Kybernetika

PARALLEL PROBABILISTIC SEARCHING AND SORTING ALGORITHMS

IVAN KRAMOSIL

ACADEMIA
PRAHA

Kybernetika

ROČNÍK/VOLUME *26* (1990)

Časopis Československé kybernetické společnosti při ČSAV/The journal of the Czechoslovak Association for Cybernetics under the auspices of the Czechoslovak Academy of Sciences

Vydává/Published by:

Ústav teorie informace a automatizace Československé akademie věd/Institute of Information Theory and Automation of the Czechoslovak Academy of Sciences

Redakce/Editorial Office:

Pod vodárenskou věží 4, 182 08 Praha 8

PARALLEL PROBABILISTIC SEARCHING AND SORTING ALGORITHMS

RNDr. Ivan Kramosil, DrSc.

Příloha časopisu Kybernetika/Supplement to the journal Kybernetika

PREFACE

Many interesting and important tasks of artificial intelligence can be solved converting or reducing them appropriately either to searching problems, when an element of a desired property is to be found in a large set, or to sorting problems, when the set in question is to be endowed by a structure. As a rule, time and space complexity or other demands connected with the resulting searching or sorting problem(s) ultimately influence the total time, space or otherwise quantified complexity of the program as a whole, in which searching or sorting occur as subprograms, and very often are decisive for applicability of the main program. The main attention, from the theoretical and practical viewpoint as well, is focused to the cases when a blind systematic exhaustive searching or sorting "by force" is intractable either theoretically (the universe of discourse is infinite) or practically (this universe being finite but too large).

Under realistic restrictions concerning the abilities of the testing and sorting devices (oracles) being at hand and without a more detailed knowledge concerning the basic set (universe of discourse) in question, the nature of its elements or its possible a priori structuralization, the time complexity of searching and sorting algorithms may increase rather rapidly with the cardinality of the basic set increasing.

The time, space or other demands connected with searching and sorting algorithms can be reduced either by a sophisticated use of specific features of particular searching or sorting problems, or using some general principles as randomization or parallelization. Our effort, in what follows, will be oriented just in this latter direction. Namely, our attention will be focused to an appropriate combination of both the principles in order to obtain parallel probabilistic algorithms for certain searching and sorting problems, with sufficient statistical qualities, and working under realistic conditions and within reasonable time limitations.

It is not the aim of this case study to survey exhaustively everything what has been done, or at least what the author knows to have been done, in the domain in question in general till now. Rather, we would like to present systematically author's own results in this direction, and to offer an introductory study into the domain of parallel probabilistic algorithms. For these sakes, as well as in order to make the text self-explanatory, a rather elementary and detailed formalization of classical as well as non-classical kinds of algorithms is presented in Chapters 2 and 3. The argumentations and reasoning of probabilistic nature are mostly the very elementary combinatorial ones, some exceptions being covered by references of textbooks or monographies on probability theory. Mathematical models, statements and conditions under which they hold are always presented at the level of formalization common in mathematical texts, proofs are introduced only supposing they have not been published yet (e.g., the results in Chapters 5 and 6), being replaced by references otherwise.

Keeping in mind the sequential way of publication of this study, references are presented immediately after each chapter to made its use more convenient for the reader. Besides the items namely quoted in the text, references introduce some textbooks and monographies, most of them of undergraduate level, which can serve as sources for preliminary knowledge from mathematical logic, theory of recursive functions and probability theory. The list of these textbooks and monographies is far from being exhaustive and most of its items may be successfully replaced by other ones, perhaps more easily accessible in a particular reader's position. However, when forming the list of references, the accessibility of particular titles served as the main choosing criterion; it is also why Czech and Russian translations are introduced supposing their existence is known to the author. Finally, the supplementary list of references contains some items which stand in relatively close connections to the investigated problems and which may serve either for an interesting and useful confrontation or as an inspiration for further development. This supplementary list of references will close the last chapter.

The author hopes this study to be of certain use for specialists in artificial intelligence, applied theory of algorithms and statistical decision making.

Prague, June 1988

Ivan Kramosil

CONTENTS

1. INTRODUCTION

Instead of beginning this study with some vague, proclamative and rather philosophical statements concerning the parallel probabilistic algorithms we shall introduce them, in this chapter, from the point of view of their possible applications in the so called knowledge systems. Both these notions, "parallel probabilistic algorithms" and "knowledge systems", have rather a lot in common. Or, both of them denote very young domains of applied science under a very rapid development in our days, both of them are very frequented in contemporary science as well as in many considerations of various provenience and degree of seriousness and, finally, there is a certain shadow of fashionness and sensationalism in both these notions. All this makes justifiable our assumption that the reader has already met both the notions in question, on the other hand, it is not possible to avoid the case that the context of this meeting has not supported too much the reader's serious and moderate image about both the domains. Hence, let us use this introductory chapter in order to limit, more correctly, the field of our interests in this study keeping in mind, of course, that a higher-level preciseness can be given only after having introduced the most necessary formal apparatus (in the next two chapters). At the very beginning, let us describe, in more details, both the notions introduced above.

The term "knowledge systems" is often understood as synonymous with that of "expert systems", but we do not consider this identification as happy or even appropriate, at least in our context. It is caused by the fact that the notion "expert systems" seem to be too charged with the idea that the pieces of knowledge contained in and handled by the system are exclusively the expressions of subjective and rather vague knowledge of human beings-experts, if not taking into account some more anthropomorphic imaginations resulting from this idea.

In the most general form to which we shall always refer in this work, knowledge system consists of two parts: of a collection of basic or outcoming pieces of knowledge or data (databases), and of a deductive or, more generally, inference mechanism or inference machine, which enables to deduce knowledge from the given database. In a non-trivial case, of course, also other pieces of knowledge than those explicitly put in the database can be obtained. The ways in which the pieces of knowledge are inscribed in the database can be rather various: assertions or statements within the framework of a natural, but also formalized language, tables of function values, graphs (including the well-known semantic nets), or even other objects or structures. An integral part of databases are data charged with an uncertainty or vagueness and also the ways in which this uncertainty may be expressed are very different: verbal expressions like "maybe", "perhaps", "probably", "we cannot avoid that...", etc., operators of appropriate modal logics, evaluations of degree with which an assertion holds or of degree of belief of a subject that the assertion holds, where the degrees can be of numerical, as it is the most common case, but also of non-numerical nature, and so on. Vagueness of an assertion, to distinguish it from the uncertainty,

corresponds rather to certain ambivalence of the used terms which results in a plurality of possible semantical interpretation of knowledge contained in the database.

Also the inference machines can be very different, either from the viewpoint of their nature, or when considering the degree of their inference abilities. The most simple inference method consists in the verification, whether an assertion, put on the input of the system as a question, can be found in the database. If it is the case, the answer to the question is affirmative (positive), the answer being negative otherwise. Let us neglect, for an instant, the evident fact that when the database is very large or its elements are hard to access, even this "inference machine" is far from being simple. The reader familiar with the foundations of the PROLOG programming language immediately notes, that the deduction power of this language reduces just to this exhaustive searching mechanism supposing that the database does not contain any formula with a free indeterminate. By the way, in its full powers it is just the PROLOG language which can be seen as a good example of a non-trivial, but theoretically as well as practically effective and realizable inference machine. As another extremal example of such a machine we may take, in case the data are written as well-formed formulas of first or higher-order predicate logic, the full deductive apparatus of the logic in question with all consequences concerning the undecidability and incompleteness of those calculi.

Let us remark, that all the inference machines mentioned till now deal exclusively with the so called "certain information", i.e. all the pieces of knowledge put into the database are taken as unambiguous and surely valid ones and only those new data are derived from the given ones, which follows with logical necessity. However, a very topical problem of contemporary theoretical and applicational research effort in the domain of knowledge systems is that of deducing some knowledge from uncertain premises, namely how to derive the (numerical or non-numerical) degree of validity or degree of belief for the deduced assertion or answer on the ground of the degrees of belief or validity ascribed to the premises used during the deduction. The so called *extensional* inference machines try to deduce the degree of validity, of the resulting answer or statement using *only* the degrees of validity of premises and a universal (in the framework of the knowledge system in question) computational or combinational rule working over the degrees of validity ascribed to the premises. The resulting relatively rather low computational complexity of extensional inference mechanisms is the most often introduced argument in favour of such systems. On the other hand, the extensionality of an inference machine is hardly compatible with the possibility to interprete the degrees of validity or belief as probabilities. Such an interpretation is possible only under some very strong and hard to defend assumptions concerning the statistical dependence of the observed events and measured quantities taken as random events and random variables. Hence, it is almost natural that among the *intensional* (i.e. non-extensional) inference mechanisms just the systems consequently based on the probabilistic models and approaches have already reached the stage of full theoretical foundations and

following experimental verification. The basic assumption of probabilistic inference machines can be expressed in such a way that the simultaneous probability distribution is the only and completely exhaustive description of all the relations among the variables which the pieces of knowledge in the database describe. The degrees of validity (weights) of the assertions posed as questions to the system, or which are output by the system as its answers, are then reduced to the values of corresponding conditional probabilities. From the point of view of practical calculation the problem is, of course, very far from being trivial and its practical solvability requests some compromises, namely when replacing the simultaneous distribution in question by its appropriate and theoretically justified approximation or simplification.

In the connections with the sakes and intentions of this study, however, it will be interesting to analyze the inference machines still from another point of view. The greatest part of those machines or mechanisms consist, in fact, in the application of a computational rule or formula to *appropriate* arguments. Once these arguments having been known, given or found, the application of the rule or the computation itself is usually relatively simple and not too much time, space or other expenses consumpting. On the other hand, the justification that just the rule or computational formula in question are the proper, adequate or reasonable ones may request a great portion of theoretical effort and large and complicated theoretical constructions. The problem consists in the fact that the arguments must be looked for in very large sets or collections, containing a very small portion of "appropriate" ones, if any. As a rule, moreover, the set of potential candidates is not structured in a way enabling to simplify substantially the seeking for "appropriate" arguments, hence, nothing better than the blind exhaustive searching can be recommended, at least within the framework of classical deterministic and sequential algorithms (cf. Chapters 2 and 3 below). As a classical example let us consider the well-known *modus ponens* deduction rule, if A and $A \rightarrow B$ ("A implies B", "if A then B") hold, the the formula B holds as well. Having already given or found the formulas A and $A \rightarrow B$, it is not difficult to verify that the later formula has the form of implication, that the antecedent of the latter formula $(A \rightarrow B)$ is identical with the former one (A), if considering formulas as finite sequences of elementary symbols (letters), finally, to "cut" (hence, "cut rule" in English for modus ponens) the consequent B of the latter formula and to joint it with the set of formulas already known to be valid. Another problem is crucial: having been given a formula B as a hypothesis, which is to be verified, or as a question which is to be answered by the knowledge system, how to find within the set of formulas having been already verified or proved, such a formula A that the formula $A \rightarrow B$ is also within this set (or how to prove that such an A does not exist). When the investigated set is very large and when no rule exists or is known which could help as a hint or heuristic during the searching process (and this is typical for the modus ponens rule), the searching problem is very difficult. If the extent of the checked set quickly (say, exponentially) increases with the size or complexity of the original task, the exhaustive searching is practically intractable.

A similar situation occurs when considering the well-known *resolution principle* for automated theorem proving: two clauses $(A_1, A_2, \ldots, A_n, C)$ and $(B_1, B_2, \ldots, B_m, \neg C)$ where $\neg C$ stands for "negation of C", are to be replaced by single clause $(A_1, \ldots, A_n, B_1, \ldots, B_m)$, and it is very simple supposing the two former clauses are given or known. What makes troubles is to find two clauses of this form in a large set of the already proved clauses (in the case of the propositional calculus). Still more difficult problem is that of finding two clauses together with a substitution for free indeterminates of these clauses which converts them into formulas to which the resolution rule can be already applied (in the case of predicate calculus). Last but not least, the problem to find a substitution for indeterminates occurring in formulas of PROLOG language enabling to match the resulting formula with one already being in the database, belongs also to this class of tasks. The last two examples, i.e. resolution principle and PROLOG, are interesting and important also because of the fact that they demonstrate the possibilities as well as the limits of possibilities of practical computer implementation of tasks leading to searching problems in large databases.

Now, we may already close the first part of this introductory explanation, having already created a sufficient background to turn our attention back to the notion of parallel probabilistic algorithms. On an informal level, parallel algorithms can be described as computational programs, during the implementation of which two or more computations run simultaneously and independently, hence on two or more computational devices, with the further processing of the particular results by another computational device-processor. The most simple and notorically knowr attempt in the direction of parallelization is the bracketing of arithmetical expressions; the expressions within mutually disjoint pairs of brackets can be evaluated independently, hence, simultaneously, in parallel. What matters for the next computation are just the resulting values evaluated inside particular pairs of brackets. Consider an example the triviality of which may seem perhaps too exaggerated. When evaluating the expression $(5 + 4) \times (3 + 6)$, the task to compute $5 + 4$ may be delivered to one processor; another one *simultaneously* computing $3 + 6$, consecutively realizing the final multiplication. Evidently, in a trivial case like this one the loss, following from the necessity to divide the computation among the two processors and then to cummulate their results, significantly exceeds the savings reached due to parallelism. However, taking into consideration a computation consisting in the multiplication of numerical values of two or more define integrals of some hardly integrable functions, the savings of time reached by a parallel evaluation of particular definite integrals on particular processors will become very evident and remarkable.

In the context of our explanation on knowledge systems and keeping in mind the above mentioned separation of the work of inference machine into its "searching" and "computational" parts, the parallelization can be immediately seen to be a very hopeful tool in order to make the work of inference machine substantially more rapid and effective. The testing, whether some elements of the basic space under investigation are "appropriate" arguments of the used decision or deduction rule

or function, in the positive case followed by the computation, deduction or decision making in question, all this can be performed independently, hence, simultaneously for different candidates from the universe of discourse. The limits of this parallelization are given by the extent of this basic universe (theoretically), by the expenses connected with the obtaining of the processors, and by the time or other demands connected with the necessity to ensure the necessary synchronization of their work together with an inspection and cummulation of their results (practical limitations). Some of these problems will be dealt with in the next chapters. So we may already specify the subject of this study by saying that our interest will be focused to parallel algorithms for searching tasks resulting as subtasks during the implementation of inference machines in knowledge systems. The most simple kinds of such algorithms will be investigated in Chapter 4. Chapter 5 deals with their modifications in the case of testing oracles with limited reliability and with generalized loss functions.

When considering some problems solved by knowledge systems, it may be of great importance not only to find appropriate (in a sense) elements in a large set of potential candidates, but also to know the (relative) frequency of these appropriate elements in this set. Also this problem can be solved in parallel, and we shall investigate the degree of soundness of such a parallelization in Chapter 6.

Parallel algorithms can also serve as a tool to introduce a structure in large sets, according to certain criteria, with the aim to make easier their further searching or investigating. In Chapter 7 we shall deal in more details with algorithms introducing a linear ordering in a set with respect to decreasing or increasing values of a numerical or ordinal criterion the values of which are ascribed to the elements of the set in question. Finally, Chapter 8 deals with parallel probabilistic searching algorithm with a certain degree of co-operation among their processors; this co-operation is supposed to be of stochastic nature. To make this survey of the contents complete let us turn back to Chapters 2 and 3. They introduce an appropriate formal apparatus to describe and to handle the notion of algorithm or effective computational procedure in the classical deterministic sense (Chapter 2), as well as their non-deterministic and, namely, probabilistic variants (Chapter 3).

In order to specify, once more, the subject of our interests, let us say that we shall consider only parallel algorithms of probabilistic nature. Probabilistic algorithm can be understood as the usual algorithm with an additional argument (input, parameter), which is obtained as the result of a random sample made from an appropriately chosen sampling space and according to an appropriate probability distribution defined over this space (cf. Chapter 3 for more details). Using an input or parameter sampled at random makes the computation qualitatively more rapid and effective, and with a probability large enough, but not the unit one in general, the result of the computation or decision making is the correct one. The positive probability of an error or failure is, hence, a penalty we have to pay for the pleasure of much more rapid computation. It is a matter of purely extra-mathematical nature to judge, how large probability of error can be taken as acceptable according to the

nature of the problem or computation in question and according to the resulting time, space or other expenses savings. What is important in our context is the fact, that in probabilistic algorithms for searching problems the random sampling of the side input value can be often reduced to a sequence of more simple and statistically independent random samples which can be, therefore, taken in parallel. Also the computation or evaluation following these independent random samples can be, in a more or less degree, divided into a number of independent computations which can be realized, for particular samples or their subgroups, simultaneously and in parallel. Hence, probabilistic algorithms with independent random samples can be considered as important potential outcome when building parallel algorithms and we shall take profit of this possibility several times in the sequel.

So, let us summarize once more: *in what follows, we shall investigate parallel probabilistic algorithms for searching over large database collections, for computation or estimation of numerical characteristics of such collections and for introducing a structure into such collections, with the aim to arrive at substantial simplifications of searching subprograms of inference machines in knowledge systems.* Of course, such a specification is very far from giving an exhaustive list of all possibilities of applications of parallelism and parallel algorithms. Let us mention, by the way the possibilities of parallelization in order to make more effective some matrix operations (multiplications and inversions of matrices). However, including parallel algorithms of this and similar kinds into the scope of our considerations, we would cause this study to be intractably large and we would also menace its thematical homogenity. It is why we have decided to limit ourselves to the domain described above and during all this study we shall consider ourselves being kept by this promise.

2. MATHEMATICAL MODELS OF CLASSICAL ALGORITHMS

In this chapter we would like to show, how the notions of effective computability and decidability can be formalized and investigated within the framework of an appropriate mathematical apparatus. Or, developing theoretical foundations of nondeterministic, parallel and probabilistic algorithms, we shall often use as building stones the notions and constructions offered by the classical theory of deterministic and sequential algorithms. To tell the truth, we shall do so because of the simple fact that no other, principally different conception of nondeterminism is at hand, at least not at a level of a sufficient and detailed enough mathematical formalization. It is why this chapter is devoted to a very brief recapitulation of basic notions of the classical theory of algorithms. Doing this, we shall take profit of the excellent Davis' monograph [3], not aiming, of course, to duplicate it neither in the extent nor in the degree of mathematical perfection. On the other hand, we do not feel as the best solution to replace all this chapter by giving a simple reference to [3], as we would like to offer to the reader at least the first insight, even if not completely

formalized, into the notions and ideas in question. So, this chapter should be taken as something like a compromise between the two extremities, in spite of all the problems involved by all compromises in general. It is also why we do not separate and formalize, in this chapter, explicitly definitions, statements and informal comments.

During a very long period, in fact, since its antic origins till the end of the last century (and still later in some applications), the intuitive and informal conception of effectiveness and effective computability has seemed to be quite satisfactory for mathematics. It was the intuition and erudition of mathematicians who was believed to be able to distinguish an effective computation from a noneffective one case by case without referring to a general theory of effective computability. The crisis in mathematics involved by the occurrence of paradoxes in set theory resulted in the Hilbert's program of remedy of mathematics through their re-formulation into a finitely axiomatizable deductive theory with formally described, exclusively combinatorial and, hence, semantically irrelevant and meaningless deduction rules. The failure of this conception and its principial nonrealizability was proved by Gödel's results from the beginning of the fourth decade of this century. They implied also, as far as the computation theory is concerned, that a strict and purely combinatorial definition of the notion of effective calculation is necessary in order to assure the soundness and consistency of proposed computational structures. But, on the other side, it plays a role of a principial limitation defending to suggest a universal system enabling to realize all intuitively valid derivations "under one roof". The occurrence of first computers, in a rather short time distance, offered also another, quite practical motivation of building a formalized theory of computations. Or, only the computations formalized in such a way can be, at least potentially, realized by an appropriate technical device. From the fourth decade of 20th century originates also the conception of Turing machine which we introduce here as one of a number of mathematically equivalent formalizations of the notion of "effective computation". This choice is caused by our personal belief that this conception in the best way joins the demand of mathematical perfection with the possibility of an informal and intuitive interpretation.

Like other models, the notion of Turing machine is built recurrently and inductively; some elementary operations are claimed to be effective by definition (at the formal level) and by repealing to their evident effective realizability (at an informal level). Similarly, some ways how to combine operations are proclaimed to be effective in the sense that the application of such a way of combination to effective operations results in another effective operation. In more details, Turing machine can be described as follows.

Consider an infinite, both-sides unlimited *tape* divided into infinite number of *boxes*. Every box can be either empty (in other words said, it contains a symbol b = *blank*), or it contains just one symbol or letter of a finite nonempty *alphabet* A, an alphabet with a single letter suffices. The "active" part of Turing machine is called

head and in every moment (the work of the machine is supposed to be executed in a series of discrete moments) the head is in one of *internal states* $q_0, q_1, q_2, \ldots$, moreover, the head is situated just over one box of the tape and "reads" its contents. Depending on the internal state of the head and of what is read, the machine chooses among the following actions.

(1) it erases the contents of the read box and inscribes in it either a letter from A or a "blank" (i.e. the box is empty). Specially, the contents may be unchanged – "rewritten by the former contents", in formal. Then the head changes its internal state, the new state may be, of course, in some cases identical with the former one. A formal description of this action can be given by a quadruple $\langle q_i S_j S_k q_l \rangle$, where q_i is the original and q_l the resulting state, and S_j (S_k, resp.) is the former (the newly inscribed, resp.) contents of the read box. Hence, the quadruple $\langle q_i S_j S_j q_i \rangle$ corresponds to an "empty action" or non-activity, the quadruple $\langle q_i S_j S_j q_l \rangle$, $l \neq j$, deals with the case when the machine, having read the box under inspection, changes its internal state without changing the contents of the box. When reading the same box once more, the machine may, of course, choose another action due to the foregoing change of its internal state.

(2) having read the contents of the box, it is left without any change, but the head itself moves to the *right* to be placed under the neighbour box, moreover, the head changes its internal state (again, of course, with a special possibility not to do so). As a formal counterpart of this action may serve the quadruple $\langle q_i S_k R q_l \rangle$, where q_i (q_l, resp.) is the original (the resulting, resp.) state of (the head of) the Turing machine and S_k is the original and unchanged contents of the box read at the beginning of the action.

(3) an operation analogous to the just described one, but with the move of the head of the machine to the *left*, as a formal description may serve the quadruple $\langle q_i S_j L q_l \rangle$.

From the formal point of view, Turing machine is completely determined by a finite set of quadruples of the types introduced above supposing that the consistency of application of the machine is assured. It is why we must demand, when defining a Turing machine, that there is no pair of quadruples inside the corresponding set of quadruples, which would agree in the first two items but which would differ in the other two items, i.e. a pair of quadruples $\langle q_i S_j x_1 x_2 \rangle$, $\langle q_i S_j y_1 y_2 \rangle$ with $\langle x_1 x_2 \rangle \neq \neq \langle y_1 y_2 \rangle$. This condition assures that, no matter which the internal state and the content of the read box may be, *at most one* quadruple is applicable.

An *instantaneous description* of the Turing machine is the expression $\alpha q_j \beta$, where α and β are finite sequences consisting of the letters of the alphabet A or of b's (blanks), formally, $\alpha, \beta \in (A \cup \{b\})^*$. The interpretation is as follows: when the instantaneous description is $\alpha q_j \beta$, the machine is in the internal state q_j and in the read box occurs the first (the leftmost) symbol of the sequence β. The other symbols from β are inscribed, sequentially, in the next boxes to the right from the read one. All other

boxes till more to the right, not occupied by the symbols from β are supposed to be empty. Similarly, the sequence α describes the contents of the boxes situated to the left from the read one with the last (rightmost) symbol of α inscribed into the box immediately preceding the read one. All boxes more left from those occupied by symbols from α are supposed to be empty. Evidently, this definition covers only cases when only a finite number of boxes are occupied by letters from A, but because of the fact that we shall investigate only configurations resulting from finite words over the alphabet A by a finite number of operations, we may accept this definition as sufficient. Its *non-constructive* generalization to the case with $\beta \in (A \cup \{b\})^{\infty}$ and α taken as an inversely written word from the same set would not be too difficult.

Now, supposing the quadruple $\langle q_i S_j X q_l \rangle$ is applicable to the instantaneous description $\alpha q_S \beta$, i.e., if $q_i = q_S$ and $\beta = S_j * \beta^*$, this application uniquely determines the new (resulting) instantaneous description $\alpha' q_l \beta'$, where

(a) if $X = S_k$, then $\alpha' = \alpha$ and $\beta' = S_k * \beta^*$,

(b) if $X = R$, then $\alpha' = \alpha * S_j$ and $\beta' = \beta^*$.

(c) if $X = L$, then $\alpha' = \alpha^*$ and $\beta' = S_n * \alpha$, where $\alpha = \alpha^* * S_n$.

In all cases, $*$ denotes the concatenation operation. So we may leave, from this moment on, any intuitive images of "head", "tape" and "internal state", taking Turing machine as a consistent finite set of quadruples of the form $\langle q_i S_j X q_l \rangle$, $X \in \{A \cup \{b\}, R, L\}$ which defines a partial mapping of the space of instantaneous descriptions into itself. Supposing an original instantaneous description φ_0 is given, the application of the Turing machine to this description can yield the three following results.

(a) no quadruple from the set of quadruples defining the machine ψ in question is applicable to φ_0, hence, the work of the machine over φ_0 terminates before it started with the result $\tilde{\psi}(\varphi_0)$ defined by φ_0.

(b) there exists a sequence $\varphi_0, \varphi_1, \ldots, \varphi_n$ such that $\psi(\varphi_i) = \varphi_{i+1} \neq \varphi_i$ for $i = 0, 1, \ldots, n-1$, and no quadruple is applicable to φ_n, i.e., $\psi(\varphi_n)$ is not defined. In such a case the machine ψ stops (terminates its work) after n steps with the result $\tilde{\psi}(\varphi_0) = \varphi_n$.

(c) there exists an infinite sequence $\varphi_0, \varphi_1, \varphi_2, \ldots$ such that $\psi(\varphi_i) = \varphi_{i+1} \neq \varphi_i$, $i = 0, 1, \ldots$, then $\tilde{\psi}(\varphi_0)$ is not defined.

The just introduced apparatus of Turing machines can be simply used in order to define and describe the effective computations of functional values. Let us introduce just the case of functions defined in the set of non-negative integers $\mathfrak{N} = \{0, 1, 2, \ldots\}$ and mapping this set into itself, as the generalization to other countable argument and value spaces is merely a matter of technical routine. For $n \in \mathfrak{N}$, let $\tilde{n}$ be a code of n in the alphabet A. The coding is supposed to be fixed, e.g. if $A = \{|\}$, then $\tilde{n} = |^{n+1}$, i.e. $n+1$ strokes, if card $A \geq 2$, then $\tilde{n}$ is the card A-adic code of n, etc. The Turing machine ψ begins to work over the (conventionally stated) instantaneous description $bq_0\tilde{n}$ and the function $f_\psi(n)$ is defined as follows: if $\tilde{\psi}(bq_0\tilde{n})$ is defined and $\tilde{\psi}(bq_0\tilde{n}) = bq_i\tilde{m}$ for some q_i and some $m \in \mathfrak{N}$, then $f_\psi(n) = m$, otherwise,

$f_\psi(n)$ is undefined. The definition can be immediately generalized to functions defined over vectors of non-negative integers; if the argument is a k-tuple $\langle n_1, n_2, \ldots, n_k\rangle$ of non-negative integers, the original instantaneous description is of the form $bq_0\tilde{n}_1b\tilde{n}_2b\tilde{n}_3b \ldots b\tilde{n}_k$ and the value $f_\psi(n, n_2, \ldots, n_k)$ is defined by $\langle m_1, m_2, \ldots, m_k\rangle$ if and only if

$$\tilde{\psi}(bq_0\tilde{n}_1b\tilde{n}_2b \ldots b\tilde{n}_k) = q_i\tilde{m}_1b\tilde{m}_2b \ldots b\tilde{m}_k \tag{2.1}$$

for an internal state q_i. Through a simple transformation and without any loss of generality we may achieve, that in case of halting the machine is situated in a specific "final" or "terminal" internal state. A function f: $\mathfrak{N} \to \mathfrak{N}$ (f: $\mathfrak{N}^k \to \mathfrak{N}^k$, resp.) is called *effectively computable* or *partially recursive*, if there exists Turing machine ψ such that $f = f_\psi$, the adjective "partially" reflects the fact that f need not be defined for each argument value from $\mathfrak{N}$ or $\mathfrak{N}^k$.

The construction of Turing machines together with the following definition of the class of effectively computable functions was described, above, as it is just this class of functions which is accessible for computers or other technical devices. Hence, the description of a computation or decision problem in the form of function effectively calculable by a Turing machine is a necessary condition for its computer solvability. It may be taken as quite intuitive to take a function computable by a Turing machine as an effectively computable one in the common sense. Or, supposing that $f(n)$ is defined, we are able to obtain this value after a finite number of operations, even if the number of these operations cannot be known a priori, and each of these operations seems to be elementary enough to be taken as intuitively effective. Of course, an open question remains, whether *each* function taken as effectively computable in the intuitive sense can be also computed by a Turing machine. Evidently, a question like this cannot be answered, neither positively nor negatively, within the framework of a mathematical formalism, or it confronts a formal conception with an informal one. On the other hand, only formal conceptions can be compared at a mathematical level, and possibly their identity or difference may be stated. The positive answer to the question just mentioned, i.e. the assumption that the class of functions computable by Turing machines covers *all* functions taken as computable in informal sense, is generally accepted as the so called *Church* or *Church-Turing thesis*. The generality of its acceptance refers rather to practical applicability. As far as the acceptance of this thesis at level of theoretical argumentation is concerned, e.g. when replacing a technically difficult routine construction of a Turing machine for a given function by simply referring to the intuitive computability of this function, the positions of various authors differ. Because of the orientation of this text toward practical algorithms, and due to the fact that the greatest part of work which will be mentioned or referred to in what follows accept the Church thesis, we shall accept it as well.

The reader who does not believe or accept the Church thesis, evidently cannot be persuaded or forced to do so on the ground of a purely mathematical argumenta-

tion and she or he may refuse all proofs based on this thesis by ascribing them just the status of an informal hint or heuristic. This position is defended, e.g. by Davis himself in [3], who introduces Church thesis, but does not take profit of it, giving careful and detailed constructions of all Turing machines necessary for his explanations. From the point of view of pure and correctly formalized mathematics it is quite a legitimate position and the main argument against it is, and necessarily must be, again of an informal and heuristic nature. Namely, there has not been given or discovered, till now, an intuitively computable function not computable by an appropriate Turing machine. Nevertheless, even when accepting the Church thesis, we must always keep in mind its special nature and the serious consequences following from such a decision.

As an important practical argument in favour of Church thesis let us mention the well-known fact that there are several. independently developed, formalizations of the notion of "effectively computable function". These conceptions are equivalent in the sense that the class of functions declared to be effectively computable is the same and is identical with the class of functions computable by Turing machines. Among the well-known alternative formulations are Markov (or normal) algorithms, Post machines (or grammatics) and partially recursive functions. Let us mention the last ones in more details.

Consider functions ascribing natural numbers to finite sequences of such numbers. *Elementary functions* are, by definition, the following ones:

(a) *constants*, i.e. functions of the form $f(x_1, x_2, \ldots, x_n) = k$, for all n, $k \in \mathfrak{N}$, $n \geqq 1$.

(b) *projection functions*, i.e. functions of the form $f(x_1, x_2, \ldots, x_n) = x_j$ for all $j, n \in \mathfrak{N}, 1 \leqq j \leqq n$.

(c) *successor function* $f(x) = x + 1$. The class of *primitive recursive functions* is then defined as the smallest class of functions containing all elementary functions and closed with respect to the *composition* and *primitive recursion*. Hence, if $f(y_1, \ldots, y_m)$ is a primitive recursive function of m arguments and $g_1(x_1, \ldots, x_n)$, $g_2(x_1, \ldots, x_n)$, ..., $g_m(x_1, \ldots, x_n)$ are primitive recursive functions, each of them with n arguments, then also

$$f(g_1(x_1, \ldots, x_n), g_2(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \tag{2.2}$$

is a primitive recursive function of n arguments (composition). If $f(x_1, \ldots, x_n)$ and $g(x_1, \ldots, x_n, y, z)$ are two primitive recursive functions, then the function h of $n + 1$ arguments, defined by

$$\begin{aligned} h(x_1, x_2, \ldots, x_n, 0) &= f(x_1, x_2, \ldots, x_n)\,, \\ h(x_1, x_2, \ldots, x_n, y + 1) &= g(x_1, \ldots, x_n, y, h(x_1, \ldots, x_n, y))\,, \end{aligned} \tag{2.3}$$

is also primitive recursive (primitive recursion).

As can be proved in a rather routine matter, each primitive recursive function is computable by a Turing machine (the corresponding construction having been done,

e.g., by Davis in [3]). However, the class of primitive recursive functions can be easily seen not to cover the class of all functions computable by Turing machines, e.g., because of the fact that each primitive recursive function of n arguments is defined on the whole space $\mathfrak{N}^n$. So partial functions, undefined for certain n-tuples from $\mathfrak{N}^n$, cannot be primitive recursive, but some functions of this kind can be defined and computed by Turing machines (within their definition domains). As an example of total, effectively computable, but not primitive recursive function we may take the well-known Ackermann (or Ackermann-Peter) function (cf. [2], e.g.). So we add, to the rules already accepted, a new *minimization rule.* A function h of $n-1$ arguments $x_2, x_3, \ldots, x_n$, $n \geqq 2$, is called to be obtained by minimalization from a function f of arguments $y, x_2, \ldots, x_n$, if

$$h(x_2, x_3, \ldots, x_n) = \min \{y: y \in \mathfrak{N}, f(y, x_2, \ldots, x_n) = 0\} \tag{2.4}$$

supposing that for $x_2, x_3, \ldots, x_n$ in question such an y exists, if it is not the case, $h(x_2, \ldots, x_n)$ is not defined. Now, the class of *partially recursive functions* is the smallest class containing all primitive recursive functions and all functions obtained when the minimalization rule is applied to a primitive recursive function. *Just one* application of this rule can be proved to be sufficient to obtain the class in question. As already mentioned above, the class of partially recursive functions is identical with the class of functions computable by Turing machines. Accepting the Church thesis, it is identical also with the class of intuitively computable functions.

The formulation presented above could involve an idea or impression that the apparatus of Turing machines as introduced is not general enough in the sense that each effectively computable function requests its own "particular" Turing machine, "independent" of the machines enabling to compute other functions. The notion of *universal Turing machine* proves this idea not to be quite correct. For example, let us consider Turing machines which compute functions of one argument. From the formal viewpoint they are nothing else than finite sets of quadruples, hence, finite sequences of symbols. So, they can be effectively ordered (lexicographically, say), enumerated, and their order numbers (indices) can be encoded in an appropriate alphabet A, e.g., by the word having the same index in the fixed lexicographical ordering of the free monoid $A^* = \bigcup_{n=0}^{\infty} a^n$. There exists a Turing machine over the alphabet A, which works as follows: given $x \in \mathfrak{N}$, it decomposes x into a pair $\langle x_1, x_2 \rangle \in \in \mathfrak{N} \times \mathfrak{N}$ using a fixed effective decomposition rule. Then the machine takes x_1 as the index of a Turing machine, namely ψ_{x_1} with respect to the defined ordering, it generates the set of quadruples corresponding to ψ_{x_1} and, finally, applies ψ_{x_1} to the argument value x_2 and computes $\psi_{x_1}(x_2)$. Hence, this "universal" Turing machine can "simulate" the computation of a machine ψ over an argument value x by working over argument value $y(\psi, x)$ defined in such a way that its fixed decomposition yields the index (often called Gödel number) of ψ and the argument value over which ψ is to operate. Informally, $y(\psi, x)$ corresponds to the input se-

quence for a computer, which the computer itself divides into *program*, i.e. the code of the algorithm which is to be executed and which is written in a special alphabet called *programming language*, and into *data*, i.e. values which are to be substituted for the free indeterminates of the program and to which the program is to be applied. Let us notice the fact that from the highly theoretical and abstract notion of (universal) Turing machine we have arrived very close to the realistic and highly practical notion of computer. Otherwise said, universal Turing machine can be seen as an idealization of computers, abandoned of the limitations resulting from the fact that each computer is a physical finite system (device). Still briefly said: universal Turing machine is a usual computer with unlimited memory and unlimited execution time (unlimited means infinite in the potential sense).

The last phrase seems to re-capitulate everything important, what the practically and rather toward applications oriented reader should keep in mind when reading the next chapters. In orientation toward a reader whose interests are more mathematical and theoretical, the aim of this chapter was to show, in which sense the notions like "algorithm", "algorithmical", "effectively computable" or "effectively decidable", etc. are to be understood. In what follows, these expressions will be used very often as basic, elementary and no more analyzed building stones of our further considerations and constructions. The explanation presented in this chapter has been, as already mentioned, taken from the already classical Davis' monography, up to a notion of oracle which is introduced in the next chapter. Several most important monographies or textbooks dealing with the classical theory of algorithms and attainable in our conditions are listed below, cf. [1]–[7].

REFERENCES

[1] M. A. Ajzerman: Logika, automaty, algoritmy (Logic, Automata, Algorithms — in Czech). Academia, Prague 1971.

[2] C. Calude: Theories of Computational Complexity. North Holland, Amsterdam 1988.

[3] M. Davis: Computability and Unsolvability. McGraw-Hill, New York 1958.

[4] Z. Manna: Mathematical Theory of Computation. McGraw-Hill, New York 1974. Czech translation: SNTL, Prague 1981.

[5] J. Mikloško and V. E. Kotov: Algorithms, Software and Hardware of Parallel Computers. Springer-Verlag, Berlin and Veda, Bratislava 1984.

[6] H. Rogers: Theory of Recursive Functions and Effective Computability. McGraw-Hill, New York 1967. Russian translation: Mir, Moscow 1972.

[7] K. Wagner and G. Wechsung: Computational Complexity. VEB Deutscher Verlag der Wissenschaften, Berlin 1986.

3. MATHEMATICAL MODEL OF NONDETERMINISTIC, PARALLEL, PROBABILISTIC AND BAYESIAN ALGORITHMS

Considering once more, and from a certain distance point of view, the notion of algorithm together with the way in which this notion has been formalized, three basic attributes of the classical paradigma of theory of algorithms, and computational processes in general, arise. A step by step omitting of these attributes will bring us to non-classical conceptions of algorithms which will be the main subjects of our investigations in this chapter.

(a) *Correctness and reliability of the result.* Supposing an algorithm pretends to the role of tool for computation of a function f over its definition domain D, then this algorithm must for every value x from D, this value being introduced on the input of the algorithm, produce in a finite time the value $f(x)$ as the corresponding output. In case the function f takes its values in a continuous set of real numbers, the algorithm must be able to produce, in a finite time and given $x \in D$ and natural number $n \geqq 1$ as input values, a value y which differs from $f(x)$ by less than $1/n$. No other incorrectness or unreliability of the result is admitted. If, in spite of this demand, the computation uses a procedure or device charged by such an incorrectness or unreliability, such an approximation can be justified only by reason of utiliary and extra-mathematical nature. These reasons, if applied, are in a position of ontologically independent side inputs and cannot be defended within the framework of classical theory of algorithms and computational processes.

(b) *Sequential nature of the algorithm work.* No matter how elementary or complicated (in an informal sense) may be the operations declared as atomic and non-analyzed during the description and implementation of the algorithm, the algorithm is always executed as a linear sequence of operations. Hence, when executing an operation – a member of a sequence of operations, all foregoing operations are supposed to be already executed and their results to be known and to be at the disposal of the algorithm. The evaluation or estimation of computational complexity of the algorithm (of the corresponding computational process, resp.) is also based on the sequential understanding. Even in case some operations are executed in parallel at the level of their practical hardware realization, this fact is not taken into consideration at the theoretical level. Besides other reasons, this sequential conception reflects the classical paradigma of mathematical work taken as an individualistic intellectual creative activity or effort of human subject which cannot share his work with somebody else by an appropriate co-ordination of their efforts.

(c) *Worst-case analysis – the minimax criterion of quality of algorithms.* This attribute is very close to that one introduced ad (a) above and consists in the fact that the quality of algorithm which computes the values of a function f over its domain D is uniquely determined by that case, i.e., by that value $x \in D$, which is the worst from the viewpoint of the criterion in question. E.g., the time, space or in other way quantified demands connected with the computation of $f(x)$ are, in

this case, the highest ones. It does not matter how "larger" or "important" subset in D is represented by these extremum cases, it does not matter as well, whether these cases are "typical" or "non-typical", or even "pathological" from the point of view of an intended practical application of the algorithm. Supposing that these extremum cases cannot be avoided from considerations by an appropriate re-definition of the function in question (by an explicit narrowing of its definition domain, e.g.), we have to accept, when accepting the classical paradigma of algorithm theory and computational complexity theory, the decisive role of extremum cases when classifying the qualities of algorithms and computational processes.

We shall not take into consideration, at the moment, a number of practical and extra-mathematical reasons for which the three principles introduced above cannot be defended, or at least cannot be taken as rational, in many practical situations. Some considerations of this kind can be found in [5] and we shall discuss them in the following chapters when introducing some particular classes of non-classical algorithms. The aim of this chapter, however, is to present appropriate theoretical foundations for nondeterministic algorithms general enough to cover all the classes of algorithms mentioned in the title of this chapter. Moreover, we would like to make clear in which sense the non-classical algorithms can be understood as special cases of nondeterministic algorithms. The notion "nondeterministic algorithm" will be used throughout this work in spite of the fact that some authors (cf. [10], e.g.) refuse it in principle as "lapsus linguae" or "contradictio in adjecto", i.e., as a contradiction, postulating, in this way, determinism as an attribute of algorithmicity. However, it should be clear, in what follows, when this term is used in an informal and rather alegorical sense, when it is used in a formal sense, and which is its precise formalization in the case in question.

Let us briefly survey some possibilities how to introduce nondeterminism into our considerations. Because of the chosen way of explanation the shortest way to nondeterministic models of computational processes goes through the notion of *nondeterministic Turing machines*. Such a machine is defined, again, as a finite set of quadruples of the form $\langle q_i S_j X q_l \rangle$, $X \in A \cup \{R, L\}$, but this time without the consistency condition. Hence, the set of quadruples may contain, simultaneously, $\langle q_i S_j X q_l \rangle$ and $\langle q_i S_j Y q_r \rangle$ with $\langle X, q_l \rangle \neq \langle Y, q_r \rangle$. This definition is as close as possible to the formal model introduced above, even if it is, maybe, not too intuitive; an equivalent and better known formulation will be given later. The disadvantage of this model consists in the evident fact, that without a decision rule solving the problem which among two or more quadruples applicable in an instant will be actually applied, the model admits only a "parallel" interpretation. It means that the computational process splits into two or more simultaneously executed branches each of them starting by the application of one of the applicable quadruples. As we shall consider as useful, for the sake of our further explanation, to be able to individualize particular branches of computation and to parametrize them appro-

priately by elements of a parameter space, we shall not take nondeterministic Turing machines as our outcoming point in what follows.

Another model of nondeterminism is introduced in [2] under the notion of A_0-*Turing machine*, where $A_0 \subset \mathfrak{N} = \{0, 1, 2, \ldots\}$ is a subset of the set of all natural numbers. A_0-Turing machine is a finite set of quadruples of the form $\langle q_i S_j X q_l \rangle$ satisfying the consistency condition as in the foregoing chapter, however, with $X \in A \cup \{R, L\} \cup \{q_0, q_1, \ldots\}$. The quadruples $\langle q_i S_j S_k q_l \rangle$, $\langle q_i S_j R q_l \rangle$ and $\langle q_i S_j L q_l \rangle$ have the same interpretation as above, the quadruple $\langle q_i S_j q_k q_l \rangle$ means: if the internal state of the (head of) the Turing machine is q_i and the read box on the tape contains S_j, verify, first, whether the instantaneous description on the tape corresponds to a natural number with respect to a chosen and fixed way of coding of natural numbers in the alphabet A. If it is the case and if this number is in A_0, the new internal state of the machine will be q_k, its new state being q_l otherwise. In both the cases, the contents of the tape is left without any change. Informally, A_0-Turing machine contains an oracle which is able, in each step, to answer, whether the natural number, the A-code of which the machine has obtained, belongs to A_0, or not, and the further work of the machine depends on this answer. The class of functions computable by A_0-Turing machines is the class of A_0-*partial recursive functions*. This class of functions is defined in the same way as the class of partial recursive functions in the foregoing chapter with the only exception that the characteristic function (identifier) χ_{A_0} of the set A_0 is classified as an elementary function (together with constants, projection function and successor function). Here $\chi_{A_0}: \mathfrak{N} \to \{0, 1\}$, $\chi_{A_0}(n) = 1$ iff $n \in A_0$. Evidently, if A_0 is a recursive subset of $\mathfrak{N}$, i.e., if χ_{A_0} is a total (for each $n \in \mathfrak{N}$ defined) recursive function, then the class of A_0-partial recursive functions agrees with the class of partial recursive functions. If A_0 is not recursive, the class of partial recursive functions is substantially enriched (at least by the function χ_{A_0}).

As can be easily seen, the same behaviour and result of the machine can be achieved by considering another, side input of the Turing machine in question. The values of this side input are 0 or 1 and the next run of the computation depends on these values. So we have arrived at the model which will play the basic role in our further considerations.

Consider a function f defined on a subset D_f of a set $\mathscr{A}$ and taking its values in a set $\mathscr{B}$. Suppose that the elements of the sets $\mathscr{A}$ and $\mathscr{B}$ can be encoded (enumerated) in an effective one-to-one way, by natural numbers or finite sequences of natural numbers. So we may speak, after all, about (partial) recursiveness of the function f or other functions defined through f. Let $\mathscr{V}$ be a nonempty set the elements of which can be, again, encoded by natural numbers or finite sequences of such numbers. A function G_f taking the Cartesian product $\mathscr{A} \times \mathscr{V}$ into $\mathscr{B}$ is called a *nondeterministic algorithm for (computation of the function)* f, if the function G_f is partial recursive and if there exists, for each $x \in D_f$, at least one $y \in \mathscr{V}$ such that $G_f(x, y) = f(x)$.

The demand that there exists at least one value $y \in \mathscr{Y}$ enabling to compute the value $f(x)$ through the computation of the value $G_f(x, y)$, is rather weak. So, the practical use of the notion of nondeterministic algorithms conceived in this way depends on the three following factors:

(a) on the accessibility of a value $y = y(x) \in \mathscr{Y}$ such that $G_f(x, y) = f(x)$, given $x \in \mathscr{A}$, or on the achievability of an oracle which is able to generate such a value;

(b) on the degree in which the use of an auxiliary value y reduces the computational complexity (suppose, for a moment, that this notion has been already appropriately defined) for $G_j(x, y)$, if compared with the computational complexity of $f(x)$, if f is partially recursive and can be computed directly, i.e. not through $G_f(x, y)$;

(c) on our abilities to recognize the value $f(x)$ among values $G_f(x, y)$ for various y's supposing we are not sure that a given $y \in \mathscr{Y}$ is such that $G_j(x, y) = f(y)$.

Hence, an application of a nondeterministic algorithm G_f as a tool for computation of values of the function f can be seen as prospective in two cases. Either, in case the set $V(x) \subset \mathscr{Y}$ of auxiliary values giving a successful computation of $f(x)$ through $G_f(x, \cdot)$, i.e. the set

$$V(x) = V(G_f, x) = \{y: y \in \mathscr{Y}, G_f(x, y) = f(x)\} \tag{3.1}$$

is a sufficiently large or prevailing majority of the set $\mathscr{Y}$ constituting subset of the set $\mathscr{Y}$. If this sufficient cardinality of $V(x)$ is defined in such a way that, taking an appropriately defined random sample from the set $\mathscr{Y}$, with a probability sufficiently close to one an element from $V(x)$ is sampled, we arrive at the notion of probabilistic algorithm (cf. below for more details). The other situation when nondeterministic algorithms may be of practical use is, that the set $\mathscr{Y}$ is relatively small and it is within our powers to compute simultaneously (in parallel), using a number of identical computational devices (processors), the values $G_f(x, y)$ for all $y \in \mathscr{Y}$, in the optimal case for each particular value of y on a separate processor. Now, of course, the demand ad (c) above is of importance, as we must be able to recognize which of the processors has computed the correct and desired value of $f(x)$. This approach leads to the idea of parallel algorithms which are also described and investigated below in more details. Perhaps, the reader may be surprised that it is as late as now when the notion of parallel algorithms, even if contained in the title of this work, for the first time comes into the scene. Briefly said, we have preferred, in what has been already told, first of all to describe motivations and ways of reasoning leading to parallelism, postponing the introduction of the parallel algorithm till the time when we are able to offer not only an intuitive description with a non-negligible danger of misinterpretation, but also a certain formalized background. Moreover, our intention is to pick up the difference between more theoretically conceived non-deterministic algorithms, when the inspection of outputs is outside the scope of the algorithm, and practically taken parallel algorithms when this inspection cannot be negliged.

Now, we may specify probabilistic algorithms in details as a particular subclass of the class of nondeterministic algorithms. As we have already noted, the specifica-

tion consists in the fact that instead of simple non-emptiness of the set $V(G_f, x)$ for all x from the domain of f this set will be requested to be "sufficiently large". If the set $\mathscr{V}$ is finite, the relative frequency of elements from $V(x)$ in $\mathscr{V}$, i.e. the ratio card $V(x)$/card $\mathscr{V}$ of cardinalities of both the set could be considered as a criterion. However, for an infinite set $\mathscr{V}$ or for more complicated or sophisticated sampling models (mechanisms), which will be investigated below, this simple quantitative criterion can be seen or easily proved to be unsatisfactory. It is why we shall consider, from the very beginning, a more a general approach based on the abstract measure theory. The basic notions of measure theory and probability theory can be found, e.g., in [3], [4], [8], [9], or elsewhere.

A system $\mathscr{B}_0$ of subsets of the set $\mathscr{V}$ is called a *sigma-field* supposing it is closed with respect to the set-theoretical operations of difference and countable union. Hence, if E, F are sets from $\mathscr{B}_0$, then also $E - F = \{x: x \in E, x \notin F\}$ is in $\mathscr{B}_0$ and if $E_1, E_2, \ldots$ is an infinite sequence of sets from $\mathscr{B}_0$ then also their union $\bigcup_{i=1}^{\infty} E_i$ is in $\mathscr{B}_0$. A function μ mapping $\mathscr{B}_0$ into the unit interval $\langle 0, 1\rangle$ of real numbers is called a *probabilistic measure* defined on $\mathscr{B}_0$, if it is a non-negative, normed and sigma-additive function on $\mathscr{B}_0$. Hence, $0 \leqq \mu(E) \leqq 1$ for each $E \in \mathscr{B}_0$, $\mu(\mathscr{V}) = 1$, and for each infinite sequence $E_1, E_2, \ldots$ of mutually disjoint sets from $\mathscr{B}_0$ we have $\mu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=0}^{\infty} \mu(E_i)$. As the most simple and, in a sense, extremum cases of sigma-fields defined over $\mathscr{V}$ let us introduce the two-element sigma field $(\emptyset, \mathscr{V}\}$ ($\emptyset$ being the empty subset of $\mathscr{V}$), and the sigma-field $\mathscr{P}(\mathscr{V})$ (sometimes denoted by $2^{\mathscr{V}}$) of all subsets of the set $\mathscr{V}$.

So we may define *probabilistic algorithm for a computation of a function* f: $\mathscr{D}_f \to \mathscr{B}$, $\mathscr{D}_f \subset \mathscr{A}$ *with the probability of error uniformly majorized by a positive value* ε *and with respect to a set* $\mathscr{V}$ *of side inputs, a sigma-field* $\mathscr{B}_0 \subset \mathscr{P}(\mathscr{V})$ *and a probabilistic measure* μ *defined on* $\mathscr{B}_0$. In what follows, the expression will be substantially abbreviated supposing the parameters are evident from the context. It is defined as a total recursive mapping from the Cartesian product $\mathscr{A} \times \mathscr{V}$ into $\mathscr{B}$ and such that, for each $x \in D_f$, the set $V(G_f, x)$ $(= \{y: y \in \mathscr{V}.\ G_f(x, y) = f(x)\})$ is in $\mathscr{B}_0$ and $\mu(V(G_f, x)) \geqq 1 - \varepsilon$. Because of the simple fact that only nonempty sets can be of positive probabilistic measure, $\varepsilon < 1$ yields that $V(G_f, x)$ is not empty for each $x \in D_f$. Hence, the definition actually species a subclass of nondeterministic algorithms.

The probability measure μ over $\mathscr{V}$ enables to measure the extent of some (i.e. measurable) subsets of the set $\mathscr{V}$ in an "absolute" sense. Because of the fact that during practical applications of probabilistic algorithms elements of the set $\mathscr{V}$ are accessible just through random samples, we would like to re-define the measure μ on $\mathscr{V}$ in such a way that the values of μ for subsets from $\mathscr{B}$ correspond to probabilities with which the sampled value is in the subset in question. Such a modification can be achieved when introducing the notion of *probabilistic* (random, stochastic) *oracle*.

Any triple $\langle \Omega, \mathscr{S}, P \rangle$, such that Ω is a nonempty set, $\mathscr{S}$ is a nonempty σ-field of subsets of the set Ω and P is a probability measure on $\mathscr{S}$, is called *probability space*, the pair $\langle \Omega, \mathscr{S} \rangle$ is called *measurable space*. As special examples of probability space and measurable space we may consider the structure $\langle \mathscr{V}, \mathscr{B}, \mu \rangle$ and $\langle \mathscr{V}, \mathscr{B} \rangle$ defined above. Probabilistic oracle is a random variable X defined on $\langle \Omega, \mathscr{S}, P \rangle$ and taking their values in $\langle \mathscr{V}, \mathscr{B} \rangle$. This is nothing else than a measurable mapping which takes Ω into $\mathscr{V}$ hence, such a mapping that the image of each set from $\mathscr{B}$ is in $\mathscr{S}$, formally,

$$\{\{\omega: \omega \in \Omega, X(\omega) \in E\}: E \in \mathscr{B}\} \subset \mathscr{S} . \tag{3.2}$$

Now, probabilistic algorithm, which computes a function f with a probability of error uniformly majorized by a value $\varepsilon > 0$ and with respect to a probabilistic oracle X, is a recursive mapping $G_f: \mathscr{A} \times \mathscr{V}^* \to \mathscr{B}$ such that, for each $x \in D_f$,

$$P(\{\omega: \omega \in \Omega, X(\omega) \in V(G_f, x)\}) =$$
$$= P(\{\omega: \omega \in \Omega, G_f(x, X(\omega)) = f(x)\}) \geqq 1 - \varepsilon . \tag{3.3}$$

Evidently, this definition agrees with the one above supposing the probability measure μ is replaced by the measure μ_X defined, for each $E \in \mathscr{B}$, by

$$\mu_X(E) = P(\{\omega: \omega \in \Omega, X(\omega) \in E\}) . \tag{3.4}$$

It is not the aim of this work to investigate, in more details, theoretical and abstract features of probabilistic algorithms and some particular algorithms will be discussed in the next chapters. However, let us turn back, very briefly, to the items a)–c) presented in the beginning of this chapter and let us reconsider them from the point of view of probabilistic algorithms. The limitations imposed by (a) are evidently overcrossed; the computation of the value $G_f(x, y)$ may be wrong, i.e. this value may differ from $f(x)$, the only we need is that the probability of such an error should be small enough for all $x \in D_f$. Using our notation, the item (a) corresponds to the demand $V(G_f, x) = \mathscr{V}^*$ for all $x \in D_f$. However, our conception of probabilistic algorithms respects the demand (c), or, the minimal value ε_0, for which $G_f(x, y)$ is a probabilistic algorithm with uniformly majorized probability of error, is given by the simple relation

$$\varepsilon_0 = 1 - \inf \{\mu(V(G_f, x)): x \in D_f\} . \tag{3.5}$$

Hence, the quality of algorithm is uniquely determined by its worst-case quality, i.e., by the $x \in D_f$ for which the probability measure $\mu(V(G_f, x))$, or the probability of sampling an element from $V(G_f, x)$, are the minimal ones. When calling this idea by *minimax principle* we were inspired by the statistical hypothesis testing theory, where, in case the losses arising from a decision making cannot be minimized uniformly over all sampling space, we must be satisfied when minimizing the potential maximum value of these losses. Nevertheless, the already mentioned statistical hypothesis testing theory offers at least still another solution, known as *Bayes principle*. When applying this principle to our problem how to compute $f(x)$ using

$G_f(x, y)$, and when supposing, for the sake of simplicity, that $D_f = \mathscr{A}$, the situation may be described as follows.

Not only the auxiliary argument y, but also the main argument x, used when $G_f(x, y)$ is to be computed, is supposed to be sampled at random. As the quality criterion for the probabilistic algorithm we take not the worst case, i.e. minimum, but the "average", "typical" one, or, using mathematical terms, the expected value of the probability $\mu(V(G_f, x))$. Formally written, we suppose to have defined a σ-field $\mathscr{C}$ of subsets of the set $\mathscr{A}$ together with a probability measure ν on $\mathscr{C}$. The *Bayes risk* $\varrho(G_f)$ is then defined by

$$\varrho(G_f) = \int \mu(V(G_f, \mathrm{d}\nu)) \,. \tag{3.6}$$

If the set $\mathscr{A}$ is finite or countable, i.e. $\mathscr{A} = \{a_1, a_2, \ldots\}$, then ν is uniquely defined through the values ν_i ascribed to each a_i (to singletons $\{a_i\}$, more correctly), and (3.6) can be rewritten into a more lucid form

$$\varrho(G_f) = \sum_{a_i \in \mathscr{A}} \nu_i \, \mu(V(G_f, a_i)) \,. \tag{3.7}$$

When requesting $\varrho(G_f)$ to be majorized by ε we arrive, clearly, to a substantial weakening of the original demand $\mu(V(G_f, x)) \leqq \varepsilon$ for all $x \in D_f$ ($x \in \mathscr{A}$ in our case). A function $G_f(x, y)$ satisfying the demand $\varrho(G_f) \leqq \varepsilon$ is called *bayesian probabilistic algorithm which computes the function f with probability of error majorized by* ε *and with respect to apriori probabiltty distribution* ν. As can be seen, when replacing probabilistic algorithms with uniformly majorized probabilities of error by their appropriate bayesian modifications, we arrive at a substantial, even exponential speed-up of the computation. So the time needed by the bayesian algorithm is just a logarithmic function of the time needed by probabilistic algorithm with uniformly majorized probability of error, some examples will be presented in the next chapters. However, we must always keep in mind, that the quality of a bayesian probabilistic algorithm ultimately depends on the apriori probability distribution ν in question. E.g., when choosing ν appropriately, we may completely avoid the influence of a given proper subset of $\mathscr{A}$ as far as the quality of the algorithm is concerned. We simply ascribe to this subset (to all its elements, if $\mathscr{A}$ is finite or countable) the zero value of the apriori measure ν. Of course, this is a much more general and philosophically deeper problem of bayesian approach in probability theory, mathematical statistics and in monographs, textbooks and special papers from those domains we can find much more profound and sophisticated discussions going far beyond the intended scope of this work. For our sakes we may be satisfied by the quotation that Bayes algorithms are of use and importance in such cases, when the nature of the solved problem or the domain of intended application yields sufficiently strong and sharp theoretical or practical reasons for an actual choice of the apriori distribution ν, or at least for its limitation to a relatively narrow class of probability distributions over the input arguments, i.e. over particular instances of a more general problem in question. In certain situations, supposing the domain of the function f is finite,

the *Laplace principle* can be considered as acceptable. Roughly said, this principle reads: "the lack of reasons for preferring A to B or B to A is a sufficient reason for taking A and B as equivalent". This leads to the equiprobable apriori distribution. i.e., $\nu(\{a\}) = (\text{card } D_f)^{-1}$ for each $a \in D_f$. In every case, we must keep in mind that no choice of the apriori distribution can be justified by argumentation inside the theoretical model of nondeterministic algorithms. Such a choice will always play the role of an ontologically independent side input with all the philosophical and methodological consequences.

As we have already mentioned, within the framework presented above, parallel algorithms can be understood as actual variants of potentially conceived nondeterministic algorithms. This phrase should be interpreted in such a way that considering nondeterministic algorithm we suppose that just one of the possible variants of the computational process will be actually realized (an oracle decides which of them), with all other alternatives resting in the sphere of potentiality which is taken as qualitatively different from the modus of actual realization. In the case of parallel algorithms all paths are realized simultaneously and through different, separated, but as a rule identical from the viewpoint of their computational abilities, devices (processors). This approach is of practical use namely when the set $\mathscr{V}$ is relatively small, at least if compared with time saving or other advantages following from such a parallelization. A common example is that consisting in separation of a computation into several cases which can be solved independently, hence, simultaneously. This point of view stands rather close to the common understanding of parallel algorithms as implementational realizations of nondeterministic algorithms with nondeterministic algorithms taken, on the other hand, as appropriate theoretical models of parallel algorithms. It follows, from what we have said, that this approach cannot be completely refused, however, its weak point consists in the fact that certain idealizations, perhaps acceptable at the theoretical level, but not justifiable at the level of implementation, are projected into the notion of parallel algorithms. Let us mention them very briefly.

The idea of a pure potentiality of non-realized computational paths leads to a complete neglection of time or other demands connected with a rational and systematic distribution of particular values $y \in \mathscr{V}$ to particular processors which will compute the value $G_f(x, y)$ using the commonly given input x and a program for the function G_f. Similarly, the idealized approach neglects the time and effort necessary to inspect the results yielded by particular processors, to verify, which of them offered the correct result, and to send this result to the main processor in order to output it. In both the cases, the idealization neglects time and other demands following from a non-effectively and existentially formulated part of the definition of nondeterministic algorithms as presented above: "*there exists* $y \in \mathscr{V}$ such that $G_f(x, y) = f(x)$". As a rule, should this problem be explicitly stated, then only at the hardware level, as a technical problem to be solved by proposing an appropriate computer architecture. This approach is supported by the fact that the first variants

of parallel processors were projected as space compact devices of relatively small and negligible dimensions and with a practically instant and faultless connections between various components. However, let us consider a system of computers, separated in a large space and with limited or noised possibilities of communication. Evidently, the problem of inspection and cumulation of results yielded by particular computers can be immediately seen as far from being a trivial one.

Similarly, when accepting some realistic assumptions, another non-trivial problem is that whether two or more processors can share the same item in a database in the same time instant. A much more sharp form of this problem occurs when a processor not only reads the item, but subjects it to some further manipulations e.g., replaces it, either in the computer storage or even in the real world. Suppose, e.g., that the set of objects or items is to be re-arranged with respect to a given criterion. In such a case, of course, the operations of two processors over the same item may be contradictory, hence, not realizable simultaneously. In what follows, this problem will be investigated in more details when re-arranging a given set with respect to the values of a given numerical criterial function and the most simple conflict control strategy will be accepted: supposing two or more processors ask for the same item from the database simultaneously, none of them reaches it, in this step of computation. Even if we could and should consider some finer, less drastic and more flexible conflict control strategies, the unlimited increase of the number of processors (of the degree of parallelization) is evidently seen not to improve the situation. It even makes it worse, so that the optimalization of the number of processors becomes, again, a non-trivial theoretical and practical problem.

So far, we have not mentioned the reasons for which an "indirect" computation of the value $f(x)$ through the computation of $G_f(x, y)$ for all or for some $y \in \mathscr{V}$ should be interesting or desirable in spite of risks (in the case of probabilistic algorithms) or of implementation difficulties (in the case of parallel algorithms) connected with such a decision. In case the function f itself is not recursive, the situation is clear: an appropriate side input offers the only possibility how to compute effectively the value $f(x)$. If there is an algorithm enabling a direct computation of $f(x)$, the reason to choose an "indirect" computation is that it is of substantially lower computational complexity. What does it mean "substantially lower" may be, of course, interpreted in different ways and during a practical application this interpretation will depend on the nature of the problem in question. However, from a rather theoretical viewpoint at least two conceptions, a "weaker" and a "stronger" one are considered. Suppose that we have defined, somehow, the size of the input into the algorithm which computes $f(x)$. Hence, the size of the value x is expressed by natural numbers. E.g., supposing that x is a natural number, its size may be defined by the length of its binary or decadic code, or it may be the number of edges or vertices in a graph, rank of a matrix, cardinality of a set, etc., Let $F(n)$ be an integer-valued function with the property that for each input x of the size n the time computational complexity of the computation of $f(x)$ is majorized by $F(n)$. Let the time compu-

tational complexity for the value $G_f(x, y)$ be majorized by the value $H(n)$ of a function H for all x of the size n and for all $y \in \mathscr{Y}$. The time computational complexity of the function G_f is, in the weak sense, substantially lower than the time computational complexity of f, if $H(n)$ belongs to the $o(F(n))$-class, hence, $\lim_{n\to\infty} H(n)/F(n) = 0$. The time computational complexity of the function G_f is, in the strong sense, substantially lower than that of f, if $H(n)$ belongs to the $\mathcal{O}(\log(F(n))$-class, hence, if there exists $K < \infty$ such that $H(n) < K \log F(n)$ for almost all $n \in \mathfrak{N}$. This latter, stronger conception is important because of the fact that the so called *exponential speed-up*, achieved in this case, is usually inaccessible, for many practically as well as theoretically important tasks, by improving the algorithm within the framework of the classical (sequential and deterministic) paradigma, hence, without accepting nondeterminism and (or) a risk of unreliability of the algorithm. A close connection with the *P-NP problem*, and *NP-completeness* – two attractive and topic domains contemporary computer science is evident. The stronger interpretation of what "substantial reduction of computational complexity" means is sometimes presented in a weakened form demanding that $H(n)$ belongs to $\mathcal{O}((\log F(n))^k)$-class for a fixed $k > 0$. Evidently, this notion still remains substantially stronger than the weak version presented above. However, we shall not solve the problem, how efficient the reduction of computational complexity may be when applying the principles of nondeterminism, at this level of generality. In every of particular cases of nondeterministic and probabilistic algorithms investigated below it is explicitly introduced, which reduction of computational complexity and under which conditions can be achieved. Hence, it is the reader or the user himself, who is to compare the offered reduction with the corresponding risks and to judge their acceptability from the viewpoint of an intended application.

So far in this chapter we have always taken parallel and probabilistic algorithms as close notions mainly because of their derivation from a common ancestor – nondeterministic algorithm with a side input. The connection between both the types of algorithms can be seen, however, from a more utilitary viewpoint. Consider a probabilistic algorithm $G_f(x, y)$ which computes a function f, where the set $\mathscr{Y}$ of side inputs is the set of all finite sequences of elements of a set $\mathscr{D}$. Hence, $\mathscr{Y} = \mathscr{D}^*$, or, what yields the same, $\mathscr{Y}$ is the set of all infinite sequences of elements of $\mathscr{D}$ with the σ-field $\mathscr{F}$ generated by finite cylinders. A random variable X, defined on an abstract probability space $\langle \Omega, \mathscr{S}, P \rangle$ and taking its values in the measurable space $\langle \mathscr{D}^\infty, \mathscr{F} \rangle$, can be defined by a sequence $X_1, X_2, \ldots$ of random variables defined on $\langle \Omega, \mathscr{S}, P \rangle$ and taking their values in $\mathscr{D}$. From the practical viewpoint the most interesting case occurs when the random variables $X_1, X_2, \ldots$ are mutually statistically independent, let us consider this situation. Let there exist functions $G_f^i(x, y)$, $i = 1, 2, \ldots$, a function H, and an integer-valued function k such that, for all $x \in D_f$ and all $\omega \in \Omega$,

$$G_f(x, X(\omega)) = G(x, \langle X_1(\omega), X_2(\omega), \ldots \rangle) =$$

$$= H(G_f^1(x, X_1(\omega)), G_f^2(x, X_2(\omega)), \ldots, G_f^{k(x)}(x, X_{k(x)}(\omega))) . \quad (3.8)$$

As can be easily seen, in such a case the sample of values $X_i(\omega)$ and the computation of values $G_f^i(x. X_i(\omega))$ for $i = 1, 2, \ldots, k(x)$ can be realized simultaneously, i.e., on parallel processors. The central or main processor (supervisor) cumulates the results and computes the corresponding value of the function H (cf. [6] for a more detailed theoretical model). Probabilistic algorithms and probabilistic oracles realized by sequences of statistically independent random samples offer a new possibility for parallelization so that we have arrived at the notion of *parallel probabilistic algorithm* with which we shall meet very often in the sequel, perhaps on a less general and abstract level. The reasons for this rather late introduction of such an important notion are the same as presented at the occasion of parallel algorithms above. A further reason consists in our aim to present this notion of parallel probabilistic algorithms at a rather general level, so that all types of parallel or probabilistic algorithms described above could be easily seen to be special cases of this notion. When considering a practical utility of such algorithms we have to compare, again, the computational complexity of the functions $G_f^i(x, y)$, $k(x)$ and H, with the computational complexity of the function f and with the probability of error. This problem will not be solved here at this level of generality, but only in connection with special parallel probabilistic algorithms investigated below in the next chapters.

When closing this chapter, the three following comments seem to be worth introducing.

(1) The level of generalization and abstraction adopted in this chapter is perhaps higher than necessary for the sake of further considerations. The reason is that we would like to offer a theoretical and formalized background for further investigations, development and modification of the parallel probabilistic algorithms introduced below, and also to cover a class as wide as possible of future parallel probabilistic algorithms of various sorts.

(2) As far as the author knows, there is still no practical experience with parallel probabilistic algorithms. On the other hand side, a parallel implementation of some already successfully implemented sequential probabilistic algorithms should not be too difficult. Or, all these sequential algorithms are based on independent random samples which can be easily taken in parallel by a number of identical copies of the device executing the original sequential probabilistic algorithm, not supposing any degree of co-operation or synchronization among different processors.

(3) Our considerations concerning the implementations of parallel probabilistic algorithms have brought us very close to the so called *chaotic* and *asynchronous algorithms* (cf. [1] as a very good informal introduction). These algorithms can be taken as parallel probabilistic algorithms with appropriate physical processes in the role of the source of uncertainty and randomness. The physical laws governing these processes serve as the main tool for deriving the mathematical and computational

properties of the corresponding algorithms. A more detailed investigation of the relation between chaotic and asynchronous algorithms and the general notion of parallel probabilistic algorithm as introduced above would be worth a more detailed investigation. However, because of the fact that such an investigation would exceed the intended extent and scope of this work, let us postpone it till another occasion.

REFERENCES

[1] V. Černý: Fyzikálne aspekty v matematickej informatike (Physical aspects in mathematical informatics — in Slovak). SOFSEM 1987, pp. 81—103.

[2] M. Davis: Computability and Unsolvability. McGraw-Hill, New York 1958.

[3] W. Feller: An Introduction to Probability Theory and its Applications, I, II. J. Wiley and Sons, New York 1957 (vol. I, 2nd edition), 1966 (vol. II). Russian translation: Mir, Moscow 1964, 1967.

[4] P. Halmos: Measure Theory. Van Nostrand, London, 1968.

[5] I. Kramosil: Paralelní pravděpodobnostní algoritmy jako prezentace nového paradigmatu ve znalostních systémech (Parallel probabilistic algorithms as presentation of new paradigma in knowledge systems — in Czech). In: Uplatnění expertních — znalostních systémů ve stavebnictví, Prague 1987, pp. 6—23.

[6] I. Kramosil: Extremum-searching hierarchical parallel probabilistic algorithms. Kybernetika (Prague) *24* (1988), 2, pp. 110—121.

[7] M. Loève: Probability Theory. Van Nostrand, Princeton, 1955. Russian translation: IIL, Moscow, 1962.

[8] A. Rényi: Teorie pravděpodobnosti (Probability Theory — in Czech). Academia, Prague 1972.

[9] J. Štěpán: Teorie pravděpodobnosti — matematické základy (Probability Theory — Mathematical Foundations — in Czech). Academia, Prague 1987.

[10] V. A. Uspenskij and A. L. Semenov: What are the gains of the theory of algorithms — basic development connected with the concept of algorithm and with its application in mathematics. In: Algorithm in Modern Mathematics and its Applications — Proceedings of the Symposium, Urgench 1979, pp. 100—234.

4. PARALLEL PROBABILISTIC SEARCHING ALGORITHMS

Searching problems belong to the basic, and from the descriptional point of view the most simple ones to which many more complicated and sophisticated tasks of artificial intelligence can be converted. Moreover, the time and perhaps other demands or expreses, involved by the solution of the corresponding searching problem decide about the computational complexity and applicability of the resulting solution of the original tasks as a whole. A general model of searching problem, very simple, but sufficient for our further reasonings, can be described as follows: Consider a finite but, as a rule, very large set $A = \{a_1, a_2, \ldots, a_N\}$ of N elements and a subset $V \subset A$. We have at our disposal an oracle working on the black-box principle, which,

having been put an element from A on its input, decides in a finite and, for the sake of simplicity constant, time, whether this element belongs to V or not. Assume again, for this instant and for the sake of simplicity, that this decision is reliable and faultless. Moreover, this oracle is our only tool of "communication" with the set V. No other knowledge concerning the set V is at our disposal, neither being obtainable by, say, an analysis of the way in which the oracle works and decides.

The question whether the set V is empty or not can be answered by a sequential exhaustive searching and testing of all elements of A. This procedure requests in the worst case, supposing we take this worst case as the decisive one as common in the classical theory of computational complexity, N operations. Taking, in the first approximation, these operations as equivalent, the resulting sequential computational complexity is in the $\mathcal{O}(N)$-class. Now, consider an M-times sequentially repeated independent random sample from A, with the same probability $1/N$ of sampling ascribed to each element of A at each step. If V contains v elements, then with the probability $(1 - v/N)^M$ all samples lie outside of V. Supposing that, having tested all the elements and having seen that none of them is in V, we decide that the set V is empty, then $(1 - v/N)^M$ is the probability of error with which we can arrive at such a wrong (if $v \neq 0$) decision. This probability of error is evidently maximum, if $v = 1$, and if we want this probability to be majorized by an a priori given $\varepsilon > 0$ even in this worst case, we must have $M \geqq (\ln \varepsilon^{-1})\, N$, as a simple calculation yields. Hence, again $M = M(N)$ is in $\mathcal{O}(N)$-class with the multiplicative constant being even greater than one if $\varepsilon < \mathrm{e}^{-1} = 0{\cdot}36\ldots$, i.e. for every "reasonable" ε. The choice of the uniform probability distribution over the set A can be justified by the mentioned above Laplace principle and by the fact that we have no reasons for another probability distribution. Here we do not take into consideration the sampling without giving back the sampled elements, as this sample violates the demand for statistical independence of particular samples. Moreover, when applying the uniform probability distribution over the set of not yet sampled elements, it is equivalent to the systematic exhaustive search.

Now, suppose to have at our disposal a greater number of identical copies of the testing oracle, which may test in parallel, hence, simultaneously, a number of elements of A. If this number of identical copies is finite, the resulting time complexity is again in the $\mathcal{O}(N)$-class, no matter whether we consider the systematic deterministic exhaustive search or the randomized solution described above. When the number of processors-copies increases and equals, say, N (the cardinality of the basic set A), the resulting time complexity is in both the cases constant, i.e. independent of N, i.e. in the $\mathcal{O}(1)$-class. This assertion holds, of course, under the assumption that it is sufficient to find an element of V by at least one processor and that no time or other demands concerning the inspection of the outputs of particular processors are taken into consideration. This assumption is, as a rule, accepted in the theory of nondeterministic algorithms, but in our context we shall take it as too idealized and non-realistic and we shall not accept it in what follows. From this point of view,

of course, the problem to inspect the outputs of N processors is far from being trivial. In fact, it is a duplicate of the original problem, or at least its computational complexity is equivalent to the original one. Hence, neither a non-limited parallelism is an adequate solution to our searching problem which would reduce, either in the weak sense or in the strong sense, the time complexity of the sequential deterministic exhaustive search. Therefore, we shall investigate in this chapter, whether a sophisticated combination of parallelism and randomization into one algorithmic structure could yield such a qualitative time reduction. As can be seen, this problem can be positively solved.

So, consider the searching problem informally described above, formally defined by a pair $\langle A, V\rangle$, $A = \{a_1, a_2, \ldots, a_N\}$, $V \subset A$, and consisting in answering the question whether $V = \emptyset$ or not. Consider also an abstract probability space $\langle \Omega, \mathcal{S}, P\rangle$, natural numbers m, n, k, and a system $\mathcal{X} = \langle \{X_{ij}\}_{i=1\,j=1}^{m\quad n}, \{Z_l\}_{l=1}^{k}\rangle$ of mutually statistically independent random variables defined on $\langle \Omega, \mathcal{S}, P\rangle$. Each X_{ij} takes its values in A, each Z_l in the set $\{1, 2, \ldots, m\}$ of integers, and for all $a \in A$, $i \leqq m$, $j \leqq n$, $l \leqq k$ and $s \leqq m$,

$$P(\{\omega\colon \omega \in \Omega,\ X_{ij}(\omega) = a\}) = N^{-1}\,, \tag{4.1}$$

$$P(\{\omega\colon \omega \in \Omega,\ Z_l(\omega) = s\}) = m^{-1}\,. \tag{4.2}$$

Hence, the values $X_{ij}(\omega)$ represent independent random samples from the uniform probability distribution over the set A, the values $Z_l(\omega)$ represent independent random samples from the uniform probability distribution over the set $\{1, 2, \ldots, m\}$ of integers. The system $\mathcal{X}$ is called *two-level hierarchical parallel probabilistic searching algorithm (HPPSA) for the searching problem* $\langle A, V\rangle$. It is understood as a random variable, the value of which, given $\langle A, V\rangle$, is defined by

$$\mathcal{X}(\langle A, V\rangle, \omega) = 1 \quad \text{iff} \quad \sum_{l=1}^{k} \sum_{j=1}^{n} \chi_V(X_{Z_l(\omega),j}(\omega)) > 0\,, \tag{4.3}$$

$$\mathcal{X}(\langle A, V\rangle, \omega) = 0 \quad \text{otherwise}\,,$$

χ_V is the characteristic function or identifier of the set V. The result $\mathcal{X}(\langle A, V\rangle, \omega)$ is *correct*, if $\mathcal{X}(\langle A, V\rangle, \omega) = 1$ and $V \neq \emptyset$, or if $\mathcal{X}(\langle A, V\rangle, \omega) = 0$ and $V = \emptyset$. The *error of the first kind* occurs, if $\mathcal{X}(\langle A, V\rangle, \omega) = 0$ and $V \neq 0$, the *error of the second kind* occurs, if $\mathcal{X}(\langle A, V\rangle, \omega) = 1$ and $V = \emptyset$. Evidently, $\mathcal{X}(\langle A, V\rangle, \omega) = 1$ iff at least one among the parallel processors, represented by n-tuples $\langle X_{i1}, X_{i2}, \ldots, X_{in}\rangle$, $i \leqq m$, samples an element from V and, at the same time, at least one processor with this property is sampled during the random inspection of the outputs of particular processors realized through random sampling of their indices by random variables $Z_1, Z_2, \ldots, Z_k$. Hence, supposing that all testing oracles are completely reliable and fail-proof, the error of the second kind cannot occur in the presented model. Or, the decision that $V \neq \emptyset$ is always based on the fact that at least one element of V has been found by a first-level processor.

The *unit time computational complexity* $TCU(\mathcal{X})$ of a given *HPPSA* $\mathcal{X}$ with parameters m, n and k is defined by

$$TCU(\mathcal{X}) = \alpha_1 n + \alpha_2 k + \alpha_3 , \tag{4.4}$$

for appropriate positive (α_1, α_2) or non-negative (α_3) constants. The *logarithmic time computational complexity* $TCL(\mathcal{X})$ is defined by

$$TCL(\mathcal{X}) = \beta_1 n \log_2 N + \beta_2 k \log_2 m + \beta_3 , \tag{4.5}$$

for appropriate positive (β_1, β_2) or non-negative (β_3) constants. Hence, the unit time complexity takes each random sample taken by X_{ij} or Z_l as being of the same complexity no matter how large the sample space (A or $\{1, 2. \ldots, m\}$) may be. The logarithmic time complexity supposes the complexity of random samples to increase linearly with the logarithm of the size of the sample space in question. When taking the logarithm to the base two, the logarithmic time complexity of a random sample corresponds to the number of coin tosses, by the mean of which the random sample in question could be realized.

An easy proof (cf. [1] or [2]) yields the following assertion.

Theorem 4.1. Let $\langle A, V\rangle$ be a searching problem, let

$$\mathcal{X} = \langle \{X_{ij}\}_{i=1}^{m}{}_{j=1}^{n}, \{Z_l\}_{l=1}^{k} \rangle \tag{4.6}$$

be a *HPPSA* for $\langle A, V\rangle$ with parameters m, n and k, let $\varepsilon > 0$ be given, let $PE_I(\mathcal{X}, A, V)$ be the probability of error of the first kind, let $v = \text{card } V$, $N = \text{card } A$. If $mn \geqq v^{-1}(\ln(2/\varepsilon))\, N$ and $k \geqq (\ln(2/\varepsilon))\, m$, then $PE_I(\mathcal{X}, A, V) < \varepsilon$.

As mentioned above, if $V = \emptyset$, then an analogous assertion for the probability of error of the second kind holds trivially for all m, n, k and $\varepsilon > 0$. Because of the fact that in the conditions of Theorem 4.1 just the product $m \,.\, n$ is bound, a natural question arises, which ratio between m and n is the optimal one in order to minimize the corresponding unit or logarithmic time complexity. To arrive at a more correct formalization, the following auxiliary notions will be of use.

HPPSA $\mathcal{X}$ is called *ε-admissible* (*strongly ε-admissible* resp.) with respect to the searching problem $\langle A, V\rangle$ and w.r. to a given $\varepsilon > 0$, if $V \neq \emptyset$ and $PE_I(\mathcal{X}, A, V) < \varepsilon$ ($V \neq \emptyset$ and the conditions of Theorem 4.1 hold, resp.). *HPPSA* $\mathcal{X}$ is called *unit ε-optimal* (*unit strongly ε-optimal*, resp.). if $\mathcal{X}$ is ε-admissible (strongly ε-admissible, resp.) and for each ε-admissible (strongly ε-admissible) *HPPSA* $\mathcal{X}'$ the relation $TCU(\mathcal{X}) \leqq TCU(\mathcal{X}')$ holds. *Logarithmically ε-optimal* and *logarithmically strongly ε-optimal HPPSA's* $\mathcal{X}$ are defined in an analogous way.

Theorem 4.2. Let $\mathcal{X}$ be a *HPPSA* for the searching problem $\langle A, V\rangle$ with parameters $m(N)$, $n(N)$, and $k(N)$, where $N = \text{card } A$, let $v = \text{card } V > 0$. Then $\mathcal{X}$ is unit strongly ε-optimal, if

$$m(N) = \lceil (\sqrt{\alpha_1}/\sqrt{(\alpha_2 v)})\, \sqrt{N} \rceil , \tag{4.7}$$

$$n(N) = \lceil (\ln (2/\varepsilon))\, (\sqrt{\alpha_2}/\sqrt{(\alpha_1 v)})\, \sqrt{N} \rceil \tag{4.8}$$

$$k(N) = \lceil (\ln (2/\varepsilon))\, m(N) \rceil . \tag{4.9}$$

In the case of *TCL* explicit expressions for the values of parameters are much more difficult to obtain so that we have to introduce an implicit formulation with unspecified $o(\sqrt{N})$-functions. Hence, if $\mathscr{X}$ is logarithmically strongly ε-optimal, then

$$m(N) = (\sqrt{\beta_1}/\sqrt{(\beta_2 v \log_2 \mathrm{e})})\, \sqrt{N} + o(\sqrt{N}) , \tag{4.10}$$

$$n(N) = (\ln (2/\varepsilon))\, (\sqrt{(\beta_2 \log_2 \mathrm{e})}/\sqrt{(\beta_1 v)})\, \sqrt{N} + o(\sqrt{N}) , \tag{4.11}$$

$$k(N) = (\ln (2/\varepsilon))\, m(N) + o(\sqrt{N}) . \tag{4.12}$$

Hence, if $\alpha_1 = \alpha_2$, then for $v = 1$, $m(N) = \lceil \sqrt{N} \rceil$ and $n(N) = \lceil \ln (2/\varepsilon))\, \sqrt{N} \rceil$ according to (4.7) and (4.8). Corresponding proofs can be found, again, in [1] and [2].

Hence, there exists a *HPPSA* the probability of error of which, when solving the searching problem, is majorized by a given $\varepsilon > 0$ uniformly for each $V \subset A$. Its unit time computational complexity is $c_1 + c_2 \sqrt{N}$ with c_1 and c_2 depending on ε but independent of N, so that this complexity is in $\mathcal{O}(\sqrt{N})$-class. The logarithmic time complexity of the same algorithm is $\tilde{c}_1 (\sqrt{(N)}) \log N + \tilde{c}_2 \sqrt{(N)} + \tilde{c}_3$, hence, is in $\mathcal{O}((\sqrt{(N)}) \log N)$-class. In both the cases the reduction of time computational complexity is substantial (in the weak sense) when compared with the results achieved by simple randomization or by the maximal parallelization, when the resulting time complexity was in the $\mathcal{O}(N)$-, or $\mathcal{O}(N \log N)$-class.

The assertions of the two theorems just presented immediately involve the question, whether a multiple use of the hieararchical principle presented above would reduce the time complexity more substantially i.e. would bring this complexity into the classes $o(\sqrt{N})$ or $o((\sqrt{(N)}) \log N)$. The following example shows that this is possible.

Again, consider the searching problem $\langle A, V \rangle$, where card $A = N$, card $V = v > 0$, let $\varepsilon > 0$ be given. Suppose, for the sake of simplicity, that N is of the form 2^K, so that $K = \log_2 N$. Set $\delta = \varepsilon/K$, $N_i = N/2^i$, $i = 1, 2, \ldots, K$, and consider $N_1 = N/2$ processors of the first level. Each of these processors takes $\lceil 2 \ln (1/\delta) \rceil$ independent random samples from A with respect to the uniform probability distribution ascribing the probability of sampling N^{-1} to each element of A, in each step, and by each processor. Access conflicts are not taken into consideration, so that each element of A is accessible simultaneously to all the processors by which it has been sampled. For each $i \leqq N_1$ the output value of the ith processor equals to one supposing at least one element from V was sampled by this processor, the output value being zero otherwise. Now, set $N_2 = N_1/2 = N/4$ and consider N_2 second-level processors, each of them taking, again, $\lceil 2 \ln (1/\delta_j) \rceil$ independent random samples from the set $\{1, 2, \ldots, N_1\}$ of integers and w.r. to the uniform probability distribution ascribing the probability N_1^{-1} to each result in each sample. For each $i \leqq N_2$, the

output of the ith second-level processor takes the value one supposing that at least once a first-level processor with the unit output value was sampled by this second-level processor, in the opposite case the output value of the ith second level processor is zero. Set $N_3 = N/8$ and proceed analogously. At the Kth level we have just one Kth level processor which takes $\lceil 2 \ln (1/\delta) \rceil$ random samples from the two-element set of (indices of) the $(K-1)$-st level processors. If at least once a $(K-1)$-st level processor with the unit output value is sampled, also the (unique) Kth level processor takes one as its output value and we decide that $V \neq \emptyset$. This decision is certainly correct, as we have discovered at least one element of V at the basic level. In the opposite case the output value of the Kth level processor is zero, we take the decision $V = \emptyset$, which may be charged by a positive probability of error; let us compute this probability.

At the basic level of our hierarchy $(N/2) \lceil 2 \ln (1/\delta) \rceil$ random samples have been taken altogether from the uniform probability distribution over the set A. As this number is at least $N \ln (1/\delta)$, the same argumentation as above or a simple calculation yield that if $V \neq \emptyset$, then with a probability greater than $1 - \delta$ at least one element from V is sampled by a first-level processor. Denoting by A_1 the set $\{1, 2, \ldots, N_1\}$ of integers, and by V_1 the set of indices of those first-level processors, the output value of which is one, we obtain, that with a probability at least $1 - \delta$ the set V_1 is nonempty. The second-level processors can be considered as the first-level ones w.r. to the new searching problem $\langle A_1, V_1 \rangle$, the argumentation can be repeated by induction and after K steps we obtain, that the probability with which the (unique) K-th level processor output value is one, under the condition that $V \neq \emptyset$, equals at least

$$\prod_{i=1}^{K} (1 - \delta_i) > 1 - K\delta = 1 - \varepsilon \tag{4.13}$$

(recall that $\delta = \delta_i = \varepsilon/K$ for each $i \leqq K$). Accepting the unit criterion for measuring the time computational complexity of random samples we obtain that the total number of sequentially taken random samples, which defines the time complexity, equals $2K\lceil \ln (1/\delta) \rceil$, i.e. $\lceil 2 \ln (1/\delta) \rceil$ samples at each of the Kth levels. After an appropriate substitution this expression reads as

$$2(\log_2 N) \ln \lceil ((\log_2 N)/\varepsilon) \rceil , \tag{4.14}$$

which is in the $\mathcal{O}(\log N \log \log N)$-class for each fixed $\varepsilon > 0$, hence, it is trivially in the $o(\sqrt{N})$-class. In the case of the logarithm criterion the corresponding time complexity can be computed as follows (set $N_0 = N$):

$$\begin{aligned} &\sum_{i=0}^{K} 2\lceil \ln (1/\delta) \rceil \log_2 N_i = \sum_{i=0}^{\log_2 N} 2\lceil \ln ((\log_2 N)/\varepsilon) \rceil \log_2 (N/2^i) = \\ &= \sum_{i=0}^{\log_2 N} 2i\lceil \ln ((\log_2 N)/\varepsilon) \rceil = 2\lceil \ln ((\log_2 N)/\varepsilon) \rceil (1/2) \log_2 N(\log_2 N - 1) = \\ &= \lceil \ln ((\log_2 N)/\varepsilon) \rceil ((\log_2 N)^2 - \log_2 N) , \end{aligned} \tag{4.15}$$

and this expression is in $\mathcal{O}((\log N)^2 \log\log N)$-class, hence, again in $o((\sqrt{(N)}) \log N)$-class. As can be easily seen, our assumption that N is of the form 2^K is not substantial for the validity of the obtained qualitative result.

The hierarchical structure informally described above can be formally defined as follows.

Many-level (particularly, *K-level*, $K \geqq 1$) *hierarchical parallel probabilistic searching algorithm* (*MLHPPSA*, or *K-HPPSA*) *for the searching problem* $\langle A, V\rangle$ is the system

$$\mathcal{X} = \langle \{X_{ij}^k\}_{k=1, i=1, j=1}^{K \quad N_k \quad n_k} \rangle \tag{4.16}$$

of mutually statistically independent random variables defined on an abstract probability space $\langle \Omega, \mathcal{S}, P\rangle$ in such a way that $N_1 > N_2 > \ldots > N_K = 1$ and each X_{ij}^k takes its values in the set $A_{k-1} = \{1, 2, \ldots, N_{k-1}\}$ of integers (where $A_0 = A$) w.r. to the uniform probability distribution. I.e., for each $k \leqq K$, $j \leqq N_k$, $j \leqq n_k$, $r \leqq N_{k-1}$ (here $N_0 = N$) we have

$$P(\{\omega\colon \omega \in \Omega, X_{ij}^k(\omega) = r\}) = N_{k-1}^{-1} . \tag{4.17}$$

The vector $\langle\langle N_1, n_1\rangle, \langle N_2, n_2\rangle, \ldots, \langle N_k, n_k\rangle\rangle$ of pairs of positive integers is called *characteristics* of the *MLHPPSA* $\mathcal{X}$.

As in the case of two-level *HPPSA*'s described above $\mathcal{X}$ can be understood as a statistical decision function which solves the searching problem $\langle A, V\rangle$, i.e. as a random variable $\mathcal{X}(\langle A, V\rangle, .)$ taking its values in the binary set $\{0, 1\}$. This value will be defined by induction. Set $V_0 = V$ and define, for each $r \leqq K$,

$$V_r(\omega) = \{i\colon i \leqq N_r . (\exists j \leqq n_r)(X_{ij}^r(\omega) \in V_{r-1}(\omega))\} =$$

$$= \{i\colon i \leqq N_r, \sum_{j=1}^{n_r} \chi_{V_{r-1}(\omega)}(X_{ij}^r(\omega)) > 0\} . \tag{4.18}$$

Namely,

$$V_1(\omega) = \{i\colon i \leqq N_1, \sum_{j=1}^{n_1} \chi_V(X_{ij}^1(\omega)) > 0\} ,$$

which agrees with the definition of *HPPSA* as presented above. Moreover, for $r = K$,

$$V_K(\omega) = \{i\colon i \leqq 1, (\exists j \leqq n_k)(X_{ij}^K(\omega) \in V_{K-1}(\omega))\} , \tag{4.20}$$

so that

$$V_K(\omega) = \{1\} = A_K \Leftrightarrow \sum_{j=1}^{n_K} \chi_{V_{K-1}(\omega)}(X_{ij}^K(\omega)) > 0 , \tag{4.21}$$

$$V_K(\omega) = \emptyset \Leftrightarrow \sum_{j=1}^{n_K} \chi_{V_{K-1}(\omega)}(X_{ij}^K(\omega)) = 0 . \tag{4.22}$$

As can be easily seen, if $V_k(\omega) = \emptyset$, then also $V_l(\omega) = \emptyset$ for each $k \leqq l \leqq K$. Now,

set

$$\{\omega: \omega \in \Omega, \mathscr{X}(\langle A, V\rangle, \omega) = 1\} = \bigcap_{k=0}^{K} \{\omega: \omega \in \Omega, V_k(\omega) \neq \emptyset\}\,, \tag{4.23}$$

$\mathscr{X}(\langle A, V\rangle, \omega) = 0$ otherwise. Both the kinds of error as well as their probabilities are defined in the same way as in the case of *HPPSA*'s above. Again, the error of the second kind is impossible on the ground of the same argumentation as above. The corresponding unit and logarithmic time complexities read

$$TCU(\mathscr{X}) = \sum_{i=1}^{K} \alpha_i n_i + \alpha_0\,, \tag{4.24}$$

$$TCL(\mathscr{X}) = \sum_{i=1}^{K} \beta_i n_i \log_2 N_i + \beta_0 \tag{4.25}$$

for appropriate positive $\alpha_1, \ldots, \alpha_K, \beta_1, \ldots, \beta_K$ and non-negative α_0, β_0. In what follows, we shall investigate only the simple case with $\alpha_i = \alpha$, $\beta_i = \beta$ for all $i = 1, 2, \ldots, K$.

Denote by p_k, $k = 1, 2, \ldots, K$, the conditional probability

$$p_k = p_k(\mathscr{X}) = P(\{\omega: \omega \in \Omega, V_k(\omega) \neq \emptyset\}/\{\omega: \omega \in \Omega, V_{k-1}(\omega) \neq \emptyset\})\,, \tag{4.26}$$

supposing it is defined. As can be easily seen, if $V \neq 0$, then

$$\begin{aligned} 1 - PE_I(\mathscr{X}, A, V) &= P(\{\omega: \omega \in \Omega, \mathscr{X}(\langle A, V\rangle, \omega) = 1\}) = \\ &= P(\{\bigcap_{k=1}^{K} \{\omega: \omega \in \Omega, V_k(\omega) \neq \emptyset\}) = \\ &= \prod_{k=1}^{K} P(\{\omega: \omega \in \Omega, V_k(\omega) \neq \emptyset\}/\{\omega: \omega \in \Omega, V_{k-1}(\omega) \neq \emptyset\}) = \prod_{k=1}^{K} p_k(\mathscr{X})\,. \end{aligned} \tag{4.27}$$

If $1 - p_k \leqq \varepsilon_k$ for each $k = 1, 2, \ldots, K$, i.e. $p_k \geqq 1 - \varepsilon_k$, then

$$\prod_{k=1}^{K} p_k \geqq \prod_{k=1}^{K} (1 - \varepsilon_k) \geqq 1 - \sum_{k=1}^{K} \varepsilon_k\,, \tag{4.28}$$

hence, the probability of error of the first kind, $PE_I(\mathscr{X}, A, V)$, is majorized by $\sum_{k=1}^{K} \varepsilon_k$.

An optimalization of many levels *HPPSA*'s, i.e. the problem to find a number of levels, number of processors at each level, and number of random samples taken by each processor, which would minimize the unit or logarithmic time complexity, is much more difficult than in the two-level case and will not be solved here in all the generality. Let us limit ourselves to the so called *ε-homogeneous many level HPPSA*'s with the following properties:

(a) if the number of levels of the *HPPSA* is K, then the conditional probability that the output value of at least one processor of the ith level is one under the condition that the output value of at least one processor of the $i - 1$st level is one is at least $1 - (\varepsilon/K)$.

(b) The ratio N_{i-1}/N_i of the number of processors of the $i - 1$st and of the ith

level is the same for all $i = 1, 2, \ldots, K$ and is called the *parameter of homogeneity* of the *ε-homogeneous MLHPPSA* in question.

Hence, the algorithm informally described above is an ε-homogeneous K-level *HPPSA* the parameter of homogeneity of which is two. The following assertion can be proved (cf. again, [1] or [2]).

Theorem 4.3. Let $\langle A, V\rangle$, card $A = N$, card $V = 1$, be a searching problem, let $\varepsilon > 0$ be given. Then the parameter of homogeneity λ of that ε-homogeneous many level *HPPSA*, which solves the problem $\langle A, V\rangle$ and the unit time complexity of which is minimal among all ε-homogeneous many level *HPPSA* for the same problem, satisfies the equation

$$(\ln \ln N + \ln \varepsilon^{-1})(\ln \lambda - 1) = (\ln \lambda) \ln \ln \lambda - 1 . \tag{4.28}$$

One of the solutions λ_1 of (4.28) is approximately equal to $e = 2{\cdot}718 \ldots$ in the sense that the difference among the left-hand side and right-hand side in (4.28), having been divided by $\ln \ln N + \ln \varepsilon^{-1}$, tends to 0 as $N \to \infty$ and $\lambda \to e$.

The unit time complexity of this optimal algorithm is in the $\mathcal{O}(\log N \log \log N)$-class, as above when $\lambda = 2$. Hence, the algorithm is optimal in the sense that the multiplicative constant is smaller than in the case of ε-homogeneous *MLHPPSA*'s with non-optimal parameters of homogeneity.

As in the case of the foregoing chapter, some conclusive comments may be useful.

(1) The role of random samples in the algorithms described above might perhaps have brought to the reader's mind the well-known idea of Monte-Carlo methods or algorithms. In fact, the parallel probabilistic searching algorithms as described here may be easily taken as very simple cases of Monte-Carlo methods. Or, roughly said, Monte-Carlo methods are all procedures, when an unknown expected value of a random variable, necessary for the sakes of further computations or decision makings, is approximated by the average value computed from a appropriate (or accessible) number of corresponding random samples, and all the risks following from such an approximation are accepted. Namely, in our case, evidently the value $v = \text{card } V/\text{card } A$ is the expected value of each X_{ij}, and we use random samples in order to decide, under some risk, whether v is positive or not. In Chapter 6 we shall investigate the problem when not only the sign, but also the numerical value of v is to be estimated by appropriate hierarchical parallel probabilistic searching structures.

(2) The reader has probably already observed that the randomization minimizes the demands imposed on co-operation and synchronization, nevertheless, we consider this general phenomenon as worth an explicit repeated introducing. Even in the most simple case of the sequential probabilistic searching algorithm mentioned at the beginning of this chapter, an important advantage has been obtained at the cost of $\mathcal{O}(N)$-time complexity and a positive probability of error. Namely, no registration

of the already tested elements is necessary, each sample, if still executed, is quite independent of all former ones, which, of course, is not and cannot be the case for a deterministic exhaustive search. In the parallel case, all processors at the same level work independently, no co-operation or synchronization is necessary, and their communication with higher-level processors is of a very limited and stochastic nature. Even if we do not investigate here the hardware problems connected with architectures appropriate for the investigated algorithms, an immediate idea is that such architectures could be built from a number of identical copies of devices executing the corresponding sequential algorithms, what may be taken as a non-negligible practical advantage.

(3) Finally, just as a quite open and worth a more detailed investigating problem let us remember the connections between the randomized searching structures investigated above and some more sophisticated deterministic searching structures as, e.g., the associative and orthogonal memories.

REFERENCES

[1] I. Kramosil: Hierarchické paralelní pravděpodobnostní algorithmy (Hierarchical Paralle Probabilistic Algorithms — in Czech). Res. Rep. No. 1409, Institute of Information Theory and Automation 1986.

[2] I. Kramosil: Extremum-searching hierarchical parallel probabilistic algorithms. Kybernetika *24* (1988), 2, 110—121.

5. SEARCHING ALGORITHMS WITH LIMITED TESTING RELIABILITY AD WITH GENERALIZED LOSS FUNCTION

In this chapter we shall go on with our investigations of the hierarchical searching algorithms as investigated in Chapter 4 and with respect to the same unit and logarithmic time computational complexities as above. Let us assume, however, that each testing oracle may fail, i.e. may output the wrong answer to the question whether the tested element from A belongs to V or not. The oracle represents an extra-mathematical device which tests the elements of the basic set A, so that the reasons and forms of its failure are also of an extra-mathematical nature and cannot be investigated within the framework of this study. Roughly said, with an unreliable testing oracle the final result is qualitatively less reliable than in the case of a failure-proof oracle defined above. Or, even if some element from V is sampled, it need not be recognized. On the other side, the supervizor may report an element from V even if $V = \emptyset$, as some processor wrongly proclaimed an element to be in V. This possibility of error will be supposed to be quantifiable by the value of a probability measure and, for the sake of simplicity, this probability will be taken as the same for different elements from V, and the same but perhaps different from the former value for

different elements from $A - V$. Moreover, the errors will be supposed to be statistically independent for different samples of different elements from A as well as for different samples of the same element from A so that the possibility of error is associated rather with the sample than with particular elements.

Like as in Chapter 4, consider positive integers n, m, and k and two systems $\{X_{ij}\}_{i=1\,j=1}^{m\quad n}$, $\{Z_l\}_{l=1}^{k}$ of mutually statistically independent random variables defined on the probability space $\langle \Omega, \mathscr{S}, P \rangle$. They take their values in A (for each $i \leqq m$, $j \leqq n$) or in the set $\{1, 2, \ldots, m\}$ of integers (for each $l \leqq k$), and satisfy (4.1) (for each $a \in A$, $i \leqq m, j \leqq n$) or (4.2) (for each $s \leqq m$ and $l \leqq k$).

A new building stone of our model are two random variables $Q(0)$ and $Q(1)$, defined on $\langle \Omega, \mathscr{S}, P \rangle$ and taking their values in binary set $\{0, 1\}$. Denote

$$P(\{\omega: \omega \in \Omega, Q(1, \omega) = 0\}) = p, P(\{\omega: \omega \in \Omega, Q(0, \omega) = 1\}) = q\,. \tag{5.1}$$

In the classical Shannon information theory these two random variables define a binary channel, but not necessarily a symmetric one. Hence, the value $Q(\chi_V(a), \omega)$ is the result of testing whether $a \in V$ or not, charged with the possible probability of error; this probability is p for $a \in V$ (when $\chi_V(a) = 1$), and is q for $a \in A - V$ (when $\chi_V(a) = 0$).

Finally, consider a real μ, $0 \leqq \mu \leqq 1$, and define random variable $\delta = \delta(\mu)$, taking $\langle \Omega, \mathscr{S}, P \rangle$ into $\{0, 1\}$ in this way: set

$$\pi(\omega) = (nk)^{-1} \sum_{l=1}^{k} \sum_{j=1}^{n} Q(\chi_V(X_{Z_l(\omega),j}(\omega)), \omega)\,, \tag{5.2}$$

and define

$$\{\omega: \delta(\mu, \omega) = 1\} = \{\omega: \pi(\omega) > \mu\}\,, \quad \delta(\mu, \omega) = 0 \quad \text{otherwise}\,. \tag{5.3}$$

The structure

$$\mathscr{X} = \langle \{X_{ij}\}_{i=1\,j=1}^{m\quad n}, \{Z_l\}_{l=1}^{k}, \{Q(i)\}_{i=1,2,}, \mu \rangle \tag{5.5}$$

is called *threshold hierarchical parallel probabilistic searching algorithm* with the threshold value μ.

Similarly as above (cf. (4.3)) a seemingly complicated relation (5.2) offers an intuitive interpretation. The value $Q(\chi_V(X_{ij}(\omega)), \omega)$ can be seen as a "report", whether the at random sampled element $X_{ij}(\omega)$ from A belongs to V or not; this report can be wrong in the sense and with the probabilities described above. The value

$$\varrho(\omega) = (nm)^{-1} \sum_{i=1}^{m} \sum_{j=1}^{n} Q(\chi_V(X_{ij}(\omega)), \omega) \tag{5.5}$$

expresses the average value of elements among the sampled ones (with possible repetitions) which are *reported* to be in V. the average being taken over all samples and all processors. However, having accepted and developed the randomized hierarchical principle, we do not compute the average value of reported elements from V

over all processors, but only over those of them which have been sampled by random variables $Z_1, \ldots, Z_k$. Hence, $\pi(\omega)$ can be seen as a statistical estimation of the value $p(\omega) = (1/N) \sum_{a \in A} Q(\chi_V(a), \omega)$, like the value $\varrho(\omega)$. The statistical qualities of $\pi(\omega)$ are, in general, worse than those of $\varrho(\omega)$ (greater dispersion), but they improve with k increasing.

Take the random event $\delta(\mu, \omega) = 1$ as the decision that $V \neq \emptyset$ and as the refusal of the alternative decision that $V = \emptyset$. Then $\delta(\mu)$ can be seen as a statistical decision function accepting the hypothesis $V \neq \emptyset$ just in case the relative frequency of samples which are reported to be in V exceeds an a priori given threshold value μ. As can be easily seen, if $p = q = 0$ and $\mu = 0$, our model reduces to the one investigated in Chapter 4. If $q > 0$, then $\varrho(\mu) > 0$ and $\pi(\omega) > 0$ may occur with positive probabilities even when $V = \emptyset$.

A simple factorization yields

$$
\begin{aligned}
&P(\{\omega\colon \omega \in \Omega,\ Q(\chi_V(X_{ij}(\omega)), \omega) = 1\}) = \\
&= P(\{\omega\colon \omega \in \Omega,\ Q(1, \omega) = 1\} / \{\omega\colon \omega \in \Omega,\ \chi_V(X_{ij}(\omega)) = 1\}) \,. \\
&\,. P(\{\omega\colon \omega \in \Omega,\ \chi_V(X_{ij}(\omega)) = 1\}) + \\
&+ P(\{\omega\colon \omega \in \Omega,\ Q(0, \omega) = 1\} / \{\omega\colon \omega \in \Omega,\ \chi_V(X_{ij}(\omega)) = 0\}) \,. \\
&\,. P(\{\omega\colon \omega \in \Omega,\ \chi_V(X_{ij}(\omega)) = 0\}) = \\
&= (1 - p)\, P(\{\omega\colon \omega \in \Omega,\ X_{ij}(\omega) \in V\}) + q\, P(\{\omega\colon \omega \in \Omega,\ X_{ij}(\omega) \in A - V\}) = \\
&= (1 - p)\, v + q(1 - v)\,, \qquad (5.6)
\end{aligned}
$$

using (4.1) and denoting $v = \operatorname{card} V / \operatorname{card} A$. Denote the last expression in (5.6) by p_v and suppose that $p, q < \frac{1}{2}$. This assumption will be taken in all the rest of this chapter in order to simplify our considerations. Then p_v is an increasing function of the argument v, hence, $p_v > p_0 = q$ for each $v > 0$. So the test of the hypothesis $V = \emptyset$ against the alternative $V \neq \emptyset$, i.e. $v > 0$, can be converted into, or understood as, the test of the hypothesis that

$$
\tilde{P} = P(\{\omega\colon \omega \in \Omega,\ Q(\chi_V(X_{ij}(\omega)), \omega) = 1\}) = p_0 = q \qquad (5.7)
$$

against the alternative that $\tilde{P}_v = p_v > 0$. Random variables π and ϱ can be taken as statistics and $\delta(\mu)$ as a statistical decision function corresponding to the latter test.

Hence, the problem whether $V = \emptyset$ or $V \neq \emptyset$, i.e. whether $v = 0$ or $v \geqq 1/N$, is converted into the test whether $\tilde{P} = q$, or

$$
\tilde{P} \geqq p_{1/N} = (1 - p)\, N^{-1} + q(1 - N^{-1}) = q + (1 - p - q)\, N^{-1} > q\,, \qquad (5.8)
$$

as $p, q < \frac{1}{2}$ implies $p + q < 1$. First of all, consider the random variable ϱ defined by (5.5). Set $K = mn$, and for each $l \leqq K$,

$$
Y_l(\omega) = Q(\chi_V(X([[(l-1)/n] + 1,\ l - ([[(l-1)/n])\, n, \omega)), \omega)\,, \qquad (5.9)
$$

writing $X(i, j, \omega)$ instead of $X_{ij}(\omega)$. This enumeration defines the ordering $X_{11}(\omega), X_{12}(\omega), \ldots, X_{1n}(\omega), X_{21}(\omega), \ldots, X_{2n}(\omega), X_{31}(\omega), \ldots, X_{mn}(\omega)$ of random samples. Clearly, $\varrho(\omega) = \varrho_K(\omega) = \left(\sum_{l=1}^{K} Y_l(\omega)\right) K^{-1}$ and $Y_1, Y_2, \ldots, Y_K$ is a sequence of statistically independent and equally distributed random variables with common expected value $\tilde{\mu}$ and dispersion σ^2, each of them taking $\langle \Omega, \mathscr{S}, P \rangle$ into $\{0, 1\}$. Hence, $V \neq \emptyset$ iff $\tilde{\mu} \geqq q_0 = p_{1/N}$, $V = \emptyset$ iff $\tilde{\mu} = q < q_{1/N}$. The strong law of large numbers yields that

$$P\left(\{\omega: \omega \in \Omega, \quad \lim_{K \to \infty} \varrho_K(\omega) = \tilde{\mu}\}\right) = 1 . \tag{5.10}$$

Take $\mu \in (q, \tilde{q})$, $\tilde{q} = p_{1/N} > q$, and define a decision function δ^* by $\delta^*(\omega) = 1$ (i.e. $\tilde{\mu} \geqq \tilde{q}$ and $V \neq \emptyset$), iff $\varrho_K(\omega) \geqq \mu$, set $\delta^*(\omega) = 0$ (i.e. $\tilde{\mu} = q$ and $V = \emptyset$) otherwise, i.e. if $\varrho_K(\omega) < \mu$. Considering the probability of error connected with this decision function, the threshold values μ outside the interval $(q, \tilde{q})$ can be evidently omitted, as in such a case the total probability of error would be always greater than that for some $\mu \in (q, \tilde{q})$. Let $V = \emptyset$, i.e., $\tilde{\mu} = q$, but an error has been made, so that $\delta^*(\omega) = 1$. It is the error of the second kind, which cannot occur in the model examined in Chapter 4. Now, $\delta^*(\omega) = 1$ iff $\varrho_K(\omega) \geqq \mu$, i.e. if $\varrho_K(\omega) - \tilde{\mu} \geqq \mu - q$, and $\mu - q$ can be written as c_1/N for appropriate c_1, $0 < c_1 \leqq 1$ due to (5.8). Because the probability distribution of ϱ_K is asymptotically symmetric with respect to $\tilde{\mu}$ for N, $K \to \infty$, the probability with which $\varrho_K(\omega) - \tilde{\mu} \geqq \mu - q$ holds can be written as

$$\tfrac{1}{2} P\left(\{\omega: \omega \in \Omega, \left|\varrho_K(\omega) - \tilde{\mu}\right| \geqq c_1/N\}\right) \tag{5.11}$$

in the sense that

$$\lim_{N, K \to \infty} \frac{P\left(\{\omega: \omega \in \Omega, \varrho_K(\omega) - \tilde{\mu} \geqq \mu - q\}\right)}{\tfrac{1}{2} P\left(\omega: \omega \in \Omega, \left|\varrho_K(\omega) - \tilde{\mu}\right| \geqq c_1/N\right\})} = 1 \tag{5.12}$$

Similarly, let $V \neq \emptyset$, i.e. $\tilde{\mu} \geqq \tilde{q} > q$, but the error occurs, so that $\delta^*(\omega) = 0$. It is the error of the first kind, investigated already above, in Chapter 4. This is possible iff $\varrho_K(\omega) < \mu$, hence $\varrho_K(\omega) - \tilde{\mu} = c_2/N$ for appropriate $c_2 \in (0, 1\rangle$. The same way of reasoning as above enables to approximate the probability of this error by

$$\tfrac{1}{2} P\left(\{\omega: \omega \in \Omega, \left|\varrho_K(\omega) - \tilde{\mu}\right| \geqq c_2/N\}\right) . \tag{5.13}$$

If we want to majorize both the probabilities of error by an $\varepsilon > 0$, choosing K large enough, the optimal solution is to take μ "in the middle of $(q, \tilde{q})$", hence $\mu = \frac{1}{2}(q + \tilde{q})$, then $c_1 = c_2$. As can be easily seen, if we want to obtain just a qualitative or "order" estimation of $K = K(N)$ up to a multiplicative constant, such a specification of μ is not necessary. In every case, probability of error resulting from the use of the decision function δ^* is uniformly (with respect to all $V \subset A$) majorized by $\varepsilon > 0$ iff for some $c \in (0, 1)$ and $\varepsilon' = \varepsilon/2$

$$P\left(\{\omega: \omega \in \Omega, \left|\varrho_K(\omega) - \tilde{\mu}\right| \geqq c/N\}\right) \leqq \varepsilon' . \tag{5.14}$$

The well-known Chebyshev inequality (cf. [1], e.g.) yields

$$P(\{\omega: \omega \in \Omega, |\varrho_K(\omega) - \tilde{\mu}| \geqq \delta\}) < \sigma^2(K\delta^2)^{-1}, \tag{5.15}$$

as the expected value of ϱ_K is $\tilde{\mu}$ and its dispersion is σ^2/K. Hence, for $\delta = c/N$

$$P(\{\omega: \omega \in \Omega, |\varrho_K(\omega) - \tilde{\mu}| \geqq c/N\}) \leqq N^2\sigma^2/Kc^2 = \text{const.}\, N^2/K, \tag{5.16}$$

and this value can be majorized by ε' only if $K \geqq c_3 N^2$ for an appropriate $c_3 > 0$ (namely, for $c_3 \geqq \sigma^2/c^2\varepsilon'$). As far as the constant c_3 is concerned, the estimation for K is very rough, however, it cannot be replaced by an estimation from $o(N^2)$, as the following consideration informally proves.

When applying the central limit theorem (cf. e.g., again [1]) to random variables $Y_1, Y_2, \ldots, Y_K$, we obtain

$$P\left(\left\{\omega: \omega \in \Omega, \alpha < \frac{\sum_{i=1}^{K} Y_i(\omega) - K\tilde{\mu}}{\sigma\sqrt{K}} < \beta\right\}\right) \to \Phi(\beta) - \Phi(\alpha), \tag{5.17}$$

$-\infty < \alpha < \beta < \infty$. Here the convergence is taken in the sense that the ratio of both the sides tends to one as $K \to \infty$ and Φ is the distribution function of the normal (or Gauss) probability distribution $N(0, 1)$. Modifying and simplifying (5.17) we obtain

$$P\left(\left\{\omega: \omega \in \Omega, \alpha < \frac{(K^{-1}\sum_{i=1}^{K} Y_i(\omega)) - \tilde{\mu}}{\sigma/\sqrt{K}} < \beta\right\}\right) \to \Phi(\beta) - \Phi(\alpha), \tag{5.18}$$

$$P\left(\left\{\omega: \omega \in \Omega, -\alpha\frac{\sigma}{\sqrt{K}} < \varrho_K(\omega) - \tilde{\mu} < \alpha\frac{\sigma}{\sqrt{K}}\right\}\right) =$$

$$= P\left(\left\{\omega: \omega \in \Omega, |\varrho_K(\omega) - \tilde{\mu}| < \alpha\frac{\sigma}{\sqrt{K}}\right\}\right) \to \Phi(\alpha) - \Phi(-\alpha). \tag{5.19}$$

If $\delta = \alpha(\sigma/\sqrt{K})$, then $\alpha = \delta(\sqrt{K}/\sigma')$, hence

$$P(\{\omega: \omega \in \Omega, |\varrho_K(\omega) - \tilde{\mu}| > \delta\}) \to 1 - \Phi\left(\delta\frac{\sqrt{K}}{\sigma}\right) - \Phi\left(-\delta\frac{\sqrt{K}}{\sigma}\right). \tag{5.20}$$

If $\delta = c/N$ and if the right-hand side in (5.20) is to be majorized by $\varepsilon' > 0$, K must be choosen in such a way that

$$\Phi\left(\frac{c}{N}\frac{\sqrt{K}}{\sigma}\right) - \Phi\left(-\frac{c}{N}\frac{\sqrt{K}}{\sigma}\right) > 1 - \varepsilon'. \tag{5.21}$$

However, if $K = K(N) \in o(N^2)$, then $\sqrt{(K(N))} \in o(N)$, so that $c\sqrt{K}/(N\sigma) \to \infty$ for $N \to \infty$, hence

$$\lim_{N\to\infty} \Phi\left(\frac{c}{N}\frac{\sqrt{(K(N))}}{\sigma}\right) = \lim_{N\to\infty} \Phi\left(-\frac{c}{N}\frac{\sqrt{(K(N))}}{\sigma}\right) = \Phi(0) = \tfrac{1}{2}. \tag{5.22}$$

So, we need $K(N) \in \mathcal{O}(N^2)$ in order to assure the validity of (5.21).

As ϱ_K is the arithmetical average of statistically independent and equally distributed random variables $Y_1, Y_2, \ldots, Y_K$ the dispersion of ϱ_K is uniformly (with respect to K) the minimum one among all unbiased estimates of $\tilde{\mu}$ based on $Y_1, Y_2, \ldots, Y_K$. Evidently, $\pi = \pi_k$ belongs to such estimates, hence, for each $\delta > 0$ and each k such that $K = kn$,

$$P(\{\omega: \omega \in \Omega, |\pi_k(\omega) - \tilde{\mu}| \leqq \delta\}) \geqq P(\{\omega: \omega \in \Omega, |\varrho_k(\omega) - \tilde{\mu}| \geqq \delta\}). \quad (5.23)$$

Hence, the extent $K' = kn$ of sample necessary to majorize the left-hand side in (5.23) by a given $\varepsilon > 0$ is at least so large as in the case when the right-hand side in (5.23) is to be majorized by the same ε. So, if $\delta = c/N$, then $K' = kn \geqq cN^2$ for an appropriate $c > 0$. The unit time complexity $TCU(\mathscr{X})$ of a threshold (two-level) hierarchical parallel probabilistic searching algorithm with characteristics m, n and k is defined by $\alpha_1 n + \alpha_2 k + \alpha_3$ for appropriate $\alpha_1, \alpha_2 > 0$, $\alpha_3 \geqq 0$. As we cannot choose, simultaneously, $k = k(N) \in o(N)$ and $n = n(N) \in o(N)$, neither $\alpha_1 n + \alpha_2 k + \alpha_3$ can be in $o(N)$. Using an analogous argumentation we can prove, that the logarithmic time complexity cannot belong to $o(N \log N)$. So we have proved

Theorem 5.1. Consider the model and notations introduced by (5.4), let

$$\tfrac{1}{2} > P(\{\omega: \omega \in \Omega, Q(0, \omega) = 1\}) > 0, \quad (5.24)$$
$$\tfrac{1}{2} > P(\{\omega: \omega \in \Omega, Q(1, \omega) = 0\}) \geqq 0.$$

Then there does not exist a threshold two-level hierarchical parallel probabilistic searching algorithm for $\langle A, V\rangle$ the probability of error of which would be majorized by a given $\varepsilon > 0$ uniformly for all $V \subset A$ and the unit time complexity of which would belong to $o(N)$-class (the logarithmic time complexity of which would belong to $o(N \log N)$-class, resp.).

The negative result just proved is intuitively easy to understand. If an element outside V can be wrongly tested as being in V, the test cannot be terminated having obtained the single information that an element of V had been found. So, the relative frequency of such reports in a long series of samples must be taken into consideration. A more detailed analysis of what it means "a sufficiently long" series leads to demands incompatible with capabilities of two-level hiearrchical parallel probabilistic algorithms with $o(N)$ – ($o(N \log N)$-, resp.) time complexity. On the other hand, if all elements from $A - V$ are decided correctly and just some elements from V can be "omitted", then $o(\sqrt{N})$, re p. $o(\sqrt{N \log N})$-time complexitives are reachable, just with multiplicative constants enlarged.

Theorem 5.2. Consider the model and notation introduced by (5.4), let

$$P(\{\omega: \omega \in \Omega, Q(0, \omega) = 1\}) = 0, \quad (5.25)$$
$$\tfrac{1}{2} > P(\{\omega: \omega \in \Omega, Q(1, \omega) = 0\}) = 1 - p \geqq 0.$$

Then there exists *HPPSA* in the sense of Chapter 4 (i.e. not in the thershold sense) for the problem $\langle A, V\rangle$ the probability of error of which is uniformly majorized

by a given $\varepsilon > 0$ over all $V \subset A$ and its unit (logarithmic, resp.) time computational complexity is in $\mathcal{O}(\sqrt{N})$-class (in $\mathcal{O}(\sqrt{(N)} \log N)$-class, resp.)

Proof. Setting $v = \text{card } V$,

$$P(\{\omega: \omega \in \Omega, X_{ij}(\omega) \in V\}) = vN^{-1},$$

$$P(\{\omega: \omega \in \Omega, Q(\chi_V(X_{ij}(\omega)), \omega) = 1\}) = pvn^{-1}. \tag{5.26}$$

If $v = 0$, assertion holds trivially, if $v > 0$, then discovering of an element of V implies that $v \neq \emptyset$ and situation defined by (5.25) is the same as in the case of failure-proof two-level *HPPSA* with $v' = vp$, hence, as in the case of $\langle A_0', V' \rangle$ with card V' : : card $A_0' = v'N^{-1} = vpN^{-1}$ (e.g., $V' = V$, $A \subset A_0'$, card. $A_0' = N_1 = N/p$). As there exists a two-level *HPPSA* for $\langle A_0', V \rangle$ with time computational complexity in $\mathcal{O}(\sqrt{N_1})$-class (in $\mathcal{O}(\sqrt{N_1}) \log N_1$-class, resp., $N_1 = \text{card } A_0'$), and because of the simple fact that $\sqrt{N_1} = \sqrt{(1/p)}\sqrt{N}$ and $\sqrt{(N_1)} \log N_1 = \sqrt{(1/p)}\sqrt{(N)} \log N + \log p^{-1})$, the classes $\mathcal{O}(\sqrt{N})$ and $\mathcal{O}(\sqrt{N_1})$ ($\mathcal{O}(\sqrt{(N)} \log N)$ and $\mathcal{O}(\sqrt{(N_1)} \log N_1)$ resp.) are identical and the assertion is proved.

Idealized version of the threshold *HPPSA* is the algorithm obtained when the decision function $\delta(\delta(\omega) = 1$, if $\pi(\omega) \geqq \mu$, $\delta(\omega) = 0$ otherwise) is replaced by the decision function δ^*, where $\delta^*(\omega) = 1$, if $\varrho(\omega) \geqq \mu$, $\delta^*(\omega) = 0$ otherwise.

Theorem 5.3. Let the notations and conditions of Theorem 5.1 hold, let a function f: $\mathfrak{N} \to \mathfrak{N}$ be given, $f(N) \to \infty$ as $N \to \infty$. Then there exists two-level threshold *HPPSA* the idealized version of which has probability of error majorized by a given $\varepsilon > 0$ uniformly for all searching problems $\langle A, V \rangle$ such that card $V \geqq f(\text{card } A)$, moreover, the unit (logarithmic, resp.) time computational complexity of this algorithm is in $\mathcal{O}(N/f(N))$ (in $\mathcal{O}(N\delta \log N/f(N))$, resp.)-class, hence, in $o(N)$ (in $o(N \log N)$, resp.)-class.

Proof. The proof copies the pattern of the proof of Theorem 5.1, so we mention explicitly just the main points. We test, whether $V = \emptyset$ $(v = 0)$ against the alternative that $v \geqq f(N)$, $N = \text{card } A$, on the ground of $Y_1(\omega), Y_2(\omega), \ldots, Y_K(\omega)$, $K = nm$. Setting p and q as in (5.1), we can easily deduce, that if $v = v(N) = \text{card } V$, then

$$\begin{aligned} &P(\{\omega: \omega \in \Omega, Y_i(\omega) = 1\}) = \\ &= q\, P\{\omega: \omega \in \Omega, Y_i(\omega) \in A - V_j\} + (1 - p)\, P(\{\omega: \omega \in \Omega, Y_i(\omega) \in V_j\}) = \\ &= q(1 - v(N)/N) + (1 - p)(v(N)/N) = q + (1 - p - q)(v(N)/N) = \\ &= p_V(N). \end{aligned} \tag{5.27}$$

Conditions of Theorem 5.1 yield that $p + q < 1$, hence, we test $\tilde{\mu}\,(= EY_i) = p_0 = q$ against $\tilde{\mu} = p_{v(N)} > p_0$.

Recall that $\delta^*(\omega) = 1$ iff $\varrho(\omega) = K^{-1} \sum_{i=1}^{K} Y_i(\omega) \geqq \mu$, then there exists, for each threshold value $\mu \in (p_0, p_{v(N)})$, a constant $c_1 > 0$ such that δ^* yields an error if

$$\left|K^{-1} \sum_{i=1}^{K} Y_i(\omega) - \mu\right| \geqq \delta_N = c_1\, v(N)/N. \tag{5.28}$$

We use, again, the Chebyshev inequality and the fact that estimation of K as a function of N, obtained from this inequality, is the best one up to multiplicative constant. We obtain

$$P\left(\left\{\omega\colon \omega \in \Omega, \left|\varrho(\omega) - \mu\right| > c_1 \frac{v(n)}{N}\right\}\right) \leqq \frac{D^2 Y}{(c_1\, v(N)/N)^2\, K}\,. \tag{5.29}$$

Majorizing the right-hand side of (5.29) by $\varepsilon > 0$, we obtain

$$K = K(N) > \frac{D^2 Y}{c_1 \varepsilon} \frac{N^2}{(v(N))^2} = c_2 \frac{N^2}{(v(N))^2}\,, \tag{5.30}$$

and this function is in $(N^2/(f(N))^2)$, if $v = f(N)$ (the worst case). Hence, for $m(N) \in$ $\in \mathcal{O}(\sqrt{(K(N))})$, $n(N) \in o(\sqrt{(K(N))})$ the unit time computational complexity for $\delta^*(\omega)$ is $\alpha_1\, m(N) + \alpha_2\, n(N) + \alpha_3$, hence is in $\mathcal{O}(N/f(N)) \subset o(N)$-class. The assertion for logarithmic time complexity follows in the same way. □

Another possibility to overcome the negative results of Theorem 5.1 consists in an appropriate change of the used loss function, i.e., in a relativization of the loss suffered when the decision about the set V is wrong. As can be easily seen, probability of error is nothing else than the expected value of the most simple loss function taking the value 0, if the decision is correct, and taking the value 1 if the decision is wrong.

If $V = \emptyset$, then the wrong decision that $V \neq \emptyset$, if it is possible, i.e., if $P(\{\omega\colon \omega \in \Omega, Q(0, \omega) = 1\}) > 0$, is a qualitative decision not implying some more consequences concerning the set V or its elements. Such consequences would be, necessarily, either tautological or false. It is why the loss connected with this kind of error can be taken, in what follows, as unit or constant. On the other hand, the loss connected with the wrong decision that $V = \emptyset$, if $V \neq \emptyset$, will be taken as an increasing function, say F, of the relative frequency v/N of elements from V in A. In this moment we suppose only that F is a strictly increasing and continuous mapping of the unit interval $\langle 0, 1\rangle$ onto itself, so that $F(0) = 0$ and $F(1) = 1$.

Formally, *loss function* R maps the Cartesian product $\mathscr{D} \times \mathscr{P}(\mathscr{A})$ into the set $\langle 0, \infty)$ of reals. Here $\mathscr{D} = \{D_0, D_1\}$ is the set of possible decision (D_0 is "$V = \emptyset$", D_1 is "$V \neq \emptyset$") and $\mathscr{P}(\mathscr{A}) = \{V\colon V \subset A\}$. According to the informal considerations above we assume that

$$R(D_0, V) = F(\text{card } V/\text{card } A) \quad \text{for each} \quad V \in \mathscr{P}(A)\,, \tag{5.31}$$

hence,

$$R(D_0, \emptyset) = 0\,, \quad R(D_1, V) = 0 \quad \text{for each} \quad V \in \mathscr{P}(A) - \{\emptyset\}\,,$$
$$R(D_1, \emptyset) = \gamma > 0\,.$$

Now, considering decision functions δ or δ^*, defined by $\pi(\omega)$ or by $\varrho(\omega)$, as random variables, also the loss function can be taken as a random variable ($R(\delta(\omega), V)$ or $R(\delta^*(\omega), V)$) with values in $\langle 0, \infty)$. All random variables used to define $R(\delta(\cdot), V)$ and $R(\delta^*(\cdot), V)$ are uniquely determined by random variables X_{ij} and Z_l, hence,

the assumptions concerning the uniform probability distributions of the latter random variable define uniquely the expected values $E(R(\delta(\cdot), V))$ and $E(R(\delta^*(\cdot), V))$. These expected values depend, of course, on the parameters m, n, k of the used *HPPSA* and we shall investigate the conditions under which, given $\varepsilon > 0$,

$$E(R(\delta(\cdot), V)) < \varepsilon\,, \quad \text{or} \quad E(R(\delta^*(\cdot), V)) < \varepsilon \tag{5.32}$$

hold uniformly for all $V \subset A$.

However, if $V = \emptyset$, then

$$\begin{aligned} &E(R(\delta^*(\cdot), \emptyset) = \gamma \,.\, P(\{\omega\colon \omega \in \Omega, \varrho(\omega) \geqq \mu\}) = \\ &= (\gamma/2)\, P(\{\omega\colon \omega \in \Omega, |\varrho(\omega) - \tilde{\mu}| \geqq c/N\}) \end{aligned} \tag{5.33}$$

for some $c > 0$, if $\tilde{\mu} = q = P(\{\omega\colon \omega \in \Omega,\ Q(0, \omega) = 1\}) > 0$. For the reasons explained above, the right-hand side of (5.33) cannot be majorized by an $\varepsilon > 0$ with $K = mn \in o(N^2)$, what leads to the same unpleasant consequences as those in Theorem 5.1. It is why we shall limit ourselves only to nonempty V's when considering (5.32).

If $v = \operatorname{card} V \geqq 1$, an easy calculation yields

$$\begin{aligned} &E(R(\delta^*(\omega), V) = F(v/N) \,.\, P(\{\omega\colon \omega \in \Omega, \delta^*(\omega) = D_0\}) = \\ &= F(v/N)\, P(\{\omega\colon \omega \in \Omega, \varrho(\omega) < \mu\}) = \\ &= \tfrac{1}{2}\, F(v/N) \,.\, P(\{\omega\colon \omega \in \Omega, |\varrho(\omega) - \tilde{\mu}| \geqq c/N\}) \leqq \\ &\leqq (\tfrac{1}{2})\, F(v/N)\, \frac{D^2 Y}{(c/N)^2\, K} \end{aligned} \tag{5.34}$$

for an appropriate constant $c > 0$. Hence, to majorize the right-hand side of (5.34) by $\varepsilon > 0$, we must take $K = K(N) = c_1 N^2\, F(v/N)$ for an appropriate constant $c_1 > 0$. As $F(v/N) \to 0$ for $N \to \infty$ (v is fixed), $K(N) \in o(N^2)$. So we have proved

Theorem 5.4. For A, V, N and v as above, there exists a two-level threshold *HPPSA*, with generalized loss function defined by (5.31) and with F satisfying the conditions above, such that the idealized version of this *HPPSA* has probability of error majorized by a given $\varepsilon > 0$ uniformly for all nonempty $V \subset A$. Its unit time computational complexity is in $\mathcal{O}(N \sqrt{(F(v/N))})$-class, i.e. in $o(N)$-class (its logarithmic time computational complexity is in $\mathcal{O}(N \log N \sqrt{(F(v/N))})$-class, i.e. in $o(N \log N)$-class).

The assertion concerning the logarithmic complexity can be easily proved in the same way as above. Hence, if $F(v/N) = v/N^2$, then the unit time complexity is in $\mathcal{O}(\sqrt{v})$-class. So, for each $0 < v \leqq \operatorname{card} A$ there exists a positive integer $w(v)$ such that the idealized version of an appropriate two-level threshold *HPPSA* with unit time complexity $w(v)$ has probability of error smaller than $\varepsilon > 0$ uniformly for all $\emptyset \neq V \subset A$ and independently of N. The loss function $F(v/N) = v/N^2$ com-

pletely eliminates the negative influence of searching problems with large A's and small nonempty V's as far as the time computational complexity of parallel as well as sequential searching algorithms is concerned. What is important is the fact that this assertion holds even in case when the probability of the wrong decision about elements of $A - V$ is positive, i.e. in the case of threshold algorithms. The problem is discussed, in more details, in [2] and [3], for the case of sequential algorithms and using a slightly different terminology.

As an immediate and natural continuation of our former considerations we may consider the *Bayesian model*, when the tested set V is supposed to be sampled at random and the algorithm is classified according to the expected value of the loss function with respect to the apriori distribution. So, consider a random variable $\mathscr{V}$ taking the probability space $\langle \Omega, \mathscr{S}, P \rangle$ into $\mathscr{P}(A)$ and define the Bayesian risk $\tilde{E}(E(R(\delta^*(\cdot), \mathscr{V}(\cdot))))$ as follows:

$$\tilde{E}(E(R(\delta^*(\cdot), \mathscr{V}(\cdot)))) = \sum_{V \in \mathscr{P}(A)} [E(R(\delta^*(\cdot), V))] \cdot P(\{\omega: \omega \in \Omega, \mathscr{V}(\omega) = V\}) . \tag{5.35}$$

Supposing that the value of $E(R(\delta^*(\cdot), V))$ depends only on the cardinality of V, Bayesian risk can be read as

$$\sum_{n=0}^{N} [E(R(\delta^*(\cdot), V_n))] \cdot P(\{\omega: \omega \in \Omega, \text{card } \mathscr{V}(\omega) = n\}) , \tag{5.36}$$

where $N = \text{card } A$ and V_n is any n-element subset of A. The demand to majorize the Bayesian risk by an $\varepsilon > 0$ is, of course, much weaker than that of uniform majorization and the choose of $\mathscr{V}$ ultimately determines the quality of the algorithm in question. The next example shows than even in case of a quite natural apriori distribution the majorization of the Bayesian risk can be reached within a constant (i.e. independent of N) unit time computational complexity.

Let $U_1, U_2, \ldots, U_N$, $N = \text{card } A$, be statistically independent and equally distributed random variables, each of them taking $\langle \Omega, \mathscr{S}, P \rangle$ into $\{0, 1\}$, denote

$$\alpha = P(\{\omega: \omega \in \Omega, U_i(\omega) = 1\}) . \tag{5.37}$$

Given an ordering $A = \{a_1, a_2, \ldots, a_N\}$, set

$$\mathscr{V}(\omega) = \{a_i: a_i \in A, U_i(\omega) = 1\} \tag{5.38}$$

and suppose, to avoid the trivial cases, that $0 < \alpha < 1$. Evidently, for each $0 \leqq n \leqq N$,

$$P(\{\omega: \omega \in \Omega, \text{card } \mathscr{V}(\omega) = n\}) = \binom{N}{n} \alpha^n (1 - \alpha)^{N-n} , \tag{5.39}$$

and $E \text{ card } \mathscr{V}(\cdot) = \alpha N$. Consider the most simple two-level *HPPSA* with $p = q = 1$, i.e. with reliable testing oracles) and with parameters m, n, k. The only we have to prove is that, given $\varepsilon > 0$, there exists $K = mn$ independent of N such that the Bayesian risk that no processor finds an element of $V = \mathscr{V}(\omega)$ is majorized by ε.

This Bayesian risk reads, as can be easily seen

$$\sum_{n=1}^{N}(1-(n/N))^K\binom{N}{n}\alpha^n(1-\alpha)^{N-n}, \tag{5.40}$$

as no error can occur if $n = 0$.

Choose $\delta < \min\{\alpha, 1-\alpha\}$, the strong law of large numbers yields that

$$P(\{\omega\colon \omega\in\Omega, \lim_{N\to\infty} N^{-1}\operatorname{card}\mathscr{V}(\omega) = \alpha\}) = 1\,. \tag{5.41}$$

Hence, there exists $N_0 = N_0(\varepsilon,\delta)$ such that, for each $N > N_0$,

$$P(\{\omega\colon \omega\in\Omega, N^{-1}\operatorname{card}\mathscr{V}(\omega)\in\langle\alpha-\delta,\alpha+\delta\rangle\}) > 1-(\varepsilon/2)\,. \tag{5.42}$$

Set $M_1 = \lceil(\alpha-\delta)N\rceil$, $M_2 = \lfloor(\alpha+\delta)N\rfloor$ and divide the sum in (5.40) into three parts: from $n = 1$ to M_1, from $n = M_1+1$ to M_2, and from $n = M_2+1$ to N (the first and the third one may be empty). Replace $(1-(n/M))^K$ by 1 in the first and third sum and $1-(v/N)$ by $1-(\alpha-\delta)$ in the second one. The Bayesian risk in question can be majorized, consequently, by

$$\begin{aligned}&\sum_{n=1}^{M_1}\binom{N}{n}\alpha^n(1-\alpha)^{N-n} + \sum_{n=M_1}^{M_2} q^K\binom{N}{n}\alpha^n(1-\alpha)^{N-n} + \\ &+\sum_{n=M_2+1}^{N}\binom{N}{n}\alpha^n(1-\alpha)^{N-n} = P(\{\omega\colon \omega\in\Omega, \operatorname{card}\mathscr{V}(\omega)\langle M_1\}) + \\ &+ q^K P(\{\omega\colon \omega\in\Omega, M_1\leqq\operatorname{card}\mathscr{V}(\omega)\leqq M_2\}) + \\ &+ P(\{\omega\colon \in\Omega, \operatorname{card}\mathscr{V}(\omega) > M_2\}) \leqq \varepsilon/2 + q^K\,,\end{aligned} \tag{5.43}$$

here $q = 1-(\alpha-\delta)$. Taking $K > (\log_2(\varepsilon/2))/\log(1-\alpha+\delta)$, the right-hand side of (5.43) is majorized by ε. Hence, if $N < N_0(\varepsilon,\delta)$, the uniform majorization of the loss function implying trivially also the majorization of the Bayesian risk can be easily reached by appropriate $K = K(\varepsilon)$ independent of N, if $N \geqq N_0(\varepsilon,\delta)$, and the example is closed.

The Bayesian model is connected with all the philosophical, methodological and practical problems involved by the Bayesian approach in statistical decision making in general and it is not the aim of this work to reproduce or to enlarge this discussion. In every case, the Bayesian approach is practically justified mainly in case we have some reasons for a choice of a particular apriori probability distribution or at least for giving a preference to a narrow class of such distributions. However, such reasons or justifications must be, of course, of extra-mathematical nature, i.e., must be implied by an apriori, perhaps partial, knowledge of the set V, but such an assumption contradicts to the "black-box principle" adopted above (introduction of Chapter 4). It is why we shall not develop the Bayesian model in more details and close this chapter by considering another example which can be taken as alternative to the one introduced above. Namely, let us take such an apriori distribution that generates

equiprobable distribution on the set $\{0, 1, 2, \ldots, N\}$ of possible cardinalities of the tested set V, hence,

$$P(\{\omega: \omega \in \Omega, \text{card } \mathscr{V}(\omega) = n\}) = (N + 1)^{-1}, \quad n = 0, 1, \ldots, N. \tag{5.44}$$

Evidently, the Bayesian risk reads

$$\sum_{i=1}^{N} (N + 1)^{-1} (1 - i/N)^K. \tag{5.45}$$

Using the well-known fact that $(1 - x/N)^N \nearrow e^{-x}$ for $N \to \infty$, an easy calculation yields

$$\sum_{i=1}^{N} \frac{1}{N+1} \left(1 - \frac{i}{n}\right)^K < \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{i}{N}\right)^K =$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{i(K/N)}{K}\right)^K < \sum_{i=1}^{N} \frac{1}{N} (e^{-(iK/N)}) =$$

$$= \sum_{i=1}^{N} \frac{1}{N} \left(\frac{1}{e^{K/N}}\right)^i = \frac{1}{N} \frac{(1/e^{K/N}) - (1/e^K)}{1 - (1/e^{K/N})} = \frac{1}{N} \frac{e^K - e^{K/N}}{e^K (e^{K/N} - 1)} < \frac{1}{N} \frac{1}{e^{K/N} - 1}. \tag{5.46}$$

If $K > 1/\varepsilon$, then $K/N > 1/(\varepsilon N)$, hence,

$$\sum_{i=0}^{\infty} (K/N)^i/i! = e^{K/N} = 1 + K/N + \sum_{i=2}^{\infty} (K/N)^i/i! > 1 + 1/(\varepsilon N), \tag{5.47}$$

so that

$$1/(e^{K/N} - 1) < \varepsilon \tag{5.48}$$

and the Bayesian risk is majorized by ε uniformly for all N and all $n \leqq N$, similarly as in the example above.

REFERENCES

[1] W. Feller: An Introduction to Probability Theory and its Applications I, II. J. Wiley and Sons, New York, 1957 (vol. I, 2nd edition), 1966 (vol. II). Russian translation: Mir, Moscow, 1964, 1967.

[2] I. Kramosil: Statistical verification procedures for propositional calculus. Computers and Artificial Intelligence *2*, (1983), 3, 235—258.

[3] I. Kramosil and J. Šindelář: Computational complexity of probabilistic searching algorithms over Herbrand universes. Computers and Artificial Intelligence *4* (1985), 2, 97—110.

6. PARALLEL ALGORITHMS FOR MONTE-CARLO METHODS

In a number of problems of various nature, when taking a decision or looking for a solution such characteristics or parameters of the investigated systems are of importance, which can be defined as mean values of certain random variables. Because of many reasons of theoretical as well as practical kinds these values need not be immediately accessible (measurable or observable), they need not be even accessible through an appropriate computation. When looking for appropriate approximations or estimations of such values the following straightforward idea arises – to take several statistically independent realizations of the random variable in question, to compute the average value of these realizations and to consider this value as an estimation of the unknown expected value. The soundness of such an argumentation is guaranteed by the well-known *strong law of large numbers* according to which the average value of an increasing number of statistically independent realizations of the same random variable tends, with the probability one, to its expected value. In symbols, if $X_1, X_2, \ldots$ is a sequence of statistically independent and equally distributed random variables with expected value $EX < \infty$ and dispersion $D^2X < \infty$, defined on an abstract probability space $\langle \Omega, \mathscr{S}, P \rangle$, then

$$P(\{\omega: \omega \in \Omega, \lim_{n \to \infty} (1/n) \sum_{i=1}^{n} X_i(\omega) = EX\}) = 1 . \tag{6.1}$$

In which sense and in which degree the average value $(1/n) \sum_{i=1}^{n} X_i(\omega)$ can serve as a "good enough" approximation of EX? This is described by Chebyshev inequality, another well-known basic result of probability theory which sounds, that for each $\varepsilon > 0$

$$P(\{\omega: \omega \in \Omega, |(1/n) \sum_{i=1}^{n} X_i(\omega) - EX| \geqq \varepsilon\}) < D^2X/n\varepsilon^2 . \tag{6.2}$$

In general, as already mentioned, the term *Monte-Carlo method* covers any computational or decision-making procedure in which the expected value of some random variable is replaced by the average value computed from a number of statistically independent realizations and the resulting risks are accepted. The supposed independence of random realizations (samples) immediately involves. as in the foregoing chapters, the idea of a possible parallel realizations of these samples. A more detailed investigation of such possibilities will be our aim in this chapter. It is why we may borrow the model defined in Chapter 4 in order to consider a finite set $A = \{a_1, a_2, \ldots, a_N\}$ together with its subset V accessible only through a testing oracle which works on the "black box" principle. However, instead of investigating, whether $V = \emptyset$ or not, our aim will be to obtain the value $p = \text{card } V/\text{card } A$ or at least an estimation of this ratio, which is "good" in a sound and formally definable sense.

Supposing that one processor inspects systematically the set A, the correct value for p can be obtained in $\mathcal{O}(N)$-time. Here we neglect the time demands to assure the

systematical character of the searching process (each element tested just once), to enregister the number of elements from V, and to compute the resulting ratio. By an unlimited parallelization which uses a complete, but from the point of view of computational complexity "free of charge" co-operation and synchronization, the constant, i.e. independent of N, time complexity would be achievable. However, because of highly idealized assumptions connected with such a model we shall not go into details and concentrate our attention to algorithms of probabilistic kind. In such a case, first of all, we have to specify, what it means a "good enough" estimation of p. In what follows, we shall take (6.2) as a good quantitative measure of the quality of the estimation $(1/n)\sum_{i=1}^{n} X_i(\omega)$ of the expected value EX. According to this relation, the quality of estimation is parametrized by two values $\varepsilon > 0$, giving the maximal acceptable absolute difference between the average and expected value, and $\delta > 0$ giving the maximal value of probability with which the threshold value ε is crossed. This other parameter value, i.e. δ, will be always taken as a free parameter of our model, the value of which is determined by exclusively extra-mathematical reasons of practical and applicational nature. In case the value $\varepsilon > 0$ is also independent of the extent N of the set A, we may take

$$n \geqq D^2(\chi_V(X_i(\omega)))/\varepsilon^2\delta = p(1 - p)/\varepsilon^2\delta \tag{6.3}$$

in order to assure that for n statistically independent random samples from the uniform probability distribution over A the inequality

$$P(\{\omega\colon \omega \in \Omega, |(1/n)\sum_{i=1}^{n} \chi_V(X_i(\omega)) - p| \geqq \varepsilon\}) < \delta \tag{6.4}$$

will hold. Here $X_i\colon \langle \Omega, \mathscr{S}, P\rangle \to A$ is a sequence of statistically independent random variables such that

$$P(\{\omega\colon \omega \in \Omega, X_i(\omega) = a\}) = 1/N \tag{6.5}$$

for each $i = 1, 2, \ldots$, and each $a \in A$. As $p(1 - p) \leqq \frac{1}{4}$ for each $p \in \langle 0, 1\rangle$, we may take $n \geqq \frac{1}{4}\varepsilon^2\delta$ independent of p. The time complexity of a probabilistic (sequential or parallel) algorithm for obtaining an estimation $\bar{p}(\omega)$ of p satisfying (6.4) is, therefore, constant (independent of N). Because of the fact that this dependence will lie in the centre of our further considerations, this case will not be developed in more details.

In what follows, let us focus our attention to the ways in which so called *δ-correct* estimates of the value $p = \text{card } V/\text{card } A$ can be obtained through an appropriate hierarchical structure of parallel probabilistic algorithms. The estimate

$$\bar{p}(\omega) = (1/n)\sum_{i=1}^{n} \chi_V(X_i(\omega)) \tag{6.6}$$

of the value p is called δ-correct supposing it satisfies (6.4) for $\varepsilon = 1/2N$. If it is the case, then with probability at least $1 - \delta$ the true value of p is in the interval $(\bar{p}(\omega) -$

$-1/2N,\ \bar{p}(\omega)+1/2N)$ of reals; this interval contains at most one value of the form v/N, hence, with the probability at least $1-\delta$ the correct value.

Consider N_1 processors which work in parallel, each of them repeating statistically independent random samples from the set A with respect to the uniform probability distribution over A. The sampled elements are tested and sent to the higher-level processors together with the total number of sampled elements (with possible repetitions) and the number of samples belonging to V. Intuitively, higher-level processors can be taken in the same sense as in Chapter 4. First of all, consider the case when the higher-level processors are hierarchically structured in such a way that to each higher level processor a fixed, finite and nonempty, in order to avoid the degenerated cases in what follows, set of processors of highest lower level is ascribed (subjected, ordered). These sets define disjoint covering at each level, so that each processor, with the exception of the (supposedly unique) highest-level one, is subjected to just one processor of the lowest higher level. However, the higher-level processors are supposed not to be able to inspect the outputs of the subjected processors systematically and exhaustively, but only on the ground of a random sample from the set of the subjected processors. For the sake of simplicity let us assume, first, the uniform probability distribution over the set of (immediately) subjected processors. Hence, the activity of each higher-level processor consists in taking a finite number of statistically independent random samples from the set of its subjected processors with respect to the uniform probability distribution and without any possibility to recognize and perhaps eliminate possible repetitions. Then the higher-level processor asks each of the sampled subjected ones for the total number of samples made by this subjected processor and for the number of samples which were in V. Finally, the higher-level processor sums both the series of numbers and sends the result to its superior supposing it is asked to do so. The unique highest-level processor combines in the same way the results obtained from its subjected processors sampled at random and outputs the ratio of the two sums as an estimation of the unknown value p. In the formal description presented below we shall consider a slightly modified version of this approach, when *each* processor computes the relative frequency of samples belonging to V or of at random sampled outputs containing the information about an element of V, and only those relative frequencies are, in case the corresponding outputs were sampled, sent to the higher-level processors.

Let $K \geqq 1, N_1, N_2, \ldots, N_K$ be natural numbers (numbers of processors of particular levels) such that $N_1 > N_2 > \ldots > N_K = 1$, let $k_1, k_2, \ldots, k_K$ be positive integers. Let $\tilde{X}_{ij}$, $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, k_1$, be statistically independent random variables, defined on the fixed probability space $\langle \Omega, \mathcal{S}, P \rangle$ and taking their values in A. Let, for each $i \leqq N_1, j \leqq k_1, a \in A$,

$$P(\{\omega: \omega \in \Omega, \tilde{X}_{ij}(\omega) = a\}) = 1/N\,. \tag{6.7}$$

Set

$$X_{ij}(\cdot) = \chi_V(\tilde{X}_{ij}(\cdot))\,, \tag{6.8}$$

where χ_V is the characteristic function of the set $V \subset A$. Hence, X_{ij} are statistically independent random variables taking $\langle \Omega, \mathscr{S}, P \rangle$ into $\{0, 1\}$ in such a way that, for each $i \leqq N_1, j \leqq k_1$,

$$P(\{\omega: \omega \in \Omega, X_{ij}(\omega) = 1\}) = p\,, \tag{6.9}$$
$$P(\{\omega: \omega \in \Omega, X_{ij}(\omega) = 0\}) = 1 - p\,.$$

Set, for each $i \leqq N_1$,

$$Y_i^1(\omega) = (1/k_1) \sum_{j=1}^{k_1} X_{ij}(\omega)\,. \tag{6.10}$$

Evidently, $Y_1^1, Y_2^1, \ldots, Y_{N_1}^1$ are statistically independent random variables with the expected value EY^1 and dispersion D^2Y^1 identical for each $i \leqq N_1$ and such that

$$EY^1 = EX_{ij} = p\,, \quad D^2Y^1 = (1/k_1)\, D^2X_{ij} = (1/k_1)\, p(1 - p)\,. \tag{6.11}$$

For each $i = 1, 2, \ldots, K - 1$, let $\mathscr{R}_i$ be a decomposition of the set $\{1, 2, \ldots, N_i\}$ of integers into N_{i+1} disjoint nonempty subsets $\mathfrak{N}_1^i, \mathfrak{N}_2^i, \ldots, \mathfrak{N}_{N_{i+1}}^i$, hence,

$$\bigcup_{j=1}^{N_{i+1}} \mathfrak{N}_j^i = \{1, 2, \ldots, N_i\}\,, \quad k \neq j \Rightarrow \mathfrak{N}_k^i \cap \mathfrak{N}_j^i = \emptyset\,. \tag{6.12}$$

For each $i \leqq K - 1$, $k \leqq N_{i+1}$, $l \leqq k_{i+1}$, let $Z_{i,k,l}$ be a random variable taking $\langle \Omega, \mathscr{S}, P \rangle$ into $\mathfrak{N}_k^i$, hence, for each $s \in \mathfrak{N}_k^i$,

$$P(\{\omega: \omega \in \Omega, Z_{i,k,l}(\omega) = s\}) = (\operatorname{card} \mathfrak{N}_k^i)^{-1}\,. \tag{6.13}$$

All the random variables $Z_{i,k,l}$ are supposed to be statistically independent and we set, for each $i \leqq K - 1$, $k \leqq N_{i+1}$,

$$Y_k^{i+1}(\omega) = (1/k_{i+1}) \sum_{j=1}^{k_{i+1}} Y^i_{Z_{i,k,j}(\omega)}\,. \tag{6.14}$$

Namely,

$$Y_1^K(\omega) = (1/k_K) \sum_{j=1}^{k_K} Y^{K-1}_{Z_{K-1,1,j}(\omega)}\,. \tag{6.15}$$

This value will be denoted by $Y^*(\omega)$ with an explicit introduction of other parameters, if necessary and will be understood as a statistical estimation of the unknown value p. The quality of this estimation will be compared with the quality of the best or optimal (in the sense of the minimal dispersion) estimation of p obtainable from the set $\{\tilde{X}_{ij}(\omega)\}_{i=1}^{N_1}{}_{j=1}^{k_1}$ of observations. This estimation will be denoted by Y_0 with an explicit introduction of other parameters, if necessary and it simply reads as

$$Y_0(\omega) = (1/N_1k_1) \sum_{i=1}^{N_1} \sum_{j=1}^{k_1} X_{ij}(\omega) = (1/N_1k_1) \sum_{i=1}^{N_1} \sum_{j=1}^{k_1} \chi_V(\tilde{X}_{ij}(\omega))\,. \tag{6.16}$$

This estimation is accessible under the condition of full cooperation and synchronization of processors and its characteristics are defined by

$$EY_0 = p\,, \quad D^2Y_0 = (1/N_1k_1)\, p(1 - p)\,. \tag{6.17}$$

Clearly, $EY^* = EY_0 = p$. Well-known assertions of probability theory then yield,

that the (arithmetical) average value is the unbiased estimation of the expected value the dispersion of which is minimal among all unbiased estimations of this value, hence, $D^2Y^* \geqq D^2Y_0$. In the rest of this chapter we shall investigate the way, in which D^2Y^* depends on $k_1, k_2, \ldots, k_K$. Or, their sum $k_1 + k_2 + \ldots + k_K$ represents, in the roughest approximation, the time complexity of the parallel probabilistic algorithm which estimates the value $p = \text{card } V/\text{card } A$ and which was described above. For the sake of abbreviation we shall refer to this algorithm as to $\mathscr{A}_1$ with further parameters possibly introduced.

Lemma 6.1. Let $Y_1, Y_2, \ldots, Y_M$ be statistically independent random variables, equally distributed and taking the probability space $\langle \Omega, \mathscr{S}, P \rangle$ into the Borel line, let $EY_i = \mu < \infty$, $D^2Y_i = \sigma^2 < \infty$ for all $i \leqq M$. Let $Z_1, Z_2, \ldots, Z_l$ be statistically independent and equally distributed random variables, taking $\langle \Omega, \mathscr{S}, P \rangle$ into the set $\{1, 2, \ldots, M\}$ of reals in such a way that

$$P(\{\omega: \omega \in \Omega, Z_i(\omega) = j\}) = M^{-1} \tag{6.18}$$

for each $i \leqq k$ and $j \leqq M$. Set

$$Y^* = (1/k) \sum_{j=1}^{k} Y_{Z_j(\omega)}, \tag{6.19}$$

then

$$EY^* = \mu, \quad D^2Y^* =$$

$$= D^2((1/M) \sum_{i=1}^{M} Y_i)(1 + (M - 1)/k) = (\sigma^2/M)(1 + (M - 1)/k). \tag{6.20}$$

Proof. It is known or can be easily checked, that if X, Y are statistically independent random variables and a, b are non-negative reals such that $a + b > 0$, then

$$D^2(aX) = a^2 D^2(X), \quad D^2(X + Y) = D^2(X) + D^2(Y). \tag{6.21}$$

Hence,

$$D^2\left(\frac{1}{a + b}(aX + bY)\right) = \frac{a^2}{a^2 + b^2} D^2X + \frac{b^2}{a^2 + b^2} D^2Y. \tag{6.22}$$

By induction, for each M-tuple $\langle b_1, b_2, \ldots, b_M \rangle$ of non-negative integers such that $\sum_{j=1}^{M} b_j = k$,

$$D^2\left(\frac{1}{k} \sum_{j=1}^{M} b_j Y_j\right) = \frac{\sigma^2}{k^2} \sum_{j=1}^{M} b_j^2. \tag{6.23}$$

Denote by $A_i(\omega)$, $1 \leqq i \leqq M$, the random variable giving the number, how many times the value i has been sampled by some Z_j, $j \leqq k$. Hence

$$A_i: \langle \Omega, \mathscr{S}, P \rangle \to \{0, 1, \ldots, k\}, \tag{6.24}$$

$$A_i(\omega) = \text{card}\{j: j \leqq k, Z_j(\omega) = i\}, \tag{6.25}$$

$$\sum_{i=1}^{M} A_i(\omega) = k. \tag{6.26}$$

A simple factorization yields

$$D^2 Y^* = (\sigma^2/k^2) \sum_{\langle a_1, a_2, \ldots, a_M \rangle \in \{0,1,\ldots,k\}^M, \Sigma a_i = k} [(\sum_{i=1}^{M} a_i^2) \cdot$$
$$\cdot P(\{\omega: \omega \in \Omega, \langle A_1(\omega), \ldots, A_M(\omega) \rangle = \langle a_1, \ldots, a_M \rangle\}], \tag{6.27}$$

and after an easy calculation we obtain

$$D^2 Y^* = (\sigma^2/k^2) \sum_{i=1}^{M} \sum_{\langle a_1, \ldots, a_M \rangle \in \{0,1,\ldots,k\}^M, \Sigma a_i = k} [a_i^2 \cdot$$
$$\cdot P(\{\omega: \omega \in \Omega, \langle A_1(\omega), \ldots, A_M(\omega) \rangle = \langle a_1, \ldots, a_M \rangle\})] =$$
$$= (\sigma^2/k^2) \sum_{i=1}^{M} \sum_{j=0}^{k} \sum_{\langle a_1, \ldots, a_M \rangle \in \{0,1,\ldots,k\}^M, \Sigma a_l = k, a_i = j} \cdot$$
$$\cdot [j^2 P(\{\omega: \omega \in \Omega, \langle A_1(\omega), \ldots, A_M(\omega) \rangle = \langle a_1, \ldots, a_M \rangle\})] =$$
$$= (\sigma^2/k^2) \sum_{i=1}^{M} \sum_{j=0}^{k} j^2 P(\{\omega: \omega \in \Omega, A_i(\omega) = j\}). \tag{6.28}$$

However, when investigating the probability of the random event $A_i(\omega) = j$, the well-known binomial formulas may be of use. Or, it is just the probability of occurrences of an event with probability $1/M$ in a series of k independent and identically distributed random samples. Because of the irrelevance of the order, for each $i \leqq M$,

$$P(\{\omega: \omega \in \Omega, A_i(\omega) = j\}) = \binom{k}{j} (1/M)^j (1 - 1/M)^{k-j}, \tag{6.29}$$

and this expression can be written as the binomial coefficient $b(j, k, p)$ with $p = 1/M$. Using the identity from [1], p. 179 (Russian translation),

$$\sum_{j=0}^{k} j^2 b(j, k, p) = k^2 p^2 + kp(1 - p), \tag{6.30}$$

hence,

$$D^2 Y^* = (\sigma^2/k^2) \sum_{i=1}^{M} \sum_{j=0}^{k} j^2 b(j, k, 1/M) =$$
$$= (\sigma^2/k^2) M(k^2/M^2 + (k/M)(1 - 1/M)) = \sigma^2/M + (\sigma^2/k)(1 - 1/M) =$$
$$= (\sigma^2/M)(1 + (M - 1)/k). \tag{6.31}$$

The relations $EY^* = \mu$ and $D^2((1/M) \sum_{i=1}^{M} Y_i) = \sigma^2/M$ hold trivially, so that the lemma is proved. □

Consider the algorithm $\mathscr{A}_1^*(k, c)$ defined as a special case of the algorithm $\mathscr{A}_1$ in which N_1 is of the form c^ϱ, $c > 1$, $\varrho \geqq 1$, and for each $i = 2, 3, \ldots, \varrho + 1$ we have $N_i = N_{i-1}/c$, hence, $K = \log_c N_1 + 1$, moreover, let $k_i = k$ for each $i = 2, 3, \ldots, K$. Other parameters, i.e. $N_1, k_1, p, \ldots$ are the same as in $\mathscr{A}_1$ and will not be explicitly introduced. Let $Y^*(\mathscr{A}_1^*(k, c))$ be the random variable defined for $\mathscr{A}_1^*(k, c)$ by the

PE 4582

relation (6.15). Suppose, moreover, that to each ith level processor just c $(i-1)$st level processors are subjected, so that card $\mathfrak{N}_j^i = c$ for each $i \leqq K-1$ and $j \leqq \leqq N_{i+1}$.

Theorem 6.1. Algorithm $\mathscr{A}_1^*(k, c)$ satisfies the relation

$$D^2(Y^*(\mathscr{A}_1^*(k,c))) = p(1-p)(k_1N_1)^{-1} N_1^{\log_c(1+(c-1)/k\ln c)} < \\ < D^2(Y_0) N_1^{(c-1)/k\ln c}. \tag{6.32}$$

Proof. Considering the way in which random variables Y_j^i are defined we can easily see, that random variables $Y_1^i, Y_2^i, \ldots, Y_{N_{i+1}}^i$ are statistically independent for each $i \leqq K-1$. Hence, Lemma 6.1 yields

$$D^2(Y_j^{i+1}) = D^2(Y_1^i)(1/c)(1+(c-1)/k) \tag{6.33}$$

for each $j \leqq N_{i+2}$. The supposed statistical independence of random variables $\tilde{X}_{ij}$ implies that X_{ij} are also statistically independent, so that

$$D^2(Y_j^1) = (1/k_1)\, p(1-p) \tag{6.34}$$

for each $j \leqq N_1$. Combining (6.33) nad (6.34) we obtain

$$D^2(Y_1^K) = p(1-p)(1/k_1)(1/c^{K-1})(1+(c-1)/k)^{K-1} = \\ = p(1-p)(1/k_1N_1)(1+(c-1)/k)^{\log_c N_1}, \tag{6.35}$$

as $K = \log_c N_1 + 1$. A simple modification of the right-hand side in (6.35) yields

$$D^2(Y^*(\mathscr{A}_1^*(k,c))) = p(1-p)(1/k_1N_1)\left(c^{\log_c(1+(c-1)/k)}\right)^{\log_c N_1} = \\ = p(1-p)(1/k_1N_1) N_1^{\log_c(1+(c-1)/k)} = \\ = p(1-p)(1/k_1N_1) N_1^{\ln(1+(c-1)/k)/\ln c} < \\ < p(1-p)(1/k_1N_1) N_1^{(c-1)/k\ln c} = D(Y_0) N_1^{(c-1)/k\ln c}, \tag{6.36}$$

where we have taken profit of (6.16), (6.17), and of the simple fact that $\ln(1+x) < x$ for each $x > 0$. The assertion is proved. □

Let us recall that, given $\delta > 0$, $V \subset A$ as above with card $A = N$, the algorithm $\mathscr{A}_1$ is called *δ-correct* with respect to the problem $\langle A, V\rangle$, if it yields δ-correct estimates of the relative frequency of elements of V in A, i.e., if

$$P(\{\omega: \omega \in \Omega, \left|Y^*(\mathscr{A}_1)(\omega) - (\text{card } V)/N\right| < (1/2N)\}) \geqq 1-\delta. \tag{6.37}$$

Intuitively said, $\mathscr{A}_1$ is δ-correct if with a probability at least $1-\delta$ the value of the form v/N which is the closest to the estimate $Y^*(\mathscr{A}_1)(\omega)$, is also the true value of the parameter $p = \text{card } V/\text{card } A$.

Theorem 6.2. For each $\varepsilon > 0$, $\delta > 0$, $c > 1$ and each problem $\langle A, V\rangle$ there exists a δ-correct algorithm $\mathscr{A}_1^*(k, c)$ the unit time computational complexity of which is in $o(N^\varepsilon)$-class.

Proof. Choose $0 < \varepsilon_1 < \varepsilon$. The value $Y^*(\mathscr{A}_1^*(k, c))$ can be understood as a sin-

gleton set of statistically independent and identically distributed random variables, so that the Chebyshev inequality reads

$$P(\{\omega\colon \omega \in \Omega, \left|Y^*(\mathscr{A}_1^*(k, c))(\omega) - p\right| \geqq (1/2N)\}) < \\ < D^2(Y^*(\mathscr{A}_1^*(k, c)))/4N^{-2}\,. \tag{6.38}$$

Given $\varepsilon > 0$ and $c > 1$, choose k such that $(c-1)/k \ln c \leqq \varepsilon_1/2$, hence

$$k \geqq 2(c-1)/\varepsilon_1 \ln c\,. \tag{6.39}$$

Moreover, set $N_1 = N^2$, and

$$k_1 = \lceil \delta^{-1} N^{2(c-1)/k \ln c} \rceil = \lceil \delta^{-1} N_1^{(c-1)/k \ln c} \rceil\,, \tag{6.40}$$

where $\lceil x \rceil$ is the upper integer part of x. (6.32) then yields

$$D^2(Y^*(\mathscr{A}_1^*(k, c)))/\tfrac{1}{4}N^{-2} = 4\,D^2(Y^*(\mathscr{A}_1^*(k, c)))\,N_1 < \\ < 4p(1-p)(k_1 N_1)^{-1} N_1^{1+(c-1)/k \ln c} \leqq 4\delta p(1-p) \leqq \delta\,, \tag{6.41}$$

as $p(1-p) \leqq \frac{1}{4}$ for each $p \in \langle 0, 1\rangle$. Combining (6.39) and (6.41) we obtain

$$P(\{\omega\colon \omega \in \Omega, \left|Y^*(\mathscr{A}_1^*(k, c))(\omega) - p\right| > 1/2N\}) < \delta\,, \tag{6.42}$$

and the δ-correctness of the algorithm $\mathscr{A}_1^*(k, c)$ is proved. Its unit time complexity then reads as

$$k_1 + Kk = k_1 + (\log_c N_1)\,k = k_1 + (2 \log_c N)\,k\,. \tag{6.43}$$

Now, $k_1 = \lceil \delta^{-1} N^{2(c-1)/k \ln c} \rceil$ and $\varepsilon_1 \geqq 2(c-1)/k \ln c$, hence, $k_1 = k_1(N) \in \mathcal{O}(N^{\varepsilon_1})$. At the same time, $2 \log_c N$ is in $o(N^{\eta})$ for each $\eta > 0$, $c > 1$, hence

$$k_1(N) + (2 \log_c N)\,k \in \mathcal{O}(N^{\varepsilon_1}) \subset o(N^{\varepsilon})\,, \tag{6.44}$$

as $\varepsilon_1 < \varepsilon$. The assertion is proved. □

Our aim is to go on in our effort to weaken, step by step, the supposed kind and degree of co-operation and synchronization among processors of the same level working in parallel and constituting hierarchical parallel algorithms for Monte-Carlo methods. Let us consider one of the possible immediate modifications of the algorithm $\mathscr{A}_1$ defined and investigated above. Informally said, the modification consists in the fact that each higher-level processor, when taking random samples from the set of its subjected lower-level ones, may with a positive probability wrongly sample some processor outside of this set of its "subjects". In such a case the information obtained from this "foreign" processor is incorporated into information set on the output of the higher-level processor. I.e., this higher-level processor has no possibility to realize its mistake and to take some correct measure. Random samples are considered, as above, to be the only tools of communication between processors of different levels. Hence, this model violates the assumption of disjointness of the sets, from which the higher-level processors take their random samples and preferences given by each processor to its subjected ones will be only of statistical character. For the sake of simplicity we suppose the equiprobable probability distribution

on both the subsets of lower-level processors (i.e. the "subjected" as well as the "other" ones). Suppose, that a higher-level processor has c subjected processors from the total number C of lower-level processors ($1 \leqq c < C$). Then it communicates with the lower-level processors through random samples such that, in each sample, with the probability $(1 - \varepsilon)/c$ each among the subjected processors is sampled, and with the probability $\varepsilon/(C - c)$ each among the other processors is sampled. The value $\varepsilon > 0$ can be seen as a quantitative measure of incompleteness of the cooperation, in the case of $\mathscr{A}_1$ we have $\varepsilon = 0$.

When describing this algorithm formally (we shall denote it by $\mathscr{A}_2$, explicitly introducing the parameters important in the given context), let us take profit of the description of algorithm $\mathscr{A}_1$ in those parts where both the algorithms are analogous. Consider, hence, again a set $A = \{a_1, a_2, \ldots, a_N\}$ and its subset V with the aim to estimate the value card V/card A, consider integers K, $N_1 > N_2 > \ldots > N_K$, $k_1, k_2, \ldots, k_K$, and a system $\{\tilde{X}_{ij}, i = 1, 2, \ldots, N_1, j = 1, 2, \ldots, k_1\}$ of statistically independent random variables taking the probability space $\langle \Omega, \mathscr{S}, P \rangle$ into A and satisfying (6.7), so that the random variables X_{ij}, defined by (6.8), satisfy (6.9). Also the variables $Y_i^1(\cdot)$, $i \leqq N_1$, are defined as in the case of algorithm $\mathscr{A}_1$, i.e. by (6.10), hence, their expected values and dispersions satisfy (6.11).

For each $i = 1, 2, \ldots, K - 1$, let $\mathscr{R}_i$ be a decomposition of the set $\{1, 2, \ldots, N_i\}$ of integers into N_{i+1}, disjoint nonempty subsets $\mathfrak{N}_1^i, \mathfrak{N}_2^i, \ldots, \mathfrak{N}_{N_{i+1}}^i$, moreover, let a real ε_i, $0 < \varepsilon_i < 1$, be given. For each $i \leqq K - 1$, $k \leqq N_{i+1}$ and $l \leqq k_{1+1}$, let Z_{i,k,l,ε_i} be a random variable taking $\langle \Omega, \mathscr{S}, P \rangle$ into $\{1, 2, \ldots, N_{i+1}\}$ and realizing the random sample intuitively described above, i.e. "random sample from the uniform probability distribution over $\mathfrak{N}_k^i$ with possibility of error the probability of which is ε_i". Hence,

$$P(\{\omega: \omega \in \Omega, Z_{i,k,l,\varepsilon_i}(\omega) = s\}) = (1 - \varepsilon_i)(\text{card } \mathfrak{N}_k^i)^{-1}, \tag{6.45}$$

if $s \in \mathfrak{N}_k^i$,

$$P(\{\omega: \omega \in \Omega, Z_{i,k,l,\varepsilon_i}(\omega) = s\}) = \varepsilon_i (\text{card } (\{1, 2, \ldots, N_i\} - \mathfrak{N}_k^i))^{-1}, \tag{6.46}$$

if $s \in \{1, 2, \ldots, N_i\} - \mathfrak{N}_k^i$.

All the random variables Z_{i,k,l,ε_i} are supposed to be statistically independent. For each $i \leqq K - 1$, $k \leqq N_{i+1}$ define random variable $Y_{k,\varepsilon_i}^i(\omega)$ by the relation (6.14) with $Z_{i,k,j}(\omega)$ replaced by $Z_{i,k,j,\varepsilon_i}(\omega)$. Our attention will be focused, again, to the random variable $Y_{1,\varepsilon_{K-1}}^K$, defined by the relation analogous to (6.15). This value will be denoted by $Y_\varepsilon^*(\omega)$ with possible explicit mentioning of other parameters and will be taken, again, as a statistical estimation of the unknown value $p =$ $=$ card V/card A. Analogously as in the case of algorithm $\mathscr{A}_1$ we shall investigate the dispersion $D^2 Y_\varepsilon^*$ of this estimation, the validity of the relation $D^2 Y_\varepsilon^* \geqq D^2 Y_0$ is clear. Namely, we shall seek for the values $k_1, k_2, \ldots, k_K$, the sum of which approximates the time computational complexity of $\mathscr{A}_2$, such that the dispersion $D^2 Y_\varepsilon^*$ was small enough to estimate the relative frequency of elements from V in A in the sense specified above.

Because of the fact that, for each $i \leqq K-1$, $k \leqq N_{i+1}$, $l \leqq k_{i+1}$, the random variable Z_{i,k,l,ε_i} takes with a positive probability *every* value from $\{1, 2, \ldots, N_i\}$, the same random variable $Y^i_{j,\varepsilon_{i-1}}$, $j \leqq N_{i+1}$, may occur in *two* (or more) sums defining random variables $Y^{i+1}_{k_1,\varepsilon_i}(\omega)$. $Y^{i+1}_{k_2,\varepsilon_2}(\omega)$. Hence, if $i \geqq 2$, then random variables Y^i_{j,ε_i}, $j \leqq N_{i+1}$, are not statistically independent and this fact makes a computation or even estimation of the value $D^2 Y^*_\varepsilon$ very difficult. So, we have to be satisfied with estimations of much rougher nature than in the case of algorithm $\mathscr{A}_1$. Let us introduce two auxiliary assertions for the upper bounds of dispersions of random variables will be of use in what follows.

Lemma 6.2. Let $Y_1, Y_2, \ldots, Y_M$ be random variables taking the probability space $\langle \Omega, \mathscr{S}, P \rangle$ into Borel line $\langle R, \mathscr{B}_0 \rangle$, then

$$D^2\big((1/M)\sum_{i=1}^{M} Y_i\big) \leqq \max_{i=1,\ldots,M} D^2 Y_i \,. \tag{6.47}$$

Proof. A simple computation yields

$$\begin{aligned} &D^2(Y_1 + \text{const.}) = E(Y_1 + \text{const.})^2 - (E(Y_1 + \text{const.}))^2 = \\ &= EY_1^2 + 2\,\text{const.}\,EY_1 + (\text{const.})^2 - (EY_1)^2 - 2\,\text{const.}\,EY_1 - (\text{const.})^2 = \\ &= EY_1^2 - (EY_1)^2 = D^2 Y_1 \,, \end{aligned} \tag{6.48}$$

So, we may limit ourselves to the case when $EY_1 = EY_2 = \ldots = EY_M = 0$. In such a case,

$$\begin{aligned} &D^2\big((1/M)\sum_{i=1}^{M} Y_i\big) = E\big((1/M)\sum_{i=1}^{M} Y_i\big)^2 = (1/M)^2\, E\big(\sum_{i=1}^{M} Y_i\big)^2 = \\ &= (1/M)^2\, E\big(\sum_{i=1}^{M}\sum_{j=1}^{M} Y_i Y_j\big) = (1/M)^2 \sum_{i=1}^{M}\sum_{j=1}^{M} E(Y_i Y_j) \leqq \\ &\leqq (1/M)^2 \sum_{i=1}^{M}\sum_{j=1}^{M} \sqrt{\big((EY_i^2)\,(EY_j^2)\big)} \leqq \\ &\leqq (1/M)^2 \sum_{i=1}^{M}\sum_{j=1}^{M} \sqrt{\big(\max_{i=1,\ldots,M} (EY_i^2)^2\big)} = \max_{i=1,\ldots,M} (EY_i^2) = \max_{i=1,\ldots,M} D^2 Y_i \,, \end{aligned} \tag{6.49}$$

using the Schwartz inequality

$$\int fg \,\mathrm{d}\mu \leqq \sqrt{\big(\big(\int f^2 \,\mathrm{d}\mu\big)\big(\int g^2 \,\mathrm{d}\mu\big)\big)} \tag{6.50}$$

hence, in terms of random variables,

$$E(Y_i Y_j) \leqq \sqrt{\big((EY_i^2)\,(EY_j^2)\big)} \,. \qquad \square \tag{6.51}$$

Lemma 6.3. Let X, Y, U be random variables taking the probability space $\langle \Omega, \mathscr{S}, P \rangle$ into the Borel line, such that the pairs $\langle X, U \rangle$ and $\langle Y, U \rangle$ of random variables are statistically independent. Let, moreover,

$$P\big(\{\omega\colon \omega \in \Omega, U(\omega)) = 1\}\big) = p \,, \quad P\big(\{\omega\colon \omega \in \Omega, U(\omega) = 0\}\big) = 1 - p \,. \tag{6.52}$$

Then

$$D^2(UX + (1 - U)\,Y) = p\,D^2X + (1 - p)\,D^2Y + p(1 - p)\,(EX - EY)^2 \tag{6.53}$$

Proof. Using the assumption that random variables X, U and Y, U are statistically independent, a simple calculation yields

$$D^2(UX + (1 - U)\,Y) = E(UX + (1 - U)\,Y)^2 - (E(UX + (1 - U)\,Y))^2 =$$
$$= E(U^2X^2 + 2U(1 - U)\,XY + (1 - U)^2\,Y^2) - (EU\,EX + E(1 - U)\,EY)^2. \tag{6.54}$$

Evidently,

$$EU = p\,,\quad E(1 - U) = 1 - p\,,\quad U^2 \equiv U\,,\quad U(1 - U) \equiv 0\,,$$
$$(1 - U)^2 \equiv 1 - U\,, \tag{6.55}$$

so that

$$D^2(UX + (1 - U)\,Y) = p\,EX^2 + (1 - p)\,EY^2 - (p\,EX + (1 - p)\,EY)^2 =$$
$$= p\,EX^2 + (1 - p)\,EY^2 - p^2(EX)^2 - (1 - p)^2\,(EY)^2 - 2p(1 - p)\,EX\,EY =$$
$$= p(EX^2 - (EX)^2) + p(1 - p)\,(EX)^2 + (1 - p)\,(EY^2 - (EY)^2) +$$
$$+ p(1 - p)\,(EY)^2 - 2p(1 - p)\,EX\,EY =$$
$$= p\,D^2X + (1 - p)\,D^2Y + p(1 - p)\,(EY - EY)^2 \tag{6.56}$$

and the assertion is proved. □

Similarly as above we shall consider the algorithm $\mathscr{A}_2^*(k, c)$, resulting as a special case of algorithm $\mathscr{A}_2$, supposing that $N_1 = c^\varrho$, $c > 1$, $\varrho \geqq 1$, $N_i = N_{i+1}/c$ for each $i = 2, 3, \ldots, \varrho + 1$, $\operatorname{card} \mathfrak{N}_j^i = c$ for each $i \leqq K - 1$ and $j \leqq N_{i+1}$, and $k_2 = k_3 = \ldots = k_K = k$. The random variable Y_ε^* will be denoted, in this case, by $Y_\varepsilon^*(\mathscr{A}_\varepsilon^*(k, c))$ and we shall study its dispersion.

Theorem 6.3. Consider the algorithm $\mathscr{A}_2^*(k, c)$ and suppose that, for each $i = 1, 2, \ldots, K$, $\varepsilon_i = \varepsilon_{K-1}/c^{K-i-1}$. Then

$$D^2(Y_\varepsilon^*(\mathscr{A}_2^*(k, c))) \leqq \frac{p(1 - p)}{k_1}\left(\frac{c}{N_1}\right)^{\varkappa(k, N_1)}, \tag{6.57}$$

where

$$\varkappa(k, N_1) \approx \mathrm{e}^{k\varepsilon_1}\left(\frac{c - 1}{c \ln c}\right)\left(\frac{k - 1}{k}\right) \tag{6.58}$$

in the sense that ratio of both the sides in (6.58) tends to one as $N_1 \to \infty$.

Proof. Random variables $Y_1^1(\cdot), Y_2^1(\cdot), \ldots, Y_{N_1}^1(\cdot)$ are defined by arithmetical averages of values of disjoint sets of statistically independent random samples, hence, they are statistically independent as well. Let random variables $Y_{1,\varepsilon_{i-1}}^i(\cdot), Y_{2,\varepsilon_{i-1}}^i(\cdot), \ldots, Y_{N_i,\varepsilon_{i-1}}^i(\cdot)$ be also statistically independent. For each $i \leqq K$, $j \leqq N_i$, evidently

$$EY_{j,\varepsilon_{i-1}}^i = p = P(\{\omega\colon \omega \in \Omega,\ \tilde{X}_{kl}(\omega) \in V\})\,. \tag{6.59}$$

In order to assure the statistical independence of random variables $Y_{1,\varepsilon_i}^{i+1}$, $Y_{2,\varepsilon_i}^{i+1}$, ..., $Y_{N_{i+1},\varepsilon_i}^{i+1}$, a sufficient condition is that the sets

$$\{Z_{1,\varepsilon_i}^{i,1}(\omega), Z_{2,\varepsilon_i}^{i,1}(\omega), \ldots, Z_{k,\varepsilon_i}^{i,1}(\omega)\}, \tag{6.60}$$

$$\{Z_{1,\varepsilon_i}^{i,2}(\omega), Z_{2,\varepsilon_i}^{i,2}(\omega), \ldots, Z_{k,\varepsilon_i}^{i,2}(\omega)\}, \ldots, \{Z_{1,\varepsilon_i}^{i,N_{i+1}}(\omega), Z_{2,\varepsilon_i}^{i,N_{i+1}}(\omega), \ldots, Z_{k,\varepsilon_i}^{i,N_{i+1}}(\omega)\}$$

must be disjoint. A sufficient condition for this property to hold is, that each processor takes random samples only among its subjected processors, hence, for each $j \leqq N_{i+1}$,

$$\{Z_{1,\varepsilon_{i-1}}^{i,j}(\omega), Z_{2,\varepsilon_{i-1}}^{i,j}, \ldots, Z_{k,\varepsilon_{i-1}}^{i,j}(\omega)\} \subset \mathfrak{N}_j^i. \tag{6.61}$$

Denote by A_j the random event defined by (6.61), then the assumed statistical independence of random variables $Z_{k,\varepsilon_{i-1}}^{i,j}(\cdot)$ implies

$$P(A_j) = (1 - \varepsilon_i)^k, \quad P(\bigcap_{j=1}^{N_{i+1}} A_j) = (1 - \varepsilon_i)^{kN_{i+1}}. \tag{6.62}$$

If the random event $\bigcap_{j=1}^{N_{i+1}} A_j$ occurred, then, for each $j \leqq N_{i+1}$, $Y_{j,\varepsilon_i}^{i+1}(\omega)$ is the arithmetical average of k statistically independent random samples with the same dispersion $D^2 Y_{1,\varepsilon_{i-1}}^i$ for each of them. Moreover, the samples are taken from a c-element set with respect to the uniform probability distribution over this set. So, Lemma 6.1 yields, in such a case

$$D^2 Y_{j,\varepsilon_i}^{i+1} = \left(\frac{1}{c}\right) D^2 Y_{1,\varepsilon_{i-1}}^i \left(1 + \frac{c-1}{k}\right). \tag{6.63}$$

Supposing that the event $\bigcap_{j=1}^{N_{i+1}} A_j$ did not occur, we cannot avoid that $Y_{j,\varepsilon_i}^{i+1}(\omega)$ is defined by an arithmetical average of statistically dependent random samples with the dispersion $D^2 Y_{j,\varepsilon_{i-1}}^i$ for each of them, Lemma 6.2, then yields $D^2 Y_{j,\varepsilon_i}^{i+1} \leqq D^2 Y_{1,\varepsilon_{i-1}}^i$. Omitting the indices j and ε_i, for the sake of simplicity, and using Lemma 5.3 with $EX = EY$ we obtain the following recurrent relations

$$\begin{aligned}
&D^2 Y^{i+1} \leqq \\
&\leqq (1 - \varepsilon_i)^{kN_{i+1}} (1/c)\, D^2 Y^i (1 + (c-1)/k) + (1 - (1 - \varepsilon_i)^{kN_{i+1}})\, D^2 Y_i = \\
&= D^2 Y_i [1 - (1 - \varepsilon_i)^{kN_{i+1}} + (1 - \varepsilon_i)^{kN_{i+1}} ((1/c) + (c-1)/ck)] = \\
&= D^2 Y_i [1 - (1 - \varepsilon_i)^{kN_{i+1}} (1 - (1/c) - ((c-1)/ck))] = \\
&= D^2 Y_i [1 - (1 - \varepsilon_i)^{kN_{i+1}} ((ck - k - c + 1)/ck)] = \\
&= D^2 Y_i [1 - (1 - \varepsilon_i)^{kN_{i+1}} ((c-1)/c)((k-1)/k)].
\end{aligned} \tag{6.64}$$

Combining the obtained results, we have

$$D^2(Y_\varepsilon^*(\mathscr{A}_2^*(k,c))) = D^2 Y^K \leqq (D^2 Y^1) \prod_{i=1}^{K-1} \left[1 - (1 - \varepsilon_i)^{kN_{i+1}} \left(\frac{c-1}{c}\right)\left(\frac{k-1}{k}\right)\right], \tag{6.65}$$

where $D^2 Y^1 = p(1-p)/k_1$. Using the relation $\ln(1-x) < -x$ for $0 < x < 1$

we obtain

$$\ln D^2 Y^K \leqq \ln\left(p(1-p)/k_1\right) + \sum_{i=1}^{K-1} \ln\left(1 - (1-\varepsilon_i)^{kN_{i+1}}\left(\frac{c-1}{c}\right)\left(\frac{k-1}{k}\right)\right) <$$

$$< \ln\left(p(1-p)/k_1\right) - \sum_{i=1}^{K-1} (1-\varepsilon_i)^{kN_{i+1}}\left(\frac{c-1}{c}\right)\left(\frac{k-1}{k}\right) =$$

$$= \ln\left(p(1-p)/k_1\right) - \sum_{i=1}^{K-1}\left(\left(1 - \frac{N_{i+1}\varepsilon_i}{N_{i+1}}\right)^{kN_{i+1}}\left(\frac{c-1}{c}\right)\left(\frac{k-1}{k}\right)\right). \tag{6.66}$$

Because of the kind of approximation for $\varkappa(k, N_1)$ which is to be proved, the following approximation

$$\left(1 - \frac{N_{i+1}\varepsilon_i}{N_{i+1}}\right)^{N_{i+1}} \approx \mathrm{e}^{-N_{i+1}\varepsilon_i} \tag{6.67}$$

may be used. As we have assumed $\varepsilon_i = \varepsilon_{K-1}/c^{K-i-1}$ for each $i = 1, 2, \ldots, K-1$, and $N_{i+1} = N_i/c$, we obtain $N_{i+1}\varepsilon_i = N_K\varepsilon_{K-1} = \varepsilon_{K-1}$ for each $i < K$. So we have

$$\ln D^2 Y^K < \ln\left(p(1-p)/k_1\right) - (K-1)\,\mathrm{e}^{-k\varepsilon_{K-1}}\left(\frac{c-1}{c}\right)\left(\frac{k-1}{k}\right), \tag{6.68}$$

hence,

$$DY^K < \left(p(1-p)/k_1\right)\mathrm{e}^{-(K-1)\mathrm{e}^{-k\varepsilon_{K-1}}((c-1)/c)((k-1)/k)}$$

$$= \frac{p(1-p)}{k_1}\left(\mathrm{e}^{-(\log_c N_1 - 1)}\right)^{\mathrm{e}^{-\varepsilon_{K-1}k}((c-1)/c)((k-1)/k)}$$

$$= \frac{p(1-p)}{k_1}\left(\mathrm{e}^{-\ln N_1 \log_c \mathrm{e} + 1}\right)^{\mathrm{e}^{-\varepsilon_{K-1}k}((c-1)/c)((k-1)/k)}$$

$$= \frac{p(p-1)}{k_1}\left(\left(\frac{1}{N_1}\right)^{\log_c \mathrm{e}}\right)^{\mathrm{e}^{-\varepsilon_{K-1}k}((c-1)/c)((k-1)/k)}$$

$$= \frac{p(1-p)}{k_1}\left(\left(\frac{1}{N_1}\right)^{\log_c \mathrm{e}} c^{\log_c \mathrm{e}}\right)^{\mathrm{e}^{-\varepsilon_{K-1}k}((c-1)/c)((k-1)/k)}$$

$$= \frac{p(1-p)}{k_1}\left(\frac{c}{N_1}\right)^{\mathrm{e}^{-\varepsilon_{K-1}k}((c-1)/c\ln c)((k-1)/k)} \tag{6.69}$$

and the assertion is proved. □

The notion of δ-correctness for the algorithm $\mathscr{A}_2$ with respect to a given $\delta > 0$ and problem $\langle A, V\rangle$ with card $A = N$ is defined in the same way as in the case of algorithm $\mathscr{A}_1$, hence, by (6.37). We shall prove also an assertion describing the time complexity of a δ-correct algorithm $\mathscr{A}_2^*(k, c)$, hence, this assertion can be seen as an analogy of Theorem 5.1.

Theorem 6.4. For each $\varepsilon > 0$ and $\delta > 0$ there exist $k, c, k_1 \in \mathfrak{N}$ and $\varepsilon_1 > 0$ such

that for each $\varepsilon_{K-1} \leqq \varepsilon_1$ and for each algorithm $\mathscr{A}_2^*$ with parameters k, c, k_1 and ε_{K-1} holds: $\mathscr{A}_2^*$ is δ-correct with respect to the problem $\langle A, V\rangle$ and the unit time computational complexity of $\mathscr{A}_2^*$ belongs to $o(N^\varepsilon)$-class, where $N = \operatorname{card} A$. □

Proof. An easy calculation yields

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{x-1}{x\ln x}\right) = \frac{x\ln x - (x-1)(\ln x - 1)}{(x\ln x)^2} =$$

$$= \frac{\ln(1+(x-1)) + 1 - x}{(x\ln x)^2} < \frac{x - 1 + 1 - x}{(x\ln x)^2} = 0\,, \tag{6.70}$$

as $x > 1$. I.e. $x = 1 + \eta, \eta > 0$, implies that $\ln(1+\eta) = \eta - f(\eta)$, $f(\eta) \in o(\eta)$, $f(\eta) > 0$. Hence, $(c-1)/c\ln c$ is a decreasing function of c for $c > 1$, moreover,

$$\lim_{c\to 1+} \frac{c-1}{c\ln c} = \lim_{\eta\to 0+} \frac{(1+\eta)-1}{(1+\eta)\ln(1+\eta)} =$$

$$= \lim_{\eta\to 0+} \frac{1}{1+\eta-(f(\eta)/\eta) - f(\eta)} = 1\,. \tag{6.71}$$

Hence, $(c-1)/c\ln c < 1$ for $c > 1$, setting

$$\varkappa(k, c, \varepsilon_{K-1}) = \mathrm{e}^{-\varepsilon_{K-1}k}\left(\frac{c-1}{c\ln c}\right)\left(\frac{k-1}{k}\right), \tag{6.72}$$

we obtain that $\varkappa(k, c, \varepsilon_{K-1}) \in (0, 1)$, as $\varepsilon_{K-1} > 0$. The assertion of Theorem 6.3 can be written as

$$D^2 Y_\varepsilon^*(\mathscr{A}_2^*(k, c)) < \frac{cp(1-p)}{k_1 N_1} N_1^{1-\varkappa(k,c,\varepsilon_{K-1})} \tag{6.73}$$

Setting $k_1 = k_1(N) = QN^{1-\varkappa(k,c,\varepsilon_{K-1})}$ for an appropriate Q we assure that, like as in the proof of Theorem 6.2.,

$$D^2(Y_\varepsilon^*(\mathscr{A}_2^*(k, c)) < \delta/4N^2 \tag{6.74}$$

for given k, ε_{K-1} and c. Hence, there exists a δ-correct algorithm $\mathscr{A}_2^*(k, c)$ with respect to the problem $\langle A, V\rangle$ and with unit time computational complexity $k_1 + k\log_c N$, i.e., in $o(N^{\varkappa(k,c,\varepsilon_{K-1})})$-class.

The only we have to prove is that for each $\varepsilon > 0$ there exist k, c and ε_1 such that $\varepsilon_{K-1} \leqq \varepsilon_1$ implies $\varkappa(k, c, \varepsilon_{K-1}) > 1 - \varepsilon$. Evidently, for each $L > 0$,

$$\lim_{c\to 1} \frac{c-1}{c\ln c} = \lim_{k\to\infty} \frac{k-1}{k} = \lim_{\varepsilon_K\to 0+} \mathrm{e}^{-\varepsilon_K L} = 1\,, \tag{6.75}$$

so that we may uniquely define

$$c_0 = \inf\{c: (c-1)/c\ln c \geqq 1 - (\varepsilon/3)\}\,, \tag{6.76}$$

$$k_0 = \min\{k: k \in \mathfrak{N}, (k-1)/k \geqq 1 - (\varepsilon/3)\}\,, \tag{6.77}$$

$$\varepsilon_1 = \sup\{\varepsilon: \mathrm{e}^{-\varepsilon k_0} \geqq 1 - (\varepsilon/3)\}\,. \tag{6.78}$$

Explicit expressions for c_0, k_0, and ε_1 could be easily settled, but they are not important in what follows. Setting $k = k_0$, $c = c_0$, and $\varepsilon_{K-1} \leqq \varepsilon_1$ we obtain that

$$\varkappa(k, c, \varepsilon_{K-1}) = \mathrm{e}^{-\varepsilon_{K+1}k}\left(\frac{c-1}{c \ln c}\right)\left(\frac{k-1}{k}\right) \geqq \mathrm{e}^{-\varepsilon_1 k_0}\left(\frac{c_0-1}{c_0 \ln c_0}\right)\left(\frac{k_0-1}{k_0}\right) \geqq$$

$$\geqq (1 - (\varepsilon/3))^3 > 1 - 3(\varepsilon/3) = 1 - \varepsilon\,. \tag{6.79}$$

So, $1 - \varkappa(k, c, \varepsilon_{K-1}) < \varepsilon$, hence,

$$o(N^{1-\varkappa(k,c,\varepsilon_{K-1})}) \subset o(N^{\varepsilon}) \tag{6.80}$$

and the theorem is proved. □

Let us close this chapter by emphasizing the fact that an investigation of parallel probabilistic algorithms the outcomes of which are statistically independent random samples, but which manipulate with these samples in a way leading to not necessarily independent random variables, is very difficult and the results having been obtained till now are not too numerous. So, the results achieved in this chapter are just very rough approximations of the characteristics in question. Their precisation, weakening of assumptions, as well as some alternative hierarchical structures, perhaps more adequate under certain types of statistical dependences, should and could be a matter of further investigations in the future.

Let us also emphasize the fact, that the results presented above are not parametrized by a fixed number of processors of various levels. They are deduced for the optimum numbers of processors, respecting the explicitly introduced dependences among numbers of processors of different levels, say, the exponential decrease of this numbers with the increasing level. The optimality is taken in a purely theoretical sense and from the viewpoint of minimal time computational complexity of the resulting hierarchical computational procedure. A practical applicability of such hierarchies is a matter of further considerations.

REFERENCES

[1] W. Feller: An Introduction to Probability Theory and Its Applications I, II. J. Wiley and Sons, New York 1957 (vol. I, 2nd edition), 1966 (vol. II). Russian translation: Mir, Moscow 1964, 1967.

7. PARALLEL PROBABILISTIC ALGORITHMS FOR LINEAR ORDERING

The problem, how to order the elements of a finite, but as a rule very large set, with respect to a numerical (or, in general, ordinal) criterion, represents very often an important subproblem of more sophisticated tasks of artificial intelligence. The time complexity with which this subgoal can be solved may be decisive for the applicability of a procedure solving the main problem. The time complexity necessary to order a finite set with respect to a given criterion depends, besides some other factors, on the cardinality of the arranged set and on the tools we have at our disposal. The first of these two factors will be, in this chapter, always taken as a free parameter. It measures the complexity of an instance of the solved general problem with respect to which the time computational complexities or other characteristics of the proposed solutions and algorithms are related. As far as the tools are connected, just a very simple case will be taken into consideration, It is an oracle, which samples at random two elements of the investigated set and computes and compares the value of the criterion for these two elements. If their actual ordering does not correspond to these criterion values, the positions of the two elements are mutually changed. Then the procedure goes on with another pair of elements sampled at random until the quality of the resulting sequence is sufficiently good in some statistically motivated and defined sense, the more exact formulations will be given below. In the center of our considerations will be the situation where we have a number of identical mechanisms of this kind working simultaneously over the same set or rather sequence of elements. We have to avoid possible data access conflicts among different processors, but no other kind of co-operation or synchronization will be supposed.

A formal description of the problem, tools and outcome situation in question can be presented as follows.

Let $\mathscr{A}$, $\mathscr{B}$ be nonempty sets, let $\leqq$ be a linear ordering on $\mathscr{B}$. Hence, for each $x, y, z \in \mathscr{B}$, $x \leqq x$, $(x \leqq y) \wedge (y \leqq x) \Rightarrow (x = y)$, $(x \leqq y) \wedge (y \leqq z) \Rightarrow (x \leqq z)$, and $(x \leqq y) \vee (y \leqq x)$. The usual conventions for $x \geqq y$, $y > x$ and $x < y$ are adopted. Let f be a function (criterion) defined on $\mathscr{A}$ and taking its values in $\mathscr{B}$.

Let $A_0 = \langle a_1^0, a_2^0, \ldots, a_N^0 \rangle \in \mathscr{A}^N$ be an N-tuple of elements of $\mathscr{A}$. A_0 is *ordered with respect to* f supposing that $i \leqq j$ implies $f(a_i^0) \leqq f(a_j^0)$ for all $i, j \leqq N$. The *degree of ordering* of A_0 with respect to f is denoted by $Q_f(A_0)$ and defined by

$$Q_f(A_0) = \frac{\operatorname{card}\left(\{\langle i, j\rangle : i, j \leqq N, i < j, f(a_i) > f(a_j)\}\right)}{\operatorname{card}\left(\{\langle i, j\rangle, i, j \leqq N, i < j\}\right)}, \tag{7.1}$$

where the denominator is evidently equal to $\frac{1}{2}N(N-1)$. So, A_0 is ordered iff $Q_f(A_0) = 0$, so that the term "degree of disordering" would be perhaps more adequate. The index f will be omitted in what follows, as the criterion is taken as fixed.

For each $i, j, N \in \mathfrak{N} = \{0, 1, 2, \ldots\}$, $0 < i, j \leqq N$, the operator $S(i, j)$ on $\mathscr{A}^N$ is

defined as follows:

$$[(i \geqq j) \vee ((i < j) \wedge (f(a_i) \leqq f(a_j)))] \Rightarrow$$
$$S(i,j)(\langle a_1, a_2, \ldots, a_N\rangle) = \langle a_1, a_2, \ldots, a_N\rangle . \tag{7.2}$$
$$[(i < j) \wedge (f(a_i) > f(a_j))] \Rightarrow S(i,j)(\langle a_1, a_2, \ldots, a_N\rangle) =$$
$$= \langle a_1, \ldots, a_{i-1}, a_j, a_{i+1}, \ldots, a_{j-1}, a_i, a_{j+1}, \ldots, a_N\rangle . \tag{7.3}$$

So, a_i and a_j are interchanged, if $i < j$ and $f(a_i) > f(a_j)$, in all other cases the N-tuple is unchanged.

Suppose, moreover, that for each $i = 1, 2, \ldots$ we have at our disposal a pair $\langle \alpha_i, \beta_i\rangle$ of random variables, mapping the probability space $\langle \Omega, \mathcal{S}\ P\rangle$ into the set $\langle 1, 2, \ldots, N\rangle$ of integers. We write α_i^N and β_i^N to express this fact explicitly, if necessary. All the random variables $\alpha_1, \alpha_2, \ldots, \beta_1, \beta_2, \ldots$ are supposed to be statistically independent and equally distributed, for the sake of simplicity (and due to the Laplace principle) they are supposed to generate equiprobable distribution on $\{1, 2, \ldots, N\}$. I. e.,

$$P(\{\omega; \omega \in \Omega, \alpha_i(\omega) = j\}) = P(\{\omega: \omega \in \Omega, \beta_i(\omega) = j\}) = 1/N \tag{7.4}$$

for each $i = 1, 2, \ldots, j \leqq N$. So we may define by induction, taking A_0 as above,

$$A_i = A_i(\omega), = S(\alpha_i(\omega), \beta_i(\omega))\, A_{i-1} , \tag{7.5}$$

We may ask, given ε, δ, $0 \leqq \varepsilon, \delta \leqq 1$, $A_0 \in \mathcal{A}^N$, which is the minimum $K(\varepsilon, \delta, A_0)$, supposing it exists, such that

$$P(\{\omega: \omega \in \Omega, Q(A_K(\omega)) \leqq \varepsilon\}) \geqq 1 - \delta . \tag{7.6}$$

The formulation just presented evidently corresponds to the case of sequential algorithm with the single oracle described informally above and formally by the operator $S(\alpha_i(\omega), \beta_i(\omega))$. Before investigating this case in more details, a formalization of the case with more operators working in parallel will be useful. However, it is possible, under such conditions, that two or more processors at the same time instant sample, test and intend to replace the same element of the sequence A_i in question. Instead of a more detailed investigation, if, when and under which conditions this can be consistently executed, such a case will be consequently excluded throughout all this chapter. So, if an element is sampled by two or more processors in the same time instant, it is put on its original position without comparing or even changing it with the (or some of the) other sampled element(s). The work continues with the next random samples. A formal description of this situation can read as follows.

Two pairs $\langle i, j\rangle$ and $\langle k, l\rangle$ of positive integers are called *contraverse* if they are not set disjoint, i.e. if $\{i, j\} \cap \{k, l\} \neq \emptyset$. Define an operator φ on the set $(\mathfrak{N} \times \mathfrak{N})^*$ of finite sequences of pairs of non-negative integers as follows: if $\langle\langle a_1, b_1\rangle, \langle a_2, b_2\rangle, \ldots, \langle a_L, b_L\rangle\rangle \in (\mathfrak{N} \times \mathfrak{N})^L$, then the index $i \leqq L$ is called contraverse, if $\langle a_i, b_i\rangle$ is contraverse with some $\langle a_j, b_j\rangle$, $j \neq i, j \leqq L$, evidently, j is also contraverse. $\varphi(\langle\langle a_1, b_1\rangle, \langle a_2, b_2\rangle, \ldots, \langle a_L, b_L\rangle\rangle)$ is defined as the sequence of pairs

of integers resulting from $\langle\langle a_1, b_1\rangle, \ldots, \langle a_L, b_L\rangle\rangle$ when erasing all members with contraverse indices. As immediately follows, no two pairs in $\varphi(\langle\langle a_1, b_1\rangle, \ldots, \langle a_L, b_L\rangle\rangle)$ are contraverse, but $\varphi(\langle\langle a_1, b_1\rangle, \ldots, \langle a_L, b_L\rangle\rangle)$ may be the empty sequence Λ.

Let us consider, now, that for each $i = 1, 2, \ldots$ we have at our disposal $L(i) > 0$ pairs $\langle\alpha_{i,1}, \beta_{i,1}\rangle \langle\alpha_{i,2}, \beta_{i,2}\rangle, \ldots, \langle\alpha_{i,L(i)}, \beta_{i,L(i)}\rangle$ of random variables, each of them mapping $\langle\Omega, \mathscr{S}, P\rangle$ into $\{1, 2, \ldots, N\}$. Again, we shall suppose that these random variables are mutually statistically independent and generate the equiprobable distribution on $\{1, 2, \ldots, N\}$. Hence, (7.4) is supposed to hold for each α_{ij} and β_{ij}, $i = 1, 2, \ldots, j = 1, 2, \ldots, L(i)$. Again, we may define by recursion

$$A_i = A_i(\omega) =$$
$$= S(b_1(\omega), \tilde{b}_1(\omega))\, S(b_2(\omega), \tilde{b}_2(\omega)) \ldots S(b_{S_i}(\omega), \tilde{b}_{S_i}(\omega))\,(A_{i-1})\,, \tag{7.7}$$

where

$$\langle\langle b_1(\omega), \tilde{b}_1(\omega)\rangle, \langle b_2(\omega), \tilde{b}_2(\omega)\rangle, \ldots, \langle b_{S_i}(\omega), \tilde{b}_{S_i}(\omega)\rangle\rangle =$$
$$= \varphi(\langle\langle\alpha_{i,1}(\omega), \beta_{i,1}(\omega)\rangle, \langle\alpha_{i,2}(\omega), \beta_{i,2}(\omega)\rangle, \ldots, \langle\alpha_{i,L(i)}, \beta_{i,L(i)}(\omega)\rangle\rangle)\,, \tag{7.8}$$

so that $0 \leqq S_i \leqq L_i$. If $S_i = 0$, hence if $\langle\langle b_1(\omega), \tilde{b}_1(\omega)\rangle, \ldots, \langle b_{S_i}(\omega), \tilde{b}_{S_i}(\omega)\rangle\rangle = \Lambda$, we set $A_i = A_{i-1}$. Also in this parallelized case our main effort will be to assure the validity of (7.6). As can be easily seen, the optimal choice of the number $L(i)$ of processors working in parallel and the way in which $L(i)$ depends on N is far from being trivial. An unlimited increasing of the number of processors does not make the situation better but rather worse. It is because of the increasing number of data access conflicts, in the sense specified above, which prove the retarding influence as far as the speed of the work of the algorithm is concerned.

In order to compare and classify the advantages of a sophisticated parallelization let us return, for a while, to the sequential case as described above with the aim to obtain an explicit expression or estimation for the value $K(\varepsilon, \delta, A_0)$ assuring the validity of (7.6). Or, under some simplifying, but in our context acceptable assumptions, $K(\varepsilon, \delta, A_0)$ can be interpreted as the unit time computational complexity of the algorithm in question. Our considerations concerning the value $K(\varepsilon, \delta, A_0)$ can be expressed, as shown below, as an appropriately formulated problem of the pure combinatoric probability theory with the results presented in the form of a particular lemma. Because of the fact that the proof of this lemma uses an appropriate approximation through the normal (Gauss) probability distribution, let us recall very briefly this notion (cf. [3] or other textbook or monography for more details).

Consider, again our fixed probability space $\langle\Omega, \mathscr{S}, P\rangle$ together with the Borel line $\langle E, \mathscr{B}\rangle$. I.e., $E = (-\infty, \infty)$ and $\mathscr{B}$ is the minimum σ-field of subsets of E containing all semi-open intervals. A mapping X taking Ω into E is called a *random variable with normal (Gauss) probability distribution with zero expected value and unit dispersion, in short*: $N(0, 1)$-distribution, if X is measurable, i.e.

$$\{\{\omega\colon \omega \in \Omega, X(\omega) \in B\}\colon B \in \mathscr{B}\} \subset \mathscr{S} \tag{7.9}$$

(a sufficient condition reads

$$\{\{\omega\colon \omega \in \Omega, X(\omega) \leqq x\}\colon x \in E\}\} \subset \mathscr{S})), \tag{7.10}$$

and if, for each $x \in E$,

$$\Phi(x) = P(\{\omega\colon \omega \in \Omega, X(\omega) \leqq x\}) = \frac{1}{\sqrt{(2\pi)}} \int_{-\infty}^{x} e^{-y^2/2}\, dy\,. \tag{7.11}$$

In such a case, in fact,

$$EX = \int_{-\infty}^{\infty} X\, d\Phi = 0\,, \quad D^2X = \int_{-\infty}^{\infty} (X - EX)^2\, d\Phi = \int_{-\infty}^{\infty} X^2\, d\Phi = 1\,. \tag{7.12}$$

Given $0 < \delta < 1$, δ-quantile of $N(0, 1)$ denoted by $\alpha_{\delta, N(0,1)}$ or simply α_δ in the sequel is defined by

$$P(\{\omega\colon \omega \in \Omega, X(\omega) \leqq \alpha_{\delta, N(0,1)}\}) = \delta \tag{7.13}$$

where X is a random variable with $N(0, 1)$-distribution. Evidently, if $0 < \delta < \frac{1}{2}$, then $\alpha_{\delta, N(0,1)} < 0$.

Lemma 7.1. Consider the Bernoulli schema with the parameter p, $0 < p < 1$. It is a sequence $X_1, X_2, \ldots$ of statistically independent and identically distributed random variables taking the probability space $\langle \Omega, \mathscr{S}, P\rangle$ into $\{0, 1\}$ and such that, for each $i = 1, 2, \ldots$,

$$P(\{\omega\colon \omega \in \Omega, X_i(\omega) = 1\}) = p\,. \tag{7.14}$$

Set $S_n(\omega) = \sum_{i=1}^{n} X_i(\omega)$. Given a positive integer N and positive reals α, δ, $\delta < \frac{1}{2}$, define

$$n^*(\alpha, N, p) = \min\{n\colon P(\{\omega\colon \omega \in \Omega, S_n(\omega) > \alpha N\}) > 1 - \delta\}\,. \tag{7.15}$$

Then

$$n^*(\alpha, N, p) = (\alpha/p) N + f(\alpha, N, p, \delta)\,, \tag{7.16}$$

where $f(\alpha, N, p, \delta) \in o(N)$ (i.e. $\lim_{N\to\infty} N^{-1} f(\alpha, N, p, \delta) = 0$) for each $\alpha > 0$, $p \in (0, 1)$ and $\pi \in (0, \frac{1}{2})$. A non-asymptotic version of this assertion reads: for each $\gamma > 0$ there exists $N_0(\gamma)$ such that $N \geqq N_0(\gamma)$ and $n = (\alpha/p) N^{1+\gamma}$ imply $n \geqq n^*(\alpha, N, p)$, so that $P(\{\omega\colon \omega \in \Omega, S_n(\omega) > \alpha N\}) > 1 - \delta$. Explicitly,

$$N_0(\gamma) = (1 - \alpha_\delta \sqrt{(1 - p)}/\sqrt{\alpha})^{[\min\{\gamma, \frac{1}{2}(\gamma+1)\}]^{-1}} \tag{7.17}$$

will do.

Proof. Evidently, if

$$S_n^*(\omega) = (S_n(\omega) - np)\,(np(1 - p))^{-1/2}\,, \tag{7.18}$$

then

$$P(\{\omega\colon \omega \in \Omega, S_n(\omega) > \alpha N\}) =$$

$$= P\left(\left\{\omega\colon \omega \in \Omega, S_n^*(\omega) > \frac{\alpha N - np}{\sqrt{(np(1 - p))}}\right\}\right). \tag{7.19}$$

For N fixed,

$$\lim_{n\to\infty} (\alpha N - np)\,(np(1-p))^{-1/2} = -\infty\,. \tag{7.20}$$

As the $N(0, 1)$ distribution is symmetric with respect to zero value, the approximation (5.1) (Chapter VII in [3]) may be used, so that

$$P(\{\omega\colon \omega \in \Omega,\ S_n(\omega) > \alpha N\}) \sim 1 - \Phi\left(\frac{\alpha N - np}{\sqrt{(np(1-p))}}\right). \tag{7.21}$$

Hence, the demand

$$P(\{\omega\colon \omega \in \Omega,\ S_n(\omega) > \alpha N\}) > 1 - \delta \tag{7.22}$$

reduces to

$$\Phi\left(\frac{\alpha N - np}{\sqrt{(np(1-p))}}\right) < \delta \tag{7.23}$$

or to

$$\frac{\alpha N - np}{\sqrt{(np(1-p))}} < \alpha_{\delta, N(0,1)}\,. \tag{7.24}$$

A simple modification yields

$$\alpha N - \alpha_\delta \sqrt{(np(1-p))} < np\,. \tag{7.25}$$

If $n = (\alpha/p)\,N$, then

$$\lim_{N\to\infty} \frac{\alpha N - \alpha_\delta \sqrt{(np(1-p))}}{np} = \lim_{N\to\infty} \frac{\alpha N - \alpha_\delta \sqrt{((\alpha/p)\,Np(1-p))}}{\alpha N} = 1, \tag{7.26}$$

hence, the minimal n^* satisfying (7.25) satisfies (7.16) as well.

For $\gamma > 0$ and $n = (\alpha/p)\,N^{1+\gamma}$ (7.25) reduces to

$$\alpha N^{1+\gamma} + \alpha_\delta \sqrt{((\alpha/p)\,p(1-p))}\,N^{(1+\gamma)/2} > \alpha N\,, \tag{7.27}$$

or to an equivalent relation

$$N^\gamma + \alpha_\delta \sqrt{((1-p)/\alpha)}\,N^{(1+\gamma)/2} > 1\,. \tag{7.28}$$

If $0 < \gamma \leqq 1$, then $\gamma/2 - \frac{1}{2} \leqq 0$ and $N^{(1+\gamma)/2} \leqq \frac{1}{2}$, if $\delta < \frac{1}{2}$, then $\alpha_\delta < 0$. Hence, (7.28) holds supposing that

$$N^\gamma > 1 - \alpha_\delta \sqrt{((1-p)/\alpha)}\,, \tag{7.29}$$

so that,

$$N > (1 - \alpha_\delta \sqrt{((1-p)/\alpha)})^{1/\gamma}\,. \tag{7.30}$$

If $\gamma > 1$, then $\gamma/2 - \frac{1}{2} > 0$ and $N^{(\gamma-1)/2} > 1$, an analogous reasoning then implies that (7.28) holds supposing that

$$N^\gamma > N^{\gamma/2-1/2} - (\alpha_\delta \sqrt{((1-p)/\alpha)})\,N^{\gamma/2-1/2} =$$
$$= N^{\gamma/2-1/2}(1 - \alpha_\delta \sqrt{((1-p)/\alpha)})\,, \tag{7.31}$$

so that

$$N^{\gamma-(\gamma/2-1/2)} = N^{\gamma/2+1/2} > (1 - \alpha_\delta \sqrt{((1-p)/\alpha)}), \tag{7.32}$$

hence,

$$N > (1 - \alpha_\delta \sqrt{((1-p)/\alpha)})^{1/(\gamma/2+1/2)}, \tag{7.33}$$

Clearly, if

$$N > (1 - \alpha_\delta \sqrt{((1-p)/\alpha)})^{\max\{1/\gamma, 1/(\gamma/2+1/2)\}}, \tag{7.34}$$

then (7.28) holds in both the cases. As

$$\max\{1/\gamma, 1/(\gamma/2 + \tfrac{1}{2})\} = 1/\min\{\gamma, \gamma/2 + \tfrac{1}{2}\}, \tag{7.35}$$

(7.34) agrees with (7.17) and the assertion is proved. □

When analyzing in more details the lemma above and its proof, the reader can easily discover the discrepancy consisting in fact, that (7.16) and (7.17) have been stated in a categoric form, but when proving them, the approximation (7.20) was used. A more detailed analysis of this approximation and its proof (cf. e.g. [3]) shows, however, that (7.16) holds in the stated form and the lower estimate introduced in (7.17) could differ from its correct value just by an additive constant, i.e. by a value depending only on γ. In the sequel we shall focus our attention to the asymptotical behaviour of the estimate n^* as the linear function of N with the multiplicative constant α/p. It is why we have admitted the use of (7.20) not to charge our explanation by technical difficulties irrelevant for what follows.

The next theorem solves the problem stated above (cf. (7.6)).

Theorem 7.1. Consider a sequence $A_0 \in \mathscr{A}^N$ and a sequence $\{\langle \alpha_i, \beta_i \rangle\}_{i=1}^{\infty}$ of pairs of statistically independent random variables defined on $\langle \Omega, \mathscr{S}, P \rangle$ and satisfying (7.4). Let $A_0, A_1(\omega), A_2(\omega), \ldots$ be generated by (7.9) from A_0 through $\{\langle \alpha_i, \beta_i \rangle\}_{i=1}^{\infty}$. Set

$$n^*(\delta, \varepsilon, N) = \min\{n: P(\{Q(A_n(\omega)) \leqq \varepsilon\}) \geqq 1 - \delta\}, \tag{7.36}$$

for $\varepsilon, \delta > 0$ given, let $\varepsilon < Q(A_0)$ to avoid trivial cases. Then

$$((Q(A_0) - \varepsilon)/Q(A_0)) N^2 + f_1(\delta, \varepsilon, N) \leqq n^*(\delta, \varepsilon, N) \leqq$$
$$\leqq ((Q(A_0) - \varepsilon)/\varepsilon) N^2 + f_2(\delta, \varepsilon, N), \tag{7.37}$$

where $f_i(\delta, \varepsilon, N)$, $i = 1, 2$, are appropriate $o(N^2)$-functions for each $0 < \delta < \frac{1}{2}$, $0 < \varepsilon < Q(A_0)$. Moreover, for each $\gamma > 0$, for $n = ((Q(A_0) - \varepsilon)/\varepsilon) N^2$ and for each

$$N > (1 - \alpha_{\delta, N(0,1)} \sqrt{((1-\varepsilon)/(Q(A_0) - \varepsilon))})^{(1/2)[\min\{\gamma, (1/2)(\gamma+1)\}]^{-1}}, \tag{7.38}$$

the inequality

$$P(\{\omega: \omega \in \Omega, Q(A_n(\omega)) \leqq \varepsilon\}) > 1 - \delta \tag{7.39}$$

holds.

Proof. Denote a pair $\langle a_i, a_j \rangle$ as *undesirable* in $A_0 = \langle a_1, a_2, \ldots, a_n \rangle$, if $i < j$ and $f(a_i) > f(a_j)$, clearly, there are just $Q(A_0)(1/2) N(N-1)$ undesirable pairs

in A_0 and our aim is to reduce this number to $(\varepsilon/2)\,N(N-1)$ at most. So, we have to remove at least $(Q(A_0)-\varepsilon)\,(\frac{1}{2}N(N-1))$ undesirable pairs from A_0. Moreover, there exists A_0 such that each undesirable pair must be sampled before its deleting. The probability of sampling of an undesirable pair is at most $p_1 = Q(A_0)\,\frac{1}{2}N(N-1)$: $: N^2$ as the number of undesirable pairs in A_i does not increase with i increasing. At the same time, this probability is at least $(\varepsilon/2)\,N(N-1)/N^2$ supposing the goal was not achieved yet. Evidently, each pair $\langle \alpha_i, \beta_i \rangle$ of random variables samples each pair from $\{1, 2, \ldots, N\} \times \{1, 2, \ldots, N\}$ with the same probability N^{-2}. Hence, denoting by S_n^1 (S_n^2, resp.) the number of successes in n sample of the Bernoulli sequence with the probability p_1 (p_2, resp.) of success, we obtain

$$P(\{\omega\colon \omega \in \Omega,\, S_n^1(\omega) \geqq \alpha N^2\}) > P(\{\omega\colon \omega \in \Omega,\, Q(A_n(\omega)) < \varepsilon\}) > \\ > P(\{\omega\colon \omega \in \Omega,\, S_n^2(\omega) \geqq \alpha N^2\}), \tag{7.40}$$

where

$$\alpha = (Q(A_0) - \varepsilon)\,\tfrac{1}{2}N(N-1)/N^2\,. \tag{7.41}$$

Hence,

$$\alpha/p_1 = (Q(A_0) - \varepsilon)/Q(A_0)\,, \quad \alpha/p_2 = (Q(A_0) - \varepsilon)/\varepsilon\,, \tag{7.42}$$

and Lemma 6.1 immediately implies (7.37). The relation (7.38) follows either immediately from (7.17) when replacing N by N^2, p by ε and $Q(A_0) - \varepsilon$ by α, or by a deduction analogous to that of (7.17). □

The value $Q(A_n(\omega))$ is a random variable. So, instead of the quality of ordering criterion defined by (7.6) we may ask, whether $EQ(A_n(\cdot)) < \varepsilon$; here E is the expected value operator. In general, both the criteria are not comparable, i.e., neither of them follows from the other one. Or, let $Q(A_n(\omega)) = \varepsilon_1 > 0$ on a set the probability of which is $1 - \delta_1$, $\delta_1 < \delta$, let $Q(A_n(\omega)) = 0$ otherwise, then $EQ(A_n(\cdot)) = (1-\delta_1)\,\varepsilon_1 = \\ = \varepsilon_1 - \delta_1\varepsilon_1 < \varepsilon$ for ε_1 and δ_1 appropriately chosen, but (7.6) does not hold. On the other side, let $Q(A_n(\omega)) = \varepsilon_1 < \varepsilon$ with the probability $1 - \delta$, $Q(A_n(\omega)) = 1$ otherwise. Then (7.6) holds, but $EQ(A_n(\cdot)) = (1-\delta)\,\varepsilon_1 + \delta = \varepsilon_1 - \varepsilon_1\delta + \delta > \varepsilon$ for appropriate ε_1 and δ.

Theorem 7.2. Let A_0, $\{\langle \alpha_i, \beta_i \rangle\}_{i=1}^{\infty}$, $\varepsilon > 0$ and $A_1(\omega), A_2(\omega), \ldots$ be as in Theorem 7.1. If

$$K = K(\varepsilon, A_0) \geqq (\ln (Q(A_0)/\varepsilon))\,N^2\,, \tag{7.43}$$

then $EQ(A_K(\cdot)) < \varepsilon$. Moreover, if K_1 satisfies $EQ(A_{K_1}(\cdot)) < \varepsilon$, then $|K_1(N) - \\ - K(\varepsilon, A_0)\,(N)| \in o(N^2)$.

Proof. Consider a sequence A_i, $i \geqq 0$. If $EQ(A_i(\cdot)) = 0$, the assertion holds trivially, so let $Q(A_i) > 0$. Hence, there are $Q(A_i)\,B_N$, $B_N = \frac{1}{2}N(N-1)$, undesirable pairs in A_i in the sense of the proof of Theorem 7.1. Using the same argumentation as above we can see that the probability of sampling and removing of an undesirable pair is at least $p_N = Q(A_i)\,B_N/N^2$ in each step. Hence, in A_{i-1} the number of undesirable pairs will be smaller than in A_i with the probability p_N at least, so that,

this number will be $Q(A_i)\,B_N - 1$ at most. With the complementary probability the number of undesirable pairs rests unchanged. Applying the operator of expected value we obtain

$$EQ(A_{i+1}(\cdot)) \leqq (EQ(A_i)\,B_N - 1)\,B_N^{-1}p_N + EQ(A_i)\,(1 - p_N) =$$
$$= EQ(A_i) - p_N B_N^{-1} = EQ(A_i)\,(1 - N^{-2})\,, \tag{7.44}$$

hence,

$$EQ(A_K(\cdot)) \leqq Q(A_0)\,(1 - N^{-2})^K\,. \tag{7.45}$$

A well-known inequality yields, for $K = \delta N^2$,

$$(1 - N^{-2})^K < \mathrm{e}^{-\delta}\,, \tag{7.46}$$

hence, relation (7.43) holds, if $\mathrm{e}^{-\delta} \leqq \varepsilon/Q(A_0)$, i.e. if $\delta \geqq \ln\,(Q(A_0)/\varepsilon)$ and the assertion is proved. The necessity of $K_1(N) \geqq K(\varepsilon, A_0)\,(N) - f(N), f(N) \in o(N^2)$, immediately follows from the fact that there is $A_0 \in \mathscr{A}^N$ such that equality holds in (7.44) and from a more detailed analysis of the approximation (7.46). □

So, as can be easily seen, also in case the quality criterion is changed the unit time complexity of the investigated sequential probabilistic algorithm rests in the $o(N^2)$-class. A substantial improvement could be reached by application of the Bayesian approach, taking also the initial sequence A_0 as sampled at random and replacing both the criteria introduced above by their expected values with respect to the a priori distribution in question. Or, till now, we have tacitly accepted the worst-case analysis or, what is the same, the minimax principle, supposing that having sampled and mutually replaced an undesirable pair, only this one undesirable pair was removed from the sequence. Such cases, of course, exist, but their number is very small when compared with the number of cases, when removing of one undesirable pair from the sequence implies that also some other undesirable pairs disappear. Then the quality of the ordering increases more rapidly than we supposed above. An appropriate a priori distribution, e.g. the uniform one, would convert this fact into a significant reduction of the time complexity. However, because of the reasons mentioned in Chapters 3, 4 and 6, we shall not investigate the Beyesian approach in more details. We shall rather concern our attention to the parallel probabilistic algorithms, as we have mentioned above when introducing the corresponding formalized model. The data access conflicts will be solved as described in this formalization; if two or more processors want to operate on the same data item in the same step, the item is blocked and is not accessible in this step (time instant). The next combinatorial lemma will be of use in the sequel.

Lemma 7.2. Let $\mathscr{A} = \{a_1, a_2, \ldots, a_N\}$ be a nonempty set, let $X_1, X_2, \ldots$ be a sequence of statistically independent and identically distributed random variables. Each of them maps the probability space $\langle \Omega, \mathscr{S}, P \rangle$ into $\mathscr{A}$ and generates the uniform probability distribution on $\mathscr{A}$. Set, for each $i = 1, 2, \ldots$,

$$q(i, \omega) = \{a_j : j \leqq N, (\exists!\, k \leqq i)\,(X_k(\omega) = a_j)\}\,, \tag{7.47}$$

where $\exists!\ k$ stands for "there is just one k such that...". Hence, $q(i, \omega)$ denotes the set of elements from $\mathscr{A}$, sampled just once by the random variables $X_1, X_2, \ldots, X_i$. Then there exists an $o(N)$-function $f(N)$ such that

$$\sup \{\mathsf{E} \text{ card } q(i, \cdot): i = 1, 2, \ldots\} = N/\mathrm{e} + f(N), \tag{7.48}$$

where $\mathrm{e} = 2{\cdot}718\ldots$ Informally, the expected value of the relative frequency of elements sampled just once does not exceed $1/\mathrm{e} = 0{\cdot}36$ no matter how large the set $\mathscr{A}$ and the random sample in question may be.

Proof. Cf. [4]. □

Theorem 7.3. Let $A_0 \in \mathscr{A}^N$, let $A_1(\omega), A_2(\omega), \ldots$ result from A_0 by (7.7), using a sequence $\{\langle \alpha_{ij}, \beta_{ij} \rangle\}$, $i = 1, 2, \ldots$, $j = 1, 2, \ldots, L(i)$ of pairs of statistically independent random variables satisfying (7.4) for each i, j within the given scopes. For given $\varepsilon > 0$ and for $\mathscr{L} = \{L(i)\}_{i=1}^{\infty}$ define

$$K(\varepsilon, A_0, N, \mathscr{L}) = \min \{k: \mathsf{E}Q(A_k(\cdot)) \leqq \varepsilon\}. \tag{7.49}$$

Then there exists a non-negative $o(N)$-function $f(N)$ such that, for each $\mathscr{L}$,

$$K(\varepsilon, A_0, N, \mathscr{L}) \geqq \mathrm{e}^2 (\ln (Q(A_0)/\varepsilon)) N + f(N). \tag{7.50}$$

Proof. Cf., again, [4]. □

The obtained result seems to be interesting and non-trivial. It shows, that in the absence of a sophisticated synchronization and co-operation among processors working in parallel an increase of their number makes their work not better, but worse. The reason is simple: an increasing number of processors block other processor including themselves by demanding the same data and this disadvantage quickly overcomes the speed-up resulting from the increasing number of processors. Supposing to have at disposal a full and "free of charge" cooperation among the processors, the sequence A_0 could be re-arranged in a constant time independent of N, by N^2 processors. Each of them tests and, if necessary and after a negotiation with other processors, interchanges one pair of elements from A_0. However, even this informal and verbal description of this co-operation reflects how complicated it must be. Let us remark, that the non-trivially optimal unit time complexity is reached with $N + o(N)$ processors working in parallel. In order (qualitative) sense this value is the square root of the cardinality of the investigated set $\{1, 2. \ldots, N\} \times \{1, 2, \ldots, N\}$ as well as the square root of the number of undesirable pairs. This number is at least $(\varepsilon/2) N^2$ throughout the work of the algorithm, as after having reached this value, in average, the algorithm stops its work. Let us recall similar results about quadratic speed-up reached as the optimal one by two-level hierarchies of parallel probabilistic searching algorithms, as proved above. It holds under the condition that the inspection of outputs of processors and cumulation of their results is not a negligible or hardware matter. Namely, we have supposed that time demands increase linearly with the number of processors working in parallel, and perhaps logarithmically with the cardinalities of the sample spaces in question. Hence, the results of this

chapter, namely, Theorem 7.3, support and specify the general result from [6], proving the quadratic speed up to be the best one accessible by two-level hierarchical parallel re-arranging of probabilistic algorithms with statistically independent samples. It is worth mentioning that the increase of the number of processors with the square root of the size of the instance in question of the solved problem is taken as the upper bound from the practical point of view, cf. [1], e.g. So, that our result can be seen as interesting also from this position.

Let us close this chapter by a result, showing that under a modification of the notion of contraverse pair the lower bound for the unit time computational complexity of our parallel probabilistic algorithm could be reduced to $\mathcal{O}(1)$, i.e. to a value independent of N. Two pairs $\langle i, j\rangle$, $\langle k, l\rangle$ of integers will be called *strongly contraverse*, if $i \neq k$ *and* $j = l$ (not $i = k$ *or* $j = l$ as above). Evidently, the possibility of blocking is reduced, so that a greater number of processors can be used in parallel when re-arranging the sequence A_0.

Theorem 7.4. Consider the model investigated in Theorem 7.3 just with contraversion replaced by strong contraversion, set

$$c_1 = 1/\ln\left(1/(1 - 1/\mathrm{e})\right) \doteq \mathrm{e} \tag{7.51}$$

Then for each $\mathscr{L}$,

$$K(\varepsilon, A_0, N, \mathscr{L}) \geqq c_1 \ln\left(Q(A_0)/\varepsilon\right). \tag{7.52}$$

Proof. Let $\mathscr{B}_i = \langle\langle \alpha_{ij}(\omega), \beta_{ij}(\omega)\rangle\rangle_{j=1}^{L(i)}$, let $\mathscr{C}_i = \varphi\mathscr{B}_i$ according to (7.7). Taking $\mathscr{B}_i$ as a random sample from the uniform probability distribution over $M = N^2$-element set we obtain, that $\langle \alpha_{ij}(\omega), \beta_{ij}(\omega)\rangle \in \mathscr{C}_i$ iff

$$\begin{aligned}
&\langle \alpha_{ij}(\omega), \beta_{ij}(\omega)\rangle \in q_0(L(i), \omega) = \\
&= \{\langle a_s, a_r\rangle : \langle s, r\rangle \in \{1, 2, \ldots, N\} \times \{1, 2, \ldots, N\}, (\exists!\, k \leqq L(i)) \times \\
&\times \left(\langle \alpha_{ik}(\omega), \beta_{ik}(\omega)\rangle = (a_s, a_r\rangle)\right\}.
\end{aligned} \tag{7.53}$$

Applying Lemma 7.2 to $M = N^2$ we obtain

$$E \operatorname{card} q_0(L(i), \cdot) \leqq N^2/\mathrm{e} \tag{7.54}$$

and this optimum is reached with $L(i) = N^2 + q(N)$, $q(N) \in o(N^2)$, processors working in parallel. Using the same argumentation as in the proof of Theorem 7.3 (cf. [4]) we obtain, that the expected number of undesirable pairs eliminated in one step is at least

$$(N^2/\mathrm{e})\, Q(A_i)\, \tfrac{1}{2}N(N - 1)/N^2. \tag{7.55}$$

Hence, the relative frequency of undesirable pairs eliminated in one step is at most $1/\mathrm{e}$, so that

$$EQ(A_{i+1}(\cdot)) \geqq EQ(A_i(\cdot))\,(1 - \mathrm{e}^{-1}), \tag{7.56}$$

$$EQ(A_i(\cdot)) \geqq Q(A_0)\,(1 - \mathrm{e}^{-1})^i, \tag{7.57}$$

Hence, if

$$i \geqq \ln\left(Q(A_0)/\varepsilon\right)/\left(\ln\left((1 - \mathrm{e}^{-1})\right)^{-1}\right) =$$
$$= \ln\left(Q(A_0)/\varepsilon\right)/\ln\left(1 + \mathrm{e}^{-1}\right) = \mathrm{e} \ln\left(Q(A_0)/\varepsilon\right), \tag{7.58}$$

then

$$Q(A_0)\left(1 - \mathrm{e}^{-1}\right)^i \leqq EQ(A_i(\cdot)) \leqq \varepsilon. \qquad \square \tag{7.59}$$

Let us emphasize the purely theoretical speed-up obtained by this modification of the notion of contraversion which should not be over-estimated. When applying this algorithm we had to solve the problems arising when two or more processors want to replace an element in different way. To synchronize such a task in a consistent way is far from being trivial and free of charge. It is why we consider our original model as the much more adequate one from the point of view of practical application.

In [7] and [8] a very interesting parallel probabilistic algorithm is presented which looks for the elements of a large finite sequence or set which are important from the viewpoint of an ordinal or numerical criterion. E.g., the smallest or the greatest element, the kth element from the bottom or from the top, given a positive integer k, etc. However, the supposed abilities of the basic processing units in [7] and [8] are different from our ones so that the results are not immediately comparable with those achieved here. They are not presented here in more details because of the shortage of space necessary to introduce an alternative mathematical formalism. In every case, [7] and [8] may serve for a useful confrontation. Finally, other sorting algorithms which could serve as an outcome or a motivation for other parallel probabilistic searching and sorting algorithms can be found in the practically oriented monographies [1], [2], [5] or elsewhere.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading 1974. Russian translation: Mir, Moscow 1979.

[2] H. Barringer: A Survey of Verification Techniques for Parallel Programs. (Lecture Notes in Computer Science 191.) Springer-Verlag, Berlin—Heidelberg—New York 1985.

[3] W. Feller: An Introduction to Probability Theory and its Applications I., II. J. Wiley and Sons, New York 1957 (vol. I, 2nd edition), 1966 (vol. II). Russian translation: Mir, Moscow 1964, 1967.

[4] I. Kramosil: Parallel probabilistic ordering algorithms with a simple conflict control strategy. In: Aplikace umělé inteligence AI 88, Prague 1988, pp. 39—46.

[5] J. Mikloško and V. E. Kotov: Algorithms, Software and Hardware of Parallel Computers. Springer-Verlag, Berlin—Heidelberg—New York and Veda, Bratislava 1984.

[6] J. H. Reif: On synchronous parallel computations with independent probabilistic choice. SIAM J. Comput. *13* (1984), 1, 46—56.

[7] R. Reischuk: A fast probabilistic parallel sorting algorithm. In: 22nd IEEE Symp. on Foundat. of Comp. Sci., Nashwille, Tennessee 1981.

[8] R. Reischuk: Probabilistic parallel algorithms for sorting and selection. SIAM J. Comput. *14* (1985), 2, 396—409.

8. SOME MODIFICATIONS OF PARALLEL PROBABILISTIC SEARCHING ALGORITHMS

In Chapter 4 we investigated hierarchical parallel probabilistic searching algorithms under some simplifying conditions which have been motivated mainly by our aim to make the model as transparent as possible and easy to describe and handle from the mathematical point of view. A modified model, resulting when abandoning the condition of safe and failure-proof work of each testing device, was investigated in Chapter 5. In this chapter we would like to review, very briefly and referring to more detailed papers, two other modifications of the elementary model explained in Chapter 4. The first of them takes into consideration possible data access conflicts, at least at the basic level, the other modification supposes that there is a possibility of co-operation among the first-level processors, even if this co-operation may be rather limited and of stochastic kind. In both the cases we limit ourselves to two-level hierarchies, postponing the investigation of corresponding many-level structures till another occasion.

Let us turn back to the model explained in Chapter 4. There, random samples were taken as non-conflict in the sense that if two or more processors sample the same element from A in the same time instant or step, all of them may and will test it. Let us accept, now, a more realistic assumption which is, in a sense, something like a dual extremum: if two or more processors sample the same element from A in one step, this element is blocked and it is not accessible to any processor at this step. Moreover, the processors are not supposed to be able to distinguish this case from that one where $x \in A - V$ has been sampled, so that they output zero value in both the situations. Each first-level processor is able to repeat its activity in the next time instant or step according to the model explained in Chapter 4, to test the elements which were sampled and not blocked, and to cumulate, on its output, the information whether at least one element from A has been found in a finite sequence of samples. If it is the case, the processor outputs a unit value, it outputs zero value otherwise. The higher-level processor works in the same way as in Chapter 4, i.e., without considering the possibility of data access conflicts. The testing oracles of all levels are supposed to be reliable, i.e. they work, like as in the model introduced in Chapter 4, without any danger of failure. The interpretation of the outcome value of the (unique) highest-level processor is the same as in Chapter 4. Again, the unit output value of this processor proves the set V to be nonempty without any doubts, the zero output value is taken as the decision that V is empty, but this decision is, in non-trivial cases, charged with a positive probability of error, which at the first sight, must be at least as large as in the conflict-free case.

Let $\{\{X_{ij}\}_{i=1}^{m}{}_{j=1}^{n}, \{Z_l\}_{l=1}^{k}\}$ be the system of random variables defined in Chapter 4 and satisfying (4.1) and (4.2). Set, for $i \leqq m$, $j \leqq n$, and $\omega \in \Omega$,

$$\gamma(i, j, \omega) = \chi_V(X_{ij}(\omega)) \prod_{k=1, k \neq i}^{m} [1 - \chi_{\{X_{ij}(\omega)\}}(X_{kj}(\omega))], \tag{8.1}$$

and define, on the fixed abstract probability space $\langle \Omega, \mathscr{S}, P \rangle$, a random variable $\mathscr{X}_0$ taking its values from $\{0, 1\}$ in this way:

$$\mathscr{X}_0(\omega) = \mathscr{X}_0(\langle A, V \rangle, \omega) = 1 \quad \text{iff} \quad \sum_{l=1}^{k} \sum_{j=1}^{n} \gamma(Z_l(\omega), j, \omega) > 0\,, \tag{8.2}$$

$$\mathscr{X}_0(\omega) = 0 \quad \text{otherwise}\,.$$

As can be easily seen, $\gamma(i, j, \omega) = 1$ iff $X_{ij}(\omega)$ is in V and differs from all $X_{kj}(\omega)$ with $k \neq i$. Supposing that $X_{i1}(\omega), X_{i2}(\omega), \ldots, X_{in}(\omega)$ are samples from A taken by the ith processor, then $\gamma(i, j, \omega) = 1$ iff the ith processor samples an element from V in the jth step and no other processor samples the same element at the same step. Hence, the ith processor tests this element and outputs the unit value. Moreover, $\sum_{j=1}^{n} \gamma(i, j, \omega) > 0$ holds iff the event just described occurs at least once in the sequence of n steps. But only the processors the indices of which are sampled at random by variables $Z_1, Z_2, \ldots, Z_k$ are asked for their output values, so that $\mathscr{X}_0(\omega) = 1$ iff there is at least one among the sampled first-level processors which outputs unit value, hence, $\mathscr{X}_0(\omega)$ is nothing else than the output value of the unique second-level processor, as defined informally above. The more simple case of conflict-free samples from Chapter 4 can be obtained by simply setting $\gamma(i, j, \omega) = \gamma_V(X_{ij}(\omega))$. Evidently,

$$\sum_{l=1}^{k} \sum_{j=1}^{n} \gamma(Z_l(\omega), j, \omega) \leqq \sum_{l=1}^{k} \sum_{j=1}^{n} \chi_V(X_{Z_l(\omega), j}(\omega))\,. \tag{8.3}$$

Considering a searching problem $\langle A, V \rangle$ and a two-level HPPSA defined in this chapter, if $V \neq \emptyset$, then the random event $\mathscr{X}_0(\omega) = 0$ can be interpreted as an error. The following assertion offers certain estimations for the probability of this error.

Theorem 8.1. Let $\langle A, V \rangle$ be a searching problem with card $A = N$, card $V = v > 0$, let $\mathscr{X} = \mathscr{X}(A, V, m, n, k)$ be a two-level HPPSA for $\langle A, V \rangle$, then

$$\left(1 - \frac{v}{N}\right)^{nk} < P(\{\omega: \omega \in \Omega, \mathscr{X}_0(\omega) = 0\}) < \left(1 - \frac{m}{N}\left(1 - \frac{1}{N}\right)^{m-1}\right)^{n} + \left(1 - \frac{1}{m}\right)^{k}. \tag{8.4}$$

Proof. Cf. Theorem 1 in [1]. □

Theorem 8.2. Let the notations and conditions of Theorem 8.1 hold. For each $\varepsilon > 0$ there exists reals $c_1(\varepsilon)$, $c_2(\varepsilon)$ and $c_3(\varepsilon)$ independent of N such that if $N \geqq 4c_1^2$, $m \geqq \lceil c_1 \sqrt{N} \rceil$, $n \geqq \lceil c_2 \sqrt{N} \rceil$ and $k \geqq \lceil c_3 \sqrt{N} \rceil$, then

$$P(\{\omega: \omega \in \Omega, \mathscr{X}_0(\omega) = 0\}) < \varepsilon\,. \tag{8.5}$$

Proof. Cf. Theorem 2 in [1]. □

The following assertion proves that the results for m, n and k stated in Theorem 8.2 are the best possible ones in the qualitative sense.

Theorem 8.3. Let $\mathscr{X}$ be a HPPSA for a searching problem $\langle A, V \rangle$ with $A = N$, card $V = 1$. If $n = n(N)$ and $k = k(N)$ are such that $nk \in o(N)$, then there exists,

for each $\varepsilon < 1$, $N_0 = N_0(\varepsilon)$ such that, for all $N \geqq N_0$

$$P(\{\omega\colon \omega \in \Omega, \mathscr{X}_0(\omega) = 0\}) > \varepsilon\,. \tag{8.6}$$

Proof. Cf. Theorem 3 in [1], the same holds also for the conflict-free algorithm defined by the random variable $\mathscr{X}(\langle A, V\rangle, \cdot)$ in (4.3). □

The interpretation of Theorems 2 and 3 is similar to that of the assertions proved in Chapters 4 and 5. Hence, accepting the simplifying assumptions that each random sample from A needs α time units and each random sample from $\{1, 2, \ldots, m\}$ needs β units independently of N and m, the expression $\alpha n + \beta k + \text{const}$ may serve as a very rough estimation of the total time complexity of the HPPSA $\mathscr{X}(m, n, k)$. Now, Theorem 2 claims that the corresponding probability of error can be kept below a given $\varepsilon > 0$ with $\alpha n(N) + \beta k(N)$ in the $\mathcal{O}(\sqrt{N})$-class, moreover, according to Theorem 3 it cannot be reduced to the $o(\sqrt{N})$-class. In both cases, what is actually needed is that the product of n and k must be in $\mathcal{O}(N)$, but when taking $m(N) = d_1 N^q$, $k(N) = d_2 N^{1-q}$, $0 < q < 1$, $q \neq \frac{1}{2}$, then $\alpha n + \beta k$ is in $\mathcal{O}(N^{\max\{1-q,q\}})$ and this result would be qualitatively worse than that with $q = \frac{1}{2}$. To summarize, the time complexity is in the same class as in the conflict-free case investigated in Chapter 4. A more detailed optimization of the expression $\alpha n + \beta k$ within the $\mathcal{O}(\sqrt{N})$-class, i.e., the computation of the multiplicative constants minimizing this expression, would be a purely technical matter.

Let us also mention briefly also the case of many-level hierarchies. Many-level hierarchical parallel probabilistic searching algorithm (for the searching problem $\langle A, V\rangle$) will be defined as in Chapter 4, cf. (4.16) and (4.17), but the corresponding random variable $\mathscr{X}$, enabling to understand this algorithm as a statistical test for the emptiness of V, will be modified in a way reflecting the possibility of data access conflicts. The intuitive definition of the modified random variable $\mathscr{X}_0$ reads as follows: Set

$$V_0 = V, \tag{8.7}$$

$$V_r = V_r(\omega) = \{i\colon i \leqq N_r, \sum_{j=1}^{n_r} \gamma(i, j, \omega) > 0\}\,,$$

where

$$\gamma(i, j, r, \omega) = \chi_{V_{r-1}(\omega)}(X_{ij}^r(\omega)) \prod_{k=1, k\neq i}^{m} (1 - \chi_{\{X^r{}_{ij}(\omega)\}}(X_{kj}^r(\omega)))\,. \tag{8.8}$$

This agrees, for $\gamma(i, j, \omega) = \gamma(i, j, 1, \omega)$, with the definition of HPPSA above, when

$$V_1(\omega) = \{i\colon i \leqq N_1, \sum_{j=1}^{n_1} \gamma(i, j, \omega) > 0\}\,, \tag{8.9}$$

for $r = K$ we obtain

$$V_K(\omega) = \{1\} = A_K \quad \text{iff} \quad \sum_{j=1}^{n_K} \gamma(1, j, K, \omega) > 0\,,$$

$$V_K(\omega) = \emptyset \quad \text{iff} \quad \sum_{j=1}^{n_K} \gamma(1, j, K, \omega) = 0\,. \tag{8.10}$$

Evidently, $V_k(\omega) = \emptyset$ implies $V_l(\omega) = \emptyset$ for each $k \leqq l \leqq K$. Finally, set

$$\{\omega\colon \omega \in \Omega, \mathscr{X}_0(\omega) = 1\} = \bigcap_{k=0}^{K} \{\omega\colon \omega \in \Omega, V_k(\omega) \neq \emptyset\}\,, \tag{8.11}$$

$$\mathscr{X}_0(\omega) = \mathscr{X}_0(\langle A, V\rangle, \omega) = 0 \quad \text{otherwise}\,.$$

As before, the random event $\mathscr{X}_0(\omega) = 1$ is taken as the decision that $V \neq \emptyset$ and this decision is always correct, being based on the positive testing of at least one element from V by a first-level processor. The random event $\mathscr{X}_0(\omega) = 0$ is taken as the decision that $V = \emptyset$ and it may be charged by an error, or, elements from V can be either disregarded by first-level processor or the report about their finding can be disregarded by higher-level processors. Hence, the value $P(\{\omega\colon \omega \in \Omega, \mathscr{X}_0(\omega) = 0\})$ may be taken, if $V \neq \emptyset$, as the probability of error connected with the algorithm in question. Trying to keep this probability below a given $\varepsilon > 0$ uniformly for all nonempty $V \subset A$, we shall use this assertion.

Lemma 8.1. For each $\varepsilon > 0$, $\delta > 0$ there exists a natural number $n_0 = n_0(\varepsilon, \delta)$ independent of N such that for all $n \geqq n_0$ and for $m = \lfloor \delta N \rfloor - 1$

$$\left(1 - \frac{m}{N}\left(1 - \frac{1}{N}\right)^{m-1}\right)^{n} < \varepsilon\,. \tag{8.12}$$

Proof. Cf. Lemma 1 in [1]. □

The just referred proof of Lemma 1 in [1] or an immediate computation yield that the threshold value n_0 reads $e^{\delta}\delta^{-1} \ln(1/\varepsilon)$. As can be easily seen, an analogous threshold value for conflict free random samples is $n_0 = \delta^{-1} \ln(1/\varepsilon)$. Hence, the corresponding time computational complexity increases (as $e^{\delta} >$ for $\delta > 0$) when admitting data access conflicts, but the increase is only a multiplicative one, as e^{δ} does not depend on N.

So, let us take a δ, $0 < \delta < 1$, and suppose, just for the sake of simplicity of the following considerations, that $N = \operatorname{card} A$ is of the form $(1/\delta)^K$. Set $N_0 = N$, $N_i = \delta N_{i-1}$, $i = 1, 2, \ldots K$, hence, $N_k = \delta^k N_0$, and consider N_i processors of the ith level. Given $\varepsilon > 0$, set

$$\varepsilon_1 = \varepsilon/K\,, \quad n_0(\delta, \varepsilon) = (e^{\delta}/\delta)(\ln(1/\varepsilon_1))\,. \tag{8.13}$$

Each of the first-level processors, there are N_1 in total, takes n_0 independent and sequential random samples from the uniform probability distribution over the basic set A, and those among the sampled elements which were not sampled, simultaneously, by another processor are tested as far as their membership in the set V is concerned. If we denote

$$\alpha(m, n, N) = \left(1 - \frac{m}{N}\left(1 - \frac{1}{N}\right)^{m-1}\right)^{n}, \tag{8.14}$$

then with probability at least $1 - \alpha(N_1, n_0, N)$ at least one first-level processor takes the unit output value, i.e. reports an element from V. Moreover, n_0 and N_1

are chosen in such a way that

$$1 - \alpha(N_1, n_0, N) \geqq 1 - \varepsilon_1 = 1 - (\varepsilon/K)\,. \tag{8.15}$$

There are N_2 second-level processors and each of them takes n_0 independent and sequential random samples from the uniform probability distribution over the set A_1 of (indices of) outputs of the first-level processors, hence, card $A_1 = N_1 = \delta N$. For those (indices of the) first-level processors among the sampled ones which were not sampled, simultaneously, by another processor, the sampling second-level processor tests whether their output values were 1 or not, i.e. tests the membership of the sampled first-order processor in question in the subset V_1 of A_1 defined by (8.9). If the result of this test is positive, then the report about the sampled element of V occurs on the output of the corresponding second-level processor and this output takes the unit value (otherwise, the zero value). If at least one first-level processor discovered an element from V, so that $V_1 \neq \emptyset$, then with probability at least $1 - \alpha(N_2, n_0, N_1)$, an element from V is reported also at the second level. But, as can be easily seen,

$$\alpha_2(N_2, n_0, N_1) = \alpha(N_1, n_0, N)\,, \tag{8.16}$$

so that

$$1 - \alpha(N_2, n_0, N_1) \geqq 1 - (\varepsilon/K)\,. \tag{8.17}$$

Now, the induction step is evident: there are N_3 third-level processors and each of them looks for a report about an element from V among n_0 non-collising random samples from the (outputs of the) second-level processors. Supposing such a report is found, the corresponding third-level processor outputs unit value and so on. Combining the corresponding conditional probabilities and using the supposed statistical independence of all random variables in question we obtain that if $V \neq \emptyset$, then with probability at least

$$(1 - \varepsilon_1)^K > 1 - K\varepsilon_1 = 1 - \varepsilon \tag{8.18}$$

the report about an element from V reaches the output of the unit Kth, i.e., the highest level processor. Hence, the probability of error is majorized by ε.

At each level we have taken n_0 sequential samples and the operations on different levels are also sequential, so that, in total,

$$\begin{aligned} &Kn_0(\log_{1/\delta}N)\,(\mathrm{e}^\delta/\delta)\ln(\varepsilon/\log_{1/\delta}N)^{-1} = \\ &= (\log_{1/\delta}N)\,(\mathrm{e}^\delta/\delta)\ln(\varepsilon/\log_{1/\delta}N - \ln\varepsilon) \end{aligned} \tag{8.19}$$

sequential samples have been taken. Taking the unit time complexity for each sample, independent of the cardinality of the corresponding sample space, and neglecting the other operations, expression (8.19) approximates the time complexity of the suggested special algorithm answering, within the probability of error uniformly majorized by the given $\varepsilon > 0$, the question whether $V = \emptyset$ or not. As in the conflict-free case, this complexity is, again, in the $\mathcal{O}(\log N \log\log N)$-class, the only difference being represented by the multiplicative constant $\mathrm{e}^\delta > 1$.

A more detailed optimization of the suggested many-level hierarchical parallel probabilistic searching algorithm including the optimization of the corresponding multiplicative constant, as well as a more detailed investigation of many-level hierarchical parallel probabilistic searching algorithms in general, i.e. with different n_i's for different levels and with N_i's not necessarily in the form of a geometric sequence, all these questions would be of great interest and would deserve further research, but the limited extend of this work forces us to postpone such an investigation till another occasion.

Let us turn back, now, to the two-level conflict-free hierarchical parallel probabilistic searching algorithms as introduced in Chapter 4, but now we shall consider the possibility when each processor may, at least partially, take profit of the successes reached by other processors, or it may take profit of an oracle which simulates the "expected" or "average" behaviour of other processors. A more detailed informal description seems to be worth introducing after an appropriate formalization.

Let

$$\mathscr{U}^* = \langle \{X_{ij}, Y_{ij}, U_{ij}\}_{i=1,j=1}^{m,n}, \{Z_l\}_{l=1}^{k} \rangle \tag{8.20}$$

be a structure consisting of mutually statistically independent random variables defined on the abstract probability space $\langle \Omega, \mathscr{S}, P \rangle$, taking their values in A (for X_{ij}), in the set $\{1, 2, \ldots, m\}$ of integers (for Y_{ij} and Z_l), and in the binary set $\{0, 1\}$ (for U_{ij}), and such that, for a fixed value Q, $0 \leqq Q \leqq 1$, for each $i \leqq m, j \leqq n, r \leqq m, l \leqq k$, and $a \in A$,

$$P(\{\omega\colon \omega \in \Omega, X_{ij}(\omega) = a\}) = 1/N\,, \tag{8.21}$$

$$P(\{\omega\colon \omega \in \Omega, Y_{ij}(\omega) = r\}) = P(\{\omega\colon \omega \in \Omega, Z_l(\omega) = r\}) = 1/m\,, \tag{8.22}$$

$$P(\{\omega\colon \omega \in \Omega, U_{ij}(\omega) = 1\}) = Q\,, \tag{8.23}$$

we shall omit the symbols $\ldots \omega\colon \omega \in \Omega, \ldots$, if no misunderstanding menaces. Define binary random variables W_{ij}, $1 \leqq i \leqq m, 0 \leqq j \leqq n$, as follows: $W_{i,0}(\omega) = 0$ for each $i \leqq m, \omega \in \Omega$,

$$W_{ij}(\omega) = \operatorname{sign}\left[\chi_V(X_{ij}(\omega)) + W_{i,j-1}(\omega) + W_{Y_{ij}(\omega),j-1}(\omega)\, U_{ij}(\omega)\right] \tag{8.24}$$

for $j > 0$, recall that χ_V is the characteristic function or identifier of the subset V of A, and $\operatorname{sign}(x) = -1$ for $x < 0$, $\operatorname{sign}(x) = 1$ for $x > 0$, $\operatorname{sign}(0) = 0$. Now, set

$$\mathscr{U}_0^*(\omega) = \operatorname{sign}\left[\sum_{l=1}^{k} W_{Z_l(\omega),n}(\omega)\right]. \tag{8.25}$$

Informally, the ith processor or ($i \leqq m$) takes in the jth ($j \leqq n$) sequential step three statistically independent random samples: an element $X_{ij}(\omega)$ from A, the value $W_{Y_{ij}(\omega),j-1}(\omega)$ of the $Y_{ij}(\omega)$-th processor in the $(j-1)$st step, and an auxiliary result (0 or 1). The ith processor takes the unit output value ($W_{ij}(\omega) = 1$) iff either it took already this value in the $(j-1)$st step, or if it sampled an element from V, or, finally, if it is sampled a processor which reached already the unit value and is willing (with

the probability Q) to share this knowledge. The processors having terminated their activites, the supervizor samples some of them, asks for their final output values $W_{Z_l(\omega),n}(\omega)$ and computes $\mathscr{Y}_0^*(\omega)$ which can be understood as a statistical decision function solving the problem whether $V = \emptyset$ or not. As can be easily seen, if $V = \emptyset$, then $\mathscr{Y}_0^*(\omega) = 0$ (or if $\mathscr{Y}_0^*(\omega) = 1$, then after all, $X_{ij}(\omega) \in V$ for at least one $i \leqq m$, $j \leqq n$ must hold), if $\mathscr{Y}_0^*(\omega) = 1$, then $V \neq \emptyset$ for the same reason. If $V \neq \emptyset$, then the value $\mathscr{Y}_0^*(\omega) = 0$ represents an error the probability of which is to be, uniformly for all $V \subset A$, majorized by a given threshold value, choosing appropriately n, m and k.

Let

$$W_{ij}^*(\omega) = \operatorname{sign}\left[\chi_V(X_{ij}(\omega)) + W_{i,j-1}^*(\omega)\right], \tag{8.26}$$

evidently, $W_{ij}^*(\omega) \leqq W_{ij}(\omega)$ for each $\omega \in \Omega$, $i \leqq m$, $j \leqq n$. In fact, W_{ij}^* agrees with W_{ij} iff $Q = 0$ in (8.23), so that no information sharing among processors is possible. Moreover, random variables W_{ij}^* are statistically independent (which is obviously not the case for W_{ij}'s) so that, setting $v = (\operatorname{card} V)/N$ and applying the expected value operator E, we obtain by an easy calculation that

$$\begin{aligned} &EW_{Y_{ij}(\cdot),j}(\cdot) = \sum_{s=1}^{m}\left[EW_{s,j}(\cdot)\, P(\{Y_{ij}(\omega) = s\})\right] \geqq \\ &\geqq \sum_{s=1}^{m}\left[EW_{s,j}^*(\cdot)\, P(\{Y_{ij}(\omega) = s\})\right] = EW_{s,j}^*(\cdot) = \\ &= P(\{W_{s,j}^*(\omega) = 1\}) = 1 - (1 - v)^j . \end{aligned} \tag{8.27}$$

Given $V \subset A$, let $\{T_{ij}\}_{i=1,j=1}^{m,n}$ be a system of mutually and with respect to each X_{ij}, U_{ij} and Z_l statistically independent random variables taking $\langle \Omega, \mathscr{S}, P\rangle$ into $\{0, 1\}$ in such a way that, for all $i \leqq m, j \leqq n$

$$P(\{T_{ij}(\omega) = 1\}) = EW_{Y_{ij}(\cdot),j} \geqq 1 - (1 - v)^j , \tag{8.28}$$

Hence, random variables T_{ij} "simulate", in the sense of their expected values, random variables $W_{Y_{ij}(\cdot),j}$ and, because of their supposed statistical independence, simplify the computations. Consider the structure $\mathscr{Y}$ defined in the same way as in $\mathscr{Y}^*$, but with Y_{ij}'s replaced by T_{ij}'s and set

$$W_{i,0}^0(\omega) = 0 \quad \text{for each} \quad \omega \in \Omega , \tag{8.29}$$

$$W_{i,j}^0(\omega) = \operatorname{sign}\left[\chi_V(X_{ij}(\omega)) + W_{i,j-1}^0(\omega) + T_{ij}(\omega)\, U_{ij}(\omega)\right], \tag{8.30}$$

for $j > 0$. $\mathscr{Y}_0(\omega)$ is defined by (8.25), just with W_{ij} replaced by W_{ij}^0. So, random variables T_{ij} play the role of an oracle which, "from the God-like position", knows the actual state of things concerning the set V and "helps" to each processor in the degree or measure which is "in average" the same as if the processor asked for help its colleague sampled at random by Y_{ij}. If $V = \emptyset$, then $v = 0$, hence $T_{ij}(\omega) = 0$ for all $i \leqq m$, $j \leqq n$, $\omega \in \Omega$, so that W_{ij}^0 reduces to W_{ij}^* and $\mathscr{Y}_0(\omega) = 0$ for each $\omega \in \Omega$. On the other hand, $\mathscr{Y}_0(\omega) = 1$ does not imply that $X_{ij}(\omega) \in V$ for some $i \leqq m$, $j \leqq n$, as the knowledge about the existence of an element in V might follow from

the oracle's consultation. Nevertheless, if $V \neq \emptyset$, the result $\mathscr{Y}_0(\omega) = 0$ can be taken as an error and we would like to minimize its probability uniformly for all $V \subset A$.

Theorem 8.4. For each $\varepsilon > 0$ there exist a natural number N_0 and real numbers $c_1(\varepsilon)$, $c_2(\varepsilon)$ and $c_3(\varepsilon)$ such that, for $m = \lceil c_1 \sqrt[3]{N} \rceil$, $n = \lceil c_2 \sqrt[3]{N} \rceil$, $k = \lceil c_3 \sqrt[3]{N} \rceil$, for all $\emptyset \neq V \subset A$, and for all $N = \operatorname{card}(A) \geqq N_0$,

$$P(\{\omega: \omega \in \Omega, \mathscr{Y}_0(\omega) = 0\}) < \varepsilon . \tag{8.31}$$

Proof. Cf. the Main Assertion in [2]. □

When choosing m, n and k in $\mathcal{O}(\sqrt[3]{N})$, it is an "almost optimal" choice in the following sense.

Let $\emptyset \neq V \subset A$, let $v = (\operatorname{card} V)/N$, set $q_{in} = P(\{W_{in}^0(\omega) = 0\})$. As $W_{ij}^*(\omega) \leqq W_{ij}^0(\omega)$, $q_{in} \leqq P(\{W_{in}^*(\omega) = 0\}) = (1 - v)^n$. Moreover,

$$\begin{aligned} q_{i1} &= P(\{W_{i1}(\omega) = 0\}) = P(\{\chi_V(X_{i1}(\omega)) = 0\}) = \\ &= P(\{X_{ij}(\omega) \in A - V\}) = 1 - v ; \end{aligned} \tag{8.32}$$

due to the supposed statistical independence of the corresponding random variables we obtain that

$$\begin{aligned} q_{ij} &= P(\{\chi_V(X_{ij}(\omega)) = 0\} \cap \{W_{i,j-1}^0(\omega) = 0\} \cap \{T_{ij}(\omega)\, U_{ij}(\omega) = 0\}) = \\ &= P(\{\chi_V(X_{ij}(\omega)) = 0\})\, P(\{W_{i,j-1}^0(\omega) = 0\})\, [1 - P(\{T_{ij}(\omega)\, U_{ij}(\omega) = 1\})] = \\ &= (1 - v)\, q_{i,j-1}[1 - [P(\{U_{ij}(\omega) = 1\})\, P(\{T_i(\omega) = 1\})]] = \\ &= (1 - v)\, q_{i,j-1}[1 - Q(1 - (1 - v)^j)] . \end{aligned} \tag{8.33}$$

An easy induction immediately yields that

$$q_{in} \leqq (1 - v)^n \prod_{j=0}^{n-1} (1 - Q(1 - (1 - v)^j)) . \tag{8.34}$$

This upper bound for q_{in} does not depend on i and is, considering $V \neq \emptyset$, evidently maximal iff V is a singleton, hence, iff $v = 1/N$. So, for each $i \leqq m$,

$$q_{in} \leqq q_n^* = (1 - (1/N))^n \prod_{j=0}^{n-1} (1 - Q(1 - (1 - (1/N))^j)) . \tag{8.35}$$

Theorem 8.5. Let the notations and conditions of Theorem 8.4 hold, let $Q < 1$, let $n(N) = d_1 N^\alpha$, $k(N) = d_2 N^\alpha$ for some $\alpha < \frac{1}{3}$, $d_1, d_2 > 0$, let q_n^* be defined by (8.35), then

$$\lim_{N \to \infty} (q_n^*)^k = 1 . \tag{8.36}$$

Proof. We have to prove that

$$\lim_{N \to \infty} \left[\left(1 - \frac{1}{N}\right)^n \prod_{j=0}^{n-1} \left(1 - Q\left(1 - \left(1 - \frac{1}{N}\right)^j\right)\right)\right]^k = 1 . \tag{8.37}$$

If $n(N), k(N) \in \mathcal{O}(N^{\alpha})$, $\alpha < \frac{1}{3}$, then $nk \in \mathcal{O}(N^{2\alpha}) \subset o(N)$, so that

$$\lim_{N\to\infty} \left(1 - \frac{1}{N}\right)^{nk} = 1 \tag{8.38}$$

and we have to prove that

$$\lim_{N\to\infty} \left[\prod_{j=0}^{n-1} \left(1 - Q\left(1 - \left(1 - \frac{1}{N}\right)^{j}\right)\right)\right]^{k} = 1 . \tag{8.39}$$

This assertion can be easily reduced to

$$\lim_{N\to\infty} \ln\left[\prod_{j=0}^{n-1} \left(1 - Q\left(1 - \left(1 - \frac{1}{N}\right)^{j}\right)\right)\right]^{k} =$$

$$= \lim_{N\to\infty} k \sum_{j=0}^{n-1} \ln\left(1 - Q\left(1 - \left(1 - \frac{1}{N}\right)^{j}\right)\right) = 0 . \tag{8.40}$$

But, for $0 < x < 1$,

$$0 > \ln(1 - x) > -\frac{x}{1 - x}, \tag{8.41}$$

hence,

$$|\ln(1 - x)| < \frac{x}{1 - x}, \tag{8.42}$$

so that, instead of (8.40), we have to prove that

$$\lim_{N\to\infty} k \sum_{j=0}^{n-1} \frac{1(1 - (1 - 1/N)^{j})}{1 - Q(1 - (1 - 1/N)^{j})} = 0 . \tag{8.43}$$

As $Q(1 - (1 - 1/N)^{j}) < Q < 1$, (8.43) reduces to

$$\lim_{N\to\infty} k \sum_{j=0}^{n-1} Q\left(1 - \left(1 - \frac{1}{N}\right)^{j}\right) = 0 . \tag{8.44}$$

Omitting Q as a multiplicative constant, an easy calculation yields

$$k \sum_{j=0}^{n-1} \left(1 - \left(1 - \frac{1}{N}\right)^{j}\right) = k\left(n - \sum_{j=0}^{n-1} \left(1 - \frac{1}{N}\right)^{j}\right) =$$

$$= k\left(n - \frac{1 - (1 - 1/N)^{n}}{1/N}\right) = kn - kN\left(1 - \left(1 - \frac{1}{N}\right)^{n}\right) \tag{8.45}$$

$$= kn - kN\left[-\sum_{j=0}^{n-1} \binom{n}{j} (-1)^{j} \left(\frac{1}{N}\right)^{j}\right] =$$

$$= kn - kN\left[\binom{n}{1}\left(\frac{1}{N}\right) - \binom{n}{2}\left(\frac{1}{N}\right)^{2} - \sum_{j=3}^{n} \binom{n}{j} (-1)^{j} \left(\frac{1}{N}\right)^{j}\right] =$$

$$= kN\binom{n}{2}\left(\frac{1}{N}\right)^{2} + kN \sum_{j=3}^{n} \binom{n}{j} (-1)^{j} \left(\frac{1}{N}\right)^{j} .$$

Denoting the last expression in (8.45) by S, we can easily obtain that

$$S < k\frac{n(n-1)}{2} + kN\sum_{j=3}^{n}\binom{n}{j}\left(\frac{1}{N}\right)^j < \frac{kn^2}{2N} + kN\sum_{j=1}^{n}\left(\frac{n}{N}\right)^j, \tag{8.46}$$

as $\binom{n}{j} < n^j$. Setting $n = d_1N^\alpha$, $k = d_2n^\alpha$, we obtain that

$$S < \left[\frac{d_1^2d_2}{2}\right]\frac{N^{3\alpha}}{N} + \sum_{j=3}^{n} d_2N^{1+\alpha}\left[\frac{d_1N^\alpha}{N}\right]. \tag{8.47}$$

Taking N_0 such that, for $N \geqq N_0$, $d_1N^\alpha/N < \frac{1}{2}$, we have

$$S < \left[\frac{d_1^2d_2}{2}\right]\frac{N^{3\alpha}}{N} + d_2N^{1+\alpha}\frac{(d_1(N^\alpha/N))^3}{1 - d_1(N^\alpha/N)} <$$

$$< \left[\frac{d_1^2d_2}{2}\right]\frac{N^{3\alpha}}{N} + 2d_1^3d_2N^{4\alpha-2}, \tag{8.48}$$

and this expression tends to 0 for $N \to \infty$ and $\alpha < \frac{1}{3}$. But, at the same time,

$$S > \frac{k(n-1)^2}{2N} - kN\sum_{j=3}^{n}\binom{n}{j}\left(\frac{1}{N}\right)^j, \tag{8.49}$$

and both the items in (8.49) tend to 0 for $N \to \infty$ and $\alpha < \frac{1}{3}$ for the same reasons as above. Hence, $S \to 0$ as well and the assertion is proved. □

Let us turn back, for a moment, to the original model with real, i.e. non-simulated, co-operation among processors and with decision function $\mathscr{Y}_0^*$ defined by (8.25). If $Q > 0$, the computation of parameters under which $P(\{\mathscr{Y}_0^*(\omega) = 0\}) < \varepsilon$, for $V \neq \emptyset$, is much more difficult than in the simulated case above. Or, using the oracle T_{ij}, the probability with which a processor is given the information that $V \neq \emptyset$ does not depend on whether an element from V has been already found or not. On the other hand, to obtain $\mathscr{Y}_0^*(\omega) = 1$, we ultimately need $X_{ij}(\omega) \in V$ for some $i \leqq m, j \leqq n$. In fact, setting

$$A_{m,n}(\omega) = \bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\}, \tag{8.50}$$

we obtain that

$$P(\{\mathscr{Y}_0^*(\omega) = 0\}) = P(\{\mathscr{Y}_0^*(\omega) = 0\}/\{A_{m,n}(\omega) = \emptyset\})\,P(\{A_{m,n}(\omega) = \emptyset\}) +$$

$$+ P(\{\mathscr{Y}_0^*(\omega) = 0\}/\{A_{m,n}(\omega) \neq \emptyset\})\,P(\{A_{m,n}(\omega) \neq \emptyset\}) \geqq$$

$$\geqq P(\{A_{m,n}(\omega) = \emptyset\}) = (1 - v)^{mn}, \tag{8.51}$$

as $A_{m,n}(\omega) = \emptyset$ implies $\mathscr{Y}_0^*(\omega) = 0$, which was not the case for $\mathscr{Y}_0$. Hence, we urgently need $mn \in \mathcal{O}(N)$ to be sure that $P(\{\mathscr{Y}_0^*(\omega) = 0\}) < \varepsilon$ for all $\emptyset \neq V \subset A = \{a_1, a_2, \ldots, a_N\}$, so that an analogy of the results from above with $m, n \in \mathcal{O}(\sqrt[3]{N})$ is impossible. But, even if the condition $mn \in \mathcal{O}(N)$ is the same as in the case with $Q = 0$, the fact

that the knowledge about an element from V can propagate, more or less quickly, among the processors, suggests an idea to optimize the product mn with $n \in o(\sqrt{N})$ and with $k \in o(\sqrt{N})$ as well. Because of the fact that the possible statistical dependences among the corresponding random variables make the direct computation very difficult, let us introduce a modified model which will be used to estimate the time computational complexity of the original model for $Q > 0$.

The modification consists in separating the samplings from A from the samplings of auxiliary processors and in postponing the consultation phase. So, having m processors formalized by random vectors $\langle X_{i,1}, X_{i,2}, \ldots, X_{i,n} \rangle$, $i \leqq m$, first of all each processor takes n sequential samples from A, independent of each other as well as of the samples taken by other processors, and tests the sampled elements as far as their membership to V is concerned. Set, for $i \leqq m$,

$$\tilde{W}_{i,0}(\omega) = \operatorname{sign}\left[\sum_{j=0}^{n} \chi_V(X_{ij}(\omega))\right], \tag{8.52}$$

so that $\tilde{W}_{i,0}(\omega) = 1$ iff $X_{ij}(\omega) \in V$ for at least one $j \leqq n$, $\tilde{W}_{i,0}(\omega) = 0$ otherwise. Evidently, for each $i \leqq m$

$$P[\{\tilde{W}_{i.0}(\omega) = 0\}] = (1 - v)^n, \tag{8.53}$$

let us denote this value by $q_0 = q_0(n)$. Take a system $\{Y_{ij}, U_{ij}\}_{i=1, j=1}^{m, r}$ of random variables satisfying the conditions described after (8.20) and including (8.21) to (8.23) and set

$$\tilde{W}_{ij}(\omega) = \operatorname{sign}\left[\tilde{W}_{i,j-1}(\omega) + \tilde{W}_{Y_{ij}(\omega),j-1}(\omega)\, U_{ij}(\omega)\right], \tag{8.54}$$

$j = 1, 2, \ldots, r$. The intuition behind this definition is like that in the case of (8.24). Hence, $\tilde{W}_{ij}(\omega) = 1$ iff either $\tilde{W}_{i,j-1}(\omega) = 1$ or if Y_{ij} sampled (the index of) a processor which took the unit value in the $(j-1)$st postponed auxiliary step and is willing to share this knowledge, i.e. $U_{ij}(\omega) = 1$.

Lemma 8.2. Let $r \geqq n$, let W_{ij} be defined by (8.24), then for all $i \leqq m$, $j \leqq n$, $\tilde{W}_{ij}(\omega) = 0$ implies $W_{ij}(\omega) = 0$, hence,

$$P(\{\tilde{W}_{ij}(\omega) = 0\}) \leqq P(\{W_{ij}(\omega) = 0\}). \tag{8.55}$$

Proof. Let us prove that $W_{ij}(\omega) = 1$ implies $\tilde{W}_{ij}(\omega) = 1$. For $j = 0$ it holds trivially, suppose the validity for $j - 1$. (8.54) yields that, if $X_{ij}(\omega) \in A - V$,

$$\begin{aligned}
&\{\omega\colon \tilde{W}_{ij}(\omega) = 1\} = \{\omega\colon \tilde{W}_{i,j-1}(\omega) = 1\} \cup \\
&\cup \bigcup_{l=1}^{m} (\{\omega\colon \tilde{W}_{l,j-1}(\omega) = 1,\ Y_{ij}(\omega) = l,\ U_{ij}(\omega) = 1\}) \supset \\
&\supset \{\omega\colon W_{i,j-1}(\omega) = 1\} \cup \bigcup_{l=1}^{m} (\{\omega\colon W_{l,j-1}(\omega) = 1, Y_{ij}(\omega) = l, U_{ij}(\omega) = 1\}) = \\
&= \{\omega\colon W_{ij}(\omega) = 1\}.
\end{aligned} \tag{8.56}$$

If $X_{ij}(\omega) \in V$, we obtain

$$\{\omega: X_{ij}(\omega) \in V, \tilde{W}_{i,j}(\omega) = 1\} = \bigcup_{a \in V} (\{\omega: X_{ij}(\omega) = a, \tilde{W}_{i,j}(\omega) = 1\}) \supset$$

$$\supset \bigcup_{a \in V} (\{\omega: X_{ij}(\omega) = a, W_{ij}(\omega) = 1\}) = \{\omega: X_{ij}(\omega) \in V, W_{ij}(\omega) = 1\} \quad (8.57)$$

and the lemma is proved. □

New, let $Z_1, Z_2, \ldots, Z_k$ be the same random variables as above and set

$$\mathscr{Y}_1^*(\omega) = \operatorname{sign}\left[\sum_{l=0}^{k} \tilde{W}_{Z_l(\omega),r}(\omega)\right]. \quad (8.58)$$

An easy calculation yields

$$P(\{\mathscr{Y}_1^*(\omega) = 0\}) = P\left(\bigcap_{l=1}^{k} \{\tilde{W}_{Z_l(\omega),r}(\omega) = 0\}\right) =$$

$$= P\left(\bigcap_{l=1}^{k} \bigcup_{s=1}^{m} \{\tilde{W}_{s,r}(\omega) = 0, z_l(\omega) = s\}\right) \leqq$$

$$\leqq P\left(\bigcap_{l=1}^{k} \bigcup_{s=1}^{m} \{\tilde{W}_{s,r}(\omega) = 0, Z_l(\omega) = s\}\right) = P(\{\mathscr{Y}_0^*(\omega) = 0\}), \quad (8.59)$$

so that the values for m, n and k, necessary to keep $P(\{\mathscr{Y}_1^*(\omega) = 0\})$ below a given threshold value, are also necessary to keep $P(\{\mathscr{Y}_0^*(\omega) = 0\})$ below the same threshold value.

Because of computational difficulties connected with an explicit expression for $P(\{\mathscr{Y}_1^*(\omega) = 0\})$, let us appropriately approximate $\mathscr{Y}_1^*$ by another random variable $\mathscr{Y}_2^*$. Set $\tilde{W}_{i,0}^*(\omega) = \tilde{W}_{i,0}(\omega)$ for all $\omega \in \Omega$, for $j > 0$ set

$$\tilde{W}_{ij}^*(\omega) = \operatorname{sign}\left[\tilde{W}_{i,j-1}^*(\omega) + \tilde{W}_{Y_{1j}(\omega)}^*(\omega)\, U_{ij}(\omega)\right], \quad (8.60)$$

so that, in the jth auxiliary step, only one processor with the index $Y_{1j}(\omega)$ is sampled at random and if its value is the unit, it is shared with all other processors with the probability Q; the random events of sharing are statistically independent for different processors. Random variable $\mathscr{Y}_2$ is defined by (8.58), but with $\tilde{W}_{Z_l(\omega),r}(\omega)$ replaced by $\tilde{W}_{Z_l(\omega),r}^*(\omega)$. As can be easily computed, the corresponding conditional probabilities read as

$$P(\{\tilde{W}_{ij}^*(\omega) = x\}/\{\tilde{W}_{i,j-1}^*(\omega) = y\}) = P(\{\tilde{W}_{ij}(\omega) = x\}/\{\tilde{W}_{i,j-1}(\omega) = y\}) \quad (8.61)$$

for all $x, y \in \{0, 1\}$, and in this sense $\mathscr{Y}_2^*$ approximates $\mathscr{Y}_1^*$.

Theorem 8.6. For each $\varepsilon > 0$ there exist real numbers c_1, c_2, c_3, c_4 independent of N such that, for $m = \lceil c_1 N^{2/3} \rceil$, $n = \lceil c_2 \sqrt[3]{N} \rceil$, $k = \lceil c_3 \sqrt[3]{N} \rceil$ and $r = \lceil c_4 \sqrt[3]{N} \rceil$,

$$P(\{\mathscr{Y}_2^*(\omega) = 0\}) < \varepsilon \quad (8.62)$$

for all $\emptyset \neq V \subset A = \{a_1, a_2, \ldots, a_N\}$.

Remark. As n, r, and k are the numbers of samples which are to be taken subsequently, their sum $n + r + k$ may serve as a first and very rough approximation

of the computational complexity of the parallel probabilistic searching algorithm with co-operation of stochastic type defined by the random variable $\mathscr{Y}_2^*$. Theorem 8.6 claims this complexity to be in the $\mathcal{O}(\sqrt[3]{N})$-class, as in the case of simulated co-operation investigated above, but the necessary number of processors increases quadratically (in the $\mathcal{O}(N^{2/3})$-class) when compared with the simulated case with m in $\mathcal{O}(\sqrt[3]{N})$. The reason for this increase is simple: in the case of $\mathscr{Y}_i^*$, $i = 0, 1, 2$, processor cannot take profit of the oracle's knowledge that $V \neq \emptyset$ without having actually sampled at least one element from V.

Proof of Theorem 8.6. First of all, suppose that there is $i \leqq m, j \leqq n$ such that $X_{ij}(\omega) \in V$ and denote, for $j = 0, 1, \ldots, r$,

$$u_j = u_j(\omega) = \operatorname{card}\{i: i \leqq m, \tilde{W}_{ij}^*(\omega) = 0\} = m - \sum_{i=0}^{m} \tilde{W}_{ij}^*(\omega), \tag{8.63}$$

$$v_j = v_j(\omega) = \operatorname{card}\{i: i \leqq m, \tilde{W}_{ij}^*(\omega) = 1, \tilde{W}_{i,j-1}^*(\omega) = 0\} =$$

$$= u_{j-1}(\omega) - u_j(\omega) = \sum_{i=0}^{m} \tilde{W}_{ij}^*(\omega) - \sum_{i=0}^{m} \tilde{W}_{i,j-1}^*(\omega). \tag{8.64}$$

If $\tilde{W}_{Y_{1j}(\omega),j-1}^*(\omega) = 1$, then the number of unit values occurring for the first time in the jth auxiliary step depends just on the results of "consultation random events" $U_{ij}(\omega)$, so that v_j has the binomial probability distribution with the probability Q of success. Abbreviating $\tilde{W}_{Y_{1j}(\omega),j-1}^*(\omega)$ by $L_j(\omega)$, we obtain

$$P(\{v_j(\omega) = k\}/\{u_{j-1}(\omega) = m_1, L_j(\omega) = 1\}) = \binom{m_1}{k} Q^k (1-Q)^{m_1-k}, \tag{8.65}$$

so that the well-known relation concerning the expected value of the binomial probability distribution yields

$$E(v_j(\cdot)/\{u_{j-1}(\omega) = m_1, L_j(\omega) = 1\}) = Qm_1. \tag{8.66}$$

(8.64) implies that, for all $j \leqq r$, $\omega \in \Omega$,

$$u_j(\omega) = u_{j-1}(\omega) - v_j(\omega), \tag{8.67}$$

so that

$$E(u_j(\cdot)/\{u_{j-1}(\omega) = m_1, L_j(\omega) = 1\}) = m_1 - Qm_1 = (1-Q)\, m_1. \tag{8.68}$$

Now,

$$E(u_j(\cdot)/\{L_j(\omega) = 1\}) =$$

$$= \sum_{m_1=0}^{m} [E(u_j(\cdot)/\{L_j(\omega) = 1, u_{j-1}(\omega) = m_1\})\, P(\{u_{j-1}(\omega) = m_1\})] =$$

$$= (1-Q) \sum_{m_1=0}^{m} m_1\, P(\{u_{j-1}(\omega) = m_1\}) = (1-Q)\, Eu_{j-1}(\cdot). \tag{8.69}$$

If $L_j(\omega) = 0$, then no new processor takes the unit value in the jth step, so that

$$E(u_j(\cdot)/\{L_j(\omega) = 0\}) = Eu_{j-1}(\cdot). \tag{8.70}$$

The probabilities of the corresponding conditioning events read as follows:

$$P(\{L_j(\omega) = 1\}/\{u_{j-1}(\omega) = m_1\}) = (m - m_1)/m = 1 - (m_1/m)\,, \tag{8.71}$$

so that

$$P(\{L_j(\omega) = 0\}/\{u_{j-1}(\omega) = m_1\}) = m_1/m\,. \tag{8.72}$$

Computing the expected values and setting $u_j^*(\omega) = m^{-1}\, u_j(\omega)$ we obtain that

$$P(\{L_j(\omega) = 1\}) = \sum_{m_1=0}^{m} (1 - (m_1/m))\, P(\{u_j(\omega) = m_1\}) = 1 - Eu_j^*(\cdot)\,, \tag{8.73}$$

$$P(\{L_j(\omega) = 0\}) = \sum_{m_1=0}^{m} (m_1/m)\, P(\{u_j(\omega) = m_1\}) = Eu_j^*(\cdot)\,. \tag{8.74}$$

Combining (8.69), (8.70), (8.73), and (8.74), we have

$$\begin{aligned} Eu_j^*(\cdot) &= E(u_j^*(\cdot)/\{L_j(\omega) = 1\})\, P(\{L_j(\omega) = 1\}) + \\ &+ E(u_j^*(\cdot)/\{L_j(\omega) = 0\})\, P(\{L_j(\omega) = 0\}) = \\ &= (1 - Q)\, Eu_{j-1}^*(\cdot)\,(1 - Eu_{j-1}^*(\cdot)) + (Eu_{j-1}^*(\cdot))^2 = \\ &= (1 - Q)\, Eu_{j-1}^*(\cdot) + Q(Eu_{j-1}^*(\cdot))^2\,. \end{aligned} \tag{8.75}$$

Evidently, $u_j^*(\omega) \leqq u_{j-1}^*(\omega)$ for each $j \leqq r$, $\omega \in \Omega$, so that $Eu_j^*(\cdot) \leqq Eu_{j-1}^* \leqq Eu_0^*(\cdot)$. Hence,

$$(Eu_j^*(\cdot))/(Eu_{j-1}^*(\cdot)) = (1 - Q) + Q\, Eu_{j-1}^*(\cdot) \leqq (1 - Q) + Q\, Eu_0^*(\cdot)\,, \tag{8.76}$$

and

$$Eu_j^*(\cdot) \leqq ((1 - Q) + Q\, Eu_0^*(\cdot))^j\,. \tag{8.77}$$

Recalling the definition of u_0^* we obtain

$$Eu_0^*(\cdot) = m^{-1}\, E \operatorname{card}\{i\colon i \leqq m,\ \tilde{W}_{i,0}^*(\omega) = 0\}\,. \tag{8.78}$$

Random events $\tilde{W}_{i,0}^*(\omega) = 0$ are identical with $\tilde{W}_{i,0}(\omega) = 0$ and are, for different i's, statistically independent with the same probability $(1 - v)^n$, so we obtain

$$Eu_0^*(\cdot) = P(\{\tilde{W}_{i,0}^*(\omega) = 0\}) = (1 - v)^n \leqq (1 - 1/N)^n\,, \tag{8.79}$$

as $V \neq \emptyset$. So

$$Eu_j^*(\cdot) \leqq [(1 - Q) + Q(1 - 1/N)^n]^j\,. \tag{8.80}$$

Consider the random variables $Z_1, Z_2, \ldots, Z_k$ which sample the (indices of the) processors after the rth auxiliary step, i.e., each T_l samples among the values $\tilde{W}_{i,r}^*(\omega)$, $i \leqq m$. Due to the supposed statistical independence of random variables Z_l, X_{ij}, Y_{ij}, U_{ij} and due to their equiprobable distribution, for each $l \leqq k$

$$\begin{aligned} &P(\{\tilde{W}_{Z_l(\omega),r}^*(\omega) = 1\}/\{\operatorname{card}\{i\colon \tilde{W}_{i,r}^*(\omega) = 0\} = m_1\}) = \\ &= P(\{\tilde{W}_{Z_l(\omega),r}^*(\omega) = 1\}/\{u_r^*(\omega) = m_1/m\}) = 1 - (m_1/m)\,, \end{aligned} \tag{8.81}$$

so that, using the same simple computation as above,

$$P(\{\tilde{W}_{Z_l(\omega),r}^*(\omega) = 1\}) = 1 - Eu_r^*(\cdot)\,. \tag{8.82}$$

Recalling our initial assumption and taking, once more, profit of the supposed statistical independence of the corresponding random variables we obtain that

$$P(\{\mathscr{Y}_2^*(\omega) = 0\}/\{\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\} \cap V \neq \emptyset\}) =$$

$$= P(\bigcap_{i=1}^{m}\{\bar{W}_{Z_i(\omega),r}(\omega) = 0\}) = [Eu_r^*(\cdot)]^k \leqq [(1 - Q) + Q(1 - 1/N)^n]^{kr} =$$

$$= [(1 - Q(1 - (1 - 1/N)^n))]^{kr} < (1 - (Qn/N))^{kr}, \tag{8.83}$$

as $(1 - x)^n > 1 - nx$ for each $0 < x < 1$. Hence, in order to have, for some $\varepsilon_1 > 0$, the conditional probability in (8.83) majorized by ε_1, a sufficient condition reads

$$\left(1 - \frac{Qn}{N}\right)^{kr} < \varepsilon_1, \tag{8.84}$$

hence

$$kr \ln (1 - (Qn/N)) < \ln \varepsilon_1. \tag{8.85}$$

But, $\ln (1 - x) < -x$ for each $0 < x < 1$, so that a sufficient condition for (8.83) to be majorized by ε_1 reads

$$krQnN^{-1} > \ln \varepsilon_1^{-1} \tag{8.86}$$

This condition can be easily satisfied with $n = c_2 \sqrt[3]{N}$, $k = c_3 \sqrt[3]{N}$, $r = c_4 \sqrt[3]{N}$, where c_i, $i = 2, 3, 4$, depend on Q and ε_1, but not on N, say $c_i > \sqrt[3]{(Q^{-1} \ln \varepsilon_1^{-1})}$ will do. This choice is optimal in the sense that $n + r + k$ is in $\mathcal{O}(\sqrt[3]{N})$, which is not the case for other possibilities when $n + r + k$ is in $\mathcal{O}(N^\alpha)$ for some $\alpha > \frac{1}{3}$.

Obviously,

$$P(\{\mathscr{Y}_2^*(\omega) = 0\}/\{\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\} \cap V \neq \emptyset\}) = 1, \tag{8.87}$$

so that, after a simple factorization,

$$P(\{\mathscr{Y}_2^*(\omega) = 0\}) < P(\{\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\} \cap V = \emptyset\}) +$$

$$+ P(\{\mathscr{Y}_2^*(\omega) = 0\}/\{\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\} \cap V \neq \emptyset\}) \leqq (1 - 1/N)^{mn} + \varepsilon_1, \tag{8.88}$$

for n, k, r defined as above. So, taking $\varepsilon_1 = \varepsilon/2$ we need $mn > (\ln (2/\varepsilon))\, N$ to keep $(1 - 1/N)^{mn}$ below $\varepsilon/2$, so that the optimal solution is $m = c_1 N^{2/3}$ for an appropriate $c_1 = c_1(\varepsilon, Q)$. The theorem is proved. □

Let us reconsider the proof of Theorem 8.6 from another point of view. If there are $i \leqq m$, $j \leqq n$ such that $X_{ij}(\omega) \in V$, then the worst case is that with just one such a pair $\langle i, j \rangle$. In this case $u_0^*(\omega) = (m - 1)/m = 1 - m^{-1}$, setting into (8.77) we obtain

$$P(u_j^*(\cdot)/\{\text{card}\,(\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\} \cap V) = 1\}) \leqq$$

$$\leqq ((1 - Q) + Q(1 - 1/m))^j = (1 - Q/m)^j. \tag{8.89}$$

Using the same computation as above,

$$P(\{\mathscr{Y}_2^*(\omega) = 0\}/\{\text{card}\,(\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\{X_{ij}(\omega)\} \cap V) = 1\}) \leqq (1 - Q/m)^{kr}, \quad (8.90)$$

hence, to majorize this conditional probability by ε_1, we need

$$kr \ln (1 - Q/m) < \ln \varepsilon_1 , \quad (8.91)$$

which evidently holds if $krQ/m > \ln \varepsilon_1^{-1}$. This can be satisfied with $k = c_3 \sqrt[3]{N}$, $r = c_4 \sqrt[3]{N}$, $m = c_1 N^{2/3}$, supposing that $c_1 c_2 / c_3 > Q^{-1} \ln \varepsilon_1^{-1}$. To keep also $(1 - 1/N)^{mn}$ below $\varepsilon - \varepsilon_1 > 0$, we need $n = c_2 \sqrt[3]{N}$ for an appropriate c_2. Such a choice solves the case when $X_{ij}(\omega) \in A - V$ for all $i \leqq m$, $j \leqq n$, by reducing its probability below a fixed threshold value, say, $\varepsilon/2$.

REFERENCES

[1] I. Kramosil: Hierarchies of parallel probabilistic searching algorithms with possible data access conflicts. Problems Control Inform. Theory *18* (1989), 6, 381–395.

[2] I. Kramosil: A simulation of partial stochastic co-operation in parallel probabilistic searching algorithms. In: Artificial Intelligence and Information-Control Systems of Robots 89 — Proceedings of the conference held at Štrbské Pleso, 6.—10. 11. 1989 (I. Plander, ed.), North Holland, Amsterdam 1989, pp. 159—162.

SUPPLEMENTARY REFERENCES

[1] S. G. Akl: Parallel Sorting Algorithms. Academic Press, Orlando 1985.

[5] Y. Azar and U. Vishkin: Tight comparison bounds on the complexity of parallel sorting. SIAM J. Comput. *16* (1987), 3, 458—464.

[3] P. Bachman and Phan-Mink-Dung: Nondeterministic computations — structure and axioms. Elektron. Informationsverarb. Kybernet. *22* (1986), 5—6, pp. 243—261.

[4] G. Bobrow and A. Collins (eds.): Representation and Understanding. Academic Press, New York 1975.

[5] D. J. Boxma: A probabilistic analysis of multiprocessor list scheduling: the Erlang case. Comm. Statist. Stochastic Models *1* (1985), 2, 209—220.

[6] M. Broy: On the Herbrand-Kleene universe for nondeterministic computations. Theoret. Comput. Sci. *36* (1985), 1, 1—19.

[7] M. Broy: A theory for nondeterminism, parallelism, communication, and concurrency. Theoret. Comput. Sci. *45* (1986), 1, 1—61.

[8] C. L. Chang and R. T. C. Lee: Symbolic Logic and Mechanical Theorem Proving. Academic Press, New York—London 1974 (Russian translation: Mir, Moscow 1982).

[9] A. Church: Introduction to Mathematical Logic I. Princeton Univ. Press, Princeton, New Jersey 1956 (Russian translation: ILL, Moscow 1960).

[10] R. Davis and D. B. Lenat: Knowledge-Based Systems in Artificial Intelligence. McGraw-Hill, New York 1982.

[11] J. Gill: Computational complexity of probabilistic Turing machines. SIAM J. Comput. *6* (1977), 4, 675—695.

[12] M. Karmarkar, R. M. Karp, G. S. Lueker and A. M. Odlyzko: Probabilistic analysis of optimum partitioning. J. Appl. Probab. *23* (1986), 3, 626—645.
[13] G. A. P. Kindervater and J. K. Lenstra: An introduction to parallelism in combinatorial optimization. In: Parallel Computers and Computations, CWI Syllabi 9, Math. Centrum Amsterdam, 1985, pp. 163—184.
[14] L. Kronsjö: Computational Complexity of Sequential and Parallel Algorithms. J. Wiley and Sons, Chichester 1985.
[15] L. Kučera: Kombinatorické algoritmy (Combinatorial Algorithms — in Czech). SNTL, Prague 1983.
[16] M. Luby: A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput. *15* (1986), 4, 1036—1053.
[17] Z. Manna and R. Waldinger: Special relations in automated deduction. J. Assoc. Comput. Mach. *33* (1986), 1, 1—59.
[18] A. N. Maslov: Verojatnostnyje mašiny Turinga i rekursivnyje funkcii (Probabilistic Turing machines and recursive functions — in Russian). Dokl. Akad. Nauk SSSR *203* (1972), 5, 1018—1020.
[19] D. Mitra: Probabilistic models and asymptotic results for concurrent processing with exclusive and non-exclusive locks. SIAM J. Comput. *14* (1985), 4, 1030—1051.
[20] Parallelnaja obrabotka informacii, vol. 2 (Parallel Information Processing — a collection of papers — in Russian). Naukova dumka, Kijev 1985.
[21] Sborník: Expertní systémy — principy, realizace, využití (Proceedings: Expert Systems — principles, realizations, applications, V. Zdráhal, V. Mařík, eds.). ČSVTS FEL ČVUT, Prague 1984.
[22] Sborník: Metody umělé inteligence a expertní systémy (Proceedings: Methods of Artificial Intelligence and Expert Systems, Z. Zdráhal, V. Mařík, eds.). ČSVTS FEL ČVUT, Prague 1985.
[23] J. R. Shoenfield: Mathematical Logic. Addison-Wesley, Reading 1967 (Russian translation: Nauka, Moscow 1975).
[24] J. R. Smith: Parallel algorithms for depth-first searches — planar graphs. SIAM J. Comput. *15* (1986), 3, 814—830.
[25] J. Sztrik: A probability model for priority processor-shared multiprogrammed computer systems. Acta Cybernet. *7* (1986), 3, 329—340.
[26] Wang Hao: A Survey of Symbolic Logic. North-Holland, Amsterdam and China Press, Peking 1962.
[27] D. A. Watermann and L. Hayes-Roth (eds.): Pattern-Directed Interference Systems. Academic Press, New York 1978.
[28] G. Winskel: Category theory and models for parallel computations. In: Category Theory and Computer Programming (Lecture Notes in Comp. Sci. 240), Springer-Verlag, Berlin—Heidelberg—New York 1987, pp. 266—281.
[29] M. Zaionc: Nondeterministic programs definable in typed lambda-calculus. Fundam. Informaticae *8* (1985), 1, 63—72.

CONTENTS