(n)|/|P(n)|$ in Shannon's theorem results in the following theorem.

Yablonski's Theorem. With probability 1, the sequence of boolean functions obtained by the probabilistic algorithm above satisfies the conditions of Task 3.

The interpretation is that probabilistic algorithms are able to solve Task 3 without *perebor*. For deterministic algorithms, Yablonski proposes that one need consider only algorithms satisfying some properties particularly connected to the Task 3 under consideration. These properties come from the argument that as soon as a circuit is constructed for some boolean function f, one can assume that implicit realizations are available for all functions obtainable from f by the following operations.

1. Inserting and/or deleting dummy arguments.
2. Permutation of arguments.
3. Substitution of constants for some of the arguments.

Definition 1. A class of boolean functions is *invariant* iff it is closed under operations 1–3.

According to this motivation, the reasonable restriction to be imposed on algorithms for Task 3 is reflected in:

Definition 2. An algorithm is *regular* iff it maps the set of natural numbers *onto* an invariant class of boolean functions. Then the main result is formulated as follows.

Main Theorem. Each regular algorithm that constructs a sequence M^0 (as required by Task 3) constructs, in fact, all the boolean functions; such an algorithm is in fact a perebor of all the boolean functions.

Since Task 3 requires exactly *one function of n arguments for each n*, the formulation above could seem inaccurately stated because each sequence that is closed under the operations of inserting and deleting dummy arguments will violate the italicized condition above. In fact, the assertion is about algorithms that construct the invariant closure of M^0; the point is:

$$\textit{the invariant closure of } M^0 \textit{ must contain all boolean functions} \tag{5}$$

The intuition behind Yablonski's interpretation of (5) is that any algorithm solving Task 3 must, in the process of generating M^0, examine all the functions in the invariant closure of M^0 and therefore is doing *perebor*.

Yablonski's result aroused mixed feelings and reactions. On the one hand, the impression was that some evidence (perhaps indirect) was given to the validity of the *perebor* conjecture. Moreover, since then many people (mainly his former students) were categorical in their opinion and made public declarations to the effect that Yablonski's work was genuine proof of the *perebor* conjecture with respect to the task under consideration. On the other hand, there was criticism, and after some confusion this was my attitude as well. *Even if we accept the hypothesis that an algorithm for Task 3 must in some sense "examine" all of the functions* in the invariant closure, there is still an objection to Yablonski's interpretation of (5): namely, *perebor* is an intuitive concept that seems clearly related to the difficulty of searching a large domain *in a short time*, and property (5) says nothing about the *rate* at which the boolean functions are "examined." Indeed, it is easy to construct sequences satisfying (5) that are generated very rapidly—there is a Turing machine that prints out (the value column of) $f_n(x_1, \ldots, x_n)$ (of length 2^n) for $n = 1, 2, \ldots$ at the rate of one bit per step, where

$$f_1(x_1),\ f_2(x_1, x_2), \ldots,\ f_n(x_1, x_n), \ldots \tag{6}$$

satisfies (5). So if property (5) is to be interpreted as implying *perebor*, we have found an easy way to do *perebor*—and this contradicts the essential idea that *perebor* is what is not easy to do.

The main theorem is in fact contained in the following technical result of the paper.

Let Q be an invariant class, Q(n) its subset consisting of n argument functions, and

$$L_Q(n) \overset{=}{\underset{\text{def}}{=}} \max L(f) \quad \textit{for} \quad f \in Q(n) \tag{7}$$

Then either Q is the set of all the boolean functions, or else

$$L_Q(n) \backsim \sigma \cdot \frac{2^n}{n} \textit{ for some constant } 0 \le \sigma < 1 \tag{8}$$

Note that in this formulation, *no algorithm is mentioned at all!*

The lack of direct connection between Yablonski's result (8) and the feasibility of algorithms for Task 3 looked somewhat strange, although the algebraic arguments (such as invariance) that led to interpretations concerning *perebor* were not contested. In this context the paper of Yablonski's student, Y. Zhuravlev (1960), is worth noting. Again, the title is about the impossibility of solving some tasks by algorithms of a certain class. Unfortunately, the definitions in this paper are cumbersome, and I cannot discuss them here. Although *perebor* is not emphasized in the title nor in the text, some comparisons with Yablonski (1959*a*; *b*) are suggested. First, the tasks under consideration concern minimization of disjunctive normal forms; their trivial solutions by *perebor* are obvious. Second, locality conditions imposed on the allowed algorithms restrict access to information stored on the nodes of a graph, so that a thorough scan over the nodes of the graph is needed to collect the information. Finally, it has been hinted that this forced scanning gives evidence that *perebor* is inevitable. Therefore Zhuravlev's approach could almost be given the same *perebor* status as Yablonski's.

Discussions focused on Yablonski's approach for two reasons.

1. The explicit and persistent claim that Yablonski had proved the *perebor* conjecture; as a matter of fact, that was mirrored in the title of the paper (Yablonski 1959*a*).
2. The tasks of circuit minimization seemed to be more fundamental than minimal normal forms. People were fascinated by some peculiarities of circuit size; indeed, for a long time these peculiarities inspired the research on *perebor* phenomena and related complexity problems.

Let me explain in more detail the second point with respect to modification of Tasks 1–3 that naturally came to light during the discussion. Suppose for a given ε, we consider as in Shannon's theorem the splitting into thin P_ε^- and thick P_ε^+.

Task 4. For a given f,

1. *Existential version:* Determine whether $f \in P_\varepsilon^-$.
2. *Constructive version:* If $f \in P_\varepsilon^-$, produce a suitable Ω with $L(\Omega) \leq (1 - \varepsilon)(2^n/n)$.

In other words, the existential version of Task 4 is the decision problem for P_ε^- (and in fact for P_ε^+ as well). As before, the trivial solution is by *perebor*, and no way is known for doing it more efficiently. There seemed to be an additional reason to pay special attention to this task: because of the following "frequential and immunity effect" of the splitting P_ε^-, P_ε^+. On the one hand, there exist infinite subsets M^ε of the thin P_ε^- that are decidable by efficient algorithms; for example, $M^\varepsilon \underset{\text{def}}{=}$

$$\{x_1, x_1 \wedge x_2, x_1 \wedge x_2 \wedge x_3, \ldots\}$$

is decidable in real time by a finite automaton. On the other hand, the decision problem for the thick P_ε^+ looks difficult (presumably because *perebor* is inevitable, and the intuitive feeling is that the same holds for each infinite subset of P_ε^+. Recall that in recursion theory an infinite set is called *immune* if it does not contain any infinite recursive subset. Hence, by analogy, the property of the thick set P_ε^+ not to contain any "easily decidable" infinite subset could be characterized by some sort of immunity. This contrast between the thin but *perebor*-free (in the sense of including easily decidable infinite subsets) P_ε^- and the thick but eventually immune P_ε^+ looked impressive and close to the essence of the problem. That is why Task 4 received special attention and for a long time was the main source of reflection on *perebor*. In particular, it stimulated the search for alternatives to *perebor* via probabilistic and frequential algorithms that could utilize the density properties of the splitting.

Section 2. Algorithmic Complexity

The events discussed in this and the next sections occurred in a decade that was remarkable in many aspects. In 1960 I moved to the Akademgorodok, the Academic Center near Novosibirsk, where, through the initiative and guidance of A. A. Liapunov, the Department of Theoretical Cybernetics was organized within the Mathematical Institute. Almost all the staff of the department were former participants in the Moscow seminars, mainly students of Liapunov and Yablonski. In general, the research areas were scheduled in close collaboration with the Moscow group. In particular, research focused on complexity theory and related problems that had previously undergone rapid and intensive development under the influence of Shannon's switching theory. Just at this time, a new approach to complexity problems, manifested as a fusion of combinatorial methods inherited from switching theory with the conceptual arsenal of the theory of algorithms, was also rapidly developing. In fact, two trends were developing: computational complexity and the complexity of algorithms.

The first subject deals with the amount of computational work that is expended in performing an algorithm. Typical measures of this complexity are *functions* such as time complexity $t_M(x)$ (the number of steps performed by an algorithm M on input x), and space complexity $s_M(x)$ (the size of memory the algorithm consumes when applied to input x). The second subject deals with the size of the algorithms (programs) themselves; as in circuit complexity (but unlike computational complexity), the size is measured by a *number*—for example, the number of symbols in the program. As these trends were developing, my belief was growing that they would yield a natural basis for the investigation of the *perebor* phenomena. Moreover, my interest in computational complexity and the choice of special research topics were to some extent influenced by reflections about *perebor*. Although I have neither the intention nor the possibility to go fully into details about the development of complexity theory in the U.S.S.R., this is the right opportunity to make some points concerning the subject.

First, investigations in computational complexity appeared in the U.S.S.R. as early as 1956—that is, earlier than in the West. G. S. Tseitin, then a 19-year-old student of A. A. Markov at Leningrad University, began to study the time complexity of Markov's normal algorithms and proved nontrivial lower and upper bounds for some concrete tasks. He also discovered the existence of arbitrarily complex 0–1-valued functions (Rabin's 1960 results became available in the U.S.S.R. in 1963). Unfortunately, these remarkable results were not published by Tseitin and appeared later without proofs in a survey (Yanovskaia 1959). Independently, I considered "signalizing functions"—a version of space complexity for computations of recursive functions (Trakhtenbrot 1956). Nevertheless, these episodes did not receive any serious continuation and development until the 1960s.

In May 1962, I met Y. M. Barzdin for the first time. After graduating from the Latvian University in Riga, he came to Novosibirsk as my postgraduate student in automata theory. He quickly became my friend and main partner in research on computational complexity, and soon other people joined us, mainly students of the Novosibirsk and Latvian universities. This is how research on computational complexity started in Novosibirsk; a new young generation arose, and I had the good fortune to work with these people over a long period. It is not surprising that we were attracted by the same problems as our colleagues in the West (of course, we were working independently and in parallel), and we worked out almost the same techniques: complexity measures, crossing sequences, diagonalization, gaps, etc. All of these ideas arose quite naturally; we became most excited, and they evoked in us enthusiasm similar to that described by Hartmanis in his historical account (1981).

On the other hand, we were upset by the deterioration in our relations with the "classical" cybernetics people, mainly Yablonski. Their attitude to the introduction of the theory of algorithms into complexity affairs was quite negative. The main argument they used was that the theory of algorithms is essentially a theory of diagonalization, and therefore is alien to the complexity area that requires combinatorial constructive solutions. Hence they distrusted the role that computational complexity and algorithm complexity could play in the *perebor* subject. These scientific divergences were likely intensified by the *perebor* controversy, especially because at that time Yablonski attained influential positions in the bodies that dealt with coordination and control of mathematical investigation.*

In the summer of 1963, during a visit by A. N. Kolmogorov to the Novosibirsk University, I learned in more detail about his new approach to the complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. In the early cybernetics period, it was already clear that the essence of the problems with the minimization of boolean functions did not depend essentially on the particular models of switching circuits that were used to compute them; any other natural class of "schemes," and ultimately any natural coding of finite objects (say, finite texts), should be expected to exhibit similar phenomena. Now, unlike the former pure combinatorial approaches, the discovery by Kolmogorov (1965), and independently by Solomonoff (1964) and Chaitin (1966), of optimal coding for finite objects happened in the framework of algorithm and recursive function theory. According to this theory, the complexity of a text x is to be interpreted as the length of the shortest binary string p containing all the information that is necessary for recovering x with the help of some fixed decoding algorithm. Another related approach was developed by Markov (1964) and Kuzmin (1965). What seemed to be especially exciting was the interaction of these ideas with the *perebor* tasks—mainly with Task 4 from Section 1.

First, let us recall in more detail some notations and facts. Given a computable mapping (a *partial* recursive function) u of binary strings (codes) into binary strings (texts, finite objects), the complexity $K_u(x)$ of x with respect to the decoding algorithm u is defined as

$$K_u(x) = \begin{cases} \min \ell(p), \; u(p) = x \\ \infty, \text{ if such a } p \text{ does not exist} \end{cases} \tag{9}$$

Each p for which $u(p) = x$ holds is called a *code* or *program* by means of which u recovers x. For example, the complexity $L(f)$ of a boolean function f as defined earlier corresponds essentially to the decoding algorithm that recovers boolean functions from circuits.

* This was a time of rapid degradation of the moral climate within the Soviet mathematical community, which also affected the "computational complexity" people. Among many examples, Levin was denied his Ph.D. on the basis of political accusations. Dekhtiar's Ph.D. award was seemingly plagued by anti-Jewish feelings: the dissertation was failed, although this decision was reconsidered after protests by prominent mathematicians. Barzdin's doctorate was obstructed for many years because he didn't share some of the scientific preferences of Yablonski.

Clearly, this definition strongly depends on the particular decoding algorithm. The following remarkable fact (by Kolmogorov and Solomonoff), however, permits an invariant definition of complexity in algorithmic terms.

There exists a *partial recursive* (PR) mapping u_o (called *optimal*) such that for any other partial recursive mapping u

$$K_{u_o}(x) \leq K_u(x) + \text{const}_u \qquad (10)$$

Hence, up to additive constants all optimal codings are equivalent. Suppose that some optimal coding u_o is selected once and for all; one defines invariantly (up to an additive constant) the complexity of x to be $K(x)$, omitting the subscript u_o.

Now let us adapt to our notations some facts proved by Kolmogorov that sound like a natural counterpart of the phenomena concerning *perebor* we discussed in Section 2. They confirm both the significance of Task 4 and the belief that Kolmogorov's invariant algorithmic approach was on the right track for the *perebor* topics.

Fact A. $K(x) \leq \ell(x) + \text{const}$

Fact B. Consider the splitting $\prod_\varepsilon^+$, $\prod_\varepsilon^-$ induced by

$$x \in \prod\nolimits_\varepsilon^- \underset{\text{def}}{=} K(x) \leq (1-\varepsilon)\,\ell(x)$$

Then $\prod_\varepsilon^-$ is thin and $\prod_\varepsilon^+$ is thick (in the same sense as for the splitting P_ε^-, P_ε^+ in Task 4).

Fact C (Immunity). Let f be an arbitrary total recursive mapping from natural numbers into natural numbers, such that

$$\limsup f = \infty$$

Then the set

$$A \underset{\text{def}}{=} \{x \mid K(x) \leq f(\ell(x))\}$$

is *recursively enumerable*, and if its complementary set $\neg A$ is infinite, it does not contain any infinite recursive subset. In particular, $\prod_\varepsilon^+$ is immune.

Fact D. Let M^0 be an infinite sequence of binary strings

$$x_1, x_2, \ldots, x_n, \ldots$$

such that $K(x_n) / \ell(x_n) \to 1$. Then the closure of M^0 by including all substrings of strings in M^0 coincides with the set of all binary strings.

Clearly, the first three facts exhibit a precise version of the somewhat vague "frequential effect" and "immunity phenomenon" that came to light in connection with the *perebor* Task 4. Further, Fact D is nothing but a version of Yablonski's main theorem formulated in general terms of optimal coding instead of special coding (of boolean functions) by switching circuits. Side by side with these analogies, the following peculiarities of the switching model are worth noting.

First (a minor point), the complexity measure $L(f)$ takes into consideration only the number of edges in the graph, deliberately ignoring its topology. Therefore, in contrast to $\prod_\varepsilon^-$, which refers to the length $\ell(x)$, in the definition of P_ε^- the size $2^n/n$ is used instead of 2^n, the length of the value column for $f(x_1, \ldots, x_u)$.

The second point is the essential one. In the switching model (as in other known combinatorial models), the decoding Φ (unlike the invariant decoding u_o) is a *total computable* function; moreover, it *is easy to compute.* As a consequence, the complexity $L(f)$ is computable, while $K(x)$ is not. Hence, in Task 4 and in its genuine analogies, one should expect "immunity" to mean "hard computability" of all the subsets instead of failure of recursive enumerability. The remedy was to consider complexity with respect to decoding functions of bounded complexity; for example, for a given total computable function g and complexity measure ρ (say time or space), the related complexity is

$$K_u^g(x) \underset{\text{def}}{=} \min\{\ell(p) : u(p) = x \wedge \rho_u(p) \leq g(p)\} \quad (11)$$

Again, invariant complexity and a suitable splitting π_ε^-, π_ε^+ can be considered where

$$\pi_\varepsilon^- = \{x : K_u^g(x) \leq (1-\varepsilon)\,\ell(x)\} \qquad (12)$$

Then the analog of Task 4 will look as follows.

Task 5.

1. *Existential version.* For arbitrary x decide whether $x \in \pi_\varepsilon^-$.
2. *Constructive version.* If the answer is affirmative, produce a code p that verifies it.

Our Novosibirsk-Riga group maintained a permanent interest in algorithms and randomness. This interest was partially inspired by the investigations of *perebor* in the cybernetics period.

Many algorithmic problems encounter essential difficulties (nonexistence of algorithms or nonexistence of feasible ones), so the natural idea arises to use devices that may produce errors in certain cases. The only requirements are that the probability or frequency of the errors not exceed some acceptable level and that the procedures are feasible. In the framework of this general idea, two approaches caught our attention: probabilistic algorithms and frequential algorithms. It is assumed here and later that in the probabilistic case a device that produces zeros and ones with equal probabilities is available. Leuw, Moore, and Shannon (1956) defined the notion of probabilistic algorithms and proved the negative answer for the question: "Is it possible to compute by a probabilistic algorithm a function that is not computable by common (deterministic) algorithms?"

Frequential algorithms are suggested by the "frequential effect" related to the *perebor* Tasks 4 and 5. For example, the solution of Task 4 (existential version) can be approximated (with practically no computational work) with limiting frequency 100 percent by always returning the answer "No."

The essential features of a frequential algorithm M are generally as follows.

1. M is deterministic, but each time it is applied, it consumes a whole *suitable sequence* of inputs instead of an individual one and works out the corresponding sequence of outputs.
2. The frequency of the correct outputs must exceed a given level.

Clearly, varying the specification of suitable sequences and the acceptable frequency level, one can get different models of frequential algorithms. From one survey (McNaughton 1961) I learned about such a peculiar model and soon realized that as in the probabilistic case, it is impossible to compute functions that are not computable in the usual sense. Our efforts were then attracted by the following questions.

1. Is it possible to compute some functions by means of probabilistic or frequential algorithms with less computational complexity than deterministic algorithms require?
2. What reasonable sorts of problems (not necessarily computation of functions) can be solved by probabilistic or frequential algorithms more efficiently than by deterministic ones? Do problems exist that are solvable by probabilistic or frequential algorithms but not by deterministic algorithms?

Papers by Trakhtenbrot (1973) and Barzdin (1969; 1970) give some idea about the investigations in these areas and contain further references. Here I shall confine myself to quoting one theorem by Barzdin illustrating that for some formulations of problems one can achieve more with probabilistic machines than with deterministic ones. This theorem is based on a precise definition of the statement, "A machine M that enumerates with probability p a set that has the property Q." It claims: "There exists an immune set T such that for any $\alpha < 1$ there exists a probabilistic machine M that enumerates with probability $\geq \alpha$ an infinite subset of R."

Let us examine the relation between this theorem and Yablonski's probabilistic approach to the elimination of *perebor* in Task 3. Recall that Yablonski's theorem essentially claims that $\exists$ a probabilistic algorithm M such that $\forall\ \varepsilon > 0$, it is true that
M enumerates with probability 1 a set M^0 of boolean functions such that

$$M^0 \cap P_\varepsilon^+ \text{ is infinite} \tag{13}$$

As the proof is a straightforward consequence of the thickness of P_ε^+ (that is, of the fact that $|P_\varepsilon^+(n)|/|P(n)| \to 1$), it can be applied as well to the set $\prod_\varepsilon^+$ from Fact C. Therefore, obviously, $\exists$ a probabilistic algorithm M such that $\forall\ \varepsilon > 0$, it is true that

M enumerates with probability 1 a set M^0 of strings such that

$$M^0 \cap \prod\nolimits_\varepsilon^+ \text{ is infinite} \tag{14}$$

Because the sets $\prod_\varepsilon^+$ are immune (see Fact C), it seems that the concern should be with the more subtle question: "Is it possible to enumerate probabilistically an infinite set M^0 that satisfies $M^0 \subseteq \prod_\varepsilon^+$ instead of $|M^0 \cap \prod_\varepsilon^+| = \infty$? It turns out that all of the $\prod_\varepsilon^+$ are among the immune sets R for which Barzdin's theorem holds. Hence, in fact, $\forall \varepsilon > 0$, $\forall \alpha > 0$, $\exists$ a probabilistic machine M such that

$$M \text{ enumerates with probability} \geqslant \alpha \text{ an infinite subset } M^0 \subseteq \prod\nolimits_\varepsilon^+ \tag{15}$$

I should like to summarize the preceding discussion as follows. The investigations in the cybernetics period put forward three main facts concerning the *perebor* topics with respect to switching circuits: (1) the frequential immunity effect, (2) the avoidance of *perebor* via probabilistic algorithms, and (3) Yablonski's *perebor* interpretation of regular algorithms.

The algorithmic approach to complexity placed them in their proper perspective. It turned out that they reflect some interesting phenomena related to optimal coding of finite objects; but by themselves these phenomena do not yet imply any conclusions concerning the *perebor* conjecture. Hence, the further investigation of the conjecture in the framework of algorithmic complexity seemed to be a vital question.

Section 3. *Perebor* and Complexity

The development of algorithmic complexity created a favorable background for alternative approaches to the *perebor* topics. The general idea was that one has to take into account the computational complexity of *perebor* algorithms. Therefore, the inevitability of *perebor* should mean the nonexistence of algorithms that are essentially more efficient. Of course, this general idea does not determine beforehand some other important details; for example: (1) What complexity measure is to be preferred—time complexity, space complexity, or something else? (2) The complexity of what kind of computation has to be estimated—an absolute computation or a relative one (i.e., a reduction)? (3) What should be the meaning of the claim that one algorithm is *essentially* more efficient than the other?

It is not surprising that in the period under consideration, there were different attempts to explain *perebor* phenomena in terms of computational complexity. The first one (Trakhtenbrot 1965) deals with space complexity; the aim of the paper is explicitly declared in the introduction.

> Their main meaning—and this is just reflected in the title ["Optimal Computations and Yablonski's Frequential Effect"]—is in their relation to a general phenomenon to which Yablonski was the first to pay attention. . . . Yablonski elaborated a special system of notions (invariant classes of boolean functions, regular algorithms, etc.) . . . and proved important theorems in which one can find some indirect confirmation of his conjecture with respect to the concrete model he considered. . . .
>
> In this paper another approach is suggested which is based on the estimate of computational complexity, specifically—of space-complexity. The paper does not contain results that directly affect the concrete model of Yablonski; it rather establishes (by suitable diagonalization) the existence of such models, for which even some generalized form of Yablonski's conjecture holds.

The technical results of the paper are about constructions of 0–1-valued functions (predicates) $\Gamma(x)$, with x being a binary string, that are hard to compute.

An additional point is that the complexity of Γ as a whole and/or the complexity of the set $\Gamma^+ = \{x \mid \Gamma(x) = 1\}$ from the splitting Γ^-, Γ^+ it induces, is *precisely* characterized (hence the term *optimal computations*) by the space function φ required to compute them.

Definition

1. φ is optimal on Γ^+ (respectively, Γ^-) iff:
   (a) There is a computation of the whole Γ with space complexity φ.
   (b) φ-hardness: For each computation of Γ with some space complexity ψ, there exists a constant c such that $\forall x\ (x \in \Gamma^+$ (respectively, $\Gamma^-) \rightarrow \psi(x) \geq \varphi(x)/c)$.
2. φ is optimal on Γ iff it is optimal on both Γ^+ and Γ^-.

Remark. Note that φ-hardness of Γ^+ implies also φ-hardness for arbitrary recursive subsets of Γ^+. Hence, the intuition under condition (b) is that Γ^+ (respectively, Γ^-) is φ-immune.

In the constructions under consideration, hardness of computation does not happen because of a "complex" or "random" pattern of zeros and ones in their successive values. In fact, for an arbitrarily slowly increasing recursive function $\lambda(n)$, one can choose Γ such that the number of arguments of length n for which Γ equals zero (its census function, see below) does not exceed $\lambda(n)$. Hence, Γ can be approximated in practically no space or time with frequency approaching one by a constant function equal to one. It is worth noting that Meyer and McCreight found similar results (1971). They related these results to the construction of pseudorandom sequences, as was also done independently by Agafonov (1969). In my paper, the main point was in the following conclusion.

Given a space function φ and a thin regular set R of binary strings, there exists a recursive set of binary strings Γ^+ that satisfies the conditions:

1. *Γ^+ is thick.*
2. *Its complementary set Γ^- includes R as a subset.*
3. *φ is optimal on Γ^+.*

In other words, although Γ^- is thin, it contains an easily decidable subset—namely, the regular set R—whereas the thick complementary Γ^+ is φ-immune. Further, the comparison of the splitting Γ^+, Γ^- with the splitting $P_\varepsilon^+, P_\varepsilon^-$ in Task 4 is suggested.

> It is easy to see that there exists a Turing machine $M_{perebor}$, that works as follows: given a string of length 2^n on its tape (i.e., the code of a boolean function $f(x_1, \ldots, x_n)$), $M_{perebor}$ computes on the same zone of the tape the value $L(f)$ by the trivial *perebor*, and checks if f belongs to P_ε^-. This fact could be interpreted as a confirmation of Yablonski's conjecture, because the space-complexity $\varphi_{perebor}$ of $M_{perebor}$ is indeed optimal on P_ε^+. However in this case, such an interpretation is trivial and is not rich in content, because for each string x, $\varphi_{perebor}$ equals $\ell(x)$, the length of the string. The point is that tape estimate is too rough for moderately complex algorithms; and therefore it seems that in such cases, time-complexity is more suitable. On the other hand, the theorem shows that not only for the $\varphi_{perebor}$ above, but *for an arbitrary space function φ*, it is possible to create a similar splitting; and what is more, the regular set R is a subset of Γ_ε^-.

Note. In Trakhtenbrot (1973*b*) such a splitting for an arbitrary space function φ is described via the following modification of the splitting in Task 5:

1. In the definition of $K_n^g(x)$, the function ρ_u is assumed to measure space complexity.
2. $\Pi_\varphi^- = \{x \mid K_n^g(x) \leq \varphi(x)\}$.

In this case the solution of the modified Task 5 by *perebor* is within the space complexity φ, and φ is optimal on Π_φ^+. Unfortunately, this idea does not work for Task 5 in the case of time complexity. This is not only because "tape complexity is too rough for moderately complex algorithms," as mentioned earlier, but also because tape complexity is too rough to grasp the gap between deterministic and nondeterministic computations. Hence, it became clear that time complexity was to be used.

Meanwhile, I began to feel that another interpretation of *perebor* was worth considering. Again, let us look at how the computation of $L(f)$ is performed by *perebor*, through scanning all possible circuits and querying

$$\Phi(\Omega_1) = f\ ? \qquad \Phi(\Omega_2) = f\ ?\ \ldots \tag{16}$$

until the first adequate circuit is discovered. Incidentally, in this situation checking $\Phi(\Omega_i) = f$ is easy, and hence the inefficiency of the *perebor* algorithm as a whole results from the enormous number of cases that have to be checked. Now suppose that in some other task checking happens to be very hard, even in comparison to the computational work needed for searching through the possible cases. Should we nevertheless identify the computational complexity of the *perebor* algorithm as the complexity of its "searching" component, neglecting the expensive "checking" component? A reasonable answer seemed to be "Yes." In other words, it seemed that the essence of *perebor* is in the complexity of interaction with the "checking" mechanism, as opposed to the complexity of the checking itself. This can be formalized in terms of oracle machines or reduction algorithms; in this view, the inevitability of *perebor* is to be interpreted in terms of the computational complexity of the reduction process. This idea resulted in the following conjecture about the impossibility of eliminating *perebor* in the task of computing a function relative to its graph: Given a total function f that maps binary strings into binary strings, consider Turing machines to compute f that are supplied with the oracle G that delivers (at no cost!) the correct answers to question "$f(x) = y$?"[1] Among them is a suitable machine $M_{perebor}$ that incorporates the *perebor* strategy appropriately and computes $f(x)$ by addressing the oracle with the questions

$$f(x) = B(0)?, f(x) = B(1)?, \ldots, \quad f(x) = B(i)?\ \ldots \tag{17}$$

where $B(i)$ is the ith binary string in lexicographical order. Hence, in computing the value of $f(x)$, the machine $M_{perebor}$ spends about $\tilde{f}(x)$ steps, where $\tilde{f}(x)$ denotes the natural number represented by the string $f(x)$. I conjectured in 1966 that for a broad spectrum of functions f, no other oracle machine M can perform the computation essentially faster. As to the "graph predicates" $G(x,y) \underset{\text{def}}{=} f(x) = y$ of such functions, it was conjectured that they would not be too hard to compute.

Dekhtiar (1969) proved the conjecture for different versions of what "essentially faster" should mean. For example, it turns out that in many cases $M_{perebor}$ is optimal up to a multiplicative constant; that is, for every other oracle machine M, there exists a constant $C(M)$ such that

$$\forall x\ (t_M(x) \geq C(M) \cdot \tilde{f}(x)) \tag{18}$$

On the other hand, it is quite obvious that if nondeterministic oracle machines are allowed, the computation of the string $f(x)$ is performable in $\ell(f(x))$ steps—that is, in just as many steps as one needs to write down the correctly guessed value of $f(x)$.

Clearly, there is a considerable gap between the lower estimate $C \cdot \tilde{f}(x)$ for deterministic machines and the upper estimate $\ell(f(x))$ for nondeterministic ones, because $\ell(f(x))$ is approximately $\log(\tilde{f}(x))$. In connection with this remark, it is worth noting that Dekhtiar's construction, in particular, produces functions f that obey (18) and in addition have the following properties:

1. f is polynomially increasing; that is, for suitable c and k,

$$\forall x\ (\ell(f(x)) \leq c \cdot (\ell(x))^k) \tag{19}$$

2. $\ell(f(x))$ is not neglectably small in comparison with $\ell(x)$.

$$\limsup \ell(f(x))/\ell(x) \neq 0 \tag{20}$$

At that time, the terminology concerning P reducibility and NP problems was not yet in use; nevertheless, it seems instructive to make some comments just in these terms. Property (19) implies that the function f is computable in polynomial time by a nondeterministic machine with the oracle G that supplies answers to questions "$f(x) = y$?" On the other hand, properties (18) and (20) imply that for no machine with the oracle G is the time complexity polynomial. Hence, relative to the oracle G, the function f is in NP − P; in other words, NP ≠ P as far as computations with oracles are concerned for single-valued functions. Usually the "P = NP?" question is formulated in terms of set decidability (and not in terms of function computation as in the preceding). It is quite obvious, though, that $f \in \text{NP} - \text{P}$ implies $\hat{G} \in \text{NP} - \text{P}$, where $\langle x,y \rangle \in \hat{G} \underset{\text{def}}{=} f(x) \leq y$.

To summarize, Dekhtiar's construction includes the proof of the relativized version of the NP ≠ P conjecture. For the first time, this version was explicitly announced by Baker, Gill, and Solovay (1975), together with another theorem that claims the relativized version of the NP = P conjecture. The intention was to give some evidence to the possibility that neither NP ≠ P nor NP = P is provable in common formalized systems. As to my conjecture about the computation of a function relative to its graph and Dekhtiar's proof, they had nothing to do at that time with the ambitious hopes to prove the independence

[1] x,y may vary, but f is always the same function under consideration.

of the NP $\neq$ P conjecture. As a matter of fact, I then believed (and to some extent I do even now) that the essence of *perebor* is adequately reflected by the complexity of such relative computations that use searching through the sequence of all binary strings. Hence, being confident that the true problem is being considered (and not its relativization!), I had no stimulant to look for models in which *perebor* could be eliminated.

On the other hand, there was some impression that Dekhtiar's results might be improved and generalized with respect to the following two points.

1. *Complexity of the Graph.* The construction by diagonalization produced a predicate $G(x,y) \overset{=}{\underset{\text{def}}{}} f(x) = y$ computable within exponential time, but it seemed likely that one could manage with functions f for which the graph $G(x,y)$ is computable in polynomial time.

2. *Ordering of the Strings.* No justification was evident as to why one had to use in *perebor* the ordering of strings fixed once and for all by lexicographical ordering as in (17) or by some other way. On the contrary, it seemed natural that the ordering must take account of x—that is, that it must vary as x varies.

Questions of this sort were still under consideration when dramatic events occurred that threw a fresh light on the problem. In 1971 S. Cook and L. Levin independently elaborated a new approach to the problem that resulted in the discovery of complete NP problems.

Cook (1971) proved SAT to be NP complete; only after Karp (1972) showed many important combinatorial problems to be NP complete, were these ideas and results fully appreciated and produced a real sensation in the United States. This work became available in the U.S.S.R. no earlier than 1973. In any case, at the Conference on Complexity Theory and Development of the Foundations of Information Theory, dedicated to the 70th birthday of A. N. Kolmogorov (Tsakhkadzor, March 1973), the audience was still ignorant about the brilliant contributions of Cook and Karp. Meanwhile in 1971, Levin obtained similar results, although he used somewhat different terminology—for example, "universal *perebor* problems" instead of "complete NP problems."

Because in the West it is still not too clear exactly what Levin did and when he did it, I feel obliged to give more details of this story. Levin reported his results in 1971 in Moscow (Kolmogorov's seminar at Moscow University and Markov's seminar at the Computing Center of the U.S.S.R. Academy of Sciences) and in Leningrad (the Leningrad section of the Steklov Mathematical Institute). The results were appreciated in these spheres, but they did not produce such a sensation as Karp's work had in the United States. In Novosibirsk we only learned about Levin's results in April of 1972. At this time Levin had just completed his Ph.D. thesis (under Kolmogorov's guidance), "On the Algorithmic Approach to Probability Theory and Information Theory." In fact, the reason for Levin's visit to us that spring was to promote his thesis, then under consideration in the Novosibirsk Institute of Mathematics. In connection with this promotion Barzdin arrived at the same time from Riga. I remember that he called me from the hotel with excitement, saying, "Just now Levin told me about his new results; it is a turning point in the topics of *perebor*!" During Levin's lecture in our seminar, the audience recognized that something very important was happening; in fact, we were witness to the start of the "NP era"!

Levin's interest in *perebor* was stimulated by two factors: (1) The earlier investigations and discussions on this topic in the U.S.S.R.; this is explicitly reflected in the bibliography to his paper (Levin 1973), which was submitted in June 1972. (2) The task of the computation of Kolmogorov complexity under bounded time—that is, essentially the constructive version of Task 5.

Although Levin's 1973 paper is laconic (as are Levin's publications in general), the formulations are absolutely crisp in the Russian original. Unfortunately, the English translation is awkward and contains misrepresented and confusing formulations. [*Note:* The translation is reprinted in the Appendix of this article.] Six tasks are considered, among them a part coinciding with Karp's and a part being different; then their universality is stated in a form that is somewhat stronger than in the sense of Cook-Karp.

Naturally, the new approach promoted a revision of the former views on *perebor* and especially a reevaluation of some questions and conjectures in the context of the fundamental "NP = P?" question. Conjecture 1 on the complexity of the graph implies NP $\neq$ P; therefore, it is not easier to prove than the original conjecture NP $\neq$ P. Let us confine ourselves to some remarks concerning the status of Tasks 4–5 in view of the new situation.

Clearly, the existential versions of Tasks 4–5 are NP problems. The conjecture is that they are not solvable in polynomial time, and moreover that the dense P_ε^+ is *immune* in the following precise sense: P_ε^+ contains no subset that is decidable in polynomial time. Note that neither in Levin's and Cook's papers, nor in later works, were Tasks 4–5 proved to be NP complete. Moreover, in light of investigations started by A. Meyer and J. Hartmanis, it is highly unlikely that they will be proved to be such for some time. These investigations concern the correlation between NP sets, thin sets, and thick sets; they remind us of the "frequential effect" of Tasks 4–5 that attracted us in the U.S.S.R. They culminated in Mahaney (1980), with the following result: if NP $\neq$ P, then no complete NP set may be *sparse*. Here, sparseness of a set S means that the "census function" $\lambda(n)$, defined as the number of binary strings in S of size up to n, is bounded by some polynomial of n. Note that in specifying the "frequential effect," we formerly used other versions of sparseness—for example, the density of 0 in P_ε^- or, as in Trakhtenbrot (1965), the fact that for Γ_ε^-, $\lambda(n)$ is bounded by some (arbitrarily) slowly increasing recursive function.

Besides the NP machinery, Levin (1973) has another result (Theorem 2) that unfortunately did not attract the attention it deserved. Certainly, one reason was that people were too struck by the main result—the discovery of complete NP problems. Theorem 2 deals with the constructive version of NP problems (see Introduction) and claims:

> For any *perebor* problem $A(x,y)$ there exists an algorithm that solves it in time that is optimal up to multiplication by a constant and the addition of a number [that is polynomially] comparable with the length of x.

Note that optimality is stated only for those x's for which *there exists* a suitable y, and nothing is claimed for the other ones. (An open question promoted by Hartmanis is whether an algorithm exists that is optimal for both the affirmative and negative cases.) Unfortunately, there is no description of the optimal algorithm in Levin (1973), which is perhaps another reason that Theorem 2 was not fully appreciated. As a matter of fact, we never knew in Novosibirsk of Levin's algorithm, and that is why Sazonov later published (1980) an algorithm of his own. Levin's original idea as seen in Levin (1980) was that the optimal way to find the y for a given x was to search via *perebor* through the sequence of all binary strings. Unlike the "traditional" *perebor*, one has to abandon the fixed ordering of the strings (as, for example, in (17)) with respect to their increasing lengths; instead, the strings y are to be checked in the increasing order of $Kt(y/x)$—the complexity of y relative to x, which is defined as follows.

$$Kt(y/x) \underset{\text{def}}{=} \min\{\ell(p) + \log t_u(p,x) : u(p,x) = y\}$$

It equals the minimal sum of the length of the program p and the logarithm of the time it spends, including the time for checking $A(x,y)$. Here, u performs the decoding relative to x, and, as in the definition of Kolmogorov complexity, it can be chosen to be optimal.

Sazonov (1980) used a similar idea of "polynomially optimal" ordering of the strings to be checked. Hence, the original intuitive idea of *perebor* as a search through all the strings is rehabilitiated; moreover, *perebor* is proved to be optimal and in this sense inevitable. Of course, its precise computational complexity remains unknown; in nontrivial cases, it depends on the "NP = P?" question.

Epilogue

In the decade after the Cook-Karp-Levin discovery, the "P = NP?" problem became one of the most popular superproblems in theoretical computer science. It is beyond the scope of this paper to survey the respective development and the contributions of many outstanding computer scientists. I shall just make two remarks that are related to the story I have told.

The first concerns the participants in the *perebor* controversy. Have their views changed? There is no evidence that this has happened. Clearly, the adherents of the algorithmic approach to complexity interpreted the current developments in the area as a confirmation of their correctness. On the other hand, there was no formal reaction by their opponents on this topic, nor did they publish new results along the line of their own understanding of *perebor*. Meanwhile, other changes occurred. Many of the people who were active earlier in complexity theory (including myself) moved to other research fields, particularly to what is called in the U.S.S.R. "theoretical programming." Moreover, some of them "moved" in the literal sense; the participants of the story are now dispersed over different continents.

The final remark is about independence results in computer science—a direction of research that seems particularly significant. Recall that the first attempt in this direction (Hartmanis and Hopcroft 1976) discovered that in any sufficiently strong axiomatic system Ax, some special version of P = NP is independent of Ax. Yet this version looked somewhat artificial; later efforts (De Millo and Lipton 1980; Sazonov 1980) were dedicated to direct and natural formulations of the original P = NP problem in the axiomatic systems under consideration. Actually, activities of this sort outgrew the primary "P = NP?" problem; they concern both the technical analysis of mathematical and logical means and the more fundamental analysis of mathematical abstractions that are relevant to computer science.

It is worth noting that the importance of such an approach was advocated in the U.S.S.R. even prior to the "NP era." As early as the 1950s Liapunov argued that hard problems in theoretical cybernetics need an analysis in the spirit of descriptive set theory and foundations of mathematics. In the late 1960s Barzdin supported similar arguments with respect to provability of nontrivial lower bounds for the complexity of computations. He conjectured that they may happen to be independent of the axioms that computer scientists really have in mind.

Perhaps the most impressive repercussions of the impasse with the "P = NP?" problem are the intensified efforts to clarify the methods and axioms that computer science relies on or should rely on (see Joseph and Young 1981). I share the widely accepted opinion on the importance of this trend, and I hope that essential progress will be achieved in this area.

Acknowledgments

It was Juris Hartmanis's idea that I should prepare a paper on this topic for the *Annals of the History of Computing*. I benefited much from his own *Annals* paper as a model of "a subjective, historical account of some developments in ... theoretical computer science in which I participated" (Hartmanis 1981). Y. Gurevich, A. Meyer, and anonymous referees contributed valuable remarks and improvements. Z. Galil, M. Sharir, and A. Yehudai did their best to remove the traces of my poor English and to polish the text. It is both a duty and a pleasure to express my gratitude to all of these colleagues.

REFERENCES

Adelson-Velski, G., V. Arlazarov, and M. Donskoi. 1976. *Programirovanie Igr* (*Programming of Games*). Moscow, Nauka, 256 pp. (Russian).

Agafonov, V. 1969. "On Algorithms, Frequency and Randomness." Ph.D. dissertation, Novosibirsk Institute of Mathematics (Russian).

Baker, T., J. Gill, and R. Solovay. 1975. Relativizations of the P=NP question. *SIAM J. Computing 4*, 4, 431–442.

Barzdin, Y. 1969. On computability by probabilistic machines. *Dokl. A.N. SSSR 189*, 4; trans. *Soviet Math Dokl. 10*, 6, 1464–1465.

Barzdin, Y. 1970. On the relative frequency of solutions of algorithmically unsolvable mass problems. *Dokl. A.N. SSSR 191*, 5; trans. *Soviet Math. Dokl. 11*, 2, 459–462.

Chaitin, G. 1966. On the length of programs for computing finite binary sequences. *JACM 13*, 547–569.

Cook, S. A. 1971. The complexity of theorem proving procedures. *Proc. 3d Annual ACM Symposium on Theory of Computing*, pp. 151–158.

Dekhtiar, M. 1969. On the impossibility of eliminating PEREBOR in computing a function relative to its graph. *Dokl. A.N. SSSR 189*, 4, 748–751; trans. *Soviet Math Dokl. 14*, 1146–1148.

De Millo, R., and R. Lipton. 1980. The consistency of "P=NP" and related problems with fragments of number theory. *Proc. 12th ACM Symposium on Theory of Computing*, pp. 45–57.

Hartmanis, J., and J. Hopcroft. 1976. Independence results in computer science. *ACM SIGACT News 8*, 4, 13–24.

Hartmanis, J. 1978. "Feasible Computations and Provable Complexity Properties." *Regional Conference Series in Applied Mathematics*. Philadelphia, Arrowsmith, 62 pp.

Hartmanis, J. January 1981. Observations about the development of theoretical computer science. *Annals of the History of Computing*, Volume 3, Number 1, pp. 42–51.

Joseph, D., and P. Young. 1981. Independence results in computer science? *J. Computer and System Sciences 23*, 205–222.

Karp, R. M. 1972. "Reducibility Among Combinatorial Problems." In R. E. Miller and J. Thatcher (eds.), *Complexity of Computer Computations*, New York, Plenum Press, pp. 85–104.

Khachijan, L. 1979. A polynomial algorithm for linear programming. *Dokl. A.N. SSSR 244*, 1093–1096 (Russian).

Kolmogorov, A. 1965. Three approaches to the definition of the concept of "the quantity of information." *Problemy Peredachi Informacii 1*, 3–11 (Russian).

Kuzmin, V. 1965. Realization of boolean functions by automata, normal algorithms and Turing machines. *Problemy Kibernetiki 13*, 75–96, Moscow, Nauka (Russian).

Landweber, L., R. Lipton, and E. Robertson. 1981. On the structure of sets in NP and other complexity classes. *Theoretical Computer Science 15*, 181–200.

Leuw, K., F. Moore, and C. Shannon. 1956. "Computability on Stochastic Machines." In C. Shannon and J. McCarthy (eds.), *Automata Studies*, Princeton, Princeton University Press.

Levin, L. 1973. Universal sequential search problems. *Probl. Pered. Inform IX, 3*; trans. *Probl. Information Transmission 9*, 3, 265–266.

Levin, L. 1981. "The Randomness Conservation Laws: Information and Independence in Mathematics." Technical Report, MIT & Boston University, pp. 1–22.

Lupanov, O. 1958. On the synthesis of switching circuits. *Dokl. A.N. SSSR 119*, 1, 23–26 (Russian).

Mahaney, S. R. 1980. Sparse complete sets for NP, solution of a conjecture of Berman and Hartmanis. *Proc. 21st Symposium on Foundations of Computer Science*, pp. 54–60.

Markov, A. 1964. On normal algorithms, that compute boolean functions. *Dokl. A.N. SSSR*; trans. *Soviet Math. Dokl. 157*, 2, 262–264.

McNaughton, R. 1961. *The Theory of Automata, a Survey*. New York, Academic Press, pp. 379–421.

Meyer, A., and F. McCreight. 1971. Computationally complex and pseudorandom zero-one valued functions. *International Symposium on Theory of Machines and Computations*. New York, Academic Press, pp. 19–42.

Petri, N. 1969. The complexity of algorithms and their operating time. *Dokl. A.N. SSSR 186*, 30–31; trans. *Soviet Math. Dokl.*

Rabin, M. O. 1960. "Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets." Technical Report No. 3, Jerusalem, Hebrew University.

Sazonov, Y. 1980. A logical approach to the problem "P=NP?" MFCS, *Lecture Notes in Computer Science* No. 88, New York, Springer-Verlag, pp. 562–575.

Solomonoff, R. 1964. A formal theory of inductive inference. *Information and Control 7*, 1, 1–22.

Trakhtenbrot, B. 1956. Signalizing functions and tabular operators. *Trans. Penza Pedagogical Institute*, No. 4, pp. 75–87 (Russian).

Trakhtenbrot, B. 1963. On the frequency computations of recursive functions. *Algebra i Logika 2*, 1, 25–32.
Trakhtenbrot, B. 1965. Optimal computations and the frequential Yablonski-effect. *Algebra i Logika 4*, 5, 79–83.
Trakhtenbrot, B. 1973*a*. Frequency computations. *Trudy Mathem. Inst. Im Steklova*; trans. *Proc. Steklov Inst. Math.*, 125, 118–127.
Trakhtenbrot, B. 1973*b*. Formalization of some notions in terms of computational complexity. *Proc. Intern. Congress for Logic Methodology and Philosophy of Science* (1971), Studies in Logic and Foundations of Mathematics, Amsterdam, North-Holland, Vol. 74, pp. 205–214.
Trakhtenbrot, B. 1975. On problems solvable by successive trial. MFCS 75, Marianske Lazne, *Lecture Notes in Computer Science*, Vol. 32, New York, Springer-Verlag, pp. 125–138.
Yablonski, S. 1959*a*. On the impossibility of eliminating PEREBOR in solving some problems of circuit theory. *Dokl. A.N. SSSR 124*, 44–47; trans. *Soviet Math. Dokl.*
Yablonski, S. 1959*b*. Algorithmic difficulties in the synthesis of minimal contact networks. *Problemy Kibernetiki 2*, Moscow, Fizmatgiz, pp. 75–121 (Russian).
Yanovskaia, S. 1959. Mathematical logic and fundamentals of mathematics. *Mathematics in the USSR for 40 Years*, Moscow, Fizmatgiz, pp. 13–120 (Russian).
Zhuravlev, Y. 1960. On the impossibility of constructing minimal disjunctive normal forms for boolean functions by algorithms of certain class. *Dokl. A.N. SSSR 132*, 504–506; trans. *Soviet Math. Dokl.*
Zvonkin, A., and L. Levin. 1970. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Surveys 25*, 6, 83–124.

APPENDIX

The following translation (© Plenum Publishing Corporation, reprinted here with permission) is from "Problemy Peredachi Informatsii," Vol. 9, No. 3, July–September 1973, pp. 115–116 (original article submitted June 7, 1972). The changes shown in brackets (—— means delete; otherwise insert) are those suggested by Trakhtenbrot. Changes originally made by Levin are shown in boldface.

BRIEF COMMUNICATIONS

UNIVERSAL [~~**SEQUENTIAL**~~ SEARCH PROBLEMS

L. A. Levin

Several well-known [~~**large-scale**~~] problems of the "[~~**sequential**~~] search" type are discussed, and it is proved that those problems can be solved only in the time it takes to solve any problems of the indicated type, in general.

After the concept of the algorithm had been fully refined, the algorithmic unsolvability of a number of classical [~~**large-scale**~~] problems was proved (including the problems of the identity of elements of groups, the homeomorphism of varieties, the solvability of the Diophantine equations, etc.). These findings dispensed with the question of finding a practical technique for solving the indicated problems. However, the existence of algorithms for the solution of other problems does not eliminate the analogous question, because the volume of work mandated by those algorithms is fantastically large. This is the situation with so-called [~~**sequential**~~] ([~~**or**~~] exhaustive) search problems, including: the minimization of Boolean functions, the search for proofs of finite length, the determination of the isomorphism of graphs, etc. All of these problems are solved by trivial algorithms entailing the sequential scanning of all possibilities. The operating time of the algorithms, however, is exponential, and mathematicians nurture the conviction that it is impossible to find simpler algorithms. That conviction has been reinforced by a number of profound arguments (see [1, 2]), but no one has yet succeeded in proving it (for example, it has yet to be proved that more time is required to find mathematical proofs than is required to test them).

If we assume, however, that there exists [~~in general~~ at all] some (even if artificially formulated) [~~**large-scale**~~] problem of the [~~**sequential**~~] search type that is unsolvable by simple (in terms of the volume of computations) algorithms, then it can be shown that many "classical" [~~sequential~~] search problems have the same property (including the minimization problem, the proof-search problem, etc.). This objective comprises the [~~fundamental~~ main] results of the present note.

We say that functions $f(n)$ and $g(n)$ are comparable if for some k

$$f(n) \leqslant (g(n) + 2)^k \text{ and } g(n) \leqslant (f(n) + 2)^k$$

We give an analogous interpretation to the term "less than or comparable with."

Definition. A problem of the **sequential** search type (or, simply, a [~~**sequential**~~] search problem) [is] a problem of the following type: "For a given x find some y of length comparable with the length of x such that $A(x, y)$ holds," where $A(x, y)$ is some property to be tested by an algorithm whose operating time is comparable with the length of x. (Here we understand by an algorithm a Kolmogorov–Uspenskii algorithm, a Turing machine algorithm, or a normal algorithm; x and y are binary words.) The corresponding quasi-[~~sequential~~] search problem is the problem of determining whether such a y exists.

We consider six problems of these types. The entities with which they are concerned are encoded in a natural way by binary words. The particular choice of natural encoding is not significant here, since they all yield comparable code lengths.

Problem 1. A list [~~generates~~ determines] a finite set and a covering of that set by 500-element subsets. Find a subcovering having a prescribed cardinality (determine whether such a subcovering exists).

Problem 2. A table generates a partial Boolean function. Find a disjunctive normal form of prescribed dimensions realizing that function in [~~the~~ its] domain [~~of definition~~] (determine whether such a DNF exists).

Problem 3. Determine whether a given formula of the [**~~predicate~~ propositional**] calculus is deducible or refutable (or, equivalently, whether a given Boolean formula is equal to a constant).

Problem 4. Two graphs are given. Find a homomorphism of one onto the other (determine whether such a homomorphism exists).

Problem 5. Two graphs are given. Find an isomorphism of one into the other (onto part thereof).

Problem 6. Consider matrices composed of integers from 1 to 100 and a certain stipulation as to which integers can be vertically adjacent and which can be horizontally adjacent. When the outermost integers are given, continue them over the entire matrix, observing the given stipulation.

Let $f(n)$ be a monotonic function.

THEOREM 1. If there exists [**~~in general~~** a **~~certain large-scale sequential~~** at all] search (quasi [**~~sequential~~**] search) problem unsolvable in a time less than $f(n)$ for an argument of length comparable with n, then Problems 1–6 also have this property.

The idea of the proof is that Problems 1–6 are so-called "universal [~~sequential~~] search problems."

Definition. Let $A(x, y)$ and $B(x, y)$ determine, respectively, [~~sequential~~] search problems A and B. We say that problem A reduces to B if three algorithms $r(x)$, $p(y)$, and $s(y)$ operating in a time comparable with the length of the argument exist such that $A(x, p(y)) \equiv B(r(x), y)$ and $A(x, y) \equiv B(r(x), s(y))$ (i.e., given the A-problem in x, an equivalent B-problem in $r(x)$ is easily constructed from it). A problem to which any [~~sequential~~] search problem reduces is called "universal."

Thus, the substance of the proof of Theorem 1 is embodied in the following lemma.

LEMMA 1. Problems 1–6 are universal [**~~sequential~~**] search problems.

The method described here [~~clearly~~] provides a means for readily obtaining results of the type of Theorem 1 and Lemma 1 for [~~the majority of~~ many] important [~~sequential~~] search problems. It still remains, however, to prove the condition stipulated in Theorem 1. A great many attempts have been made in this direction for some time now, and a number of interesting results have been obtained (see, e.g., [3, 4]). Of course, the universality of various [~~large-scale sequential~~] search problems can be established without solving the indicated problem. The following is provable in the system of Kolmogorov–Uspenskii algorithms.

THEOREM 2. For any [**~~large-scale sequential~~**] search problem $A(x, y)$ there exists an algorithm that solves it in a time that is optimal up to multiplication by a constant and the addition of a number comparable with the length of x.

The author is deeply grateful to A. N. Kolmogorov, B. A. Trakhtenbrot, Ya. M. Barzdin [**~~Yu. I. Al'brton~~ Alberton**], and M. I. Degtyar' for a rewarding discussion.

LITERATURE CITED

1. S. V. Yablonskii, "Algorithmic difficulties in the synthesis of minimal contact networks," in: Problems of Cybernetics [in Russian], Vol. 2, Fizmatgiz, Moscow (1959), pp. 75–121.
2. Yu. I. Zhuravlev, "Set-theoretic methods in logic algebra," in: Problems of Cybernetics [in Russian], Vol. 8, Fizmatgiz, Moscow (1962), pp. 5–44.
3. B. A. Trakhtenbrot, "Optimal computations and the partial Yablonskii effect," Seminar [in Russian], Vol. 4, No. 5, Nauka, SO, Novosibirsk (1965), pp. 79–93.
4. M. L. Degtyar', "On the impossibility of eliminating exhaustive search in the computation of functions with respect to their graphs," Dokl. Akad. Nauk SSSR, *189*, No. 4, 748–751 (1969).