Hans J. Bremermann

Professor of Mathematics and Biophysics, University of California at Berkeley.

Programmed and operated John von Neumann's pioneering computer at Princeton in 1955. Early researches into the physical limitations of computer processes influenced theoretical cybernetics and emphasized the importance of the modern theory of computational complexity.

Recent research has been concerned to some extent with complexity theory, but is mostly concerned with understanding naturally occurring "computers" and efficient methods of solving cybernetic problems.

# *COMPLEXITY AND TRANSCOMPUTABILITY*

In this essay I wish to point out serious limitations to the ability of computers to carry out enough computations to solve certain mathematical and logical problems. The same limitations also apply to data processing by nerve nets, and thus ultimately to human thought processes.

The present era has seen advances in computing that are no less spectacular than advances in space travel. Today a single computer can do more arithmetic operations in a year, than all of mankind has done from its beginnings till 1945 when the first electronic computer became operational. I will show in the following that this achievement is insignificant when measured against the vastness of what one might call the "mathematical universe".

Computers are physical devices and as such are subject to and limited by the laws of physics. For example, no signal can travel faster than light in vacuum ($3 \times 10^8$ m/sec) and this restriction applies especially to signals inside a computer. Let $\tau_{\text{switch}}$ be the switching time of computer components. (In the fastest computers $\tau_{\text{switch}}$ is of the order of $10^{-9}$ to $10^{-8}$ sec (1 to 10 nanoseconds).) The travel time of signals between different parts of a computer is determined by the distance a signal has to travel. That means $\tau_{\text{travel}}$ = distance/velocity $\geqslant$ distance/light velocity.

The travel time of signals between different parts of a computer should not exceed the switching time. Otherwise travel time would be the limiting factor in the speed of the computer.

Hence $\tau_{\text{switch}} \geqslant \tau_{\text{travel}} \geqslant$ distance/light velocity. Hence, the distance between different parts of the computer is bounded by $\tau_{\text{switch}} \times 3 \times 10^8$ m/sec $\geqslant$ distance. Thus, if $\tau_{\text{switch}}$ is $10^{-9}$ sec, the distances in the computer are limited to 30 cm. In other words, the entire computer must be quite small. For $\tau_{\text{switch}} = 10^{-10}$ sec the maximum size would be 3 cm, etc.

Size and signal speeds are not the only limitations. More fundamental are the following: The different components of a computer communicate with each other by signals. The reception and interpretation of a signal constitutes a physical measurement. Physical measurements are governed by the uncertainty principle of quantum mechanics. One can show: the faster the measurement the larger is the energy that is required to make the signal readable with sufficiently small error probability. If the total signalling energy is limited, then there is a trade-off between the number of distinguishable signals that can be sent and the time required to identify them. It is customary to call the logarithm (base 2) of the number of distinguishable messages the *information content* (measured in bits (binary digits)). It turns out that for given energy the amount of information that can be sent is proportional to time. The proportionality factor is given by $E/h$ [$\text{sec}^{-1}$]. Where $E$ is the energy, $h$ is Planck's constant. *The amount of signal flow (in bits/sec) in a computer is thus limited by E/h, where E is the energy available for signalling*. (Bremermann, 1962, 1967, 1978; R. Thom, 1972, p. 143 (English translation Thom, 1975).)

This fundamental limit of data processing applies to computers, irrespective of the details of their construction. It can even be extended to computers other than digital machines (and thus becomes applicable to data processing by nerve nets). We will return to this question later.

Granted that computers are thus limited, what does it mean? Is the fundamental limit a serious barrier to computations of practical importance, or is it a subtlety without significant implications? To answer this question one must know something about the computational requirements of mathematical problems.

## COMPLEXITY OF COMPUTATIONS

What exactly is the role of computation in mathematical problems? This question has been asked in earnest only in recent years. Very few results have been published prior to 1962, but since 1968 many papers have appeared that deal with this subject.

It will be useful to consider some specific examples before we attempt to examine the general question. Consider a system of two linear equations in two unknowns:

$$a_{11}x_1 + a_{12}x_2 = b_1,$$

$$a_{21}x_1 + a_{22}x_2 = b_2.$$

If $a_{11} \neq 0$ we multiply the first equation by $a_{21}/a_{11}$ and subtract it from the second which gives

$$a_{11}x_1 + a_{12}x_2 = b_1,$$

$$(a_{22} - a_{12}a_{21}/a_{11})x_2 = b_2 - b_1 a_{21}/a_{11}.$$

Solving for $x_2$, substituting the result in the first equation and solving for $x_1$ requires altogether nine arithmetic operations.

If we have three linear equations in three unknowns we may proceed analogously. Multiply the first equation with $a_{21}/a_{11}$ (provided $a_{11} \neq 0$) and substract it from the second. Multiply with $a_{31}/a_{11}$ and subtract it from the third. This reduces the second and third equation to a system of two equations in two unknowns which we solve as before.

The analogous method can be applied to four, five, any number of linear equations. It is known as *Gaussian elimination* and it is an example of what mathematicians call an *algorithm*. An algorithm is a method that takes the data that come with the problem (in our case the coefficients $a_{11}$, $a_{12}$ . . .) and transforms them step by step until the numbers are obtained that constitute the solution of the problem (in our example the values of $x_1, \ldots x_n$ if we have $n$ linear equations). It can be shown that the number of arithmetic operations required to carry out the Gaussian elimination algorithm is $\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n$.

We may call this number the *computational cost* of the algorithm. As the number of equations, $n$, increases, the cost goes up, and it goes up faster than $n$. According to our formula the dominant term is $\frac{2}{3}n^3$, thus the computational cost (as measured in terms of arithmetic operations) increases as the third power of the number of equations.

There are other methods to solve systems of linear equations. For example, Cramer's rule which computes $x_1, \ldots x_n$ by means of determinants, and in turn there are algorithms for computing determinants. The popular algorithms of computing determinants by developing them with respect to the elements of a row or column multiplied with smaller subdeterminants have a computational cost of the order of $n! = n(n-1)(n-2) \ldots 1$. The number $n!$ increases much faster than $n^3$.

When we implement an algorithm on a computer we must see to it that the computational cost of an algorithm does not exceed the number of arithmetic operations that a computer can carry out within a reasonable span of time. Before the advent of electronic computers, the computational capacity of existing machines and of hand computation by humans was quite limited.

Early electronic computers could perform 100 to 1000 arithmetic operations per second. Today's computers have reached performance rates of between 10 and 100 million arithmetic operations per second, or less than $10^{13}$ arithmetic operations per day. If we use Gaussian elimination, we thus can at most solve $\sqrt[3]{(3 \times 10^{13})} \approx 3 \times 10^4$ linear equations in a day. (The current monetary cost of a day of computer time may run as high as £10,000.) (In practice the maximum number of linear equations that can be solved is smaller than $3 \times 10^4$ because of round-off errors that are introduced, since numbers must be limited to a fixed number of digits.)

To increase the number of linear equations that can be solved we may explore the possibility of (a) algorithms that require fewer arithmetic operations and (b) computers of greater speed. These two questions are quite different; we will first discuss question (a).

As we have seen, there is an algorithm for solving linear equations that requires of the order of $n!$ arithmetic operations. This algorithm is worse than Gaussian elimination because $n!$ grows much faster than $\frac{2}{3}n^3$. In fact, by the mid-1960s Gaussian elimination had empirically proven itself as the best all-purpose algorithm for solving linear equations. Can this be proven rigorously?

This is not an easy task. For each $n$ we must determine the minimum of the computational costs of all possible algorithms that solve systems of $n$ linear equations. Since the computational cost of any algorithm is positive or at most zero, the computational costs of all algorithms are bounded below by zero. Therefore, for each $n$ the minimum exists. It is some integer between 0 and $\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n$.

In the mid-sixties some mathematicians conjectured that Gaussian elimination is indeed the best algorithm for solving linear equations and tried to prove that this is the case. However, in 1968 V. Strassen (then at Berkeley, now in Zürich) described an algorithm which for large $n$ has a lower computational cost than Gaussian elimination. Its computational cost is less than $4.7\, n^{\log_2 7}$, and $\log_2 7 \approx 2.807$. For large $n$ this number is less than $\frac{2}{3}n^3$. Strassen did not prove that his algorithm is the best of all possible algorithms. His result provides a better upper bound for the minimal cost. It can easily be shown that any algorithm requires at least $n^2$ arithmetic operations (in the worst case). Thus, we have the minimal computational cost bounded by $n^2$ below and by $4.7n^{2.807}$ above (Strassen, 1969).

So far we have considered a single mathematical task, namely solving systems of $n$ linear equations in $n$ unknowns. There are numerous other tasks of widespread interest in applications, such as finding the roots of a polynomial, solving systems of non-linear equations, optimizing a function of several variables, computing solutions of differential equations, etc. Operations research, which analyzes the efficiency of business operations and production processes, has its share of high cost computational problems. Some of these are known as "travelling salesman problem", "allocation problem", "shortest path problem", etc. It would take us too far to explain all of these problems in detail. The reader is referred to textbooks on operations research or to the paper of Karp (1972) which describes these and other problems and their interrelation in a concise way.

It may suffice to describe one of them, the travelling salesman problem, which is as follows: given $n$+1 cities $A, A_1, A_2, \ldots A_n$, any pair of cities has a distance between them. Suppose a salesman wants to visit each city exactly once, except $A$, the city from which he started and to which he wishes to return at the end of his trip. The problem is to find that routing of his trip which minimizes his total mileage; that is, the sum of the distances between the cities that he has visited. Each routing is given by a sequence: $A, A_{j_1}, \ldots A_{j_n}, A$ where $A_{j_i}, \ldots A_{j_n}$ is a permutation of the cities $A_1, \ldots A_n$, and $A$ is the city where the trip starts and ends. There are $n!$ such permutations, and hence the problem can be solved by examining $n!$ mileage sums and picking the minimum. Picking the minimum of $N$ numbers can be done with $N-1$ comparisons. Thus we could solve the problem at the cost of $n!$ comparisons between numbers (and an additional arithmetic cost of computing the mileage sums). This method, however, is not practical for large $n$. For example, for $n = 100$ we have (by a formula known as Stirling's formula):

$$n! = 100! > \sqrt{(200\pi)}\left(\frac{100}{e}\right)^{100}, \text{ where } e = 2.718.$$

This computational cost exceeds the capacity of any computing resource on earth.

Better algorithms than the one just described are known, but all known algorithms have a computational cost which increases faster than any finite power of $n$, where $n$ is the number of cities. The problem of whether there exist algorithm whose computational cost is bounded by some polynomial in $n$, for all $n$, is unsolved. (If such an algorithm exists the problem is called *polynomial.* Whether the travelling salesman problem is polynomial is a famous unsolved problem, and lately Karp (1972) has shown that in this respect many of the problems of operations research are tied together. Either they are all polynomial or none of them is.)

The theory of the computational costs of mathematical numerical problems is known as *complexity theory*. Before 1959 it was virtually non-existent. In recent years it has made giant strides and it is developing into an entirely new branch of mathematics and theoretical computer science. For a sampling of recent results the reader is referred to Miller and Thatcher (1972). (Some further discussion of complexity problems is also contained in Bremermann (1974) and in the author's forthcoming lecture notes on biological algorithms (1978).)

For many practical mathematical, engineering and accounting tasks the computational costs of available algorithms and the capacities of available computers are satisfactory, but there are exceptions. We already mentioned operations research. Another area of difficulty is the numerical solution of large systems of differential equations, especially differential equations that are *stiff.* (That is, systems that combine processes of greatly differing speeds.) Partial differential equations pose a problem, as do the *ab initio* calculations of molecular configurations from Schrödinger's equation.

In another area, artificial intelligence, the excessive computational cost of known algorithms has been the main obstacle to having, for example, computers play perfect games of chess (or checkers, or Go). If at each move a player has $k$ choices, then $n$ moves comprise $k^n$ possible move sequences. This number grows exponentially with $n$. For some games (like Nim) there are shortcuts that eliminate the need for searching through all the alternative move sequences. However, for chess (a game that is considered a true intellectual challenge and not mere child's play) such shortcuts have never been found. All known algorithms involve search through an exponentially growing number of alternatives and this number, when search is pursued to the end of the game, exceeds the power of any computing device.

A similar situation prevails in mathematical logic and in other branches of mathematics where there are formalized proof procedures. The search for the proof of a (conjectured) theorem always seems to involve search through an exponentially growing sequence of alternatives of formula transformations. Quite generally: most artificial intelligence problems require computing in amounts that grow exponentially with the depth of the search. The required search effort, in most cases, exceeds any available computing resource before it has reached sufficient depth to solve the problem (cf. Nilsson, 1971).

In summary: many mathematical, logical, and artificial intelligence problems cannot now be solved because the computational cost of known algorithms (and in some cases of all possible algorithms) exceeds the power of any existing computer.

## THE FUNDAMENTAL LIMIT OF DATA PROCESSING

Granted that certain problems cannot be solved with existing computers, may we expect that eventually all problems will become solvable through advances in computer technology?

Computer performance has indeed increased dramatically since 1945, when ENIAC became operational. However, as I indicated in the introduction, computer performance cannot be improved indefinitely. The signal flow in a computer is limited by $E/h$ [bits/sec], where $E$ is the energy available for signalling. How serious is this limit?

It is easy to derive an ultimate upper bound. This bound is not meant to be realistic in the sense that it would seem practically possible to build computers that come close to this bound. Practical bounds would be much harder to derive. Thus our fundamental limit is merely a far out yardstick beyond which improvement cannot go. It is comparable to saying: astronauts cannot travel at speeds exceeding the light velocity, though in practice their speeds are much more limited. Estimates of realistic limits of the speed of space travel would be much harder to derive, having to take into consideration rocket technology, etc.

As Einstein first observed, there is an equivalence relation between mass and energy. Energy has mass, and mass, if converted, yields energy in the amount of $E = mc^2$, where $c$ is the velocity of light in vacuum and $m$ is the mass that is being converted. An atomic power plant is, in fact, a device for converting mass to energy.

Consider now a closed computing system, with its own power supply. Let $m$ be the total mass of the system. This includes the mass equivalent of the energy of signals employed in the computer, while another share of the total mass is contained in the materials of which the computer and its power supply are made.

In existing computers the structural mass by far outweighs the mass equivalent of the signal energy. It would be difficult, however, to derive a realistic upper bound for this ratio. Thus, in order to avoid complicated arguments, we simply observe. *The total mass equivalent of the energy that is invested in signals cannot exceed m, where m is the total mass of the system.*

By combining this limit with the fundamental limit of the data processing we obtain:

*No closed computer system, however constructed, can have an internal signal flow that exceeds $mc^2/h$ bits per second.* (Here $m$ is the total mass of the system, $c$ the velocity of light in vacuum and $h$ is Planck's constant.)

This limit was derived by the author in 1961 (see Bledsoe, 1961; Bremermann, 1962). An improved argument was given by the author, Bremermann, 1967. A new discussion is to be contained in Bremermann, 1978.

The numerical value of $c^2/h$ is $1.35 \times 10^{47}$ (bits per second per gram). The number is large or small, depending upon the perspective. Existing computers process no more than $\approx 10^4$ to $10^5$ bits per second per gram, and the fundamental limit appears much too large for practical purposes. When compared with the complexity of some algorithms, however, the limit appears small. The limited age and the limited size of the Universe constitute (far out) outer limits to computing. Again we choose these very unrealistic outer limits to avoid complicated arguments that could be made in order to establish more realistic and stringent bounds to the product of mass and time that could be considered as available for computing. Current estimates of the age of the physical universe run to about 20 billion years, that is $2 \times 10^{10}$ years or $6.3 \times 10^{17}$ sec. The total mass of the Universe is estimated as about $10^{55}$ gs. Thus we have $6.3 \times 10^{72}$ gram seconds as an outer limit for the mass time product.

## TRANSCOMPUTABLE ALGORITHMS

We call an algorithm *transcomputable* if its computational cost exceeds all bounds that govern the physical implementation of algorithms.

It can be shown that the exhaustive search algorithm for chess is transcomputable. The same is true for many algorithms of artificial intelligence and operations research. In fact, any algorithm whose computational cost grows exponentially with a size parameter $n$ is transcomputational for all but the first few integers $n$.

This is a rather disturbing thought and many people have chosen to ignore it. (Analogously many people have for a long time chosen to ignore the fact that earthly resources of space, air, fossil energy and raw materials are limited.) One exception to this trend has been Ross Ashby, who more than any person in the world has emphasized the consequences of this limit in many of his writings between 1962 and his death in 1972 (cf. Ross Ashby, 1967, 1968, 1972).*

Another kind of limitation to computation is *thermodynamic*. R. Laundauer (1961) has pointed out that when in the course of computation information is discarded entropy is generated which must be dissipated as heat. How much information must be discarded when computations are carried out? Initially this question was not well understood. Recently Bennett (1973) has shown that any computation can be carried out essentially in a logically reversible way which implies that it can be done with little or no entropy generation (cf. also Landauer, 1976). In Landauer and Woo (1973) both thermodynamic and quantum limitations are discussed and an extensive bibliography is given. There are many open questions.

So far we have stated the fundamental limit for *signal flow* in a computer. Readers may wonder whether there would be an escape from the limit if we consider larger classes of computers – analogue computers, special-purpose circuitry (hardwave simulations), etc. This is not the case. In a forthcoming article I am trying to show that the limit applies to any physical implementation of any kind of algorithm. In essence, the fundamental limit is identical with the uncertainty principle of quantum mechanics.

In particular, the limit applies to nerve nets, and thus, ultimately, it imposes limits on human intelligence. This statement presupposes that the human brain is subject to the laws of physics and that it cannot solve logical and mathematical problems without implementing algorithms.

We may compare the phenomenon of transcomputability with limitations that apply to space travel. In order to reach a distant point in space the traveller has to perform a motion which requires time and energy. Since both are in limited supply the accessible portion of the Universe is limited. Analogously, in order to reach knowledge of mathematical theorems, optimal moves in a (mathematical) game, or in order to explore the trajectories of differential equations, etc., computations must be performed which require time and energy. Since both are in limited supply the accessible portion of the mathematical universe is limited.

The fundamental limit has epistemological consequences, for example the following: many systems (biological or physical) are composed of *parts*. The interactions between parts obey certain laws (e.g. gravitational interaction between mass points, electromagnetic, weak, strong interactions between elementary particles, chemical interactions between molecules, etc.). The *reductionist approach* tries to derive the total system behavior (the trajectory of the system in its state space) from the laws that govern the interactions of the component parts and from the initial state and inputs to the system. For complex systems knowledge of the systems

* *Note added in print*: Recently Knuth (1976) has written an article in *Science* that clearly explains the dual problems of complexity and "Ultimate Limitations" of computing (which he derives in a different way).

trajectories can be transcomputational with respect to sequential digital computation. In that case, if an analog of the system can be obtained, put in the proper initial state and if the state of the system can be observed, then the system trajectories are predictable, provided that the analog system runs faster than the original. If no such analog system is obtainable, then prediction becomes impossible, even if all the parts and the laws governing their interactions are known.

## REFERENCES

Ashby, R. (1967) "The place of the brain in the natural world", *Currents in Modern Biology* 1, 95-104.

Ashby, R. (1968) "Some consequences of Bremermann's limit for information processing systems", in *Cybernetic Problems in Bionics* (Bionics Symposium, 1966), edited by H. L. Oestreicher and D. R. Moore, Gordon & Breach, New York.

Ashby, R. (1973) Editorial, *Behavioral Science* 18, 2-6.

Bennett, C. H. (1973) "Logical reversibility of computation", *IBM J. Res. Devel.* 17, 525-32.

Bledsoe, W. W. (1961) "A basic limitation of the speed of digital computers", *IRE Trans. Electr. Comp.* EC-10, 530.

Bremermann, H. J. (1962) "Part I: Limitations on data processing arising from quantum theory", in "Optimization through evolution and recombination" in *Self-organizing Systems*, edited by M. C. Yovits, G. T. Jacobi and G. D. Goldstein, Spartan Books, Washington, D.C.

Bremermann, H. J. (1967) "Quantum noise and information", *Proc. Fifth Berkeley Sympos. Math. Stat. a. Prob.*, Univ. Calif. Press, Berkeley, Cal.

Bremermann, H. J. (1974) "Complexity of automata, brains and behavior", in *Physics and Mathematics of the Nervous System*, edited by M. Conrad, W. Güttinger and M. Dal Cin, *Biomathematics Lecture Notes*, Vol. 4, Springer Verlag, Heidelberg.

Bremermann, H. J. (1978) "Evolution and Optimization", planned for publication in the *Biomathematics Lecture Notes* series, Springer Verlag, Heidelberg.

Karp, R. M. (1972) "Reducibility among combinatorial problems", in Miller and Thatcher (1972).

Landauer, R. (1961) "Irreversibility and heat generation in the computing process", *IBM J. Res. Devel.* 5, 183-91.

Landauer, R. (1976) "Fundamental limitations in the computational process", *Bericht Bunsengesellschaft Physikal. Chem.* 80 (to appear).

Landauer, R. and Woo, J. W. F. (1973) "Cooperative phenomena in data processing", in *Synergetics*, edited by H. Haken, B. G. Teubner, Stuttgart.

Miller, R. E. and Thatcher, J. W. (Eds.) (1972) *Complexity of Computer Computations*, Plenum Press, New York.

Nilsson, N. J. (1971) *Problem-solving Methods in Artificial Intelligence*, McGraw-Hill, New York.

Strassen, V. (1969) "Gaussian elimination is not optimal", *Numerische Math.* 13, 354-6.

Thom, R. (1972) *Morphogénèse et stabilité structurelle*, Benjamin, Reading, Mass.

Thom, R. (1975) *Morphogenesis and structural stability*, English translation of Thom (1972) by D. Fowler, Benjamin, Reading, Mass.

Knuth, D. E. (1976) "Mathematics and Computer Science: Coping with Finiteness", *Science* 194, 1235-1242.