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

Miklós Ajtai

Abstract. There is a need both from a theoretical and from a practical point of view to create computational problems (in NP) that are hard (that is, they have no polynomial time solutions). Currently there are no methods to prove that such problemx exist at all. We may assume however as an axiom, that certain problems are hard, where the choice of the problems may have historical or theoretical motivations. These problems however are usually worst-case problems, while, e.g. for cryptographic application, we need hard average-case problems. In this paper we desrcibe two different average-case problems, and their cryptographic applications, which are at least as difficult as some well-known worst-case problems concerning lattices.

1991 Mathematics Subject Classification: lattice, worst-case, average-case, complexity, basis, shortest vector 68Q15
Keywords and Phrases: lattice, worst-case, average-case, complexity, basis, shortest vector

1. Introduction. 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 problems 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 recently in this way to create average-case problems which are at least as difficult as the original worst-case problems 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 linerly independent vectors. 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 reader may find more information about lattices e.g. in [6] and [11]. The more modern algorithmic theory of lattices is described in [12].)

The two mentioned worst-case problems are the following:

(P1) *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.*

(P2) *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 **(P1)** 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 (see Ajtai [2]). 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 in [9] . Problem (**P2**) is somewhat weaker than Problem (**P1**), but it seems to be more easily applicable for cryptographic protocols, e.g. its hardness guarantees the security of a public-key cryptosystem (see Ajtai and Dwork [4]). 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 (see [10]).

2. THE CONSTRUCTION OF A RANDOM LATTICE. In this section we describe a way to generate random n-dimensional lattices so that, if we can find, with a positive probablitiy and in polynomial time a short vector in the random lattice Λ (where the probability is taken for the generation of Λ), then the worst-case problems (**P1**) and (**P2**) can be solved in polynomial time. (This assumption also implies that it ia possible to approximate the length of the shortest vector in an arbitrary lattice up to a polynomial factor in polynomial time. This is, again a worst-case problem.) The proofs of the results described in this section can be found in [2].

The definition of the random class. The definition of the lattices will depend on a parameter n. n can be any positive integer. (The meaning of n is the following: if it is possible to find a short vector easily in the random lattice generated with parameter n, then the n dimensional worst-case problem (**P1**) have a polynomial time solution.) The dimension of the random lattice will be somewhat larger, about $cn\log n$ for some constant c. The lattices in the random class will be subsets of $\mathbf{Z}^m$, that is, they will contain only vectors with integer coordinates. (m will be defined later as a function of n). We will fix an integer q as well (it will be also a function of n) and the lattices will be defined in a way that the fact whether a vector belongs to the lattice or not will depend only on the modulo q residue classes of its coordinates.

Assume that the positive integers n, m and q are fixed, for the moment in an arbitrary way, and $\nu = \langle u_1, ..., u_m\rangle$ where $u_1, ..., u_m \in \mathbf{Z}^n$ is an arbitrary sequence of length m from the elements of $\mathbf{Z}^n$. We define a latice $\Lambda(\nu, q)$ in the following way: $\Lambda(\nu, q)$ will consist of all sequences $\langle h_1, ..., h_m\rangle$ of integers of length m with the property: $\sum_{i=1}^m h_i u_i \equiv 0 \pmod{q}$.

Our definition of the random class will depend on the choice of two absolute constant c_1 and c_2. Assume that n is fixed let $m = [c_1 n\log n]$ and $q = [n^{c_2}]$. For each n we will give a single random variable λ so that $\Lambda = \Lambda(\lambda, q)$ is a lattice with dimension m. (The existence of a polynomial time algorithm which finds a short vector in Λ will imply the existence of such an algorithm which solves the mentioned problems in every lattice $L \subseteq \mathbf{R}^n$.)

First we define an "idealized" version λ' of λ, which we can define in a simpler way. The disadvantage of λ' is that we do not know how to generate λ' together with short vector in $\Lambda(\lambda', q)$. Then we define λ (in a somewhat more complicated way) so that we can generate it together with a short vector in $\Lambda(\lambda, q)$ and we will also have that $P(\lambda \neq \lambda')$ is exponentially small. This last inequality implies that if we prove our theorem for $\Lambda(\lambda', q)$ then it will automatically hold for $\Lambda(\lambda, q)$ too.

Let $\lambda' = \langle v_1, ..., v_m \rangle$ where $v_1, ..., v_m$ are chosen independently and with uniform distribution from the set of all vectors $\langle x_1, ..., x_n \rangle$ where $x_1, ..., x_n$ are integers and $0 \leq x_i < q$. To find a short vector in the lattice $\Lambda(\lambda', q)$ is equivalent of finding a solution for a linear simultaneous Diophantine approximation problem. Dirichlet's theorem implies that if c_1 is sufficiently large with respect to c_2 then there is always a vector shorter than n. (The proof of Dirichlet's theorem is not constructive, it is based on the Pigeonhole Principle applied to a set of exponential size.)

Definition of λ. We randomize the vectors $v_1, ..., v_{m-1}$ independently and with uniform distribution on the set of all vectors $\langle x_1, ..., x_n \rangle \in \mathbf{Z}^n$, with $0 \leq x_i < q$. Independently of this randomization we also randomize a $0,1$-sequence $\delta_1, ..., \delta_{m-1}$ where the numbers δ_i are chosen independently and with uniform distribution from $\{0, 1\}$. We define v_m by $v_m \equiv -\sum_{i=1}^{m-1} \delta_i v_i \pmod{q}$ with the additional constraint that every component of v_m is an integer in the interval $[0, q-1]$. Let $\lambda = \langle v_1, ..., v_m \rangle$. (If we want to emphasize the dependence of λ on n, c_1, c_2 then we will write λ_{n,c_1,c_2}.) It is possible to prove that the distribution of λ is exponentially close to the uniform distribution in the sense that $\sum_{a \in A} |P(\lambda = a) - |A|^{-1}| \leq 2^{-cn}$, where A is the set of possible values of λ. This will imply that the random variable λ' with the given distribution can be chosen in a way that $P(\lambda' \neq \lambda)$ is exponentially small.

With this definition our theorem will be formulated in the following way: "if there is an algorithm which finds a short vector in $\Lambda(\lambda, q)$ given λ as an input, then etc." That is, we allow the algorithm whose existence is assumed in the theorem to use λ.

Definitions. 1. If v is a shortest nonzero vector in the lattice $L \subseteq \mathbf{R}^n$, and $\alpha > 1$, we say that v is α-unique if for any $w \in L$, $\|w\| \leq \alpha \|v\|$ implies that v and w are parallel.

2. If k is an integer then $\mathrm{size}(k)$ will denote the number of bits in the binary representation of k, ($\mathrm{size}(0) = 1$). If $v = \langle x_1, ..., x_n \rangle \in \mathbf{Z}^n$ then $\mathrm{size}(v) = \sum_{i=1}^{n} \mathrm{size}(x_i)$. Our definition implies that for all $v \in \mathbf{Z}^n$, $\mathrm{size}(v) \geq n$.

Theorem . *There are absolute constants c_1, c_2, c_3 so that the following holds. Suppose that there is a probabilistic polynomial time algorithm $\mathcal{A}$ which given a value of the random variable λ_{n,c_1,c_2} as an input, with a probability of at least $1/2$ outputs a vector of $\Lambda(\lambda_{n,c_1,c_2}, [n^{c_2}])$ of length at most n. Then, there is a probabilistic algorithm $\mathcal{B}$ with the following properties. If the linearly independent vectors $a_1, ..., a_n \in \mathbf{Z}^n$ are given as an input, then $\mathcal{B}$, in time polynomial in $\sigma = \sum_{i=1}^{n} \mathrm{size}(a_i)$, gives the outputs z, u, $\langle d_1, ..., d_n \rangle$ so that, with a probability of greater than $1 - 2^{-\sigma}$, the following three requirements are met:*

(1.1) *if v is a shortest non-zero vector in $L(a_1, ..., a_n)$ then $z \leq \|v\| \leq n^{c_3} z$*

(1.2) *if v is an n^{c_3}-unique shortest nonzero vector in $L(a_1, ..., a_n)$ then $u = v$ or $u = -v$*

(1.3) *$d_1, ..., d_n$ is a basis with $\max_{i=1}^n \|d_i\| \leq n^{c_3}\mathrm{bl}(L)$.*

Remarks. 1. The probability $1/2$ in the assumption about $\mathcal{A}$ can be replaced by n^{-c}. This will increase the running time of $\mathcal{B}$ by a factor of at most n^c but does not affect the constants c_1, c_2 and c_3.

2. If we assume that $\mathcal{A}$ produces a vector of length at most $n^{c'}$ for some $c' > 1$ then the theorem remains true but c_1, c_2 and c_3 will depend on c'.

3. A PUBLIC-KEY CRYPTOSYSTEM. The following public-key cryptosystem was constructed by Ajtai and Dwork (see [4]). It is secure if problem (**P2**) has no polynomial time solution. Here we give only a very high level and somewhat simplified description of the cryptosystem and its mathematical background, and we refer the reader to [4] for the exact definitions and proofs.

A public cryptosystem serves an unlimited number of participants. Each of them publishes a public key and keeps a private key. The public key is available for everybody, but the private key is known only for its owner. Assume now that Alice wants to send a message to Bob, who published a public key. (We do not assume that Alice has a public or private key.) Alice, gets Bob's public key from a directory available for everybody. Then, using Bob's private key, she encodes the message and sends it to Bob through an open channel. Bob using his private key is able to decode the message, but without this private key the message cannot be decoded.

The RSA public key cryptosystem (see [14]) for example, fulfils this requirement, provided that each participant B can find a positive integer $m_B = p_B q_B$ where p_B, q_B are primes known to B, but nobody else in the knowledge of the number n_B alone is able to find the primes p_B, q_B. Since there is no known factoring algorithm which can factor in a reasonable amount of time a number n with several hundred digits we may think that the assumption is justified. Notice however that the fact that there is no such algorithm implies only that if B would be able to find the worst possible number n_B then his private key would be safe. However Bob has no way of knowing which is the "worst" number n_B. In practice the pair p_X, q_X is generated at random with some care of avoiding such pairs where the factoring of $p_B q_B$ can be easy. Therefore the assumption used in practice is that a certain (rather complicated) average-case problem is hard (with high probability).

In the cryptosystem described below the assumption is the hardness of the worst-case problem (**P2**). Still it is proved that this assumption implies that with a probability very close to one not a single message can be broken without acces to the private key.

The private key of Bob will be a sequence of equidistant $n-1$ dimensional hyperplanes in the n dimensional real space $\mathbf{R}^n$. More precisely, Bob picks a random vector u_B with uniform distribution from the n dimensional unit ball and this vector u_B is his private key. For each integer k the set of all $x \in \mathbf{R}^n$ whose inner product with u_B is k forms a hyperplane H_k. The sequence $\langle H_i \rangle$ is the sequence of hyperplanes mentioned at the beginning. Of course Bob has only the vector u_B, we mentioned the hyperplanes only to make it easier to visualize the steps in the protocol.

The public key will be a sequence of vectors $v_1, ..., v_m$, where $m = n^{c_3}$, that Bob picks at random close to the hyperplanes. More precisely assume that Q is a large cube and U is the union of the hyperplanes H_i. Bob first picks vectors $v'_1, ..., v'_m$ independently and with uniform distribution from $Q \cap U$ (with respect to the $n-1$ dimensional Lebesgue measure). Then Bob perturbs these vectors sightly at random so that they remain close to the hyperplanes. (Their distance to the closest hyperplane remains smaller than, say, n^{-8}.) The perturbed vectors are $v_1, ..., v_m$.

It is possible to prove (assuming that (P2) has no polynomial time solution) that the sequence $v_1, .., v_m$ is computationally indistinguishable from a sequence of length m whose elements are picked independently and with uniform distribution from the cube Q. Therefore by making the public key available for anybody, Bob did not give out any information about the hyperplanes.

Knowing the public key Alice is able to generate a sequence of independent random points $x_1, ..., x_i, ... \in \mathbf{R}^n$ with identical distributions and with the following properties:

(1) with high probability x_i is very close to a hypeprlane H_k, more precisely if the distance of neighboring hyperplanes is M then there is a hyperplane H_k so that the distance of x_i and H_k is smaller than $\frac{M}{n^5}$.

(2) the distribution of x_i is computationally indistinguishable in polynomial time, form the uniform distribution on a parallelepiped $\mathcal{P}$, where $\mathcal{P}$ can be computed from the public key, so it is known to everybody.

(This last property will be a consequence of the hardness of problem (**P2**).)

For the moment we accept that Alice has a way of generating such a distribution. Assume now that Alice wants to send a single bit δ to Bob. If $\delta = 0$ then Alice picks a random point y with uniform distribution on the set $\mathcal{P}$, and sends y as the message. If $\delta = 1$ then Alice generates a random point x with the distribution of the points x_i, from the $\mathcal{P}$ and sends x as the message.

Suppose that Bob gets a message z. z is an n-dimensional vector in $\mathcal{P}$. Bob computes the inner product $\alpha = z \cdot u_B$. If α is close to an integer (say closer than $\frac{1}{n^4}$) then Bob knows z is close to a hyperplane therefore he concludes that $\delta = 1$. If the distance of α from the closest integer is greater than $\frac{1}{n^4}$, then Bob concludes that $\delta = 0$. (There is a small probability, about $\frac{1}{n^4}$, that Bob makes the wrong decision.)

Finally we sketch how can Alice generate the points x_i with the required properties. $\mathcal{P}$ will be a parallelepiped determined by n vectors from the sequence $v_1, ..., v_m$, so that the parallelepiped is relatively "fat", that is, the minimal distance between its opposite sides is not too small with respect to the length of a side of Q. (Larger then, say, n^{-2} times this length.) $\mathcal{P}$ may be the first such parallelpiped with this property or Bob can designate a parallelepiped in the public key. With a very high probability such a parallelepiped always exists. Assume that $\mathcal{P}$ is the parallelepiped determined by the vectors $v_{i_1}, ..., v_{i_n}$

Alice takes a random $0,1$ linear combination w of the vectors $v_1, ..., v_m$, then reduces it to the parallelepiped $\mathcal{P}$ modulo $v_{i_1}, ..., v_{i_n}$. In other words she adds an integer linear combination of the vectors $v_{i_1}, ..., v_{i_n}$ to the vector w so that the sum x is in $\mathcal{P}$. x has a distribution with the required properties.

4. THE NP-HARDNESS OF THE SHORTEST VECTOR PROBLEM.

We cannot prove from any reasonable assumption from complexity theory (like $P \neq NP$) that the problems (P1) or (**P2**) are hard. Actually it is unlikely that Problem (**P2**) is NP-hard for $c > \frac{1}{2}$ since it would lead to a collapse in the computational hierarchy (see Goldreich and Goldwasser [8]). 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 (see Ajtai [3]). The proof of this result uses lattices constructed from logarithms of small primes. This type of lattice construction was originally used by Adleman ([1]) 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 proof of the NP-hardness result also has a combinatorial part which is a constructive/probabilistic version of Sauer's Lemma (related to the concept VC dimension). This is the most difficult part of the proof.

The NP-hardness of the shortest vector problem was conjectured by Van Emde Boas almost twenty years ago (see [5]). He proved the analogue statement for the L_∞ norm (for deterministic reductions). The shortest vector problem in L_2 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 $c < n^{\frac{1}{2}}$" is NP-hard (see [13]). (The original proof in [3] gave only a factor $1+2^{-n^\epsilon}$ which was improved first by Cai and Nerurkar (see [7]) to $1+n^{-\epsilon}$.) Micciancio 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.

REFERENCES

[1] L. Adleman, Factoring and Lattice Reduction, Manuscript, 1995.

[2] M. Ajtai, Generating Hard Instances of Lattice Problems, Proceedings 28th Annual ACM Symposium on Theory of Computing, 1996 or Electronic Colloquium on Computational Complexity, 1996, http://www.eccc.uni-trier.de/eccc/

[3] M. Ajtai, The Shortest Vector Problem is NP-hard for Randomized Reductions. Proceedings 30th Annual ACM Symposium on Theory of Computing, 1998 or Electronic Colloquium on Computational Complexity, 1997, http://www.eccc.uni-trier.de/eccc/

[4] M. Ajtai and C. Dwork, A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence, Proceedings 29th Annual ACM Symposium on Theory of Computing, 1997 or Electronic Colloquium on Computational Complexity, 1996, http://www.eccc.uni-trier.de/eccc/

[5] P. Van Emde Boas, Another NP-complete partition problem and the complexity of computing short vectors in a lattice, Tech. Report 81-04, Dept. of Mathematics, Univ. of Amsterdam, 1980.

[6] J.W.S. Cassels, *An Introduction to the Geometry of Numbers*, Springer, 1959

[7] J.-Y. Cai, A. Nerurkar, Approximating SVP to within a factor of $1 + \dim^{\epsilon}$ is NP-hard under randomized reductions, IEEE Conference on Computational Complexity (to appear), see also Electronic Colloquium on Computational Complexity, 1997, http://www.eccc.uni-trier.de/eccc/

[8] O. Goldreich, S. Goldwasser, On the Limits of Non-Approximability of Lattice Problems Electronic Colloquium on Computational Complexity, 1997, http://www.eccc.uni-trier.de/eccc/

[9] O. Goldreich, S. Goldwasser, S. Halevi, Collision-free hashing from lattice problems, Electronic Colloquium, on Computational Complexity, 1996, http://www.eccc.uni-trier.de/eccc/

[10] O. Goldreich, S. Goldwasser, S. Halevi, Public-key cryptosystems from lattice reduction problems, Electronic Colloquium on Computational Complexity, 1996, http://www.eccc.uni-trier.de/eccc/

[11] P.M. Gruber, C.G.Lekkerkerker, *Geometry of Numbers*, North-Holland, 1987

[12] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, Algorithms and Combinatorics 2, 1988

[13] D. Micciancio, The Shortest Vector in a Lattice is Hard to Approximate within Some Constant. Electronic Colloquium, on Computational Complexity, 1996, http://www.eccc.uni-trier.de/eccc/

[14] R. Rivest, A. Shamir, L. Adelman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *CACM 21*(2), pp. 120–126, 1978

Miklós Ajtai
IBM Almaden Research Center, K/53
650 Harry Road
San Jose, CA 95120
USA