Abel interview 2021: László Lovász and Avi Wigderson

Bjørn Ian Dundas and Christian F. Skau

Professor Lovász and Professor Wigderson. First, we want to congratulate you on being the Abel Prize recipients for 2021. We cite the Abel committee:

> For their foundational contributions to theoretical computer science and discrete mathematics, and their leading role in shaping them into central fields of modern mathematics.

We would first like you to comment on the remarkable change that has occurred over the last few decades in the attitude of, say, mainstream mathematics towards discrete mathematics and theoretical computer science. As you are fully aware of, not that many years ago it was quite common among many first class mathematicians to have a sceptical, if not condescending opinion, of this type of mathematics. Please, could you start, Professor Lovász?

Lovász. I think that is true. It took time before two things were realized about theoretical computer science that are relevant for mathematics.

One is simply that it is a source of exciting problems. When I finished the university, together with some other young researchers, we started a group to study computing and computer science, because we realized that it's such a huge unexplored field; questions about what can be computed, how fast and how well and so on.

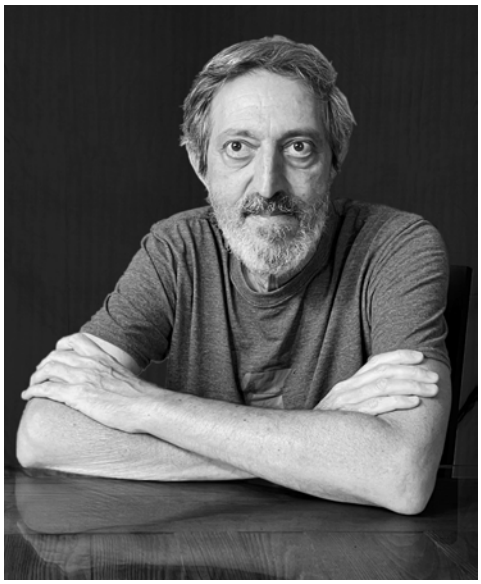
The second thing is that when answers began to come, in particular, when the notions of NP and P, i.e., nondeterministic polynomial time and polynomial time became central, we realized that the whole of mathematics can be viewed in a completely different way through these notions, through effective computation and through short proofs of existence.

For us young people these two things were so inspiring that we started to make connections with the rest of mathematics. I think it took time until other areas of mathematics also realized the significance of this, but gradually it came about. In number theory it turned out to be very important, and also in group theory these notions became important, and then slowly in a lot of other areas of mathematics.

Wigderson. Yeah, I completely agree. In fact, it's true that there was a condescending attitude among some mathematicians towards discrete mathematics. This was perhaps less so in theoretical computer science, because it existed in the realm of computer science as it was developing, and maybe people were less aware of it directly? I think that Lovász is right in that the very idea of efficient algorithms and the notions of computational complexity that were introduced in theoretical computer science are fundamental to mathematics, and it took time to realize that.

However, the real truth is that all mathematicians of all ages, they all used algorithms. They needed to compute things. Gauss' famous challenge to the mathematical community to find fast methods to test whether a number is prime and to factor integers is extremely eloquent, given the time it was written. It's really calling for fast algorithms to be developed.

Parts of discrete mathematics were viewed by some as trivial in the sense that there are only finite number of possibilities that



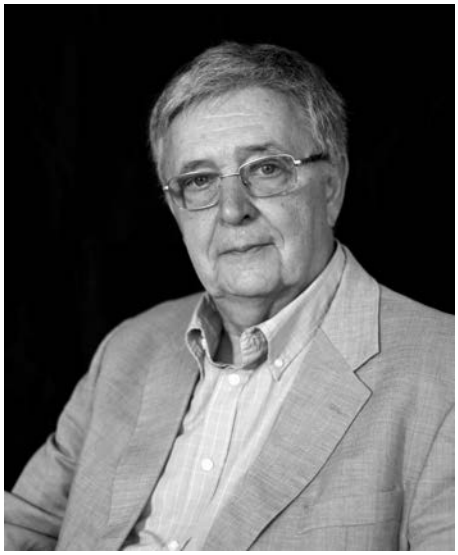
Avi Wigderson
© Peter Badge/Abel Prize

EMS MAGAZINE 122 (2021) — DOI 10.4171/MAG-54

we have to test. Then, in principle, it can be done, so what is the problem?

I think the notion of an efficient algorithm clarifies what the problem is. There may be an exponential number of things to try out, and you will never do it, right? If instead you have a fast algorithm for doing it, then it makes all the difference. The question whether such an algorithm exists becomes all important.

This understanding evolved. It caught up first with pioneers in the 70s in the area of combinatorics and in the area of discrete mathematics, because there it's most natural; at least it's easy to formulate problems, so that you can attach complexity to them. Gradually it spread to other parts of mathematics. Number theory is a great example, because there too there are discrete problems and discrete methods hiding behind a lot of famous number theoretical results. From there it gradually dispersed. I think by now it's pretty universal to understand the importance of discrete mathematics and theoretical computer science.



László Lovász
© Peter Badge/Abel Prize

Turing and Hilbert

This is admittedly a naïve question, but as non-experts we have few inhibitions, so here goes: Why is it that Turing's notion of what is today called a Turing machine captures the intuitive idea of an effective procedure, and, so to speak, sets the standard for what can be computed? How is this related to Hilbert's Entscheidungsproblem?

Wigderson. I think my first recommendation would be to read Turing's paper – in fact, to read *all* his papers. He writes so eloquently. If you read his paper on computing procedures and the Entscheidungsproblem, you will understand everything.

There are several reasons why the Turing machine is so fundamental and so basic. The first one is that it's simple – it's extremely simple. That was evident to Turing and to many others at that time. It's so simple that it could be directly implemented. And thereby he started the computer revolution. If you look at other notions of computability that people studied, Gödel and others – definitely Hilbert – with recursive functions and so on, they did not lend themselves to being able to make a machine out of them. So this was fundamental.

The second is that a few years later it was proved that all other notions of efficient computability were equivalent. So the Turing machine could simulate all of them. It encompassed all of them, but it was much simpler to describe.

Thirdly, one way Turing motivates his model is to look at what we humans do when we calculate to solve a problem, let's say multiply two long numbers. Look at what we do on a piece of paper, we abstract it and formalize it. And when we do that, we will automatically be lead to a model like the Turing machine.

The fourth reason is the universality, the fact that his model is a universal model. In a single machine you can have part of the data be a program you want to run, and it will just emulate this program. That is why we have laptops, computers and so on. There is just one machine. You don't need to have a different machine to multiply, a different machine to integrate, a different machine to test primality, etc. You just have one machine in which you can write a program. It was an amazing revolution and it encapsulates it in a really simple notion that everybody can understand and use, so that is the power of it.

Now, you asked about the relation to the Entscheidungsproblem. You know, Hilbert had a dream, and the dream had two parts: Everything that is true in mathematics is provable, and everything provable can be automatically computed. Well, Gödel shattered the first one – there are true facts, let's say, about integers, that can not be proven. And then Church and Turing shattered the second one. They showed that there are provable things that are not computable. Turing's proof is not only far simpler than Gödel's, with Turing's clever diagonal argument, it also implies the Gödel result if you think about it. This is usually the way most people teach Gödel's incompleteness theory today; well, I don't know if "most people" would agree to this, but it's using Turing's notions. So that's the connection. Turing was, of course, inspired by Gödel's work. The whole thing that led him into working on computability was Gödel's work.

Lovász. I have just one thing I would like to add. A Turing machine is really consisting of just two parts. It's a finite automaton and a memory. If you think about it, the memory is needed. Whatever computation you do you need to remember the partial results.

The memory in its simplest way is to just write it on a tape as a string. The finite automaton is sort of the simplest thing that you can define which will do some kind of, actually any kind of computation. If you combine the two you get the Turing machine. So it's also natural from this point of view.

P versus NP

Now we come to a really big topic, namely the P *versus* NP *problem, one of the Millennium Prize Problems. What is the* P *versus* NP *problem? Why is that problem the most important in theoretical computer science? What would the consequences be if* $P = NP$*? What do you envisage a proof of* $P \neq NP$ *would require of tools?*

Lovász. Well, let me again go back to when I was a student. I talked to Tibor Gallai, who was a distinguished graph theorist and my mentor. He said: Here are two very simple graph-theoretical problems. Does a graph have a perfect matching, that is, can the vertices be paired so that each pair is connected by an edge? The other one is whether the graph has a Hamiltonian cycle, i.e., does it have a cycle which contains all the nodes?

The first problem is essentially solved; there is a lot of literature about it. As for the other we only have superficial results, maybe nontrivial results, but still very superficial.

Gallai said, well, you should think about it, so that maybe you could come up with some explanation. Unfortunately, I could not come up with an explanation for that, but with my friend, Péter Gács, we were trying to explain it. And then we both went off – we got different scholarships: Gács went to Moscow for a year and I went to Nashville, Tennessee for a year. Then we came back and we both wanted to speak first, because we both had learned about the theory of P versus NP, which completely explains this. Peter Gács learned it from Leonid Levin in Moscow, and I learned it from listening in on discussions taking place at coffee tables at conferences.

The perfect matching problem is in P and the Hamilton cycle problem is NP-complete. This explained what really was a tough question. It was clear that this was going to be a central topic, and this was reinforced with the work of Karp proving the NP-completeness of lots of everyday problems. So, summing-up, the notions of P and NP they made order where there was such a chaos before. That was really overwhelming.

Wigderson. The fact that it puts an order on things in a world that looked pretty chaotic is the major reason why this problem is important. In fact, it's almost a dichotomy, almost all natural problems we want to solve are either in P, as far as we know, or are NP-complete. In the two examples Lovász gave, first the perfect matching, which is in P, we can solve it quickly, we can characterize it and do a lot of things, we understand it really well. The second example, the Hamiltonian cycle problem is a representative of an NP-complete problem.

The main point about NP-completeness is that every problem in this class is equivalent to every other. If you solve one, you have solved all of them. By now we know thousands of problems that we want to solve, in logic, in number theory, in combinatorics, in optimization and so on, that are equivalent.

So, we have these two classes that seem separate, and whether they are equal or not is the P versus NP question; and all we need to know is the answer to one of the NP-complete problems.

But I want to look at the importance of this problem from a higher point of view. Related to what I said about natural problems we want to compute, I often argue in popular lectures that problems in NP are really all the problems we humans, especially mathematicians, can ever hope to solve, because the most basic thing about problems we are trying to solve is that we will at least know if we have solved them, right? This is true not only for mathematicians. For example, physicists don't try to build a model for something for which, when they find it, they will not know if they have found it. And the same is true for engineers with designs, or detectives with solutions to their puzzles. In every undertaking that we seriously embark on, we assume that when we find what we were looking for we know that we have found it. But this is the very definition of NP: a problem is in NP exactly if you can check if the solution you got is correct.

So now we understand what NP is. If $P = NP$, this means that all these problems have an efficient algorithm, so they can be solved very quickly on a computer. In some sense, if $P = NP$ then everything we are trying to do can be done. Maybe find a cure for cancer or solve other serious problems, all these can be found quickly by an algorithm. That is why $P = NP$ is important and would be so consequential. However, I think most people believe that $P \neq NP$.

Lovász. Let me add another thought on how it can be proved that $P \neq NP$. There is a nice analogy here with constructions with ruler and compass. That is one of the oldest algorithms, but what can you construct by ruler and compass? The Greeks formulated the problems about trisecting the angle and doubling the cube by ruler and compass, and they probably believed, or conjectured, that these were not solvable by ruler and compass. But to prove this is not easy, even today. I mean, it can be taught in an undergraduate class, in an advanced undergraduate class, I would say. You have to deal with the theory of algebraic numbers and a little bit of Galois theory in order to be able to prove this. So to prove that these problems are not solvable by a specific algorithm took a huge development in a completely different area of mathematics.

I expect that $P \neq NP$ might be similar. Of course, we probably will not have to wait 2000 years for the solution, but it will take a substantial development in some area which we today may not even be aware of.

But we take it for granted that you both think that P *is different from* NP*, right?*

Wigderson. I do, but I must say that the reasons we have are not very strong. The main reason is that for mathematicians it seems obviously much easier to read proofs of theorems that are already discovered, than to discover these proofs. This suggests that P is different from NP. Many people have tried to find algorithms for many of the NP-complete problems for practical reasons, for example, various scheduling problems and optimization problems, graph theory problems, etc. And they have failed, and these failures may suggest that there are no such algorithms. This, however, is a weak argument.

In other words, I intuitively feel that $P \neq NP$, but I don't think it's a strong argument. I just believe it as a working hypothesis.

Problems versus theory

We often characterize mathematicians as theory builders or as problem solvers. Where would you place yourself on a scale ranging from theory builder to problem solver?

Wigderson. First of all, I love solving problems. But then I ask myself: Oh, this is how I solved it, but maybe this is a technique that can be applied other places? Then I try to apply it in other places, and then I write it up in its most general form, and that is how I present it. In this way I may also be called a theory builder. I don't know. I don't want to characterize myself in terms of theory builder or problem solver.

I enjoy doing both things, finding solutions to problems and trying to understand how they apply elsewhere. I love understanding connections between different problems, and even more between different areas. I think we are lucky in theoretical computer science that so many seemingly dispersed areas are so intimately connected, but not always obviously so, like with hardness and randomness. Theory is built out of such connections.

Lovász. I have similar feelings. I like to solve problems. I started out under the inspiration of Paul Erdős, who was really always breaking down questions into problems. I think that was a particular strength of his mathematics, that he could formulate simple problems that actually illustrated an underlying theory. I don't remember who said this about him: it would be nice to know the general theories that are in his head, which he breaks down into these problems that he feeds us so that we can solve them. And, indeed, based on his problems, whole new areas arose, extremal graph theory, random graph theory, probabilistic combinatorics in general, and various areas of number theory. So I started as a problem solver, but I always liked to make connections, and tried to build something more general out of a particular problem that I had solved.

László Lovász. © Hungarian Academy of Sciences/Institute for Advanced Study, Princeton, NJ, USA

Youth in Haifa

Professor Wigderson, you were born in 1956 in Haifa, Israel. Could you tell us when you got interested in mathematics and, in particular, in theoretical computer science?

WIGDERSON. I got interested in mathematics much earlier than in computer science. As a very young child, my father introduced me to mathematics. He liked to ask me questions and to look at puzzles, and I got interested. We found books that I could read, and in these there would be more problems. This was my main early interaction with mathematics. In high school we had a very good mathematics teacher who came from Ukraine, and he had a special class for interested kids. He taught us more exciting stuff, like college level stuff, and I got even more excited about mathematics. In college I got much more into it, but it's actually an accident that I got into computer science, and thereby to theoretical computer science.

After my army service, as I was applying to colleges in Israel, I thought that I wanted to do mathematics, but my parents suggested that it might be good to also have a profession when I graduated. So they said: "Why don't you study computer science, it will probably be a lot of math in it anyway, and you will enjoy it. Also, when you graduate you will have a computer science diploma." Nobody was thinking about academia at that time.

So I went to the computer science department at the Technion, and I think I was extremely lucky. I am sure that if I had gone to a math department I would have been interested in many other things, like analysis, combinatorics, geometry, and so on. Because I was in the computer science department, I took several theoretical courses. We had, in particular, a very inspiring teacher, Shimon Even, at the Technion. His courses on algorithms and complexity were extremely inspiring. When I applied to graduate school I applied for continuing to do this sort of stuff. This was how I was drawn into theoretical computer science.

But still, in an earlier interview you have described yourself as a beach bum and a soccer devotee. That contrasts rather starkly with what you have been telling us now, doesn't it?

WIGDERSON. I don't think there is a contrast. I mentioned that my dad was my main intellectual contact in mathematics. The schools in the neighbourhood of our home were not very good, it was pretty boring. The neighbourhood was situated by the beach, so everybody was at the beach. We were beach bums by definition. The weather in Israel is wonderful, so you can be altogether 300 days a year on the beach and in the water. So that was one pastime activity. The other thing you mentioned, soccer, is the easiest game to play. You need no facilities. And that was what we did, being involved in these two activities. When I was growing up I never saw myself as an intellectual. I loved math, but I also loved soccer, I loved swimming and I loved reading. And this is how I spent all my youth. There were no contrast and, if anything, it's probably good to do other things.

Youth in Budapest

Professor Lovázs, you were definitely not a beach bum.

LOVÁSZ. When I entered eighth grade, I did not have any special subject that I was particularly interested in. In the eighth grade I started to go to a math club, and I realized how many interesting problems there are. Then the teacher of the math club recommended that I should go to a particular high school which had just started and was specializing in teaching mathematically talented kids. We had ten classes of mathematics a week, which this group of students enjoyed very much, including myself. I appreciated very much the fact that I was among a fairly large group of students who had quite similar interests.

In elementary school, I was a little bit outside of the "cool" group of the class. I was not in the mainstream of the class, but in this new high school class I found myself much more at home. In fact, I felt so much at home that I married one of my classmates, Katalin Vesztergombi, and we are still together.

This high school was absolutely a great start for my life, that is how I feel about it. Before that … you just had to survive the school. I entered this new high school in the first half of the sixties. There were many good mathematicians in Budapest at that time, and they did not really have the chance to travel or to do anything outside Hungary. So they had more time, and they came to the high school and gave talks and they took a great interest in our group. We learned a lot from them. I should, of course, mention that Paul Erdős was often visiting the class and gave talks and gave problems. So it was all very inspiring.

Professor Lovász, to quote professor Wigderson: "In the land of prodigies and stars in Hungary, with its problem solving tradition, he (meaning you) stands out." We have a witness who recalls rushing home from school to watch the final of one of the competitions in which you participated on national television, where you solved a combinatorial problem in real time and won the competition. It's kind of hard to imagine doing such things now and in the West.

LOVÁSZ. You are right. That was one of the things that went on for a few years on Hungarian TV, but unfortunately it stopped. Unfortunately, because I thought it was a very good popularization of mathematics. You know, people like to watch competitions. The way it worked was that there were two glass cells, and the two students that were competing were sitting in separate cells. They got the same problem which they had to solve, and then verbally tell the solution; maybe there was a blackboard they could use as

well. I think people like to watch youngsters sweating and doing their best to win. You know, most people cannot jump over two meters, but nevertheless we watch the Olympic Games. Even today I meet people, of course older people, who say: "Oh, yeah, I saw you on TV when you were in high school, and I was in the eighth grade of elementary school, and it was so nice to watch you." It was really something quite special.

A part of this story which is both funny and charming is that you told us that the solution to the final problem, with which you won the competition, you had previously learned from the other competitor. Isn't that correct?

Lovász. Yes, that's true. But we competitors were also good friends, and we still are very good friends. Especially the two people with whom I competed in the semi-final and the final, are very good friends of mine. One was Miklós Laczkovich. He came up with the proof of Tarski's conjecture about squaring the circle. And the other one was Lajos Posa. He is very well known in math education. He did a lot on developing methods to teach talented students.

Before we leave this subject, we should also mention that you won the gold medal three years in a row 1964–65–66 at the International Mathematical Olympiad, when you were 16–17–18 years old. These are impressive results! We don't know of anyone that has such record in that competition.

Lovász. Thank you, but there are others. Someone has won it five times. You can go to the website of the International Mathematical Olympiad, and there you find a list of achievements.

Lovász local lemma

Professor Lovász, you have published several papers – we think six papers altogether – with your mentor Paul Erdős. We think we know the answer to which one of these papers is your favourite, and you can correct us if we are wrong. A weak version of the important so-called Lovász local lemma was proven in 1975 in a joint paper with Erdős – that's the paper we have in mind. The lemma itself is very important as is attested to by Robin Moser and Gábor Tardos receiving the Gödel Prize in 2020 for their algorithmic version of the Lovász local lemma. Anyway, could you tell us what the Lovász local lemma is all about?

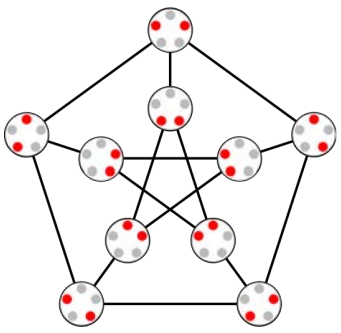
Lovász. Okay, I will try. Almost everything in mathematics, or at least in discrete mathematics, you can formulate like this: there are a number of bad events, and you want to avoid all of them. The question is whether you can give a condition so you can avoid all of these. The most basic thing is that if the probabilities of these events add up to something less than one, then with positive probability you will avoid all of them. That is a very basic trick in applications of probability in discrete mathematics. But suppose that the number is much larger, so that the probabilities add up to something very large, how do you handle that? Another special case is if they are independent events. If you can avoid each of them separately, then there is a positive probability that you avoid all of them, simply take the product of the probabilities for avoiding each one of them.

The local lemma is some kind of combination of these two ideas. If the events are not independent, but each of them is dependent only on a small number of others, and if the sum of the probabilities of those that it depends on is less than one – not the total sum, but just those it depends on – then you can still, with positive probability, avoid each of the bad events.

Maybe I should add one thing here. There was a problem of Erdős which I was thinking about, and I came up with this lemma. I was together with Erdős, actually at Ohio State for a summer school. We solved the problem, and we wrote a long paper on that problem and related problems, including this lemma. But Erdős realized that this lemma was more than just a lemma for this particular problem and made sure that it became known under my name. Normally it would be called the Erdős–Lovász local lemma, because it appeared in a joint paper of us, but he always promoted young people and always wanted to make sure that it became known if they had proved something important. I benefited from his generosity.

The Kneser conjecture

In 1955 Kneser conjectured how many colours you need in order to colour a type of naturally occurring graphs, now known as Kneser graphs. In 1978 you, professor Lovász, proved this conjecture by encoding the problem as a question of high dimensional spaces, which you answered by using a standard tool in homotopy theory, and so boosted the field of combinatorial topology. How did such a line of approach occur to you, and can you say something about the problem and your solution?



Kneser graph $K(5,2)$

Lovász. It goes back to one of these difficult problems, the chromatic number problem: how many colours do you need to colour a graph properly, where properly means that neighbouring vertices must have different colours. That is a difficult problem in general, it's an NP-complete problem.

A first approach is looking at the local structure. If a graph has many vertices which are mutually adjacent, then, of course, you need many colours. The question is: is there always such a local reason? It was already known at that time that there are graphs which have absolutely no local structures, so that they don't have short cycles, but nevertheless you need many colours to colour them. It was an interesting question to construct such graphs. For example, just exclude triangles, or more generally, exclude odd cycles from the graph. There was a well known construction for such a graph by looking at the sphere, and then connecting two points if they are almost antipodal. Then Borsuk–Ulam's theorem says that you will need more colours than the dimension, so that the almost antipodal points have different colours. That was one construction, and the other one was the construction where the vertices would be a k-element subset of an n-element set, where $n > 2k$, and you can connect two of these if they are disjoint. Kneser conjectured what would be the chromatic number of such a graph.

It was an interesting problem going around in Budapest. Simonovits, a friend and colleague of mine, brought to my attention that these problems could actually be similar, or that these two constructions could be similar. So I came up with a reduction of one into the other, but then it turned out that the reduction was more general and gave a lower bound on the chromatic numbers of any graph in terms of some topological construction. So, that is how topology came in. It took actually quite some time to make it work. As I remember it, I spent about two years to make these ideas work, but eventually it worked.

Zero-knowledge proofs

Professor Wigderson, earlier in your career you made fundamental contributions to a new concept in cryptography, namely the zero-knowledge proof, which more than 30 years later is now being used for example in blockchain technology. Please tell us specifically what a zero-knowledge proof is, and why this concept is so useful in cryptography?

Wigderson. As a mathematician, suppose you found the proof of something important, like the Riemann hypothesis. And you want to convince your colleagues that you have found this proof, but you don't want them to publish it before you do. You want to convince them only of the fact that you have a proof of this theorem, and nothing else. It seems ridiculous, it seems absolutely ridiculous, and it's contrary to all our intuition that there is a way to convince someone of something they do not believe, without giving them any shred of new information.

This very idea was raised by Goldwasser, Micali and Rackoff in 1985, where they suggested this notion. They did not suggest it for paranoid mathematicians, but they suggested it for cryptography. They realized that in cryptography there are a lot of situations, in fact, almost all situations, of interactions between agents in a cryptographic protocol, in which no one trust the other ones. Nevertheless, each of them makes claims that they are doing something, or knowing something, which they don't want to share with you. For example, their private key in a public crypto system. You know, each one is supposed to compute their public key by multiplying two prime numbers, which they keep secret. I give you a number and I tell you: here is a number; I multiplied two secret prime numbers and this is the result. Why should you believe me? Maybe I did something else, and this is going to ruin the protocol. To fix this, it would be nice if there was a way for me to convince you that that's exactly what I did. Namely, there exist two prime numbers whose product is the number I gave you. That is a mathematical theorem, and I want to convince you of it, without giving you any idea what my prime numbers are, or anything else. Goldwasser, Micali and Rackoff suggested this extremely useful notion of a zero-knowledge proof.

They gave a couple of nontrivial examples, which was already related to existing crypto systems where this might be possible. And they asked the question: what kind of mathematical statements can you have a zero-knowledge proof for? A year later, with Goldreich and Micali, we proved that it was possible for any mathematical theorem. If you want it formally, it's true for any NP-statement.

So this is the content of the theorem. I am not going to tell you the proof of the theorem, though something can be said about it. The proof uses cryptography in an essential way. It's a theorem which assumes the ability to encrypt. Why it's useful in cryptography is exactly for the reasons I described, but, in fact, it's much more general, as we observed in a subsequent paper. Given a zero-knowledge proof, you can really automate the generation of protocols that are safe against bad players. The way to get a protocol that is resilient against bad players once you have a protocol that works if everybody is honest and doing exactly what they should, is just to intervene every step with a zero-knowledge proof, in which the potentially bad players will convince the others that they are doing the right thing. It's much more complex, zero-knowledge is not enough, you have to have a way to computing with secrets.

I want to stress that when we proved this theorem, it was a theoretical result. It was clear to us from the beginning that the protocol which enables zero-knowledge proof, is complex. We thought it unlikely to be of any use in cryptographic protocols that are running on machines.

The fact that they became practically useful is still an amazing thing to me, and I think that is a good point to make about

many other theoretical results in theoretical computer science, in particular, about algorithms. People tend to complain sometimes about the notion of P being too liberal when describing efficient algorithms, because some algorithms, when they were first discovered, may have a running time that looks too large. It's polynomial time, but maybe it's n to the 10th power, and for n size a thousand, or size a million, which are problems that come up naturally in practice, it seemed useless, as useless as exponential time algorithms. But what you learn again and again, both in the field of cryptography and in the field of algorithms, is that once you have a theoretical solution with ideas in it that make it very efficient, then other people, especially if they are motivated enough, like in cryptography or in optimization, can make it much more efficient, and eventually practical. That's a general point I wanted to make.

Randomness versus efficiency

It's an amazing result, and we quote you saying: "This is probably the most surprising, the most paradoxical of the results that my colleagues and I have proved". Let us continue, professor Wigderson, with another topic to which you have made fundamental contributions. When you began your academic career in the late 1970s, the theory of computational complexity was in its infancy. Your contribution in enlarging and deepening the field is arguably greater than that of any other person. We want to focus here on your stunning advances in the role of randomness in aiding computation. You showed, together with co-workers Nisan and Impagliazzo, that for any fast algorithm that can solve a hard problem using coin flipping, there exists an almost as fast algorithm that does not use coin flipping, provided certain conditions are met. Could you please elaborate on this?

Wigderson. Randomness has always fascinated me. Specifically, the power of randomness in computation, but not only in computation. This is probably the area where I have invested most of my research time. I mean, *successful* research time! The rest would be trying to prove lower bounds or hardness results, like proving that P is different from NP, where I generally – like everybody else – failed.

So, back to randomness. Ever since the 1970s, people realized that randomness is an extremely powerful resource to have in algorithms. There were initial discoveries, like primality tests. Solovay/Strassen and Miller/Rabin discovered fast methods with randomness to test if a number is prime. Then in coding theory, in number theory, in graph theory, in optimization and so on, randomness was used all over the place. People just realized it's an extremely powerful tool to solve problems that we have no idea of how to solve efficiently without randomness. With randomness you can find the solution very fast. Another famous class of examples is Monte Carlo methods. So you explore a large chunk of problems using randomness. Without it, it seemed like it would take exponential time to solve them, and it was natural to believe that having randomness is much more powerful than not having it.

Nevertheless, mainly from motivations in cryptography, people started in computational complexity trying to understand pseudorandomness. You need randomness in cryptographic protocols for secrecy. On the other hand, sometimes random bits were not so available, and you wanted to test when random bits are good, as good as having independent coin flips – which you really assume when talking about probabilistic algorithms.

So, there was a quest to understand when a distribution of bits is as good as random. This started in cryptography with a very powerful work by Blum, Micali and Yao. Notions began to emerge which suggested that if you have computational hardness, if you somehow have a hard problem, then you can generate pseudorandom bits cheaply. So you can invest much less randomness in order to generate a lot, which is still useful, let's say for probabilistic algorithms.

This kind of understanding started in the early 1980s. It took about 20 years of work to really elucidate it and to be able to make the weakest assumptions on what hardness you need in order to have a pseudorandom outcome, which then corresponds to a full probabilistic algorithm. Parts of this were indeed developed in my papers with Nisan, and then with Babai and Fortnow, and then with Impagliazzo and Kabanets.

The upshot of this development is again a conditional result, right? You have to assume something, if you want the conclusion you stated. What you need to assume is that some problem is difficult. You can take it to be the problem of colouring graphs, you can take it to be any NP-complete problem you like, or even problems that are higher up, but you need a problem that is exponentially difficult. This is the assumption that the result is conditioned on. If you are willing to make this assumption, then the conclusion is exactly as you said, namely that every efficient probabilistic algorithm can be replaced by a deterministic algorithm which does the same thing. In fact, it does it without error and is roughly as efficient as the original one.

In other words, the power of probabilistic algorithms is just a figment of our imagination. It's only that we are unable to find deterministic algorithms that we can prove are as efficient. This result suggests that there is no such power and that randomness does not help to make efficient algorithms more efficient.

The hardness assumption that you need to make, were they something that you were expecting?

Wigderson. They are completely expected! First of all, they are expected in the sense that they were there from the beginning, specifically in the works of Blum, Micali and Yao that I mentioned, which do create pseudorandom generators that are good against efficient algorithms and which assume specific hardness assumptions like those used in cryptography. For example, that factoring

Avi Wigderson. © Andrea Kane, Institute for Advanced Study, Princeton, NJ, USA

is difficult, or that one way functions exist. These are very specific hardness assumptions, and these problems are unlikely to be NP-complete.

In my paper with Nisan, we realized that a much weaker assumption is enough. It was not enough to give the result stated at the end, because it's not efficient enough, but it already got us pretty close to the understanding that random algorithms are not as powerful as they seemingly are. It did not give the BPP = P consequence, which is the final one. This was not surprising, the connection paradigm between hardness and randomness came from the very initial studies of computation of pseudorandomness, and, if I remember correctly, the paper of Blum and Micali, or perhaps even the Ph.D. thesis of Silvio Micali, is titled: *Hardness vs. Randomness*. There is an intimate connection there – it was there from the start – and the question is how tight the connection is.

I should probably mention that the consequence of what we just discussed is that hardness implies derandomization, and the question is whether the reverse hold also. If you have a good pseudorandom generator, or if you could derandomize all probabilistic algorithms, does it mean that you can prove something like P ≠ NP? The answer is that we have partial results like that. My paper with Impagliazzo and Kabanets is one, and there is another paper with just the two of them. So there are partial results for the converse, and we don't understand it fully. But it's a fascinating connection, because these two issues seem separate from each other. I think it's a very fundamental discovery of the field, this intimate connection between computational difficulty and the power of randomness.

The LLL-algorithm

Professor Lovász, we would like to talk about the LLL*-algorithm, an algorithm which has striking applications. For instance, it's claimed that the only crypto systems that can withstand an attack by a quantum computer use* LLL*. The algorithm appears in your paper together with the Lenstra brothers on factorization of polynomials, which more or less follows the expected path of reducing modulo primes, and then using Hensel's Lemma. But as far as we understand the breakthrough from you and the Lenstra brothers was that you were able to do the lift in polynomial time by an algorithm giving you an approximation to the shorter vector in a lattice. Tell us first how the collaboration with the Lenstras came about.*

Lovász. This is an interesting story about mathematics and the role of beauty, or at least elegance, in mathematics. With Martin Grötschel and Alexander Schrijver we were working on applications of the ellipsoid method in combinatorial optimization. We came up with some general theorem that stated some equivalence of separation and optimization. Actually, these were polynomial time equivalent problems under some mild additional conditions. But there was a case where the algorithm did not work, and that was when the convex body was lying in a lower dimensional linear subspace. One could always get around this, sometimes by mathematical methods, for example, by lifting everything into a higher dimensional space. But there was always some trick involved that we wanted to avoid.

At some point I realized that we can solve this if we can solve some really ancient mathematical problem algorithmically. That was Dirichlet's result that several real numbers can be simultaneously approximated by rational numbers with the same denominator, and the question was whether you could solve this algorithmically. Now one looks at the proof and one sees immediately that the proof is the opposite of being algorithmic; it's a pigeonhole principle proof, so it just shows the existence of such an approximation. After some trial and error, I came up with an algorithm which actually computed in polynomial time such an approximation with rational numbers with common denominator.

A little bit earlier I heard a talk of Hendrik Lenstra, where he talked about similar problems, but in terms of lattices and bases reduction in lattices. Now it's easy to reduce the Dirichlet problem to a shortest lattice vector problem. So I wrote to them, and it turned out that if I could solve the Dirichlet problem, then they could factor polynomials in polynomial time.

This was actually very surprising. One would think that factoring an integer should be easier than factoring a polynomial. But it turns out that it's the other way around, polynomials can be factored in polynomial time. So that is how this joint paper came about. Then a couple of years later Lagarias and Odlyzko discovered that this algorithm can be used to break the so-called knapsack crypto system. Since then this algorithm is used a lot in checking the security of various crypto systems.

As far as we understand it has applications way beyond anything that you imagined?

Lovász. Yes, definitely. For example, shortly after it was published it was used by Andrew Odlyzko and Herman te Riele in a very extended numerical computation to disprove the so-called Mertens conjecture about the ζ-function in prime number theory. But the point that I want to stress is that the whole thing started from something that was apparently not so important. Grötschel, Schriver and I just wanted to get the nicest possible theorem about equivalence of optimization and separation. This, however, was the motivation for proving something that turned out to be very important.

The ellipsoid method

Indeed, in 1981 you published a paper together with coauthors Grötschel and Schrijver entitled "The ellipsoid method and its consequence in combinatorial optimization", a paper which is widely cited, and which you touched upon in your previous answer. There is a prehistory to this, namely a paper by a Russian, Khachiyan, containing a result that was regarded as sensational. Could you comment on this, and how your joint paper is related to his?

Lovász. Khachiyan gave the first polynomial time algorithm for linear programming using what is called the ellipsoid method today. I should say that in the Soviet Union at that time there were several other people who worked on similar results, but he proved the necessary details. So it was Khachiyan who proved that linear programming can be solved in polynomial time.

Of course, everybody got interested. In the theory of algorithms before that there existed these mysterious problems that in practical terms could always be efficiently solved, but there was no polynomial time algorithm known to them. So we got interested in it, and we realized that to apply Khachyian's method you don't have to have an explicit description of the linear program. It's enough if the linear program is given in such a way that if you ask whether a point is a feasible point, then you should be able to tell this, and you should be able to find them if any constraints are violated. That observation was made by several people, including Karp and Papadimitriou, and I think Padberg and Rao. We realized that in combinatorial optimization there are many situations like this.

Then I met Martin Grötschel, and he came up with a way to apply these methods to another old problem, namely to find the chromatic number of a perfect graph in polynomial time, which was also an unsolved problem in those days. And for that it turned out that you have to apply this ellipsoid method, not only to linear programs, but to convex programs more generally. We worked on this together with Lex Schrijver, who visited the University of Szeged for a year where we shared an office, and started to see what happens in general in convex optimization and how to apply this. This is how we came up with this result that I mentioned, the equivalence of separation and optimization, it was sort of the main outcome of this study. Eventually we wrote a monograph about this subject.

The zig-zag product

Expander graphs have been a recurring theme for the Abel Prize. Last year we had Margulis, who constructed the first explicit expander graphs, after Pinsker had proven that they existed. Gromov, who won the Abel Prize in 2009, used expanders on Cayley graphs of fundamental groups, which were relevant for the study of the Baum–Connes conjecture. Also Szemerédi, who won the Abel Prize in 2012, made use of expander graphs. In 2000, you, professor Wigderson, together with Reingold and Vadhan, presented the zig-zag product of regular graphs, which is, as far as we understand, analogous to the semidirect product in group theory, by which you gave explicit constructions of very large and simple expanders. Could we just start by asking: what is the zig and what is the zag?

Wigderson. So, maybe I should start with what is an expander graph? You should think of networks, and you should think that one desirable property of networks is that there would be sort of

fault tolerance. If some of the connections are severed you would still be able to communicate. It could be computer networks, or it could be networks of roads that you would like to be highly connected. Of course, you don't want to pay too much, so you would like these networks to be sparse, that is, you don't want to have too many connections. You want a large graph in which the degree of every vertex – that is, the number of connections to every vertex – is small, let's say constant, for example ten.

A random graph will have this property, and the whole question – this is what Pinsker realized – becomes: can you describe such graphs, and can you find them efficiently? Margulis gave the first construction using this deep algebraic concept, namely Kazhdan's property (T). They can also be built using results by Selberg and others.

Then people started to simplify the proofs. By the time I was teaching this material there were reasonably simple proofs, like the one given by Jimbo and Maruoka, and you could teach it in a class in an hour or two; it's just basically Fourier transform on finite groups. So you have everything you want, you have a very nice explicit construction, you can even prove it in a class to undergraduates, but to me it was, as with many proofs based on algebra, so mysterious. I mean, what is going on? What is really behind the fact that these are highly connected graphs? This was sort of an obsession of mine for years, and I did not know what to do with it.

In 2000, just after I moved to the IAS, I had two postdocs here, Salil Vadhan and Omar Reingold. We were working on a completely different project about pseudorandomness, where an important notion is the notion of an extractor, which has something to do with purifying randomness. I will not talk about that now, but we were trying to build better extractors. As we were doing this we realized that one of our constructions may be useful towards creating expanders. The constructions in the extractor business were often iterative, and they have a very different combinatorial nature than constructions, say, of the algebraic type. Once we realized this we understood that we had a completely different combinatorial construction of expanders, but more than that, one in which, for me, it was evident from the proof why these graphs are expanding.

This is the zig-zag result; the zig-zag name was actually suggested by Peter Winkler. The construction starts with a small graph which is expanding, and one uses it to keep boosting another graph to be an expander. So you plug this little graph in somehow, and you get a bigger expander, and then you repeat this to get a bigger one, and so on. So you can generate arbitrary large expanders. This local construction has some zig-zag picture in it if you look at it, but that is not the important thing.

There is another way of describing an expander which I think is more intuitive. An expander is a graph such that, no matter what distribution you have on the vertices, if you take a vertex from this distribution and go from this vertex to a random neighbour, the entropy of the distribution increases. That is another way to describe expanders, and this you see almost with your eyes in the zig-zag construction. You see how the entropy grows, and that is what I like about this way of looking at it.

To try to get a picture of what is going on: as far as we understand you have a graph and you place this other graph at all the vertices. Then you have to decide how to put the edges in. Then essentially what you are doing, just like in the semidirect product situation where you have the multiplication rule, you move a little bit in one of the vertices, then you jump all the way to the next vertex, and then you do the similar jump there. Is that correct, vaguely?

Wigderson. It's completely correct, and moreover the connection to semidirect products was something we realized two or three years later with Alexander Lubotzky and Noga Alon. It was sort of a challenge that I felt early on, namely that the graphs that we got were expanders, they were combinatorially generated, we understood them, and I was wondering whether our construction could be useful to construct Cayley graphs. And then with Noga Alon and Alexander Lubotzky we realized it's not just similar, but the zig-zag product is a combinatorial generalization of semidirect products of groups applied to Cayley graphs. It's more general and it specializes in the case of Cayley graphs to semidirect products. For example, because of this you can prove that Cayley graphs of groups that are not simple can be expanding with a constant number of generators. No algebraic method is known to give that.

This has been used extensively in many situations, and one of the things one perhaps should mention is that the symmetric logspace and the logspace are the same, as shown by Reingold in 2004. This seems to be an idea that really caught on. Are you still using it yourself, or have you let your "baby" grow up and run into the mathematical community?

Wigderson. I think it's great that we have a mathematical community. Many of our ideas have been taken to places beyond my imagination. There is something fundamental about this construction, and it was used like you said in this Reingold result, which can more simply be described as the logspace algorithm for connectivity in graphs. In fact, it goes back to a result of Lovász and his collaborators, and can be viewed as a randomization result.

Lovász with Karp, Aleliunas, Lipton and Rackoff showed in 1980 that if you want to test whether a large graph is connected, but you have no memory, you just need enough memory to remember where you are, then by tossing coins you can explore the whole graph. This is the random logspace algorithm for graph connectivity. Derandomizing this algorithm was another project of mine that I never got to do, but Reingold observed that if you take the zig-zag product and applied it very cleverly to their randomized algorithm, you get the deterministic logspace algorithm for the same problem. So it's a particular pseudorandom generator tailored to this. It was

also used in the new PCP-theorem of Irit Dinur. So, yeah, there is something general with this zig-zag product that other people find extremely useful.

Mutual influence

Actually, this brings us to an interesting place in this interview, because here we are seeing connections between what the two of you were doing.

Wigderson. Let me mention one of the most influential things that happened to me in my postdoc years. It was in 1985. I was a postdoc in Berkeley, and there was a workshop going on in Oregon in which Lovász gave ten lectures. I don't remember exactly what it was called, but there were lectures on optimization, geometry of numbers, etc. It was a whole week of lectures and everybody wanted to hear Lovász' talk, and everybody appreciated how extremely clear his presentation was.

But the most important thing I got out of this is what Lovász described himself when you asked him the question about the LLL-algorithm, and its relation to the work on the ellipsoid and so on. He stressed how a high level point of view, rather than one focused on a specific problem, can connect lots and lots of areas of mathematics of great importance. Lovász described to you how a question that was a bit peculiar, namely about having a more elegant solution to a problem in optimization, led to solving the lattice basis reduction problem, and how it was connected to Diophantine approximation, as well as how it connects to cryptography, both to breaking crypto systems and creating crypto systems. And, you know, you get this panoramic view where everything fits in with everything. I was extremely influenced by this, it was an amazing memorable event in my early career.

Lovász. I think I have some similar memories. The zero-knowledge proof was such a shockingly exciting thing that I learned about, and it sort of showed me how powerful these new ideas of cryptography, and theoretical computer science in general, how very powerful they are. I was always very interested in Wigderson's work on randomness, even though I was sometimes trying to go the opposite direction, and find examples where randomness really helps.

One has to add that this is sometimes a matter of the model, of the computational model. I mentioned some results about convex optimization, convex geometry, algorithmic results in high dimensional convexity, and it's a basic problem there that if you have a convex body, how can you compute the volume? One of my Ph.D. students at the time, György Elekes, came up with a beautiful proof showing that you need exponential time to approximate this volume, even within a constant factor. That was in our model in which we formulated this equivalence of optimization and separation of convex bodies given by a separation oracle. A few years later, and that is actually another thing that Wigderson said, Dyer, Frieze and Kannan came up with a randomized algorithm to compute the volume, or to approximate the volume, in polynomial time with an arbitrary small relative error.

The interesting thing is the dependence on the dimension. If the dimension is n then their algorithm took n^{29} steps. Obviously this was very far from being practical, but that started their flow of research. I was also part of it and I really liked this result, and I was quite interested in making it more efficient and understanding why the exponent is so high. And then the exponent went down nicely from 29 to 17, to 10, to 7, to 5, to 4. It stood for a long time at 4, but a year ago it went down to 3. So now this is close to being practical. It's still not, n to the cube is still not enough to be a really fast algorithm, but it's definitely not ridiculously way off.

Two comments about this example. Firstly, because it's a different computational model, provably randomness helps. It's provable that without randomness it takes exponential time, and with randomness it's now down to a decent polynomial time. And the second is that polynomial time is an indicator that this problem has some deep structure. You explore this deep structure, and eventually you can improve the polynomial time to something decent.

Graphons

Here is a question to you, professor Lovász, on a subject where you have made major contributions: what is a limit theory for graphs, and what are graph limits good for? Also, explain what a graphon is.

Lovász. I will try to be not too technical. A graph is often given by an adjacency matrix, so you can imagine it as a zero-one matrix. And now, suppose that the graph is getting bigger and bigger, and you have this sequence of matrices. We always think of these as functions on the unit square, where we just cut into smaller squares, each square carrying a zero or a one. And now these functions in some sense tend to a function on the unit square, which may be continuous, or, at least not discrete any more, and that is a graphon. So, for example, if the graph is random, so each square is randomly one or zero, then it will tend to a grey square, that is, to an identical one half graphon. So a graphon is a function on the unit square, which is measurable and symmetric, and it turns out that you can exactly define what it means that a sequence of graphs converges to such a graphon.

Now, a lot of properties of the graphs are preserved, that is, if all the graphs in the sequence have a certain property, then the limit will also have this property. For example, if all these graphs have some good eigenvalue gap – a property that expanders have – then the limit will also have a good eigenvalue gap. Here we are considering dense graphs. So you look at this space of graphons,

and then you have to prove – and there is a lot of technical details there – that the space of graphons in an appropriate metric is compact. This is very convenient to work with, because from then on you can, for example, take a graph parameter, let's say density of triangles. It can be defined in the limit graphon what the density of triangles is, and then in this limit graphon there will be a graphon which minimizes this under certain other conditions.

So you can play the usual game which you play in analysis, that studies the minimum, the minimizer, and then you try to determine whether it's a local minimum, or a global minimum. All these things that you can do in analysis, you can do in this setting, and this all has some translation back to the graph theory.

It's worthwhile mentioning that the Regularity Lemma of Szemerédi is closely related to the topology of graphons. In particular, compactness of the space of graphons implies a strong form of the Regularity Lemma.

The Shannon capacity

Professor Lovász, in 1979 you published a widely cited paper titled: "On the Shannon capacity of a graph". In this paper you determine the Shannon capacity of the pentagon by introducing deep mathematical methods. Moreover, you proved that there is a number, now called the Lovász number, which can be computed in polynomial time. The Lovász number is the upper bound of the Shannon capacity associated to a graph. Could you tell us a little more about that, and explain what the Shannon capacity is?

Lovász. I will not give a formal definition of what the Shannon capacity is, but you have an alphabet and you are sending messages composed of the letters of the alphabet. Now certain letters are confusable or confoundable, so they are not clearly distinguished by the recipient. You want to pick a largest subset of words which can be sent without danger of confusion. For any two words there should be at least one position where they are clearly distinguishable. So if the alphabet is described by the vertices of a graph, an edge between two letters means that those two letters are confusable. Shannon came up with this model, and he determined the capacity. If you are sending very long words, how many words can you send without causing confusion? That number grows exponentially, and the base of this exponential function is the Shannon capacity.

The pentagon graph was the first one for which the Shannon capacity was not known, and I introduced some technique called the orthogonal representation, which enabled me to answer this question.

This is an example of one of those things that occasionally happen, namely when you answer a question, then all of a sudden it begins to have its own life. For example, it was used to determine the chromatic number of perfect graphs. In a very different direction, recently a group of physicists found some quite interesting applications of it in quantum physics. So all of a sudden you hear that something you did has inspired other people to do something really interesting. That is very pleasing.

© Eirik Furu Baardsen/DNVA

The Erdős–Faber–Lovász conjecture

Our last mathematical question to you, professor Lovász, is about the so-called Erdős–Faber–Lovász conjecture, a conjecture that was posed in 1972. How did it come about, and what were your initial thoughts on how difficult it would be to prove it? Quite recently the conjecture has been proved by Kang, Kelly, Kühn, Methuku and Osthus. We should also add that apparently Erdős considered this to be one of his three most favourite combinatorial problems.

Lovász. The background for this problem was that there was a meeting in August 1972 at Ohio State University, where we discussed hypergraph theory, which was just beginning to emerge as an interesting topic. The idea is that instead of having a standard graph where an edge always has two endpoints, you can instead look at structures where an edge has three endpoints, or five endpoints, and so on. These are called hypergraphs, and the question was: given any particular question in graph theory, like chromatic number, connectivity, etc., how can this be generalized to hypergraphs?

One of these questions was what is called the edge chromatic number in graph theory. It's a well known variant of the chromatic number problem, in which case you colour the edges, not the vertices, and you want that edges incident with the same vertex should get different colours. And then you can ask the same question about hypergraphs and what upper bound you can give on the number of different colours needed. We came up with this observation that in all the known examples the number of vertices was an upper bound on the number of colours needed to edge-colour the hypergraph.

A few weeks after this meeting at Ohio State, I was visiting the University of Colorado, Boulder, and so was Erdős. Then Faber gave a party, and we began to discuss mathematics, that is what mathematicians usually do at parties, and so we came up with this question.

Erdős didn't really believe this to be true. I was more optimistic and thought maybe it is true. It certainly was a nice conjecture, stating that the number of vertices was an upper bound on how many colours is needed. Then we realized that the conjecture had some nontrivial special cases, like something called the Fisher inequality in the theory of block designs. And that is where we got stuck. The conjecture became more and more famous, it's a very elementary question, very simple to ask. Nobody could actually get a good grip on it. Eventually Jeff Kahn was able, maybe 10 years ago or so, to prove it with a factor of $1 + \varepsilon$, for every positive ε.

A year ago, Daniela Kühn and her students were able to prove it, at least for every large enough n. One peculiar feature of this conjecture is that you make a conjecture based on small n, and then you can prove it for very large n. And what is in between often remains a question mark. She gave a talk about it at the European Congress a couple of months ago, and it was very convincing, so I think it's now proved.

Quantum interactive proofs

In January 2020, five people, Ji, Natarajan, Vidick, Wright and Yuen announced that they had proved a result in quantum complexity theory that implied a negative answer to Connes' embedding problem in operator algebra theory. This came as a total surprise to a lot of people, included the two of us, as we are somewhat familiar with the Connes problem, a problem whose proof has withstood all attacks over the last forty plus years. That a problem which seems to have nothing to do with quantum complexity theory should find its solution within the latter field is astonishing to us. Professor Wigderson, do you have any comments?

Wigderson. Ever since this result came out I have tried to give popular lectures about the evolution of the particular field that is relevant to this result, namely interactive proofs, specifically the study of quantum interactive proofs and how it connects to the MIP* = RE result, as well as to particular questions, like the Connes embedding problem and the Tsirelson problem in quantum information theory. Of course, every particular result might be surprising, but I am not at all surprised by this connection. By now we have lots and lots of places all over mathematics where ideas from theoretical computer science, algorithms and, of course, discrete mathematics, are present and reveal their power.

As for the connection to operator algebras, and specifically to von Neumann algebras, it's not so mysterious as it may seem, because of quantum measurements involving applications of operators. The question of whether these operators commute is fundamental in the understanding both of quantum information theory and in the power of quantum interactive proofs. I was more focused on the reason that possibly a proof could be obtained in the realm of quantum interactive proofs, and not in classical quantum information theory.

If you look at the formulation of quantum interactive proofs – particularly the MIP* ones of multiple provers – and you compare them to the EPR paper, the famous Einstein–Podolsky–Rosen Gedankenexperiment testing quantum mechanics, you see the same picture. You see there a two-prover interactive proof like you see in the more recent complexity theoretic quantum interactive proofs. If you look at the history of studying such experiments or proofs, in the physics world the focus was on particular different types of problems. There are several famous ones, like the Bell inequalities. Whereas it's very natural for people studying quantum interactive proofs to study them as a collection. There is a collection of games, some games reducible to each other, and the proof that MIP* = RE is a sequence of amazing reductions and ampli-

fication results using various quantum coding theory techniques and so on, even PCP techniques. This complexity-theoretic way of looking at things builds a better understanding of how they behave as a whole, and I think that is the source of the power of this approach, and the applications come from the final result just because the objects of study are operators on a Hilbert space.

Non-commutative optimization

Professor Wigderson, you are currently working on something that appears to us to be quite different from what you have been working on earlier. You call it noncommutative optimization, and it seems to us that you are doing optimization in the presence of symmetries of certain noncommutative groups, general linear groups and stuff like that. It seems like a truly fascinating project with connections to many areas. Would you care to comment a little bit on what you are doing here?

Wigderson. First of all, it's completely true that it's very different from anything that I have done before, because it's more about algorithms than about complexity. Even more, it's using far more mathematics that I did not know about beforehand. So I had to learn, and I still have to learn much more mathematics, especially invariant theory, representation theory and some algebraic geometry that I certainly was not aware of and never needed before.

This again shows the interconnectivity in mathematics, in particular, what is used from different areas of mathematics in order to obtain efficient algorithms and for obtaining other results in discrete mathematics. This connection, of course, goes in the other direction as well and enriches these mathematical areas.

This project started from something that is very dear to me, namely the derandomization project that I have been thinking about for thirty years. One of the simplest problems which we know has a probabilistic algorithm, but that we don't know have a deterministic counterpart – I mean without assumptions – is the testing of algebraic identities. You can think of the Newton identities between symmetric polynomials, you can think of the Vandermonde identity, there are lots and lots of algebraic identities in mathematics.

If anybody conjectures an algebraic identity, what do you do, how do you check it? You may think about these as polynomials with many variables. Of course, you can not expand them and compare coefficients, because this would take exponential time since there are exponentially many coefficients. Well, there is a sure probabilistic way. What we do is just to plug random numbers into the variables and evaluate the polynomials in question, and compare the results. This will be correct with high probability. So there is a fast probabilistic algorithm for this problem of polynomial identity testing, and we don't know if a fast deterministic one exists.

About twenty years ago Kabanets and Impagliazzo realized something absolutely fundamental, namely that if you find a deterministic polynomial time algorithm for this problem, you would have proved something like P different from NP. The analogue in algebraic complexity theory is that you would have proven that the permanent is exponentially harder to compute than the determinant. In short, a hardness result which will be a breakthrough in computer science and mathematics!

First of all, I would like to say that this statement should be shocking, because a fast algorithm implies hardness of a different problem. It implies a computational hardness result, which is amazing. Even before this result it was a basic problem to try to derandomize, and there were various attempts in many special cases that I worked on and others worked on. And, of course, this result made these attempts far more important.

Some years ago the issue of what happens with polynomials or rational functions that you are trying to prove are equivalent, are not with commuting variables, but are rather with noncommutative variables. It became evident that we needed it in a project here with two postdocs, Pavel Hrubes and Amir Yehudayoff. We started working on the noncommutative version of testing algebraic identities; it's basically the word problem for skew fields, so it's a very basic problem. It became apparent from our attempts that invariant theory was absolutely crucial for this problem. So understanding the invariants of certain group actions on a set of matrices, as well as understanding the degree of the generating invariants of such actions, became essential.

So I started learning about this and kept asking people in this area, and then I started collaborating with two students in Princeton, Ankit Garg and Rafael Oliveira. Eventually, cutting a long story short, together with Leonid Gurvits we found a deterministic polynomial time algorithm for solving this problem in a noncommutative domain, for noncommutative variables. Nothing like this was known, even a randomized algorithm was not known, and it uses essentially results in invariant theory.

And then we were trying to understand what we did. For the last five years I have repeatedly attempted to better understand what we did, to understand the extent of the power of these types of algorithms. What are the problems they are related to or can solve, and what these techniques can do, and what, in general, is the meaning of this result?

I should say something about applications of this. It turns out that it captures a lot of things that seemed to be unrelated. It's useful not just for testing identities, but also for testing inequalities, like the Brascamp–Lieb inequalities. It's good for problems in quantum information theory, it's good for problems in statistics, for problems in operator theory. It seems to be very broad.

Now all these algorithms just evolve along the orbit of a group action on some linear space. That is the nature of all of them. Many of these problems we are looking at are not convex, so standard convex optimization methods don't work for them. But

these algorithms work. And what we understood was that these algorithms can be viewed as doing convex optimizations, standard first order, second order methods, that are used in convex optimization, except, instead of taking place in Euclidean space, they take place in some Riemannian manifold, and the convexity that you need is the geodesic convexity of that space.

By now we have a theory of these algorithms, but, of course, there is plenty that we don't understand. The growing number of application areas of this I find very fascinating. Of course, I am hoping that eventually it will help us to solve the commutative case and understand what works and what does not work there.

LL and AW are super heros

To our delight also some young Koreans have discovered that you are mathematical super heroes. Your two sons have a common Ph.D. advisor at Stanford, Jacob Fox, and this was seized upon by a South Korean popular science journal aimed at a younger audience, where you and your sons are depicted as various characters from Star Wars. As high profile scientists, do you feel comfortable being actual heroes with lightsabers and what not?

Lovász. I always like a good joke, so I think this was a great cartoon.

Wigderson. I also loved it, and I think that it just shows that one can always be more creative in getting younger audiences excited about mathematics in ways that you did not expect before.

© dongascience <mathdonga>

Is science under pressure?

There is a question we would like to ask that has nothing to do as such with mathematics, and that is: do you feel that science is under pressure and is this something that mathematicians can and should engage in?

Lovász. I think that is true, science is under pressure. The basic reason for that, as far as I see it, is that it has grown very fast, and people understand less and less of what is going on in each particular science, and that makes it frightening, that makes it alien. Furthermore, that also makes it more difficult to distinguish between what to believe and what not, to distinguish between science and pseudoscience. This is a real problem. I think we have to very carefully rethink how we teach students in high school. Now, mathematics is one of the areas where the teaching of it is really not up to what it could be. I guess about 90 % of the people I meet say: I have always hated mathematics.

I think we are not doing our job of teaching well. I am saying this in spite of that some of my best friends are working on trying to improve mathematics education. Many people realize that there is a problem there, but it's very difficult to move ahead. I have less experience with other areas, but just looking from the outside I can see how biology today is different from what I studied in biology in high school. It's clear that it's a huge task there in front of the scientific community.

Mathematics should play central role because a lot of sciences are using more and more mathematics, not only statistics, which is sort of standard. For example, network theory or, of course, analysis and differential equations, and quantum physics, which is really also mathematics; it's a complicated area of multilinear algebra, so to say. I think the problem is there and that we should do something about it.

On behalf of the Norwegian Mathematical Society and the European Mathematical Society, and the two of us, we would like to thank you for this very interesting interview, and again, congratulations with being awarded the Abel Prize!

Wigderson. Thank you!

Lovász. Thank you!