A Tour of Algorithmics

P.M.B. Vitányi

1. Introduction

Computer science distinguishes itself among most other sciences in that it does not deal exclusively with a given reality (like physics) nor does it deal exclusively with a man-made ideal construct (like mathematics). Instead, computer science deals with invented ideal constructs, some aspects of which have to be realized in the physical world. Like the human body there is a physical shape and mental operations being performed in this physical substrate. The physical body corresponds with the physical architecture of the computing device, and the mental equipment and operations correspond to the various algorithms the device executes. The notions of *algorithm* and *architecture* are basic to all computer programming, and so are the *complexity* issues which arise in this context. Of course, algorithms need to be expressed in particular programming languages, and checked for correctness, much like the mind and body of a human need to be checked on appropriate function. The rôle of medication is filled by logics-based semantics and related fields in computer science and is not dealt with here.

2. A Brief History

The word 'algorithm' itself derives from the 9th century Persian Abu Ja'far Mohammed ibn Mûsâ al-Khowârizmî (native of Khowârizm, today the small city of Khiva in the former Soviet Union). He is the author of a celebrated book which preserved large parts of mathematics from antiquity through

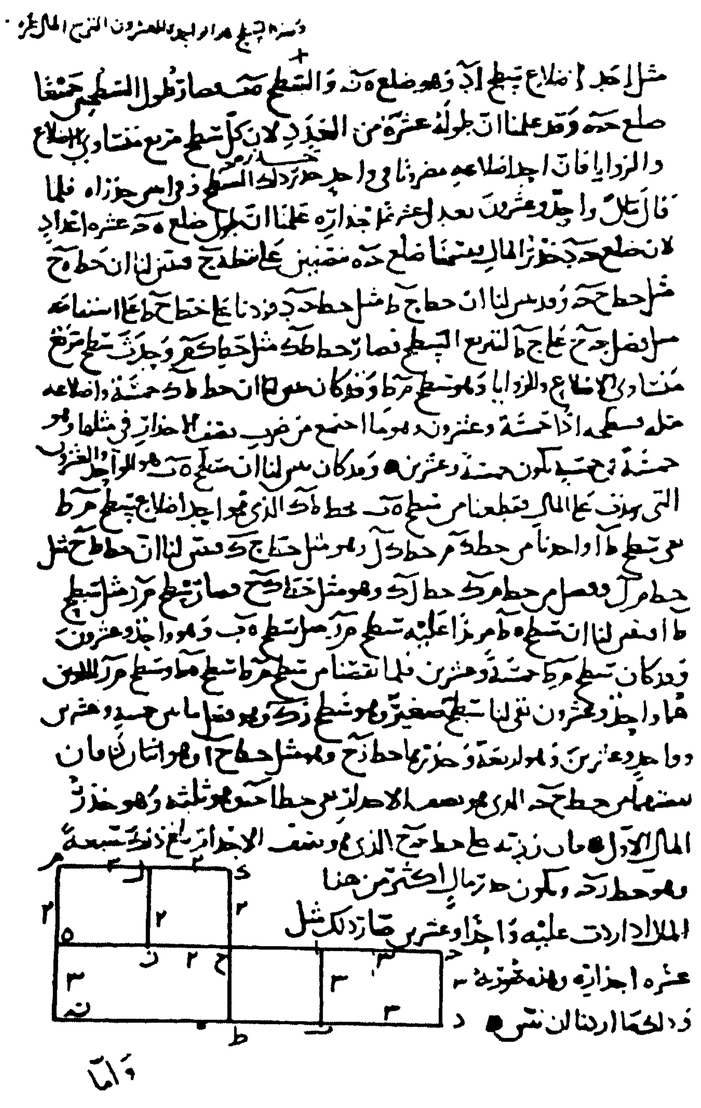

Figure 1. Part of the 9th century Arabic work 'al-jabr wa l-muqabala' by al-Khowârizmî, dealing with the solution of the quadratic equation $x^2 + 21 = 10x$.

the dark ages. Incidentally, the word 'algebra' is derived from the book's title *Kitab al jabr wa l-muqabala* which means 'Rules of restoration and reduction'. In the middle ages there was a fierce struggle between 'abacists' who calculated on the abacus or counting board, and the 'algorists' computing with pencil and paper using algorithms for addition, subtraction, multiplication and division following the teaching of al-Khowârizmî.

In his famous address to the International Mathematical Congress in 1900, D. Hilbert proposed twenty-three mathematical problems as a programme to direct the mathematical efforts in the twentieth century. The tenth problem asks for an algorithm which, given an arbitrary Diophantine equation, produces either an integer solution for this equation or indicates that no such solution exists. (In the 1970s Yu.V. Matijasevich showed that no such algorithm exists.)

The idea of a completely mechanical procedure, an *algorithm*, to find solutions to mathematical questionsgoes back at least to Hilbert. In 1931

K. Gödel proved that not every true mathematical statement is provable in a finitely axiomatized system of mathematics. With the purpose of identifying fundamental ideas immanent in Gödel's proof, in 1936 A.M. Turing exhibited an exceedingly simple type of hypothetical machine and gave a brilliant demonstration that everything that can be reasonably said to be computed by a human computer using a fixed procedure can be computed by such a machine. As Turing claimed: any process which can be naturally called an effective procedure is realized by a Turing machine. This is known as *Turing's Thesis*. Over the years, all serious attempts to give precise yet intuitively satisfactory definitions of a notion of 'effective procedure' have turned out to define essentially the same class of processes. (In his original paper, Turing established the equivalence of his notion of 'effective procedure' with A. Church's notion of 'effective calculability'.)

Church's Thesis states that, in this sense, there is an objective notion of effective computability independent of a particular formalization. According to Gödel: 'With this concept one has for the first time succeeded in giving an absolute definition of an interesting epistimological notion, i.e., one which does not depend on the formalism chosen. In all other cases treated previously...one has been able to define them only relative to a given language, and for each individual language it is clear that the one [definition] thus obtained is not the one looked for... This situation [according to Church's Thesis] is some kind of miracle.'

Thus, theoretically any formally computable function is computable by a laptop computer with indefinitely expandable memory or by a Turing machine, however clumsy the latter may be. But a computation that takes 2^n steps on an input of length n would not be regarded as *practical* or *feasible.* No computer would ever finish such a computation in the lifetime of the universe even with n merely 1000. Computational complexity theory tries to identify problems that are feasibly computable.

> If we have 10^9 processors taking 10^9 steps/second, then we can execute $3.1 \times 10^{25} < 2^{100}$ steps/year.

This shows that in practice the relevant question is whether a computation is also *feasible.* To characterize this notion of feasibility J. Edmonds in 1965 proposed a classification of computational problems in terms of polynomial time bounds on the length of the computation.

A problem is in the compexity class P if it can be solved (the answer is 'yes' or 'no') in time polynomial in the input length, and in NP if it can be solved by a so-called 'nondeterministic algorithm' in polynomial time. Informally speaking, P is the set of 'yes-no' problems where it is easy to find the answer (easy: doable by a deterministic Turing machine in polynomial time), and NP the set where it is easy to show that the answer is 'yes'.

Normally, we do not ask questions unless we can easily recognize the good answer. NP is about those questions that we are likely to want answers to.

The question: $P = NP$? is possibly the most important problem in computer science if not in mathematics. Attempts to resolve this question have thus far mainly led to reformulations or reductions. For example, the difficulty of the entire problem class NP has been reduced to a nucleus of so-called 'NP-hard' problems (S. Cook and R. Karp, 1971; L. Levin, 1973).

These notions are tied to sequential computation such as performed by a Turing machine or a Von Neumann architecture computer. However, in the past years hopes have emerged that nonclassical or nonstandard physical realizations of computers may have different properties that may help in beating the NP barrier.

Numerous computer developments together with an ever-increasing complexity of the problems handled by computers, produce challenging demands

requiring the invention of new architectures for emergent computer technologies and more efficient algorithmic designs. Research questions cover the design, construction and use of hardware, as well as applications. Solutions to these problems are sought via improved networks and parallel architectures, partially through exploitation of opportunities arising in novel applications of physics phenomena, in combination with efficient algorithms. For an overivew, see for example D. Knuth's *The Art of Computer Programming* Series published by Addison-Wesley. In this article we trace CWI-based research in this area.

3. Past Highlights

3.1. Machine complexity

In machine complexity we are interested in the variation of computing power resulting from variation of machine parameters. Three well-known open basic problems were resolved as follows.

1. It is possible to real-time simulate a fixed finite number of independent counters on-line by a one tape Turing machine, [4].

2. It requires n^2 steps to simulate n steps of a k-tape Turing machine by a one-tape Turing machine, and many such results, were discovered independently by M. Li, W. Maass and P.M.B. Vitányi. This matches the trivial upper bound and improves the previous best lower bound of $n\sqrt{\log n}$ by almost an order of magnitude.

3. For over 30 years it was conjectured that two heads on the same work tape (of a Turing machine) are more powerful in real time than two work tapes with one head each. Vitányi published in 1984 a preliminary lemma, aimed at eventually proving this result, which was accomplished very recently by T. Jiang, J. Seiferas, and Vitányi.

3.2. Computational number theory

A.K. Lenstra (then at CWI), H.W. Lenstra, Jr., and L. Lóvasz [1] showed that factorization of polynomials over the radicals into irreducible factors can be performed in deterministic polynomial time. Later, the link between cryptography and computational number theory was pursued at CWI by E. Kranakis (who authored the first monograph on public key cryptosystems while at Yale University), and by D. Chaum, who founded CWI's group on cryptographic research. (See also H.J.J. te Riele's article in this volume.)

3.3. Distributed and parallel computing

Around 1985 attention shifted from sequential computing to distributed and parallel computing. Distributed computation is related to the emergence of

computer networks: computer applications moved from single stand-alone mainframes to multiple communicating local workstations. Parallel computation arose from the quest of fundamentally improving the speed of sequential computation by using multiple processing units. Both fields generate questions of architecture of physical interconnects and topologies, and concurrent algorithms for control of interprocessor communication and applications. We give a selection of CWI related research.

1. The common approach towards synchronicity issues of multiprocessor systems was to assume that either the processors were totally synchronized or totally unsynchronized. We pioneered an approach in between: 'Archimedean time systems', which is more realistic in terms of real-time issues.

2. Many communication issues in multicomputer systems such as mutual exclusion, name server, load balancing, data integrity, voting systems, and so on, have a mutual underlying core which was identified and analyzed by S.J. Mullender and Vitányi as 'distributed match-making'.

3. A basic primitive for asynchronous interprocess communication was identified by L. Lamport as wait-free read-write shared register. He constructed the single user case. For multiple users the problem of implementing such shared memory primitives from basic available electronic components or software components becomes very difficult and possibly *a priori* impossible. Vitányi and B. Awerbuch developed the appropriate theory and gave a basic implementation now known as the Vitányi-Awerbuch register, [6]. To settle the theoretically interesting question whether such a construction can exist using only a bounded number of control bits, after several published erroneous solutions by several researchers, M. Li and Vitányi (later joined by J.T. Tromp) gave the first uncontested solution.

4. In a sequential computation one can safely ignore many physical aspects of the underlying computer system and analyse the computational complexity of a program in a purely logical fashion. This is not the case in nonsequential computation. Moreover, nonclassical or nonstandard physical realizations of computers may have totally unexpected properties. A popular model to analyse parallel algorithms is the parallel random access machine (PRAM), where many processors can read and write a single shared memory in unit time per operation.

   In fact, optimality of PRAM algorithms may be misleading, because in any physically realizable machine architecture a much simpler and unsophisticated algorithm may outperform the optimal PRAM algorithm. Do networks help with this problem? We can simulate PRAMs

fast by networks of processors communicating by message passing at the cost of a multiplicative slowdown square logarithmic in the number of processors n for simulation on a $\log n$-dimensional hypercube. However, this does not solve the problem mentioned above, since the hypercube nodes need to be order $n^{1/3}$ apart for the majority of pairs (see below). Together it turns out that rather than saving time, the simulation costs at least a logarithmic in n factor more time than the original. R. Landauer at IBM T.J. Watson Research Labs has emphasized that 'information is physical'. So is communication. We have analyzed the real physical aspects of proposed computer architectures through a sequence of papers debunking many popular misconceptions about models for parallel computations.

4. Kolmogorov Complexity and the Incompressibility Method

In parts of the research mentioned above, we and our collaborators developed a new mathematical proof technique now known as the *incompressibility method*—a basic technique such as the 'pidgeon hole' argument, 'the counting method' or the 'probabilistic method'.

The new method is based on so-called Kolmogorov complexity, a modern notion of randomness proposed by A.N. Kolmogorov in 1965 to quantify the randomness of individual objects in an objective and absolute manner. This is impossible by classical probability theory (a branch of measure theory satsifying the so-called Kolmogorov axioms formulated in 1933). Likewise, the Kolmogorov complexity of an object is a form of absolute information of the individual object. This is not possible to do by C. Shannon's information theory, since the latter is only concerned with the average information of a random source.

After we pioneered several successful applications of Kolmogorov complexity in the theory of computation, the general pattern of the incompressibility method emerged. It is a sharper relative of classical information theory and yet satisfies many of the laws of classical information theory—although with a slight error term.

Applications of Kolmogorov complexity by us and others have been given in a plethora of areas, including the theory of computation, inductive reasoning, formal language theory, computational learning theory, combinatorial theory, randomness, Gödel style incompleteness results, graph theory, Kolmogorov 0-1 Laws, theory of parallel and distributed computation, average complexity, sorting, string matching, routing in computer networks, circuit theory, complexity of tapes, stacks, queues, complexity of parallel random access machines, in physics of computation, information distance (for example in pattern recognition) and so on. A comprehensive account of both theory and applications is given in the (text)book [2].

5. Current and Future Directions

Current research at CWI is in the direction of machine learning and physical aspects of computation, while continuing work in communication infrastructure in parallel and distributed computation. The incompressibility method and other compression based techniques are used throughout.

5.1. Computational Machine Learning

It is not always realized that most of traditional statistics is about computational learning. It is always involved with algorithms to obtain the general from the particular. A novel approach in statistical learning is based on the minimum description length of hypotheses and data together—one way to express the so-called MDL principle.

In our work we follow the thread of inductive inference and pac learning and, from the other end, we examine an approach related to statistics, Bayesian reasoning, and the principle of 'minimum description length' or 'MDL' for short. It appears to us that the future of computational machine learning will involve combinations of these approaches coupled with guaranties with respect to used time and memory resources. It is clear that computational learning theory will move closer to practice and the application of principles such as MDL requires further justification. Building on our earlier work, as reported in the textbook [2], we can justify certain applications of MDL via the Bayesian approach as follows.

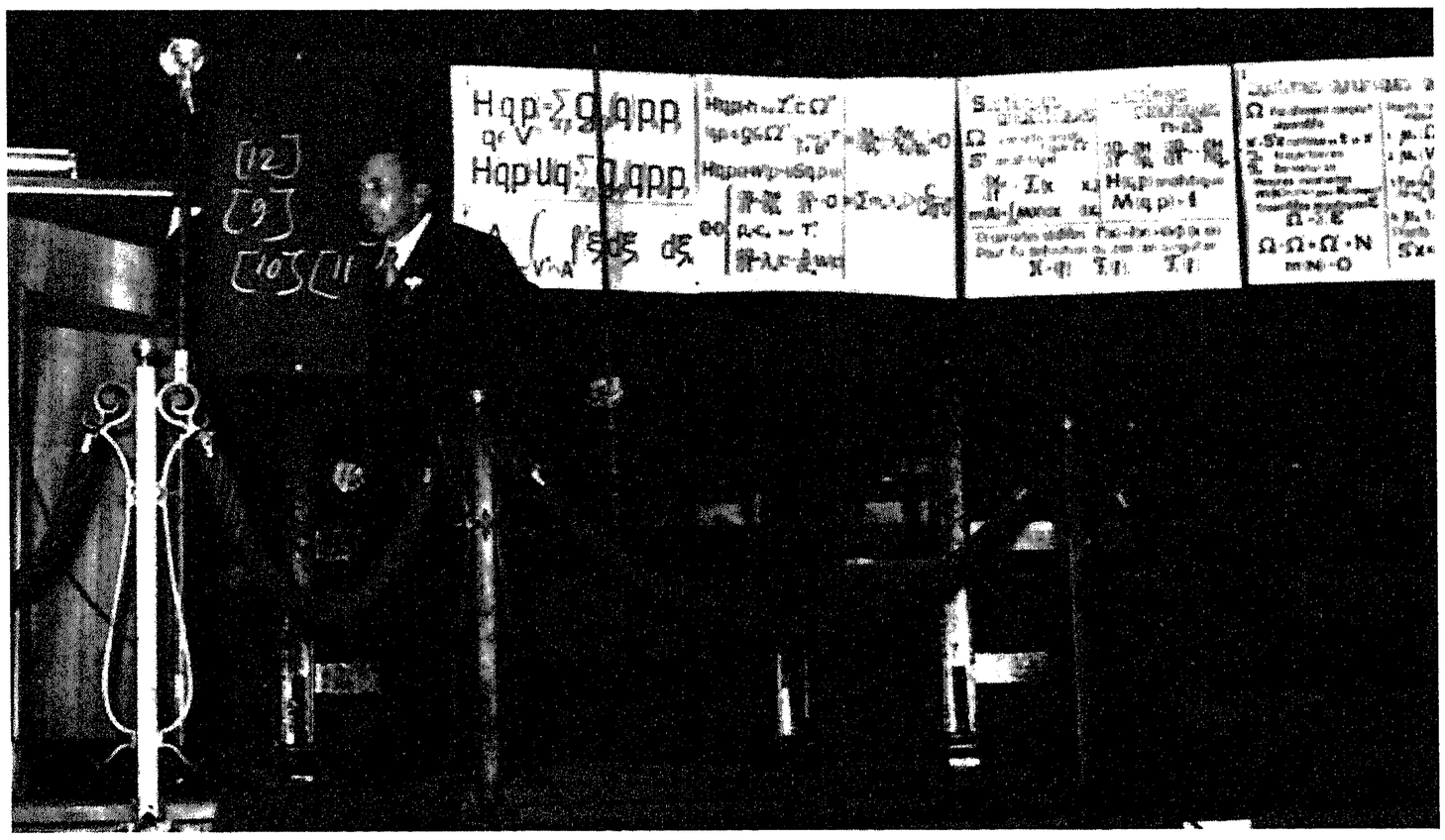

Figure 2. A.N. Kolmogorov (1903-1987) at the 1954 International Mathematical Congress in Amsterdam.

A general task of statistical learning is to select the most plausible hypothesis in the light of experimental evidence. The classic method to do so is Bayes' rule. The problem with applying Bayes' rule is that one first requires the prior probabilities of the possible hypotheses. Unfortunately, it is often impossible to obtain these.

One way out of this conundrum is to require inference of hypotheses to be completely data driven. The MDL approach embodies this idea. This approach is currently widely and successfully used in many diverse applications. MDL is usually presented as justified in and of itself by philosophical persuasion. As one of the founders, C. Wallace, remarked at an AAAI meeting at Stanford University in 1990, 'the most surprising thing about MDL is that no monster has yet sprung from the woods'. That is, the principle has not yet known spectacular failures in practice. It is important for theoretical foundation and for practical application that a firm basis for the principle is established.

In [3] we supply a rigorous justification for MDL from first principles, identify similarities and differences with Bayesian inference, and give a comparison of pac learning criteria and MDL algorithms for the practical topic of decision tree learning.

5.2. Computational Linguistics

Given a body of text (a *corpus*), we want to automatically derive a grammar for it. While systems doing so will produce unfamilar grammars for, say, natural language texts, in other contexts we do not know or care what is 'natural'. For example, the grammatical rules of entries in the *Oxford English Dictionary* are poorly described and unknown. Yet automatically extracting rules from existing texts yields a grammar which can monitor correct format of new entries. In the *Human Genome Project* an enormous corpus of genetic data has been collected and is available in data banks. Extracting a grammar from these data can be used to validate or reject

hypotheses on this material. The approach we use is primarily based on statistics of pattern conjunctions. This is used to generate the grammatical syntax rules.

Generally, the main problem of such approaches is how to judge relative goodness of alternative possibilities and similarly when to stop complicating the grammar to obtain a better fit with the data. We anticipate that using the MDL principle in this essentially data-driven process is the right thing to do. Our initial results seem to confirm this idea.

5.3. Multiple Computing Agents

Computational approaches based on the biology of the 'brain' and 'evolution' comprise research in areas of neurocomputing and adaptive computing. The employed programming techniques are referred to as 'multiple

computing agents'. Such approaches turn out to be very useful to apply to ill-defined problems which can only with great difficulty be expressed in conventional algorithmics, such as for example problems of computer vision and speech recognition.

Neurocomputing deals with the design, analysis and application of networks built from artificial neurons. Here the goal is not to imitate the human brain but to design a functional computing engine.

Genetic algorithms solve optimization problems in a way which is based on natural selection in biology. A population of possible solutions (programs) must converge in a short time to yield the best (or a very good) solution. Even more than in neurocomputing, we can talk about 'automatic programming' in this setting. The program develops by itself, governed by 'evolutionary fitness and selection' (therefore, one also talks about 'evolutionary' or 'genetic' programming). This approach is successful in contexts where it is almost impossible to write explicit programs. This is very often the case in practice. Genetic programming appears to be a popular approach with software houses, since the result is an optimal program which can be 'explained' to the client, rather than a set of optimal weights in a neural network which may perform great but have no direct intuitive explanation. At CWI we have developed a pilot implementation of novel evolutionary programs based partially on the genetic programming paradigm, with more capabilities than hitherto known (FALS), which will be extensively tested on real problems.

Automatic programming with genetic algorithms and the like for providing near-optimal solutions to ill-defined problems is widely used by software developers in commercial applications. For example, it is used by banks to judge credit card applications and by KLM to predict career planning of its pilots.

The method has become so popular because it is relatively easy to program and leads in practice almost invariably to very good performance. The reasons for this are mathematically still poorly understood. Because of its immense commercial impact it is paramount that the mathematical underpinnings of this discipline are discovered and performance guaranties can be given, and, moreover, that parameters which speed up convergence and improve quality of solutions are identified.

Our investigation in the underlying mathematical theory is based on Markov processes and focuses on the analysis and exploitation of 'rapid mixing' properties. Preliminary results point at a mode of operation where many short runs of the genetic algorithm are more likely to yield an optimal program than one very long run—in contrast to current usage. Another direction of work concerns the speed-up of convergence to high quality solutions by application of the Bayesian approach and the MDL principle.

5.4. Routing

In computer networks routing of messages (much like email over internet) is a vital item. As networks grow larger, routing information present at each particular site increases to unmanageable size. Clearly, it is sufficient to maintain a routing table at each node which says over which adjacent node a message to a target node must be routed. However, such a method requires routing tables in all nodes of size $n \log n$ and the question is how to route messages using as compact as possible routing tables. A typical method is *interval routing*. Adjacent nodes are first ordered (for example lexicographically by name), and then a set of intervals on the set of nodes, say $\{1, \ldots, n\}$, is assigned to each such adjacent node. The intervals are chosen such that together they cover $\{1, \ldots, n\}$ completely. To route a message to any target node, we first look for an interval containing the target node, and then route the message to the adjacent node associated with the interval. Clearly, for certain networks (like trees) interval routing can be very efficient in the sense of saving a lot of bits in the description of the routing information.

We use Kolmogorov complexity to determine the optimal space used by routing tables in communication networks for both worst-case static networks and on the average for all static networks. This resolves the problem of the necessary and sufficient size of all routing tables together for unrestricted routing schemes. Similarly we determine the optimum routing table size for shortest path routing on almost all graphs (the Kolmogorov random graphs which constitute a fraction of at least $1 - 1/n^3$ of all graphs). We prove that $\Theta(n^2)$ bits are sufficient and necessary for the total size of the routing tables. We show that this implies the same optimum for the average of all graphs. It turns out that our methods are applicable to many different current models used in applied routing.

5.5. Quantum Coherent Computation

New computation devices increasingly depend on particular physical properties. The theory of computation is thus becoming an increasingly interdisciplinary subject, because of the need to understand and apply physical laws in computational considerations.

Quantum coherent computation (QCC) is a new field of research that has attracted considerable attention over the last 10 years (see for example the article by S. Lloyd in the Scientific American, October 1995). Recent evidence that the proposed coherent quantum computers may be intrinsically much faster than classical computing devices makes their technological development of great economic interest. QCC may contribute to solving standing open problems in computation theory as well as increase our understanding of quantum phenomena.

The ultimate limits of the speed of computation are determined by the

energy dissipation per unit area per unit time, which increases with the density of switching elements in the computing device. Linear speed up by shortening interconnects on a two-dimensional device is attended by cubing the dissipated energy per area unit per second, and dissipation on this scale in the long run cannot be compensated for by cooling. The dissipated energy per bit operation at room temperature has decreased from 10^{-2} Joule in 1940 to 10^{-17} Joule at present, and extrapolation of current trends shows that within twenty years a further reduction below the thermal noise level of 10^{-21} Joule is required. Reducing the energy dissipation of a computation is very relevant in the development of massive multiprocessing architectures [5], where the interconnect volume essentially is a square or cube power of the processor volume. The requirement of (almost) dissipationless computation has led computer scientists to consider 'reversible' computation, for which there is no lower bound on the energy dissipation. It has been established that the functionality of reversible classical logical gates forms a proper subclass of the functionality of quantum logical gates. An efficient methodology of quantum gate construction for n-bit unitary transformations is still a subject of current research.

Since R. Feynman's statement that quantum mechanics does not impose a physical limitation on computation, the question whether quantum-based computation can be more efficient than classical (probabilistic) computation, has subsequently become an object of intensive research. This has led to a *nonclassical* emergent possible computer technology (quantum coherent computation or QCC).

The QCC approach will partially alleviate the interconnect problem (above) because a large number of different computation paths can be simultaneously followed (with appropriate probability amplitudes, to be sure) by the same single physical apparatus requiring but a tiny amount of physical space. This is the substance of Feynman's dictum 'there is room at the bottom' in the context of his proposal of QCC. Of course, since the different computation paths of a quantum computation cannot communicate as is often a main feature in a parallel distributed computation, it is only a very special type of room which is available at the bottom.

In 1994 P. Shor showed a remarkable result, which suddenly made the physical realization of a quantum computer extremely interesting. Shor provided a quantum algorithm that could do integer factoring (or discrete logarithm) with a bounded probability of error in polynomial time. On a classical computer with currently known methods the determination of prime factors can be an exceedingly difficult (exponential time) problem, although verification is trivial. This asymmetry is the basis of modern cryptography and is used to obtain secret codes used in your bank card and to transfer diplomatic messages between embassies.

Physical implementation of quantum algorithms, like Shor's, will however

face great problems. The power of quantum computation is related to the use of coherent quantum superpositions in the computation. This coherence, however, can be destroyed by the inevitable coupling to the 'environment'. W. Unruh has calculated that QCC computations using physical realizations based on spin lattices will have to be finished in an extremely short time. For example, for factoring a 1000 bit number in square quantum factoring time we have to perform 10^6 steps in less than the thermal time scale $\hbar/kT$ which at 1 K is of order 10^{-9} seconds.

Another problem is *error correction*: measurements to detect errors will destroy the computation. A novel partial method for error correction has been suggested by A. Berthiaume, D. Deutsch and R. Josza.

It will require an exploration of several candidate physical systems to obtain a more precise idea of the possibilities and limitations of the physical implementation of a quantum computer.

REFERENCES

1. A.K. LENSTRA, H.W. LENSTRA, JR., L. LOVÁSZ (1982). Factoring polynomials with rational coefficients, *Math. Ann. 261*, 515-532. See also: A.K. Lenstra, H.W. Lenstra, Jr. (1990). Algorithms in number theory. In: *Handbook of Theoret. Comp. Sci.*, J. VAN LEEUWEN (ed.), MIT Press/Esevier, Amsterdam.
2. M. LI, P.M.B. VITÁNYI (1993). *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York.
3. M. LI, P.M.B. VITÁNYI (1995). Computational Machine Learning in Theory and Praxis. In: *Lecture Notes in Computer Science, Vol. 1000*, Springer Verlag, Heidelberg. (Invited paper)
4. P.M.B. VITÁNYI (1985). An optimal simulation of counter machines. *SIAM Journal on Computing 14*, 1-33. See also: J. Seiferas, P.M.B. Vitányi (1988). Counting is easy, *J. Assoc. Comp. Mach. 35*, 985-1000.

5. P.M.B. VITÁNYI (1995). Physics and the New Computation, Prague, August 1995, Proc. 20th Int. Symp. Math. Foundations of Computer Science, MFCS'95, *Lecture Notes in Computer Science, Vol 969*, Springer-Verlag, Heidelberg. (Invited paper)
6. P.M.B. VITÁNYI, B. AWERBUCH (1986). Atomic shared register access by asynchronous hardware. *Proceedings 27th Annual IEEE Symposium on Foundations of Computer Science*, Errata, FOCS'87, 233-243.