ICM 1998, Berlin, Aug. 18–27

Abstracts of Plenary and Invited Lectures

Section:

14. Mathematical Aspects of Computer Science

1991 MS Classification: 68-Q15

Ajtai, Miklós, IBM Almaden Research Center, USA

Worst-case Complexity, Average-case Complexity and Lattice Problems

The goal of complexity theory is to describe the necessary resources, in terms of time, memory etc. for the solutions of computational problems. For cryptographic applications it would be particularly important to know that certain problems (e. g. finding the prime factors of a large integer) cannot be solved in a reasonable amount of time. (In fact the popular RSA public-key cryptosystem is based on that assumption.) Unfortunately we do not have yet any results of this type. Still we may get some information about the (relative) hardness of such problems if we accept as an axiom the hardness of a well-known computational problem which was attacked for a long time by many mathematicians without success (that is, we accept that there is no polynomial time solution of the problem in the size of the input) and prove from this axiom the hardness of other problems or the security of a cryptographic protocol. E. g. factoring integers, finding the discrete logarithm can be such problems.
Another similar solution would be to accept as an axiom that there is a problem in NP (that is a problem where the correctness of a proposed solution can be checked in polynomial time) which has no polynomial time solution. This is the famous $P \neq NP$ conjecture. There are known problems (namely each NP-complete problem) whose hardness follows from this assumption. E. g. "find a Hamilton-cycle in a given graph" is an NP-complete problem. Unfortunately these methods (either we choose the hardness of a famous problem or an NP-complete problem as an axiom) has an inherent limitation. The problems in both categories are so-called worst-case problems. That is, they are hard in the sense that finding a solution is assumed to be difficult only for some unknown values of the input and can be very easy for other values. E. g. there are integers whose factors are very easy to find. For cryptographic applications we have to present a hard instance of the problem, that is, a particular input where the solution cannot be found easily. The practical solution in the case of factoring e. g. is to pick the integer as the product of two random primes with some additional constraint to make sure that factoring is not made easier by the specific structure of the prime factors. That is, we use an average-case problem instead of a worst-case problem. We assume now that this problem whose input is chosen at random is difficult on the average (or for almost all choice of the input). We

gave up however our original requirement namely that we use only simply stated and well-studied problems. The algorithmic theory of average-case propblems are ususally only a few decades long, while the history of certain worst-case problems go back for hundreds of years. The statement of an average-case problem is also generally less clear-cut because of the many possible choices of the parameters involved in the randomization. In the case of the Hamilton cycle problem it is not even clear what would be a good randomization.

There is however a possiblitiy which unites the advantages of the two (worst-case average-case) methods. Namely we need an average-case problem which is just as difficult as a well-known worst-case problem. There are two different worst-case problems concerning short vectors for lattices which has been used in this way to create average-case problems which are at least as difficult as the original worst-case problem and can be used for various cryptographic purposes. It is important that individual random instances of these average-case problems can be created together with a known solution. To formulate these problems we need some basic defintions about lattices.

A lattice is a subset of the n-dimensional space $\mathbf{R}^n$ over the reals which consist of the integer linear combinations of n fixed lineraly independent vector. Such a set of vectors will be called a basis of the lattice. The history of finding short vectors in lattices goes back to the works of Gauss and Dirichlet. With the fundamental results of Minkowski about a hundred years ago the theory of lattices became a separate branch of number theory with a huge literature. Finding short vectors in a lattice (in various possible senses) was always one of the main goals of this theory.

The two mentioned worst-case problems are the following:

(1) *Find a basis $b_1, ..., b_n$ in the n-dimensional lattice L whose length, defined as $\max_{i=1}^{n} \|b_i\|$, is the smallest possible up to a factor of n^c, where c is constant.*

(2) *Find the shortest nonzero vector in an n dimensional lattice L where the shortest vector v is unique in the sense that any other vector whose length is at most $n^c\|v\|$ is parallel to v, where c is a sufficiently large absolute constant.*

Problem (1) is equivalent to the problem of finding a single vector shorter than a given number in a class of randomly generated lattices, with a positive probability (Ajtai). Therefore finding a short vector in a random lattice is just as difficult (with high probability) as finding a short basis in the worst-case. This random construction also gives a one-way function which leads to some cryptographic tools like pseudo-random number generators. A different cryptographic application namely a collision-free hash function were given by Goldreich, Goldwasser and Halevi. Problem (2) is somewhat weaker than Problem (1), but it seems to be more easily applicable for cryptographic protocols, e. g. its hardness guarantees the security of a public-key cryptosystem (Ajtai and Dwork). Another completely different public-key cryptosystem based on the hardness of lattice problems (without

worst-case average-case connection) was proposed by Goldreich, Goldwasser and Halevi.
The talk will also survey results which try to determine the hardness of these (or closely related) problems in the hierarchy of complexity classes. It is unlikely that Problem (2) is NP-hard for $c > \frac{1}{2}$ since it would lead to a collapse in the computational hierarchy (Goldreich and Goldwasser). However if we drop the uniqueness requirement from the problem, that is, we want to find the shortest vector (under the Euclidean norm) then the problem is NP-hard at least for randomized reductions (Ajtai). The proof of this result uses lattices constructed from logarithms of small primes. This type of lattice construction was originally used by Adleman to reduce factoring to the shortest vector problem (for the proof of correctness he used number theoretical conjectures about the distribution of smooth numbers.)
The shortest vector problem is NP-hard even in some approximate sense. Micciancio has proved recently, that the problem "find a vector which is longer than the shortest vector only by a constant factor c, where c is smaller than $\sqrt{2}$", is NP-hard. He also proved that the NP-hardness of the shortest vector problem for deterministic reductions follows from a natural number theoretic conjecture about the existence of square-free smooth numbers in long enough intervals.