CUCS-51-83

# COPING WITH COMPLEXITY

J.F. Traub
Edwin Howard Armstrong Professor
of Computer Science
and Professor of Mathematics
Columbia University

April 1983

Talk presented at a session entitled "New Directions for the Computer Sciences" at a Symposium on "Computer Culture: The Scientific, Intellectual and Social Impact of the Computer" sponsored by The New York Academy of Sciences, April 5, 1983. To appear in the Proceedings.

A common complaint about modern life is its complexity. I'm sure the word complex brings varied associations to your minds; I'll give you a few of mine:

- Designing a jet aircraft
- Managing a large organization such as a city, a corporation, a university, or even a country
- Sending a man to the moon
- Deciding where to drill for oil
- Investing in the stock or commodity markets
- Designing and maintaining a national communication system
- Trying to understand the mind of man
- Controlling the national money supply to achieve certain goals
- Designing a safe nuclear power plant

In a recent talk Bob Lucky stated that the management of complexity is a key problem. As examples of complexity he cited million word programs written by a hundred programmers and the design of chips with tens of thousands of transistors. He pointed out that we will, one day, have a billion transistors on a chip.

What are some of the features that make a problem or system complex?

Size

Arranging a schedule for an airline that flies four planes between New York and Washington can be done on the back of an envelope. On the other hand, preparing the schedule for an airline with hundreds of planes and scores of cities is a complex task. Size often leads to complexity.

Interconnections

If a system can be decomposed into subsystems, then those smaller subsystems can be independently analyzed. If, on the other hand, a system or problem has many interconnections, we must tackle it whole. If we jiggle it a bit here there may be major unforseen consequences there.

Limited and Inexact Information

For most systems and problems the available information is limited and inexact. I'll use weather prediction as an example. Thousands of weather stations make measurements at various attitudes. In addition, planes, ships, and satellites also make measurements.

Inevitably these measurements, as all measurements, will be contaminated with errors. Furthermore, since there are only a certain number of measurements the information is limited. Hence we don't know the world-wide weather at any moment.

Why is that significant? If we don't exactly know the world-wide weather at any time, this severely limits the extent to which we can do long-term prediction.

In general, making predictions or decisions under conditions of limited and inexact information adds enormously to complexity.

When people say a situation is complex they often mean they must proceed with insufficient information. We return to this theme later.

As you know, computers are widely used by corporations, governments at all levels, the military, and many professions such as medicine, law, journalism, engineering, sociology; the list goes on and on. We use computers to control complexity because of certain characteristics. They have great speed, high reliability, and large memories. If the plans for the Fifth Generation are successful, we'll have machines which draw inferences as fast as today's computers perform arithmetic. A major benefit will be to help us deal with complexity.

Although computers are used to control complexity, they are also its cause. For example, the design of a large computer or of a micro-processor on a chip are among the most complex tasks performed by man. Indeed, they are so complex that we use computers to design computers.

So far, I've discussed complexity in vague everyday language. If we're going to get a scientific handle on complexity, we have to be more precise. There is a major area within computer science devoted to the study of complexity. I want to tell you about a number of the most challenging problems and opportunities facing the field.

I'll start by describing a famous problem. You are given a set of cities and the distances between them. Determine the order in which the cities should be visited so that each city is visited exactly once, the tour ends in the starting city, and the distance traveled is as small as possible.

For obvious reasons this is called the traveling salesman problem and is an abstraction of many scheduling and layout problems which arise in practice.

Say the number of cities is six: New York, San Francisco, St. Louis, Miami, Fargo, and Santa Fe. Then the problem could be solved at sight or with a little pencil and paper work.

What if the number of cities is 50 or 100 rather than six. Then how long will it take to find the shortest tour of the cities?

Many people have puzzled over this for decades. No matter what procedures they devised for finding the shortest tour, and no matter how clever they were, they discovered that the time required to carry out the procedures took hundreds of centuries on the fastest available computers. Since computers were only invented some forty years ago, no one has computed for hundreds of centuries. What I mean is that analysis reveals that if the procedures were implemented, that is how long they would take.

One way around that seems clear - build faster computers. But that does not help significantly. The number of operations required by these procedures is so large that even the coming generation of supercomputers, computers capable of performing billions of operations per second, wouldn't make a big difference.

I've been using the word procedure to denote a method for solving a problem. From now on, I will use the word used in computer science instead of procedure - the word is algorithm.

If people keep devising algorithms to solve the traveling salesman problem and all these algorithms take centuries, a natural question arises - is there any algorithm for solving the problem faster or is the problem inherently intractable? You can see what an important question that is. If we know the problem is intractable, then we can stop looking for algorithms to solve it faster.

I'm now going to introduce the basic idea of intrinsic problem difficulty. A measure of intrinsic difficulty is called the computational complexity or for brevity, the complexity.

Let me stress a point which is often misunderstood. The complexity depends on the problem, not on any particular algorithm for solving it.

If the complexity is large, in a sense I could make technically precise, we say the problem is hard. Is the travelling salesman problem hard? On April 5, 1983 no one on earth knows.

Computer scientists have however made a remarkable discovery. There exist numerous problems, some of them extremely important, which are all equivalent from the complexity viewpoint. They are all hard or they are all easy. The common belief among experts is that they are all hard but no one knows.

These problems as said to be NP-complete. Pioneers in this work are Professor Steven Cook at the University of Toronto and Professor Richard Karp at the University of California, Berkley.

Determining whether NP-complete problems are hard or easy is one of the great questions to be decided in the future. If you want to read more, an excellent account is given by Garey and Johnson.[1]

In the traveling salesman problem we concerned ourselves with the time taken to find the shortest tour. But time is only one measure of cost. A second is the amount of memory required and a third is area. Since it is harder to produce a large VSLI chip than a small one, we are interested in the area required to solve a problem. Interesting tradeoffs have been discovered between the time and the area required to solve a problem.[2]

You may think, from the examples I've provided that complexity is bad. Complexity may indeed be desirable. For example, modern techniques for encrypting communications are based on the belief that problems such as factoring large integers have high complexity.

I've discussed at some length the traveling salesman problem and the great open question of the complexity of NP-complete problems. I now turn to an area where we are still, relatively, near the beginning but for which I expect major progress over the next decade.

Interest in this area is motivated by the fact that for most problems, information is limited and inexact. We therefore cannot find the exact answer, that is, we must live with uncertainty.

At this point you might appreciate an example. The examples are everywhere. In almost every discipline or subject - physical science, biological science, engineering, control, prediction, decision theory, tomography, economics, design of experiments, seismology,.sociology, medicine - one deals with limited and contaminated information and therefore one must be content with uncertainty in the answer.

Typically the larger the system the more likely that one doesn't have exact information, especially if the system is changing with time. Therefore one has to settle for knowing a certain outcome is probable but one cannot be sure.

The areas I've mentioned above look very different from each other. Are there laws regarding how well they can be solved which apply to all of them?

We believe the answer to be yes and a number of my colleagues and I at Columbia, as well as co-workers around the world, are trying to discover these laws and apply them to the solution of problems in many disciplines. For a moment I want to step back and take a broad view of science. Science has been called the study of invariants, seeking laws which are valid in varied domains. An archetypal example is provided by Newtonian mechanics. Before Newton, any "reasonable" person believed that apples and planets were very different objects obeying different laws. For some characteristics this is true; as you can verify by biting into an apple and a planet. But if you consider the correct quantities, which are force, mass, and acceleration, then there are laws which apply equally to apples and planets as well as to carriages, waterwheels, and buildings.

For most problems we have only limited or inexact information and consequently such problems can only be solved with uncertainty. Such problems are as different from each other as apples and planets. For such seemingly unrelated problems we believe that we have identified the basic quantities and a fundamental invariant, which we call the radius of information.

The radius of information measures the intrinsic uncertainty in the solution of a problem due to the available information. It not only tells us how well a problem can be solved, but also leads to the best algorithm and the best information. For emphasis we state the

INFORMATION PRINCIPLE There exists a quantity called the radius of information which measures the intrinsic uncertainty of solving a problem if certain information is known.

Because of the emphasis on information we call this the information-centered approach. Currently the algorithm-based approach is in widespread use. In this approach, an algorithm is created and analyzed. Then another algorithm is proposed and analyzed, and so on. One never knows how much better one might be.

We contrast this with the information-centered approach. In this approach, one merely states what problem one wants solved and indicates the type of information available. The theory then tells one the best information, the best algorithm, and the complexity.

I hope I've convinced you that if information is limited or inexact you can't solve a problem exactly. Sometimes you have complete and exact information and choose not to solve exactly. The use of artificial intelligence techniques in chess provides us with an example.

Assume you are white and wish to find a winning strategy (if it exists) against all possible moves by black. The information is complete and exact. Therefore there exists an algorithm with no uncertainty. Indeed, we have the following gedanken - algorithm. Consider all possible first moves for white and for each of these consider all possible moves for black. Continue this process for the second, third, etc. moves. If there exist one or more winning strategies against all moves by black, choose one of these strategies.

Such a "brute-force" approach would be far too expensive and we use a heuristic approach instead. We live with the uncertainty of the heuristics to decrease complexity.

Chess is an example of a common phenomenon. We choose to live with uncertainty to decrease complexity.

I've mentioned that the information-centered approach gives you the best algorithm and the computational complexity. What else can it provide? It can, for example, tell you whether a problem can be decomposed for solution on a parallel computer, or more generally, on a distributed network.

I'll use the game of twenty questions, with which you are all familiar, for illustration. I'll describe a simplified version of twenty questions. You tell me you're thinking of an integer between 1 and 16. I ask questions concerning your integer to which you respond yes or no. If you're thinking of 15 the sequence of questions and answers might be as follows:

Is the number between 1 and 8? No.

Is the number between 9 and 12? No.

Is the number between 13 and 14? No.

Is the number 15? Yes.

I obtained your number with 4 questions and it's well known that one of 16 possibilities can always be identified with 4 questions with yes or no answers.

The questions I asked were adaptive because I waited for your answer to each question and then adapted my next question to your answer. This is an example of adaptive information.

What if I asked all my question simultaneously? That means I have no opportunity to adapt my questions to your answers. This is an example of nonadaptive information

We saw that four adaptive questions are sufficient. This is the best adaptive information in that fewer than four questions are not enough. How many nonadaptive questions are required?

It turns out that four nonadaptive questions are also enough. I'll leave it for you to see what nonadaptive questions should be asked.

Thus for this problem nonadaptive information is just as powerful as adaptive information.

Why is this result of interest? During the 1980s and 1990s parallel computation, that is, computation on a machine with many processors capable of simultaneous operation, will be increasingly important.

If the information is nonadaptive, we have a natural decomposition for parallel computation. Various components of the information can be simultaneously and independently obtained.

We saw in the twenty questions example that nonadaptive information can be as good as the best adaptive information. But that's just a toy example. What about real problems?

Almost everyone believes that "usually" adaption helps. However, almost everyone is mistaken. Although there are problems where adaption helps, the information-based approach enables us to show there are many important problems for which adaption cannot help.

Although much has been accomplished in the information-centered approach, a vast amount remains to be done before we can say we have established a general theory for dealing optimally with limited and inexact information. Of special interest is the application of the information-centered approach to areas as diverse as distributed computation, control, seismology, design of experiments, and signal processing. We can provide interesting research for many people over many years.

As introduction to the information-centered theory may be found in a recent expository paper.[3] The general theory is presented in two research monographs.[4,5]

In closing I want to recapitulate. Coping with complexity is a central issue for us individually and collectively. I started this talk with examples of complex problems ranging from managing a large organization to putting a man on the moon. I then indicated some of the results of the new field call computational complexity.

I am hopeful that computational complexity will eventually provide us a general scientific framework for coping with complexity in the everyday sense of the word.

# BIBLIOGRAPHY

1. GAREY, M.R. & D.S. JOHNSON. 1979. Computers And Intractability W.H. Freeman and Company. San Francisco, C.A.

2. THOMPSON, C.D. 1980. A complexity theory for VLSI. Computer Science Department Report, Carnegie-Mellon University.

3. TRAUB, J.F. & H. WOŹNIAKOWSKI. 1983. Information and computation. To appear in Advances in Computers. Vol. 23. Edited by M. Yovits. Academic Press. New York, N.Y.

4. TRAUB, J.F. & H. WOŹNIAKOWSKI. 1980. A General Theory Of Optimal Algorithms. Academic Press. New York, N.Y.

5. TRAUB, J.F., G.W. WASILKOWSKI & H. WOŹNIAKOWSKI. 1983. Information, Uncertainty, Complexity. Addison-Wesley. Reading, M.A.