COMPUTING: YESTERDAY, TODAY, AND TOMORROW

by

JOSEPH F. TRAUB

ABSTRACT

I will talk briefly about some of the changes I've witnessed over almost half a century of computing. Then I'll discuss the unifying idea of scaling laws, with examples ranging from Moore's law to computational complexity. What are some of the implications of scaling laws for the future of computing?

I can remember vividly the moment I was hooked on computers. I can't tell you the date, which was probably in 1957, but I remember the moment. Let me give you the context.

I entered Columbia in 1954 intending to take a Ph.D. in physics. In 1955 I learned that IBM had a research lab at Columbia and I started taking courses in computing and applied mathematics. In 1957 I started working on my thesis which was the following: Experimentalists could make accurate measures of quantities such as the Lamb shift as well as relativistic corrections and the theoreticians wanted to test their theories to see if they agreed with experiments. This required solving the Schroedinger equations to obtain the wave function and using that to calculate the quantities of interest. I wanted to do this for the ground and excited states of helium. As you know, helium has two electrons, two protons, and two neutrons. It is the simplest element after hydrogen and yet the helium calculations were at the cutting edge of computing in the mid-to-late 50's.

What machines were available at IBM's research laboratory? The first machine that I used in 1955 was a plugboard machine. That is, you programmed the machine by placing wires into a plugboard. By the time I started my thesis the IBM 650 was available. This was a machine with a main memory consisting of a drum with 2000 words. That was a bit of a challenge. Of those 2000 words, 500 were used to turn the 650 into a 3 address machine and 500 for mathematical routines like the sine function. That left 500 words for my program and about 500 words for my data; I had to store three matrices, each of order 18. The secondary memory consisted of punch cards. Doing this "super-computer" calculation on a 2000 word drum memory machine was, in part, what made it a Ph.D. thesis.

Just a couple of more things before I tell you about the moment I got hooked on computing. The way that the wave function was calculated was to assume it had a certain form with adjustable parameters. The parameters were adjusted via a variational principle. One assumed values of parameters in the wave function, computed the corresponding energy level and adjusted the parameters to make the energy smaller.

I worked on parts of the program for some 6 months, planning, coding, running and debugging various modules. Now I was ready for my first complete run. My output from the 650 was punched into cards which I carried over to the printer which started to print the energies which were decreasing as the wave functione improved. The first number that I saw was the calculated ground state energy of helium which agreed with experiment to 4 decimal places. A chill ran down my spine. I was doubly amazed. I was amazed you could model nature with an equation, solve the equation and obtain results that agreed with physical experiments, and I was amazed that I could program for six months, do extensive calculations, and out would come the experimentally measured energy, good to four places. Then I watched the printout as the energy dropped, dropped, dropped, dropped which meant I was converging. That was the moment I was hooked.

After finishing my degree, I went to Bell Labs. That was a golden time at the Labs. If you were hired by the Research Division, you could work on anything you chose. The only criterion was that your work should have impact. One day in late 1959, one of my colleagues came into my office and asked how to calculate the solution of a certain problem. I could see a variety of ways to solve the problem. What was the best, that is, the optimal method? The problem was continuous and had to be solved numerically. That meant that you could only solve it approximately, to within an error ε. By optimal I mean the method which used the minimal computational resources, for example time, to compute an ε-approximation. To my surprise, there was no existing theory. I got fascinated by this question and in 1964 published a monograph on optimal iteration theory. It wasn't until 1968 that I heard Al Borodin give a talk on what he called the sexy phrase "computational complexity" and realized that's what I was working on. Hartmanis and Stearns had coined the phrase in 1965. It was independently introduced by the Soviet polymath, Kolmogorov.

For over 40 years my colleagues and I have worked on the computational complexity of continuous problems, such as high-dimensional integration, continuous optimization, partial and integral equations, and approximation. This field is today called *information-based complexity*, a name suggested in the 80's by Richard Karp.

Back to the 60's for a moment. People would ask me what do you do and if I mentioned computers they didn't know what to make of it. They thought it had something to do with numbers and they might mention that their uncle was a certified public accountant. It seems hard to believe today, but well into the 70's there was rarely a mention of computers in the popular media. If, say, the New York Times, or Newsweek, mentioned computers, that was an occasion. This changed especially after personal computers became widespread in the early 80's and became transformed with the world wide web in the 90's.

The first computer science departments were created in the mid 60's. In 1971 I was asked to head the Computer Science Department at Carnegie-Mellon University. For those of you familiar with the School of Computing at Carnegie-Mellon today, the size of the department in 1971 might surprise you. There were about 8 of us. Of course they were an extraordinary group including Herb Simon, who passed away very recently. Herb was one of the founding fathers of artificial intelligence and was to be named a Nobel laureate in economics. Then in 1979 I was asked to start the department at Columbia.

I was lucky to have started in computation in the mid 50's and to have had the opportunity to build theories, departments and journals. I like to say that in my scientific work I can just walk along and pick up diamonds; I never have to strip mine. I am a theoretical computational scientist and I'll discuss why this is specifically true in my field. But it has been true and it is true today all over computer science, telecommunications and electrical engineering. The general reason is the incredible rate of change.

I'll zoom in on my own field of computational complexity. Every time there is a major change in architecture, the rules of the game change. The rules are what we call the model of computation. It is as if you were a chess player and every decade or so there is a radical change in, say, the type of pieces, their legal moves, the type and shape of the board and perhaps even the dimension of the board changing from 2 to d. Every time the rules change you've got to master what is essentially a new game. In the first few decades, computers were sequential. Then we got parallel computers, asynchronous computers, and heterogeneous workstation farms. More recently, DNA computation, nanotube chips, and quantum computation are being considered . I will return to quantum computation later. Furthermore, the computational resources of interest vary; they include time, memory, area on the chip, and communication costs. So there are always new challenges for the theoretician and I don't see that ending.

I turn now to scaling laws as a unifying principle, both temporally (past, present and future) and across the various areas of computing and telecommunications.

As an example of a scaling law consider Moore's law which states that the number of transistors on a chip doubles every 18 to 24 months. This implies a doubling of computer power or a halving of cost over the same time. Moore's law has held for some 40 years. If we take the more conservative number of doubling every two years, we have had 20 doublings in 40 years. Thus means based on chip density, computers are 2^{20} or about 10^6 times more powerful than around 1960.

Moore's law is an example of an exponential scaling law. An exponential scaling law is of the form a^x where a is a constant and x is a variable. For

Moore's law a is the square root of 2 and x is the number of years it can be applied. There are quantities that are doubling much faster than every 18 months. For example, George Gilder claims that bandwidth is doubling every 6 months. He points out that the doubling is still in its early stages; it may take some years before we really know the doubling rate. This bandwidth doubling rate is sometimes called Gilder's law. It is also exponential, but of the form a^x with a now equal to 4. It is a much steeper exponential than Moore's law. It's important to understand what effect the much faster increase in communication power than in computational power will have on networks and how we do our computing.

I will turn next to scaling laws in computational complexity. A central question here is how must the difficulty of a problem scale with its size. I want to contrast polynomial scaling with exponential scaling. If a problem scales polynomially with size and the degree of the polynomial is say 2 or 3, then we can solve the very large problems that occur in practice. But if a problem scales exponentially, then it's impossible to solve large problems and such problems are said to be computationally intractable.

Are combinatorial problems, such as the travelling salesman problem, polynomially or exponentially hard? We don't know. Technically the question is whether P = NP and it is perhaps the most important open question in theoretical computer science. What we know, due to work initiated by Steven Cook and Richard Karp in 1971 and 1972, is that there are hundreds of combinatorial problems that scale either polynomially or exponentially. That is they are all easy or all intractable, but we don't know which. The belief of the experts is that they scale exponentially.

But we have to solve large combinatorial problems. Since these are complexity results, intractability cannot be broken by inventing a clever new algorithm. Two of the ways intractability may be broken are the following:

- Settle for an approximate solution; this sometimes breaks intractability of combinatorial problems.
- Replace the worst case assurance by a stochastic one. For example, analyze the average complexity or use randomization.

One area where exponential scaling has been proven, not just conjectured is for continuous problems, which we study in information-based complexity. For these problems the size is the number of variables, that is, the dimension. In the 50's Richard Bellman noticed that the difficulty of certain problems grew rapidly more difficult with their dimension and called this the "curse of dimensionality". That was before we started to study complexity, so it was just an observation. Now we know that if you want a worst case assurance of error at most ε, then the complexity of most continuous problems is exponential in dimension. The base of the exponential is the reciprocal of ε. For example, if you want 4 place accuracy, the base is 10^4 and if there are *d* variables then the complexity scales as 10^{4d}.

We want to solve problems with very large *d*. Path integrals where *d* is infinity occur in physics, chemistry and finance. Very high dimensional integrals with $d = 360$ must be calculated in mathematical finance. Can we break exponential scaling , that is, can we vanquish the "curse of dimensionality"?

Sometimes we can vanquish the curse and I would like to tell you how this is done for certain problems in finance on which my research colleagues and I have been working. The problem is to value financial derivatives. A financial derivative is an instrument whose value is derived from the value of an underlying asset. Alan Greenspan estimates that the notional value of all financial derivatives is some 90 trillion dollars. That is about 10 times our gross national product. So there is some interest in how to value derivatives.

An example of a financial derivative is a CMO, that is, a collateralized mortgage application. Think of it as a basket of 30 year mortgages. If I want to estimate future cash flows, I have to compute integrals in 360 dimensions -- 360 being the number of months in 30 years. If I want to be assured of an error at most ε, the complexity of this problem grows as $(1/\varepsilon)^{360}$. If we want a two place answer, the complexity is of order 10^{720}.

Of course, we cannot use that much time, so for several decades the financial community has been using Monte Carlo methods. Then this problem can be solved at cost which is the reciprocal of ε squared. The curse of dimensionality has been vanquished but it is now only the *expected* error that is less than ε. There is no such thing as a free lunch. Here, one has achieved less complexity for more uncertainty. So using Monte Carlo, the problem scales as $1/\varepsilon^2$, where ε is the expected error.

Can we do better? In the early 90's I asked a student, Spassimir Paskov, to compare quasi-Monte Carlo with Monte Carlo for a hard CMO provided by Goldman Sachs. By hard, I mean that about a million floating point operations are needed to sample the integrand at one point. So it's important to minimize the number of samples. A quasi-Monte Carlo method uses deterministic sampling with a rather small number of sample points spread as uniformly as possible. There is a very well developed theory of quasi-Monte Carlo methods which predicts that the complexity should scale as

$(1/\varepsilon)\left(\log\frac{1}{\varepsilon}\right)^d$. That's better than Monte Carlo as ε goes to zero with d fixed. But in finance ε is rather large, say one part in a hundred, and d is huge. Then this scales very badly . It scales so badly with d that in the early 90's experts believed that quasi-Monte Carlo should not be used if d was larger than, say, 12. To everyone's amazement Paskov's experiments showed that the problem scales as $1/\varepsilon$; the exponential factor in d did not appear. Anargyros Papageorgiou and others obtained similar results for a variety of financial derivatives and also for value at risk calculations. Quasi-Monte Carlo is now being used extensively in the financial community.

There is no existing theory that predicts $1/\varepsilon$ scaling for certain financial calculations, so we need to create such a theory. Note the similarity to physics. There is a well developed theory which doesn't predict the experimental evidence. That means the theory has to be refined. I believe the following to be true. Formalize what is special about finance. Then prove the complexity scales as $1/\varepsilon$ with a worst case assurance. If true, this would be a double win over Monte Carlo since Monte Carlo scales as $1/\varepsilon^2$ with only a stochastic assurance. I call this the Holy Grail theorem of mathematical finance because we have looked for it for a very long time, believe it is true, but haven't yet found it.

I will now briefly discuss the future, starting with Moore's law. It is generally believed that Moore's law will end in one or two decades for a number of reasons:

- There are physical limits to the smallness of what we can put on a chip.
- There are financial limits because the cost of building a chip manufacturing facility doubles with each chip generation.

What might come after silicon computers? DNA computation will probably be special purpose. Nanotube chips are being studied. There is much interest in quantum computing and I'll confine myself to that. Can superposition of states on a quantum computer help us solve problems which are intractable on a classical computer? The quantum algorithm which has stirred the most interest is Peter Shor's factoring algorithm. To understand the significance of this algorithm, I have to remind you of a few basics of cryptography.

A widely used cryptosystem, the RSA system, depends on the belief that factoring large integers is intractable. That is, the essence of public key cryptography is scaling where the size of the problem is the number of bits in the number to be factored. Shor gives a polynomial time algorithm for factoring on a quantum computer. Almost all the algorithm research for quantum computing has been for discrete problems such as integer factorization. But Richard Feynman proposed that quantum systems could be simulated by quantum computers. Quantum mechanics is governed by continuous models such as path integration and Schroedinger's equation. Many of the problems in science, engineering and economics require the solution of continuous models. There is a joint Columbia/MIT project (Traub and Henryk Wozniakowski, Columbia, and Seth Lloyd, MIT) on algorithms and complexity for solving continuous problems on quantum computers. One of our goals is to find important problems which are intractable on classical computers and tractable on quantum computers.

As you know, although quantum computers with few qubits have been built, it is not clear whether quantum computers will prove to be an answer to the end of Moore's law for silicon-based classical computers. A difficulty is whether the superposition of states can be maintained or, to use the interpretation of the Copenhagen school, whether we can avoid the collapse of the wave function by measurement or interaction of the computer with its environment. The thrust of our research is if quantum computers can be built, for what problems can we achieve big wins?

To summarize: it seems to me that scaling laws are a central theme for computing past, present and future. I'll end by posing four questions about scaling laws in the future:

1. Can we build quantum computers and use them to solve problems which are intractable on classical machines?
2. Are there other means by which we can continue the enormous strides in computation without the benefit of Moore's law for silicon chips?
3. How should we plan our computing and networking in light of the fact that the doubling rate of bandwidth is much shorter than that of chip density?
4. Settle the conjecture $P \neq NP$.