Extended Lattice Reduction Experiments using the BKZ Algorithm

Michael Schneider Johannes Buchmann

Technische Universität Darmstadt
{mischnei,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract: We present experimental results using lattice reduction algorithms. We choose the BKZ algorithm, that is the algorithm considered the strongest one in this area in practice. It is an important task to analyze the practical behaviour of lattice reduction algorithms, as the theoretical predictions are far from being practical. Our work helps choosing the right parameters for lattice reduction in practice. The experiments in this paper go beyond the results of Gama and Nguyen in their Eurocrypt 2008 paper. We give evidence of some facts stated in their work, concerning the runtime and the output quality of lattice reduction algorithms.

Keywords: Lattice Reduction, LLL, BKZ, Hermite-SVP

1 Introduction

Lattices have been known in number theory since the eighteenth century. They already appear when Lagrange, Gauss, and Hermite study quadratic forms. Former applications of lattice reduction are discrete optimization and integer programming, as well as factoring polynomials. Nowadays, lattices and hard problems in lattices are widely used in cryptography as the basis of promising cryptosystems.

Lattice reduction is also a useful tool in cryptanalysis. Various cryptosystems are broken using lattice reduction, e.g. knapsack systems [LO85, CJL$^+$92] as well as RSA in special settings [May10]. Further on, factoring composite numbers and computing discrete logarithms is possible using lattice reduction [Sch91, May10]. The security of lattice based cryptosystems is based on the hardness of lattice problems that are solved using lattice reduction. Therefore it is necessary to analyze the strength of lattice reduction algorithms.

A lattice is an additive subgroup of the Euclidean vector space $\mathbb{R}^n$, generated by a lattice basis. Its elements can be considered to be vectors in a vector space. Lattice reduction is basically the search for vectors that are short and nearly orthogonal. There are two basic notions of lattice reduction, namely *LLL-reduction* and *BKZ-reduction*. LLL-reduction is a special case of the stronger BKZ-reduction.

The most famous algorithm for lattice reduction is the LLL algorithm by Lenstra, Lenstra, and Lovász [LLL82], which outputs an LLL-reduced basis. Theoretically, the best algorithm to find short vectors is the *slide reduction* algorithm [GN08a]. In practice, the most promising algorithm is the BKZ algorithm by Schnorr and Euchner [SE91], that outputs a BKZ-reduced basis. A practical comparison of lattice reduction algorithms can be found in [NS06, GN08b, BLR08].

One major problem with lattice reduction algorithms is the fact that in practice, the algorithms like LLL and BKZ behave better than the theoretical worst-case analysis predicts. Therefore it is necessary to examine their practical, average-case behaviour. In 2008, Gama and Nguyen published a comprehensive summary of their experiments [GN08b], that helps analyzing the practical behaviour of the LLL and BKZ algorithm as well as the deep insertion variant of LLL. The results of this work are used widely when facts about the strength of lattice reductions algorithm is required, it became kind of a standard work.

Our Contribution. In this paper we present experimental results using lattice reduction algorithms that we collected during the last years. The focus of our work is on the BKZ algorithm, which is the algorithm most widely used in practice. We analyze the runtime of BKZ using high blocksizes, give details about the output quality of BKZ-reduced bases, and based on our observations present a strategy for lattice reduction in high lattice dimensions. With our work, we present heuristics that give further evidence to some arguments stated unproven by Gama and Nguyen in [GN08b]. Therefore, we extend the state-of-the-art in practical lattice reduction.

Organization of the Paper. Firstly, we present the necessary facts on lattices and lattice reduction in Section 2. Secondly, we present three main questions of this paper and answer them successively in Section 3. Finally, we give a conclusion and present open questions in the area in Section 4.

2 Preliminaries

A lattice L is an additive subgroup of the euclidean space $\mathbb{R}^d$. Let $n, d \in \mathbb{N}$, $n \leq d$, and let $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^d$ be linearly independent vectors. Then $L(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$ is the lattice spanned by the column matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$. $L(\mathbf{B})$ has dimension n, the matrix $\mathbf{B}$ is called a basis of the lattice. Such a basis is not unique, lattices are invariant under unimodular transformations. Therefore, every lattice in dimension $n > 1$ possesses infinitely many bases. We write L instead of $L(\mathbf{B})$ if it is clear which basis is concerned. The first successive minimum $\lambda_1(L)$ is the length of a shortest vector of a lattice. The lattice determinant $\det(L(\mathbf{B}))$ is defined as $\sqrt{\det(\mathbf{B}\mathbf{B}^t)}$. It is invariant under basis changes. For full-dimensional lattices ($n = d$) there is $\det(L(\mathbf{B})) = |\det(\mathbf{B})|$ for every basis $\mathbf{B}$. Throughout this paper, vectors and matrices are written in bold face, e.g., $\mathbf{x}$ and $\mathbf{M}$. We write $\|\mathbf{x}\|$ for the Euclidean norm of $\mathbf{x}$.

The lattices that arise in cryptography have a more special structure. Let $q \in \mathbb{N}$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. We define an m-dimensional lattice $\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \mathbf{A}\,\mathbf{v} \equiv \mathbf{0} \pmod{q}\}$. Those lattices are called *modular* or *q-ary* lattices. For efficiency reason, so-called *ideal lattices* are often used in cryptography. An ideal lattice of dimension m is the set of all points that belong to an ideal in the ring $R = \mathbb{Z}[x]/\langle f \rangle$, for a polynomial f of degree n. For special f, it allows to exhibit the lattice from only m representatives in $\mathbb{Z}_q$, instead of mn in the q-ary lattice case, where m typically is in $\mathcal{O}(n \log n)$.

As mentioned above, each lattice has infinitely many bases. Creating a basis consisting of short and nearly orthogonal vectors is the goal of lattice reduction.

Hard lattice problems. There are several problems on lattices that are supposed to be or proven to be hard [MG02]. The most famous problem is the shortest vector problem (SVP). The goal of γ-SVP is to find an (approximate) shortest non-zero vector in the lattice, namely a vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| \leq \gamma \lambda_1(L)$, where $\gamma \geq 1$ is the approximation factor. It is possible to formulate the problem in every norm, the most usual norm is the euclidean norm, that we are using throughout this paper.

As the length of the shortest vector $\lambda_1(L)$ might not be known, it is hard to control the approximation factor of SVP in practice. Therefore it is common practice to use the Hermite-SVP variant: given a $\gamma \geq 1$, find a non-zero vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| \leq \gamma \cdot (\det L)^{1/n}$. Recall that the determinant of a lattice is always known, as it can be computed from every basis. Having reduced a basis $\mathbf{B}$ one can easily calculate the reached Hermite factor using $\gamma_{\text{Hermite}} = \|\mathbf{b}_{\text{min}}\| / (\det L)^{1/n}$. The main computational problem in q-ary lattices $\Lambda_q^{\perp}(\mathbf{A})$ is the short integer solution problem (SIS): given $n, m, q, \mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and a norm bound ν, find $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A})$ with $\|\mathbf{v}\|_2 \leq \nu$. The SIS was first introduced and analyzed by Ajtai [Ajt96]. Numerous improvements to the analysis where made in, e.g., [MR07, GPV08]. For information about further lattice problems we refer the reader to [MG02] and [MR08].

Algorithms. The γ-SVP was solved by Lenstra, Lenstra, and Lovász in [LLL82] for factors γ exponential in the lattice dimension n. Their LLL algorithm has runtime polynomial in the lattice dimension. It is parameterized by a δ with $\frac{1}{4} < \delta \leq 1$, and outputs a basis whose first vector has length $\|\mathbf{b}_1\| \leq (\delta - \frac{1}{4})^{(1-n)/4} \cdot \det(L)^{1/n}$ [LLL82]. In other words, LLL provably reaches a Hermite factor of $(\delta - \frac{1}{4})^{(1-n)/4}$. An overview about applications and facts about LLL can be found in [NV10]. A typical choice for δ in practice is $\delta = 0.99$.

In [SE91] the authors introduce the BKZ algorithm, that is a blockwise variant of the LLL algorithm. BKZ is today's best algorithm for lattice reduction in practice. A BKZ instance using blocksize parameter β is called BKZ-β. It finds a lattice vector with length $\|\mathbf{b}_1\| \leq (\gamma_\beta)^{\frac{n-1}{\beta-1}} \cdot \lambda_1$, where γ_β is the Hermite constant in dimension β [Sch94]. The runtime cannot be proven to be polynomial in the lattice dimension, but practical experiments point out that the runtime of BKZ is polynomial in the lattice dimension. In fact, the runtime is exponential in the blocksize parameter β. This parameter allows a trade-off between runtime and output quality. Bigger blocksize causes shorter vectors at the expense of increased runtime, and vice versa.

Theoretically, the most promising algorithm for approximating shortest vectors is the algorithm of [GN08a]. In [SE91] the authors also proposed, besides BKZ, the deep insertion variant of LLL.

There are various algorithms that find a shortest lattice vector, not only an approximation. The algorithms of Kannan [Kan83], Fincke and Pohst [FP83], and Schnorr and Euchner [SE91] perform an exhaustive search for the shortest lattice vector. The algorithm used in practice today is the variant of Schnorr and Euchner, called ENUM. A second approach for solving the exact SVP are probabilistic sieving algorithms like the one of [AKS01], analyzed in [NV08], or the improved algorithm of [MV10b] and [PS10b]. The most practical sieving variant is presented in [MV10b], called Gauss-Sieve. In [MV10a] a deterministic, single exponential time algorithm based on Voronoi cell computations is presented.

Concerning implementations, there are two libraries available for lattice reduction: the NTL library of Shoup [Sho] and the fpLLL library of Stehlé et al. [CPS]. The fpLLL library does not offer BKZ. Therefore, for our experiments we use the NTL library, as was done in [NS06, GN08b, BLR08]. An improved LLL algorithm was presented as the L^2 algorithm by Nguyen and Stehlé [NS05]. It is implemented in the fpLLL library.

Practical behaviour. In practice however, lattice reduction algorithms behave much better than expected from theory. In the average case they find much shorter vectors than theoretical worst case bounds suggest. In [GN08b] Gama and Nguyen give a practical analysis of random lattices using the NTL library [Sho]. The authors state that a Hermite factor of 1.01^n and an approximation factor of 1.02^n in high lattice dimension (e.g. dimension 500) is within reach today, but a Hermite factor of 1.005^n in dimension around 500 is totally out of reach. The authors of [NS06] state similar facts for the LLL algorithm. In [BLR08] and [RS10], the runtime of different lattice reduction algorithms on q-ary lattices is analyzed. Those are the lattices that arise in lattice based cryptography, and their practical behaviour is not well studied to date.

3 Experiments

This section shows the experiments that were performed during the last two years. We provide answers to the following three questions:

- Which Hermite factor is reachable with a given blocksize?
- Is it possible in practice to run BKZ with big blocksizes?
- Is running BKZ with increasing blocksize a good strategy for lattice reduction?

All experiments are performed on random, full-rank lattices in the sense of Goldstein and Mayer [GM03], with bit size of the entries in the order of magnitude $10n$. For LLL-reduction we used a parameter $\delta = 0.99$ throughout all our experiments. We use the NTL library [Sho] in version 5.4.2. The experiments were performed on an AMD Opteron (2.3GHz) quad core processor, using one single core for the computation. For BKZ, we use the quad precision (QP1) variant of NTL.

3.1 Which Hermite Factor is Reachable With a Given Blocksize

It is well known that the blocksize parameter β allows a trade-off between runtime and reduction quality. Here we analyze the quality, in the shape of the Hermite factor. We provide a prediction for the Hermite factor that is reachable with a given blocksize β. This directly predicts the length of the shortest vector after BKZ-reduction.

For each dimension $n \in \{25, 50, \ldots, 250\}$ we generated 5 different, randomized bases for each of 5 random lattices. The entries of the input bases where bounded by 2^{10n}. These bases were reduced using BKZ with each blocksize $\beta \in \{5, 6, \ldots, 23\}$, independently. The reduced lattices were then used to calculate the Hermite factor reached in each case, i.e.,

$$\gamma_{\text{Hermite}} = \sqrt[n]{\|\mathbf{b}_1\| \cdot \det(L)^{1/n}} .$$

The mean values of all 25 bases in each dimension are depicted in Figure 1. We only show every fourth value of β, for clarity reason.

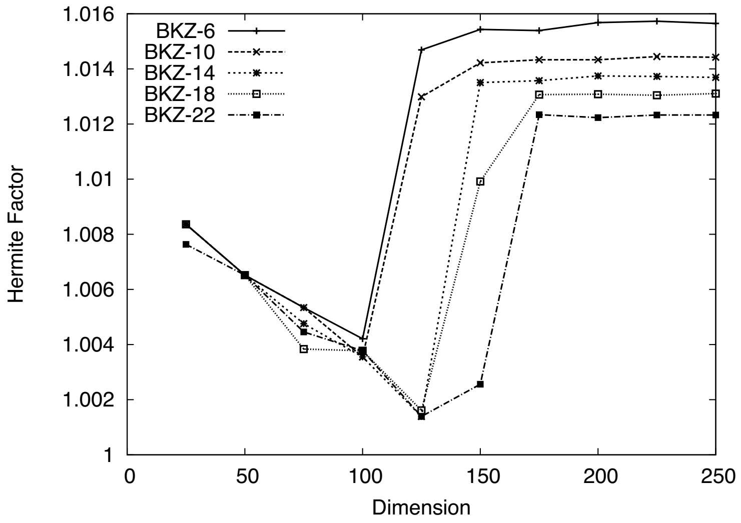

Figure 1: Hermite factor in dimension 25 to 250. The figure shows the experimental results, i.e., the average Hermite factors γ_{Hermite} reached by BKZ with different blocksize β.

One notices that in low dimensions, say less than $n = 175$, BKZ reaches very small Hermite factors. In higher dimension of $n \geq 200$, the Hermite factor for each blocksize stabilizes at a certain constant value (as was already stated in [GN08b]). In practice one is interested in higher dimensions $n \geq 200$, We extract these constants and, using least-squares fitting, we gain a fitting function (cf. Figure 2). For a given blocksize β, this function predicts the Hermite factor that can be reached with BKZ using the specified blocksize, in higher lattice dimensions. The resulting function is $f(\beta) = 1.01655 - 0.000196185 \cdot \beta$. Proposition 1 summarizes the result.

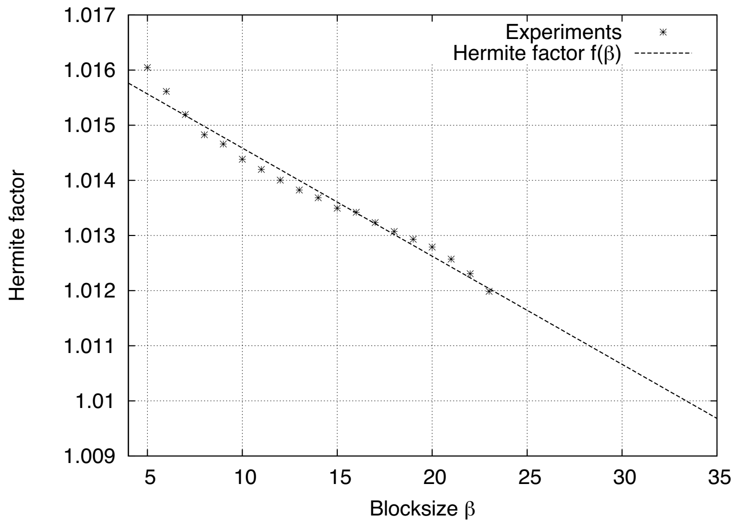

Figure 2: Mean values of the reached Hermite factors γ_{Hermite} and the fitting function $f(\beta)$. For high lattice dimensions (say $n \geq 200$), the function $f(\beta)$ predicts the Hermite factor that is reachable with a chosen blocksize β.

Proposition 1 *When BKZ is applied to a random, full-rank lattice L of dimension n using blocksize β, the output length of the shortest vector after BKZ reduction is*

$$\|\mathbf{b}_1\| \approx f(\beta)^n \cdot \det(L)^{1/n} \quad where \quad f(\beta) = 1.01655 - 0.000196185 \cdot \beta\,.$$

This analysis implies that, for example, for reaching a hermite factor of 1.005, one has to run BKZ with blocksize $\beta = 59$. Running BKZ with blocksize 50 it is possible to reach a Hermite factor $\gamma = 1.0067$. Our result is similar to that of [GN08b]. For $\beta = 20$ we compute $\gamma = 1.0126$ compared to 1.0128 ([GN08b]) and for $\beta = 28$ we get $\gamma = 1.0111$ compared to 1.0109 ([GN08b]). Therefore we extend the analysis of Gama and Nguyen when presenting our function for extrapolation of the possible Hermite factor for a given block size parameter β.

This result is very useful when attacking lattice based cryptosystems. Assume that our random lattices represent the average case of lattice reduction for ideal and q-ary lattices, that arise in lattice based cryptography. The security of most signature and encryption schemes is guaranteed as long as the SIS problem is hard for q-ary or ideal lattices, respectively [MR08, RS10]. As the lattice determinant is always known, it is easy to map a Hermite-SVP solution to a SIS solution. More precisely, when a vector $\mathbf{v} \in \Lambda_q^\perp$ with $\|\mathbf{v}\| \leq \nu$ is required, this corresponds to a Hermite factor $\gamma = \sqrt[n]{\nu/(\det \Lambda_q^\perp)^{1/n}}$ needed to render a crypto system insecure. Using our analysis, we can predict the blocksize parameter required for this. The prediction of the runtime of the chosen blocksize is the concern of our second question.

3.2 Runtime of BKZ Using Bigger Blocksizes

In [GN08b], the authors state that BKZ reduction is only practical in blocksizes β up to 30 (they present results with blocksizes up to 40). With higher blocksizes, the runtime of BKZ rises too fast to be practical, at least in high lattice dimensions. In this section, we give evidence to this argument using runtime experiments with BKZ in high blocksizes.

We run BKZ with blocksize $\beta = 40, 45$, and 50 on five random lattices of each dimension $n \in \{30, 35, \ldots, 90\}$. As stated before, higher lattice dimensions are intractable. Again, the bit size of the entries of the input bases are in the order of magnitude $10n$. The logarithmic plot of Figure 3 shows the results, including least squares fitting lines $t_\beta(x)$ that help guessing the runtime of BKZ-β in bigger lattice dimensions.

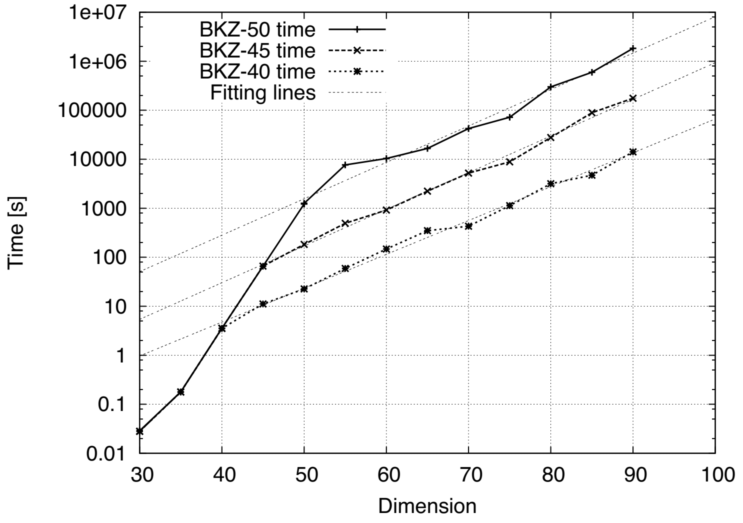

Figure 3: Runtime of BKZ-β on random lattices, for $\beta \in \{40, 45, 50\}$. The fitting lines are the linear functions $t_\beta(x)$ that can be used to predict the runtime of BKZ-β in higher dimension > 90.

For lattice dimensions $n \leq \beta$, BKZ-β behaves like BKZ-n. This can be observed in Figure 3, where the lines of all three blocksizes are the same in small dimensions. Therefore, lower dimension $n \leq \beta$ are excluded when computing the fitting functions. The resulting functions that we compute from our experiments are

$$t_{40}(x) = 10^{0.0690798 \cdot x - 2.08621} \text{ seconds},$$
$$t_{45}(x) = 10^{0.0748307 \cdot x - 1.51248} \text{ seconds},$$
$$t_{50}(x) = 10^{0.0743183 \cdot x - 0.523057} \text{ seconds}.$$

Using this we can for example predict the runtime (on state-of-the-art hardware comparable to ours) of NTL's BKZ-50 for a 108-dimension lattice to be around 1 year and for a 150-dimensional lattice around 1300 years. This concretizes the statements of Gama and Nguyen that BKZ in higher blocksizes is intractable in high lattice dimensions.

In addition, our experiments show that, when running BKZ with blocksize $\beta \geq 40$, the enumeration part takes more than 99% of the reduction time. In low blocksizes, the enumeration usually takes less then 10% of the reduction time, stated for example in [SE91, GN08b]. This is an important fact, since, during the last years, there are numerous improvements to enumeration algorithms, see [HSB+10, GNR10, DS10, PS10a], for example. These improvements will speed up BKZ in high blocksizes best, BKZ with low blocksizes will be enhanced only slightly.

3.3 Running BKZ with Increasing Block Size

It is common practice to run BKZ with increasing blocksize, that means starting with low blocksize (say, $\beta \approx 3$), fully BKZ-β reduce the basis, and increase the blocksize, until a vector of desired length is found. The question that we want to answer in this section is, if this approach is faster than directly starting with high blocksize, and if the reduction quality is concerned by the chosen strategy. Is it faster to run BKZ in increasing block size, using the pre-reduction of lower blocksizes to speed up the reduction in higher blocksize?

Again, we run BKZ on $5 \cdot 5$ random bases of dimension $n \in \{200, 300\}$, once with increasing blocksize with β from 3 up to 15, once directly running BKZ-15. We measure the total reduction time and the Hermite factor reached by the shortest output vector. The results are presented in Figure 4. The graphs show average, maximum, and minimum values of the runtime and the Hermite factor, respectively.

One notices that, concerning the runtime, starting BKZ directly with high blocksize β results in lower runtime in total. On average, the reduction finishes faster than in the increasing blocksize case. Concerning output quality, we also notice a minor advantage for the direct BKZ-15 case. This behaviour is surprising, since in both cases, the output basis is BKZ-15 reduced, and with increasing blocksize, more time was spend reducing the basis.

However, in practice it was unclear which blocksize will be necessary (and sufficient) to find a vector of suitable size. Therefore it was good practice to run BKZ with increasing blocksize, as a short vector might be reached much faster than when starting BKZ with high blocksize. With this technique, Gama and Nguyen could find the secret key of NTRU-107 lattices. The new strategy that one might apply is to use Proposition 1 to predict the blocksize needed for a desired Hermite factor. Combining the results of both sections in this paper leads to a new strategy, that will outperform the strategy with increasing blocksize.

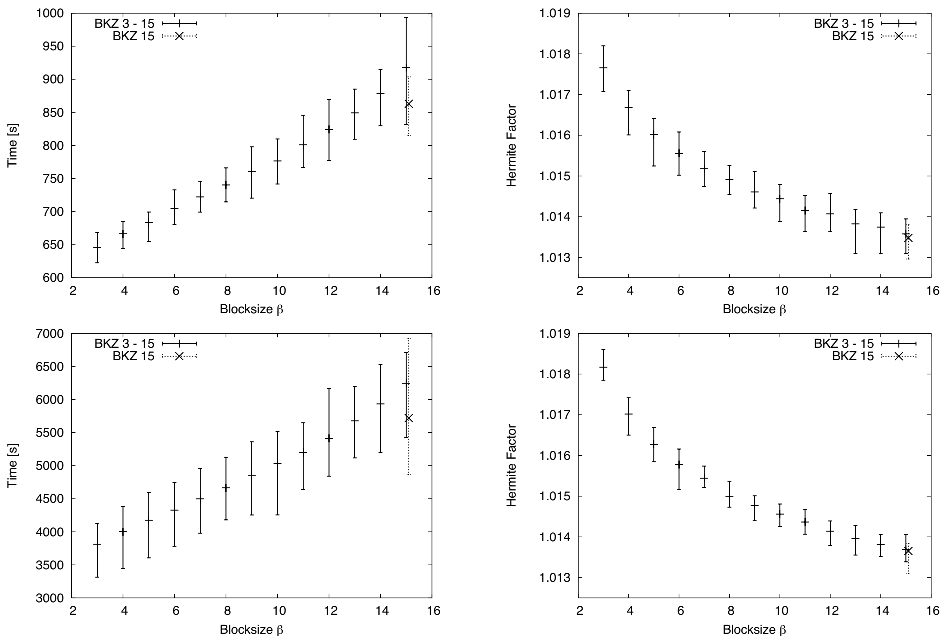

Figure 4: Running BKZ on random lattices with increasing block size β. The left pictures show the runtime, the right figures present the Hermite factor γ, both in dimensions 200 (top) and 300 (bottom). The solid data points were gained with increasing block size, the results depicted with dotted lines were started directly with blocksize $\beta = 15$. The graphs show average, maximum, and minimum values.

4 Conclusion

In this paper we have presented a prediction of the Hermite factor reachable with a given blocksize. We have shown how the choice of big blocksize effects the BKZ algorithm. These two results yield a new strategy for BKZ reduction, that replaces the strategy of increasing blocksize.

It is an open problem if lattice reduction for ideal lattices is as hard as it is for random lattices. So far, there is no algorithm that makes use of the special structure of ideal lattices, and people believe that lattice reduction in the ideal lattice case is as hard as in the regular case. In addition, it is unclear if q-ary lattices, which are more important for cryptography than random lattices, behave different to random lattices. There is need for exhaustive testing of the existing lattice reduction algorithms on q-ary and ideal lattices.

Unfortunately the NTL library is the only implementation of BKZ that is publicly available. It is evident that this version is no more up to date. Numerous improvements to SVP solvers were made during the last years. Therefore, it is necessary to test BKZ including these new enumeration and sieving variants to show its real strength today.

The *theoretical* analysis of BKZ is still in the early stages. Basic facts like worst-case runtime are still not known. There is a lot of work left to do in this area.

References

[Ajt96] Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC 1996*, LNCS, pages 99–108. ACM, 1996.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC 2001*, pages 601–610. ACM, 2001.

[BBD08] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer, 2008.

[BLR08] Johannes Buchmann, Richard Lindner, and Markus Rückert. Explicit hard instances of the shortest vector problem. In *PQCrypto 2008*, volume 5299 of *LNCS*, pages 79–94. Springer, 2008.

[CJL$^+$92] Matthijs J. Coster, Antoine Joux, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved Low-Density Subset Sum Algorithms. *Computational Complexity*, 2:111–128, 1992.

[CPS] David Cadé, Xavier Pujol, and Damien Stehlé. fpLLL - A floating point LLL implementation. Available at Damien Stehlé's homepage at école normale supérieure de Lyon, `http://perso.ens-lyon.fr/damien.stehle/index.html`.

[DS10] Özgür Dagdelen and Michael Schneider. Parallel Enumeration of Shortest Lattice Vectors, 2010. To appear in Euro-Par 2010.

[FP83] U. Fincke and Michael Pohst. A procedure for determining algebraic integers of given norm. In *European Computer Algebra Conference 1983*, volume 162 of *LNCS*, pages 194–202. Springer, 1983.

[GM03] Daniel Goldstein and Andrew Mayer. On the equidistribution of Hecke points. *Forum Mathematicum 2003, 15:2*, pages 165–189, 2003.

[GN08a] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC 2008*, pages 207–216. ACM, 2008.

[GN08b] Nicolas Gama and Phong Q. Nguyen. Predicting Lattice Reduction. In *Eurocrypt 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, 2008.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice Enumeration using Extreme Pruning. In *Eurocrypt 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206. ACM, 2008.

[HSB$^+$10] Jens Hermans, Michael Schneider, Johannes Buchmann, Frederik Vercauteren, and Bart Preneel. Parallel Shortest Lattice Vector Enumeration on Graphics Cards. In *Africacrypt 2010*, volume 6055 of *LNCS*, pages 52–68. Springer, 2010.

[Kan83] Ravi Kannan. Improved Algorithms for Integer Programming and Related Lattice Problems. In *STOC 1983*, pages 193–206. ACM, 1983.

[LLL82] Arjen Lenstra, Hendrik Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LO85] J. C. Lagarias and Andrew M. Odlyzko. Solving Low-Density Subset Sum Problems. *Journal of the ACM*, 32(1):229–246, 1985.

[May10] Alexander May. Using LLL-Reduction for Solving RSA and Factorization Problems. In Nguyen and Vallée [NV10], pages 315–348.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[MR08] Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In Bernstein et al. [BBD08], pages 147–191.

[MV10a] Daniele Micciancio and Panagiotis Voulgaris. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems based on Voronoi Cell Computations. In *STOC 2010*. ACM, 2010.

[MV10b] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *SODA 2010*, pages 1468–1480. ACM/SIAM, 2010.

[NS05] Phong Q. Nguyen and Damien Stehlé. Floating-Point LLL Revisited. In *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 215–233. Springer, 2005.

[NS06] Phong Q. Nguyen and Damien Stehlé. LLL on the Average. In *ANTS 2006*, volume 4076 of *LNCS*, pages 238–256. Springer, 2006.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve Algorithms for the Shortest Vector Problem are Practical. *J. of Mathematical Cryptology*, 2(2), 2008.

[NV10] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Springer, 2010.

[PS10a] Xavier Pujol and Damien Stehlé. Accelerating Lattice Reduction with FPGAs, 2010. To appear in Latincrypt 2010.

[PS10b] Xavier Pujol and Damien Stehlé. Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$, 2010. submitted.

[RS10] Markus Rückert and Michael Schneider. Selecting Secure Parameters for Lattice-based Cryptography. Cryptology ePrint Archive, Report 2010/137, 2010. `http://eprint.iacr.org/`.

[Sch91] Claus-Peter Schnorr. Factoring Integers and Computing Discrete Logarithms via Diophantine Approximations. In *Eurocrypt 1991*, volume 547 of *LNCS*, pages 281–293. Springer, 1991.

[Sch94] Claus-Peter Schnorr. Block Reduced Lattice Bases and Successive Minima. *Combinatorics, Probability & Computing*, 3:507–522, 1994.

[SE91] Claus-Peter Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In *FCT 1991*, pages 68–85. Springer, 1991.

[Sho] Victor Shoup. Number Theory Library (NTL) for C++. `http://www.shoup.net/ntl/`.