Random Thoughts and Prime Numbers

Jin-Yi Cai on the nature of theoretical computer science research.

Jin-Yi Cai, an ACM Fellow, received his Ph.D. in 1986 from Cornell University. He has held faculty positions at Yale University (1986-1989), Princeton University (1989-1993), and SUNY Buffalo (1993-2000), rising from Assistant Professor, Associate Professor to Professor of Computer Science. He is currently a Professor of Computer Science at the University of Wisconsin -- Madison. He received a Presidential Young Investigator Award in 1990, an Alfred P. Sloan Fellowship in 1994, and a John Simon Guggenheim Fellowship in 1998. He also received the Humboldt Research Award for Senior U.S. Scientists.

UBIQUITY: How is theory viewed today within the broader area of computer science? Is there any controversy about the role it plays?

CAI: I wouldn't say it's controversial, because it's really one of the more well-established areas; in fact, I hear some of my systems colleagues jokingly complain that an unseemly high percentage of Turing Award winners are in theory!

UBIQUITY: And your answer?

CAI: It is true that among US universities many Turing Award winners are in theory, but if you consider those who are in Europe or industry, then theory is not really the majority. I admit that it's a very significant percentage, but I would argue that every single one of them fully deserved it. They have all made undisputed, tremendous foundational contributions.

UBIQUITY: And yet a casual observer might be forgiven for wondering if there's something of a defensive position within the theory community. From some of the discussions and position papers I read it almost appears that theory is somehow under attack.

CAI: Well, let me just say that things have changed somewhat since the mid-nineties. At that time, I think it was fair to say there was a push and a debate about the service aspect of theory to other areas of CS. A few of us, though, felt there should be much more emphasis on the internal, intrinsic worth of the subject. Many efforts that eventually
become great applications for use outside of theory originated from discussions and investigations that were purely curiosity-driven. They started off driven by elegance and internal logic and beauty, by the scientific investigation into the nature of computation. Some of those ideas have had tremendous impact down the road. But of course different people feel differently in the theory community. Some felt that there should be more of an emphasis on practical and useful aspects of the subject. Some of this of course was driven by pressure from funding agencies. However, some of us felt that there was a need to expound on the intrinsic value of the subject. Just like physics, computation is an important and intellectually deep subject of study.

**UBIQUITY:** Have there been any recent major developments in theory?

**CAI:** Yes, actually, there's a recent big development in the grand old problem of deciding whether or not a prime number can be decided in polynomial time.

**UBIQUITY:** Explain.

**CAI:** Prime numbers have fascinated people from the time of antiquity. Primality is just telling if a number is a prime or not. This is not the same as factorization. Of course, to be able to factor it implies the ability to tell if it is a prime. But, not vice versa. At least we don't believe so. For people who care more about utility in everyday life than subliminal beauty, we can mention that the ubiquitous RSA scheme for encryption/decryption is based on properties of prime numbers. RSA is named after Rivest, Shamir and Adleman, three wonderful theoreticians. Their RSA scheme is used in much of today's Web traffic as a security mechanism.

**UBIQUITY:** How does the story continue?

**CAI:** About 25 years ago, Gary Miller proved that there is a polynomial time test, if we assume the Extended Riemann Hypothesis, which is an extended version of an old conjecture by Bernhard Riemann given 150 years ago. The Riemann Hypothesis is probably the most celebrated mathematical conjecture. Miller basically argued that a certain procedure looking for a witness for compositeness will be able to find the witness quickly, if the Extended Riemann Hypothesis is true, which implied a certain regularity in a number theoretic distribution. This was quickly followed up by a great observation by Michael Rabin, who realized that he could substitute randomness for the unproven hypothesis. He showed: if I randomly take a potential witness, then I have a good probability of being able to find one.

**UBIQUITY:** What is the upshot?

**CAI:** Rabin's great result -- and by coincidence, at the same time Solovay and Strassen found another probabilistic method to test primality -- was very much responsible for the great amount of initial
enthusiasm of this whole idea of probabilistic computation. Of course, the idea of probabilistic computation did not originate with them. Monte-Carlo methods had been used before. But there is a revolutionary new perspective in a quantitative sense. Rabin and Solovay-Strassen now gave a rigorously defined notion of what it meant to have a provably fast probabilistic algorithm -- that for every number, every input, the procedure is supposed to succeed, with very high probability in polynomial time.

UBIQUITY: Then it's a probability issue?

CAI: To an extent, but the notion is different from, say, my picking a random number and finding that 50 percent of the time I can do well. For example, for primes this "method" is totally useless. If I say "Half the time I can do well" -- well, half the times it is even, so of course you can do well. But the claim that for every number I can do well, meaning with polynomial time, I can tell you if it is a prime or not with high probability, this is different! There followed a period of a great deal of excitement, finding all sorts of randomized algorithms for all sorts of problems. This was followed further by a great deal of work on derandomization, where one first designs a randomized algorithm, then substitutes the randomness by deterministic means. Nowadays, there are all sorts of algorithms. Graph algorithms, for example, instead of doing a depth-first-search or some such natural steps, mimic a certain randomized process and argue that you will have succeeded when all is said and done.

UBIQUITY: Is your argument essentially that it's okay for theory to appear useless, because mere appearance belies the fact that it actually has important and useful benefits?

CAI: Yes. I would argue that in my experience it's been proven again and again that curiosity-driven research has a very good chance of paying high dividends down the road. This is true especially but not exclusively for theoretical research.

UBIQUITY: So there is an expectation that sooner or later there will be a benefit from all theoretical research?

CAI: Not exactly an "expectation," and not "all," because some of it may not. That's the nature and beauty of research. If we knew what would happen, we wouldn't do research anymore. The whole point is to discover things. My argument is that there is a good chance that curiosity-driven research will pay off, and it tends to pay big when it does. The most innovative ideas seem to come from this line of inquiry, though of course I can't predict which will be the winning bet. I'm further arguing that humanity must do research for its own dignity. We should do it for our species, for our intellectual development, and for our self-respect as scientists. Computation is governed by quantifiable laws. We must seek the truth. Look at physics, for example; when physicists look at various things, it's not always with some immediate application in mind. So much in today's world is derived out of research totally dedicated to the search of truth, beauty,
regularity, harmony, etc. It seems that historical evidence supports the belief that blue sky research almost inevitably brings about quite practical applications. I take that as an article of faith, and I think to argue otherwise is shortsighted.

**UBIQUITY:** Let's return to the primality problem itself.

**CAI:** So here is the big result that Agrawal and two of his students recently obtained. They've shown that primality testing is in polynomial time. This closed the loop, if you will. It is fascinating, starting off from a deterministic algorithm, through randomness, back to deterministic. One can perhaps take the philosophical view that if you didn't have the Rabin-type randomized algorithm you could still have shown primality is in P. The whole edifice of randomization is not needed. Minimally, in the legalistic sense, perhaps, but in fact historically it's false. They came to the deterministic algorithm by discovering other randomized algorithms. Finally they can show that it did not require any randomization, using some number theoretic estimates that were unconditionally proven.

**UBIQUITY:** Are there other instances of this excursion from deterministic thorough randomness back to deterministic? Have you found any such instances in your own work?

**CAI:** Yes. There are many wonderful results of this type. For instance, in my own joint work with D. Sivakumar, who was a graduate student at SUNY Buffalo where I was a faculty there at the time, and who is now at IBM Research. In 1978 Juris Hartmanis conjectured that no sparse sets can be complete for P under Logspace reduction unless P is equal to Logspace. Here a set of strings is called sparse if it can have at most a polynomial number of members among exponentially many at any length n. Sparse sets are considered to be sets with low information content, and they are also intimately connected with non-uniform complexity. For example, the famous Karp-Lipton Theorem says that if NP is reducible to a sparse set by P-time Turing reductions (also called Cook reductions), then the Polynomial-time Hierarchy collapses to its second level. The assumption that NP is reducible to a sparse set by Cook reductions is also equivalent to saying that every set in NP is solvable by a polynomial-sized circuit family. The Hartmanis Conjecture is concerned with the intrinsic power of computation in polynomial time versus logarithmic space, which remained open until 1995. Ogihara first made a breakthrough, showing that if P were to be reducible to sparse sets, then P is contained in polylogarithmic space.

Sivakumar and I realized that the same assumption would yield the unlikely consequence that P is contained in randomized NC2. NC2 is a parallel complexity class named after Nick Pippenger. After that we realized that the randomization could be substituted by derandomization techniques, using finite fields. Finally the derandomization technique led us to devise a totally deterministic procedure that says that the assumption implied P is contained in deterministic NC1, which is known to be contained in Logspace, thus
completely resolving the Hartmanis Conjecture of 1978. It is instructive to note that many concepts crucial in this proof were only studied in the years after, and not particularly in anticipation of, the proof of the Hartmanis Conjecture. Again, one can take the minimalist point of view and argue that randomization was not needed. But I am sure that we would never have arrived at the final proof had it not been for the randomization and derandomization process.

**UBIQUITY:** Tell us a little about your intellectual history.

**CAI:** I work in computational complexity theory, which is the study of the inherent difficulty and existence of efficient algorithms for various problems. It's in some sense more concerned with the non-existence of an algorithm, and you try to find evidence of why certain things are hard. There are theorems, called hierarchy theorems, which say that with more computational resource such as time and space, it is provably the case that one can solve more problems. So the intrinsic difficulties of computational problems are not mirages. They do not refer to our mere inability to find a good algorithm, or that even some existing algorithm provably does not work efficiently. It refers to the non-existence of any efficient algorithm for the problem.

I came to the subject of computational complexity theory, perhaps unknowingly, in a strange tortuous sense. Let me explain. I was a school kid in China. In high school, you start to study geometry, right? You consider a circumscribing circle and drop a perpendicular line from this point against that line and so on. Some magic moves later, you have a theorem. Some of the theorems are quite challenging. You have to be very clever to see what's the right thing to do. I was pretty good at it, but there were quite a few occasions where the problems stumped me. Furthermore, there were times where I got a proof but then afterwards the teacher would show us what is called the Book Proof. Then you find your proof is kind of ugly. That it's roundabout and maybe wasteful.

**UBIQUITY:** And that bothered you.

**CAI:** Yes, terribly; I don't want to be falsely modest here because I was pretty good at this. But I was never truly satisfied. Every time you are tested for your insight of one magic move. Sometimes you can produce the magic move but many times you can't. This is the experience of everyone who does mathematical proofs. As a kid, I didn't want to concede this. It was especially painful when I could not produce any proof at all. So I started to think and then hit upon the idea of a more systematic approach to it. I found that I could almost always take your input data, your problem specification, as points on a plane, and I could write them down, and then the conditions become equations; and I realized that for almost every problem I could just plug in and churn away. I just wrote down the equations and manipulated the algebra. It lacked a certain level of beauty and elegance but nonetheless, it was a surefire way of winning. I was never going to draw a blank! So I was hooked. Of course, my teacher was very distraught because this brute force process ruined some of the
delicately designed problems.

UBIQUITY: Then what happened?

CAI: Ten or 15 years later I learned that what I had observed were basically consequences of certain theorems of the logician Alfred Tarski -- theorems of decidabilities for Euclidean geometry and so on. As a high school kid, of course I was not sophisticated enough to realize that there is a general theorem of this nature, I was only observing in a very vague and informal sense that something was happening. There is this step-by-step procedure that will get the solution. You don't need cleverness. Even though cleverness may get you a very short proof. But the mechanical process is guaranteed to work. By Tarski's theorem it always works, provably so, but I didn't know that. This type of question is basically the genesis of the field of computational complexity. The question of NP versus P is whether or not anything that has a short guessing that leads you to the correct solution can be mechanically obtained in short order. The best we know is if it's in NP, then it can be done in exponential time, which is when you check out every possibility.

UBIQUITY: When and where did you first become a theoretical computer scientist?

CAI: That was in graduate school. I first went to Temple University from China, because Temple was the only place that would give me financial support. I learned a lot from a wonderful number theorist D. J. Newman, and then I studied for my Ph.D. at Cornell, where my advisor, Juris Hartmanis, was one of the founders of computational complexity theory. After that I was an assistant professor at Yale and then went to Princeton and then to SUNY Buffalo. Now I am at Wisconsin-Madison.

UBIQUITY: Have you interacted much professionally with non-theoreticians in these various posts? Do you frequently talk with people who are more applications-oriented?

CAI: I am very open to talking to other people. In my career, I have written papers with people who are in the more applied areas. Various people have different styles of research. I know wonderful people who work with everyone, who solve problems left and right -- and I know other wonderful people who concentrate in that one thing in which they excel.

UBIQUITY: You said you first started thinking about this while you were a student in China.

CAI: In retrospect, it must have been the place where I started to be bewitched by this thing. But I didn't know it at the time. It was just a way to cover my own shortcomings against this idealized perfection that you somehow could instantaneously see the proof any time.

UBIQUITY: Looking back, do you find that the intellectual style in
China is the same or different than it is here?

CAI: For my university education, I was an undergraduate at Fudan University in the mathematics department, class of 77. The emphasis was very classical: algebra, analysis, topology, all the basic things. I don't have the ability to tell if there was any difference. Furthermore, things have changed a lot today in China compared to 25 years ago. I would not say that a computer science department in China is more theoretical or more "academic," than practical. In fact, they're quite commercial these days. The whole society there is undergoing so much upheaval and everybody wants to be rich in a hurry.

UBIQUITY: And they see computing as the right tool to accomplish that?

CAI: Perhaps. But for me personally, it is the intellectual challenge that is the attraction. Let me cite a quotation by Dijkstra, who recently passed away. In his Turing Award lecture, he said that in their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

UBIQUITY: A nice quote.

CAI: Yes, such lofty sentiment. I feel it's worth repeating now and then to remind ourselves that there are more basic problems in computing that we can focus on. We could talk about funding, but my take is that there are more fundamental long-term issues than funding.

UBIQUITY: Even so, you must have some general thoughts on funding issues, especially do you observe any difference here and abroad?

CAI: I spent half a year at the University of Toronto supported by a Guggenheim Fellowship. Toronto has a wonderful computer department with many outstanding, important people in theory, as well as other areas. I learned during my stay there that the funding situation in Canada is rather different from what it is in the U.S. People get a certain amount of funding relatively easily. Also it's almost impossible to get a huge amount, so therefore, people tend to have roughly the same amount. There is less pressure to get more funding. In the U.S., theoreticians don't get millions of dollars, but millions of dollars of funding are available to certain other areas. I don't want to belittle their effort to get that. You have to be very good in your area to be able to do it.

UBIQUITY: Money rules?

CAI: Even Hardy admitted that researchers are partly motivated by money. He was saying that if someone says he is motivated just for the goodness of humanity, and not for the ambition, position, influence, money and power that to some extent come with it, he shall not believe it, or shall not think any better of it, even if he believed it. Money is
not just salary. It's also the amount of research funding you bring in, which also brings with it certain influence, position, leverage. University administrators are always looking for more funding. This is clear. You can look on the Web and see some departments carry as a badge of honor their research expenditure at X million dollars. I find it somewhat strange, that in and of itself spending a million dollars is something to be proud of. Isn't it incumbent upon those who claim they have spent more to at least prove that they have done more? How do you know these millions weren't wasted?

UBIQUITY: What is your personal motivation for doing your work?

CAI: To seek a deeper understanding of the quantitative laws that govern computation. And a certain amount of ambition and competitiveness. Not primarily money, and not primarily seeing my work applied. Certainly, I would very much like to see my work being shown to be useful, but I don't consider that as my primary motivation. I look to uncover the laws of computation, much as physicists look for the laws of nature, and try to uncover the mathematical reason why it is so. I believe that with a certain amount of curiosity and luck, and a good amount of work and perseverance, I can make some contribution. Experience seems to show that when you find something beautiful, it's very likely it will be useful later on too. In the long run, money doesn't rule. Beauty rules.