Keeping Complexity Under Rein

HENRYK WOŹNIAKOWSKI
Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw
Department of Computer Science, Columbia University
Corresponding member, PAS

Prof. Henryk Woźniakowski studies computational complexity theory. For around 20 years he has been working on tractability; he also studies the properties of the quasi-Monte Carlo method for potential applications in financial mathematics

Prof. Henryk Woźniakowski talks to Academia about the past, present, and future of computational complexity theory

Academia: What is computational complexity theory?

Prof. Henryk Woźniakowski: Complexity theory is a fairly new branch of mathematics, or, more precisely, theoretical informatics. It arose around the time of the advent of computers. People quickly realized that the difficulty of certain problems could be so great that they couldn't be solved even by the fastest computer.

Some problem-solving algorithms are extremely slow. It's not easy to determine straight away if a problem is simply too difficult to solve, or if we're just using the wrong algorithm. This has given rise to the field of computational complexity, which seeks a way to find the lowest-cost algorithms and studies the complexity of the problem.

In the past people were able to wait a long time for a solution – what was important was that the process should require as little computer memory as possible. Back then, the cost factor was computer memory. Nowadays there is generally a lot of computer memory available, whereas it is computation time that we try to reduce. Complexity theory aims to provide a good estimate for the lowest possible time required for an algorithm.

So the cost factor today really is time?

Yes, the run time of the algorithm operation. To assess complexity, we can use the worst-case scenario: assessing algorithm quality in terms of the longest time taken for all the data that interests us. It's a bit as though a student's worst grade throughout his studies was then given as his final overall grade. In such a system a student who gets straight As wouldn't mind, but another who has mostly As and a single F would fail. Such rigorous criteria classify many tasks as too difficult. An alternative is to use average complexity: assuming that we can have an occasional slip up, and just averaging the algorithm time for all data. Yet another option is to use a probabilistic scenario, and assess the algorithm after rejecting its behavior for, say, 1% of data. We also frequently consider randomized cases, when the algorithm time is estimated on the basis of average behavior for random parameters. This gives a higher probability of finding a fast algorithm. Computational complexity is usually lower – sometimes significantly so – in the average, probabilistic or randomized cases than in the worst case.

But not always?

There are problems for which the results are always bad. Certain problems suffer from the curse of dimensionality. Let's say we have a problem defined for a class of d-variable functions. In computational practice, problems where d is huge are increasingly common. For example, in financial mathematics integrals need to be calculated for functions with 360 or more variables! Computational complexity is frequently an exponential function of d. For example, if cost is defined as at least 2 to the power of d, for d=3 it would equal 8 – not that much. However, when we have 360 variables, 2 to the power of 360 in practice gives a number so high as to be close to infinity. So what do we do then? We can introduce further assumptions on those functions to sensibly reduce the function class and remove the curse of dimensionality.

Leszek Plaskota

In 2010, Prof. Henryk Woźniakowski (in the centre of the front row) co-organized the 9th International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing

So how can we apply complexity theory to a practical problem?

That brings us to financial mathematics. To start with I'll clue you in to why financiers want to calculate 360-dimensional integrals. Major loans, for example mortgages to buy a house, are usually given for 30 years. Because the markets fluctuate incredibly, banks reserve the right to slightly amend the terms of such a loan agreement every month. Over 30 years that comes to 360 rates, hence the 360 parameters. Banks want to be fair to their customers, but even more than that they want to guarantee a profit for themselves, so they calculate the expected value of the loan given to the customer. Mathematically this expected value is an integral.

Until the 1990s, integrals were calculated using (randomized) Monte Carlo methods. Completely different methods also exist, for example the quasi-Monte Carlo. In the 1990s scientists suddenly discovered that those methods are far better. It turned out that they give results that are very similar to the Monte Carlo method, but using significantly fewer steps. The riddle of this efficiency has not been fully explained to this day, although I think that we do now grasp the main elements. So – we do have 360 variables, but they vary in significance. In such problems quasi-Monte Carlo algorithms can in a way eliminate the insignificant variables; then instead of having 360, we are left with just two or three. Of course no one wants to have to precisely calculate those integrals; one percent is already quite a high degree of accuracy. I know that in Australia they are successfully calculating integrals for over 9000 variables!

Research is now being done on quantum computers. Tasks that are difficult for classical computers are not necessarily so for quantum computers. Interest in quantum computation came through Shor's algorithm. It has applications in cryptography, or, to put it more simply, in banking security. The task of finding prime factors is known to be difficult for very large integers. However, once you have those prime factors, you can easily reconstruct the original number. This principle is used in encrypting data. Shor demonstrated that by using quantum computers the problem of finding prime factors for a given number N can actually be solved at a cost proportional to log N to the power of at most 3, whereas we don't know any algorithm using standard computers whose cost is a power of log N.

So it's very low-cost?

Extremely so. I think that's why so many governments are worried that whoever owns the first quantum computer will be able to "unlock" everyone's bank passwords. Billions of dollars have been channeled into research, though unfortunately there is still little technological progress towards actually building quantum computers. But if such a computer does exist one day, we will have a new computational model and a new instance of computational complexity, with all the consequences that entails.

And do you see a future for complexity theory in quantum computing?

As a man of advanced age I don't think it will happen soon. Tomorrow – in the biblical sense – perhaps we'll have such computers. Research into new technologies and new methods of building computers appeals to me, although I don't know if it will ever work. The mathematical model of quantum computing is itself very interesting. But will it ever become a reality? I don't know. Probably partially so.

Interview by Agnieszka Pollo